San José State University
Department of Computer Science

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

Nondeterministic Finite Automata

(Part 3)

Lecture 11

Day 11/31

CS 154

Formal Languages and Computability

Spring 2019

Agenda of Day 11

- Quiz +
- Summary of Lecture 10
- Lecture 11: Teaching ...
  - Nondeterministic Finite Automata (Part 3)

NAME	Alan M. Turing		
SUBJECT	CS 154	TEST NO.	Quiz +
DATE	02/28/2019	PERIOD	1 / 2 / 3

TEST RECORD	
PART 1	123
PART 2	
TOTAL	

Your **list #** goes here!

Take-Home Exam!

Quiz +

Use Scantron

Summary of Lecture 10: We learned ...

NFAs

- We introduced a new class of automata.

  Nondeterministic Finite Automata

- Very similar to DFAs
- The same building blocks
- NFAs are interesting because ...

  ... their transition graphs are simpler.

- We added two new abilities that DFAs could not have.
- We called them "two violations".
- So, NFAs behavior are similar to DFAs except for those two violations.

NFAs Behavior

1. When NFAs have zero transition, ...

   ... they halt.

2. When there are more than one transition, ...

   ... they start parallel processing.

When NFAs halt

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

OR

- It has zero transition. ≡ **z**

$$(\mathbf{c} \lor \mathbf{z}) \leftrightarrow \mathbf{h}$$

Any question?

Summary of Lecture 10: We learned ...

Accepting/Rejecting Strings

- A string is accepted iff ...

  ... at least one process accepts it.

  - For NFAs, **(h ∧ c ∧ f)** ↔ **a** is valid for accepting strings by one process.
  - Recall that for DFAs, we changed **(h ∧ c ∧ f)** ↔ **a** to **(c ∧ f)** ↔ a because h and c have the same value.
  - But for NFAs, **h** and **c** might have different values.
- A string is rejected iff ...

  ... all processes reject it.

λ-transition

- Short circuit is ...

  ... an edge with no input symbol.
- We represent it with symbol λ.
- The transition is called λ-transition.
  - In fact λ means "NO symbol".
- λ-transition in automata theory means ...

  ... the machine may unconditionally transit.

  - In the same timeframe, without consuming input.

Any question?

Summary of Lecture 10: We learned ...

NFAs

- The sub-rule of the following transition is ...

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

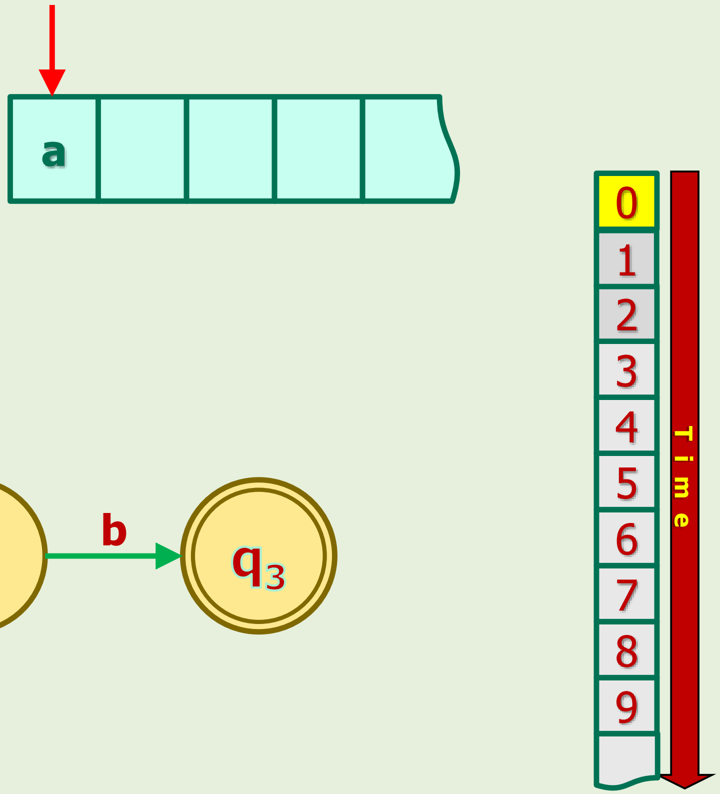
- As a general rule, when NFAs encounter multiple choices, they start parallel processing.

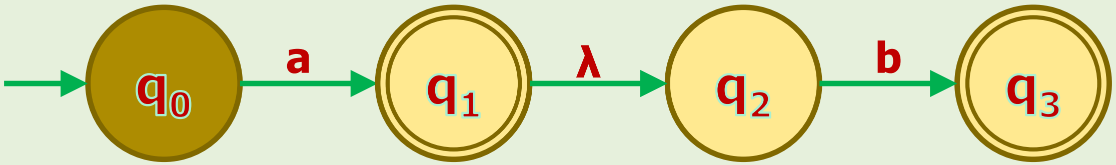
Any question?

λ-Transitions in Action

λ-Transitions in Action

Example 12: Starting Configuration

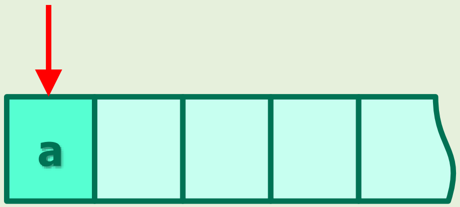


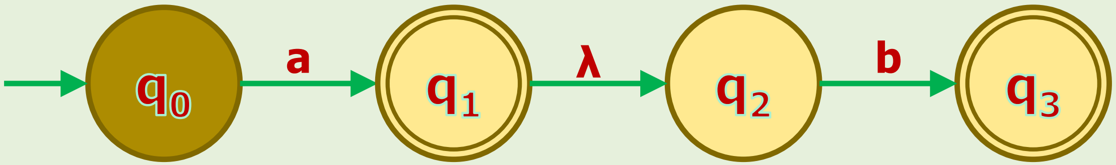


- Process #1 (main) starts normally.

λ-Transitions in Action

Example 12 (cont'd)

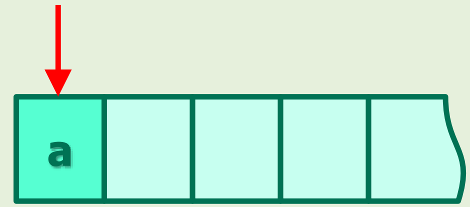




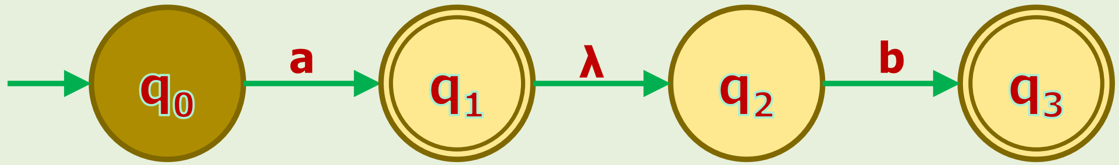
- Input tape reads 'a' and sends it to the control unit.
- The control unit makes a decision based on $\delta\ (q_0\ ,\ a) = \{q_1\ ,\ q_2\}$

λ-Transitions in Action

Example 12 (cont'd)

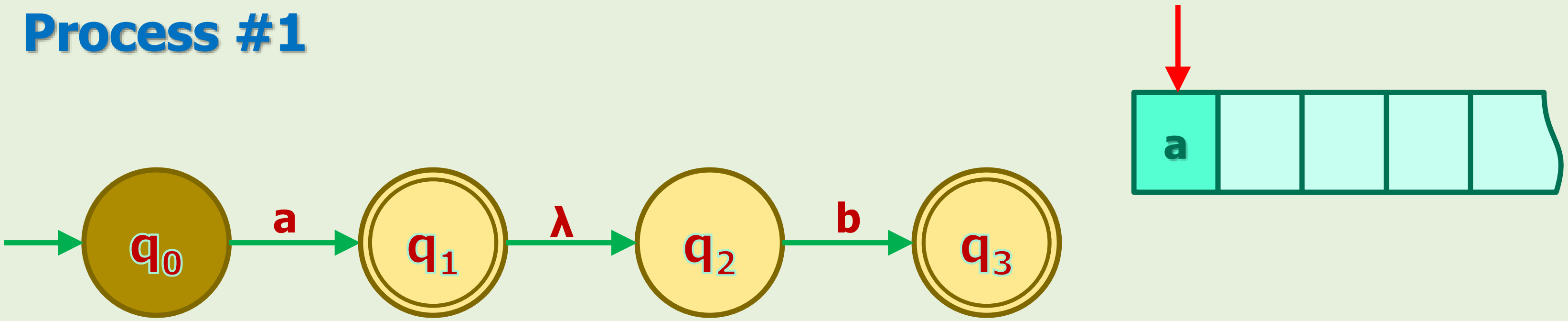


$\delta(q_0, a) = \{q_1, q_2\}$



- It encounters two possibilities: transition to q_1 or q_2.
- So, parallel processing starts!

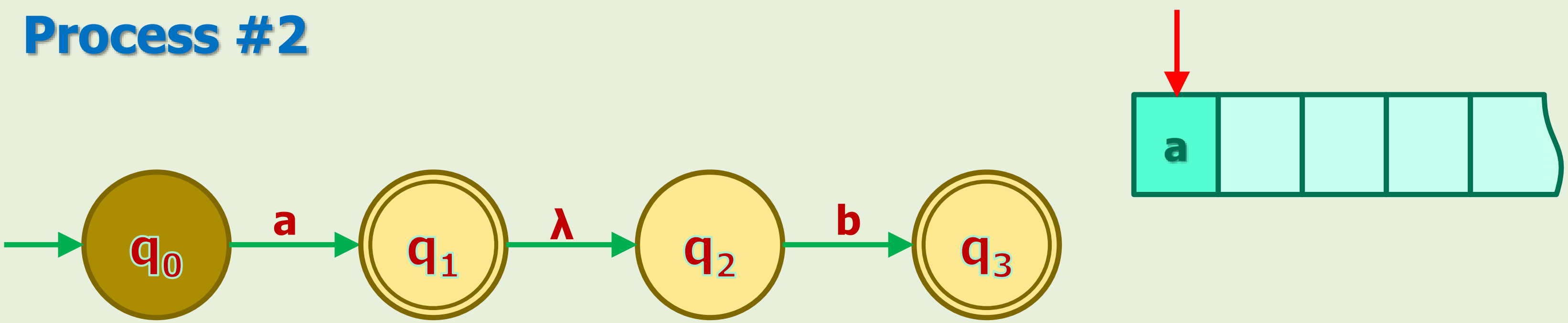
Process #1



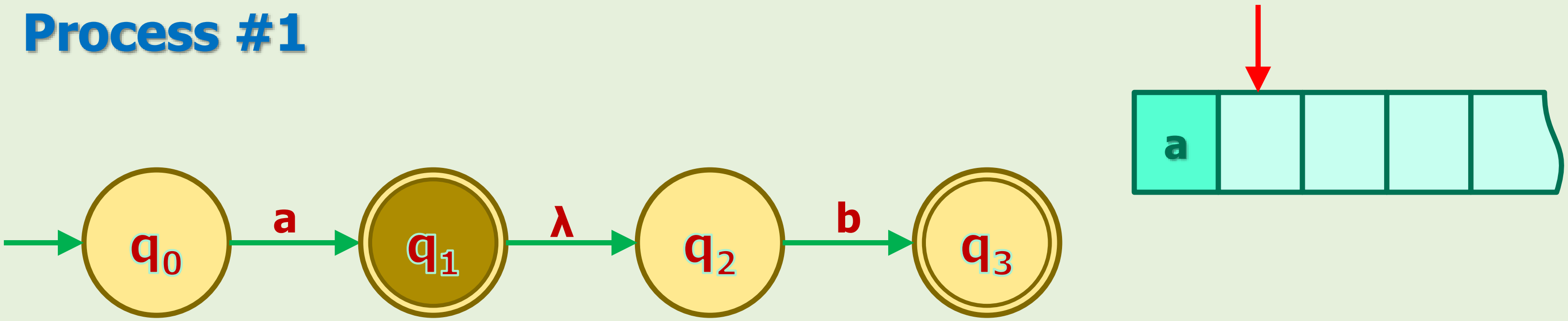
$\delta\ (q_0\ ,\ a) = \{q_1\ ,\ q_2\}$

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

Process #2



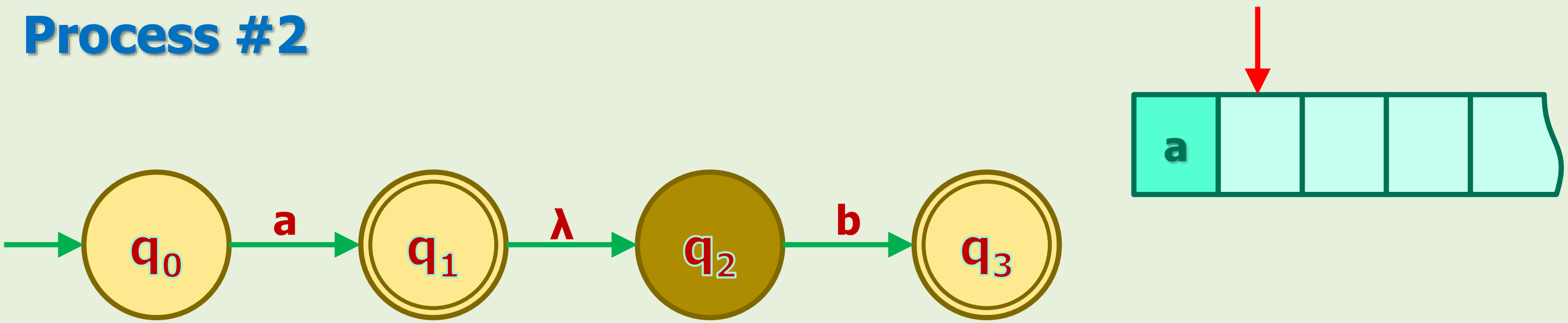
Process #1



$\delta(q_0, a) = \{q_1, q_2\}$

This process consumes 'a' and transits to q_1.

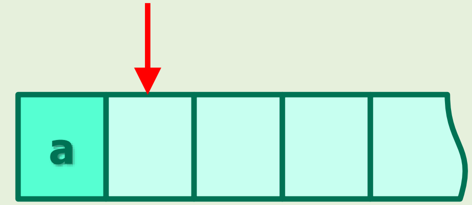
Process #2

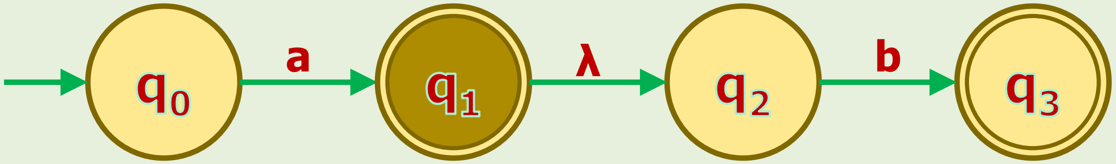


This process consumes 'a' and transits to q_2.

This is the end of timeframe 1.

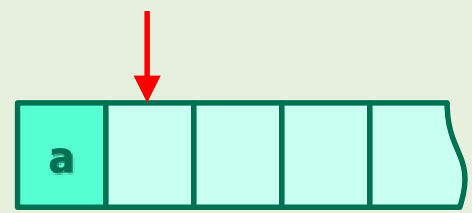
Process #1

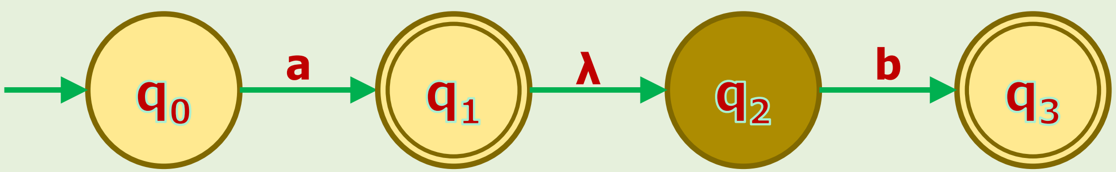




Process #1 is out of symbol and has to halt.

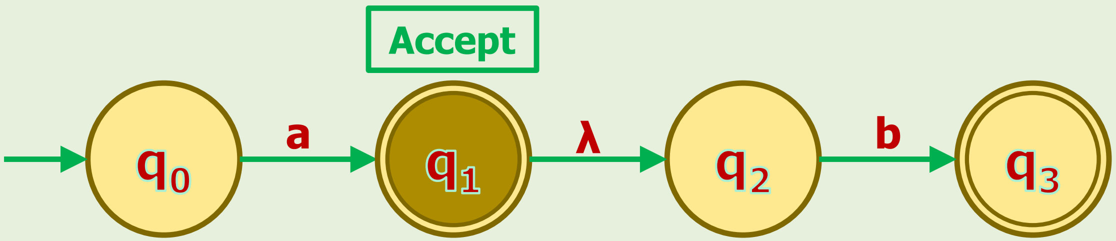
Process #2

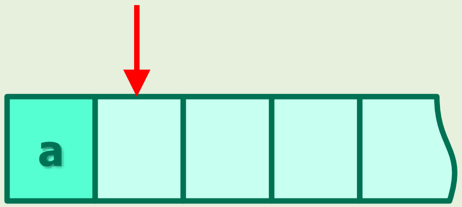




Process #2 is out of symbol and has to halt.

Process #1

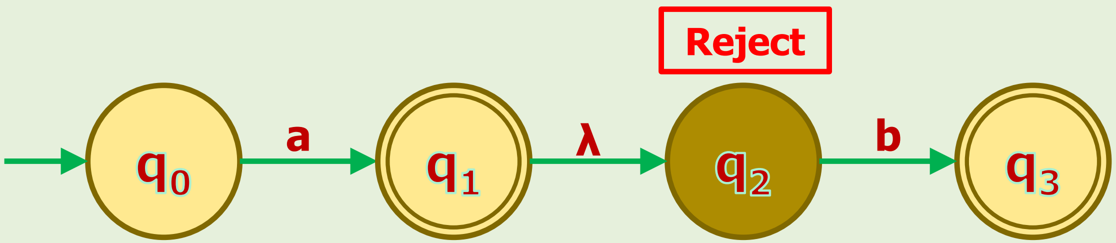


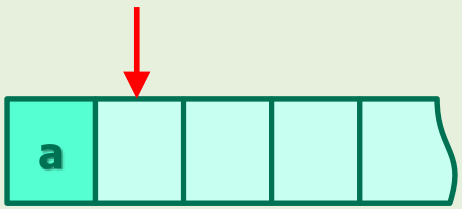


Process #1 halts in an accepting state AND all symbols are consumed.

So, process #1 accepts w.

Process #2





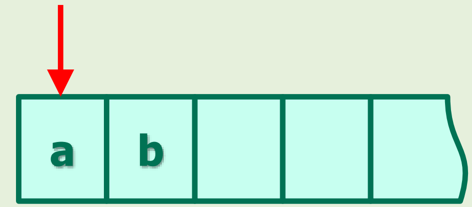
Process #2 halts in a non-accepting state.

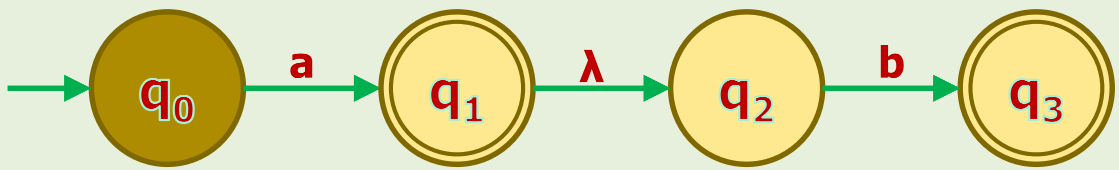
So, process #2 rejects w.

Overall Accepted

λ-Transitions in Action

Example 13: Starting Configuration

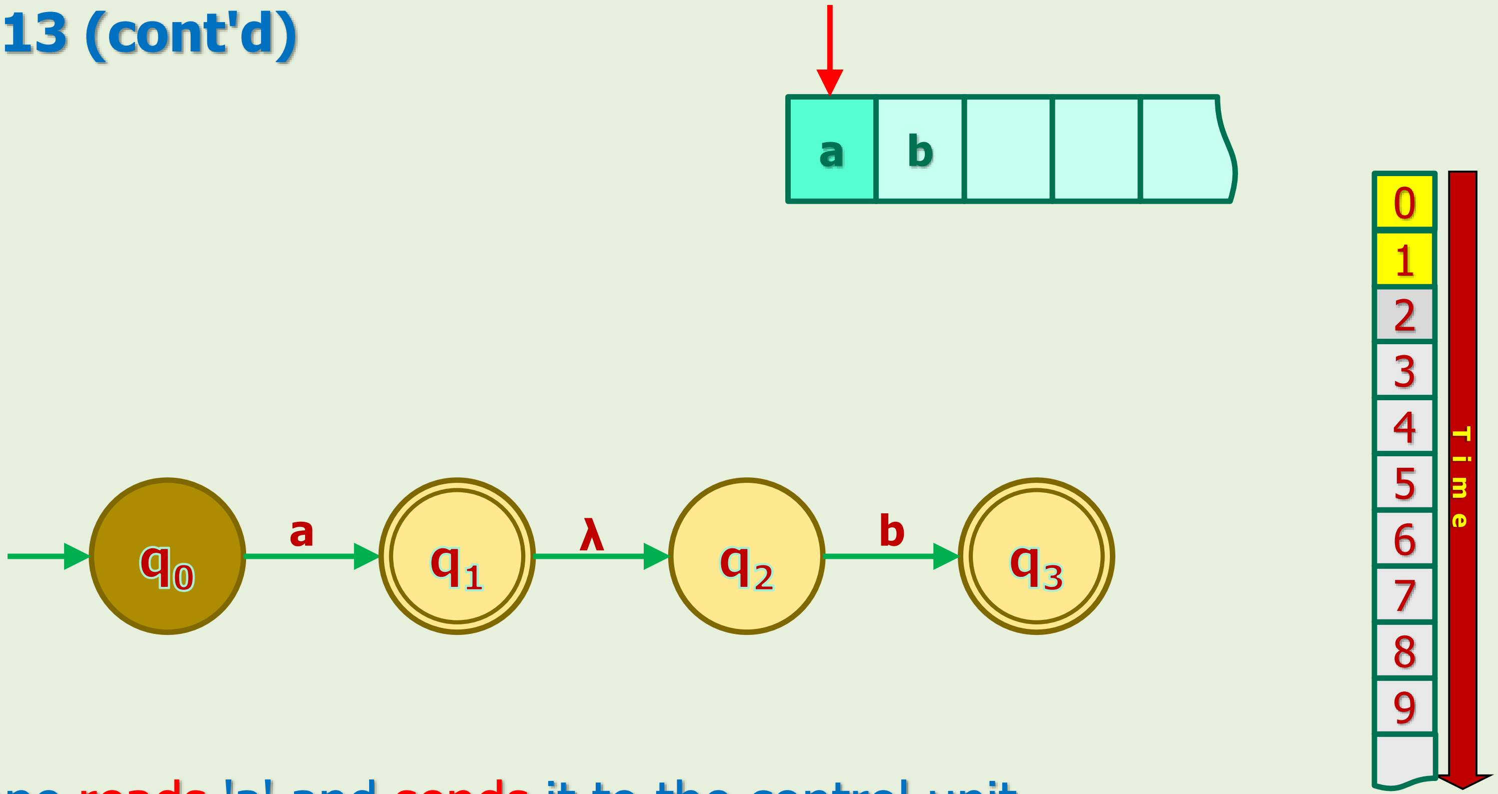




- Process #1 (main) starts normally.

λ-Transitions in Action

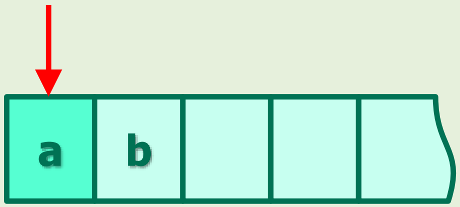
Example 13 (cont'd)



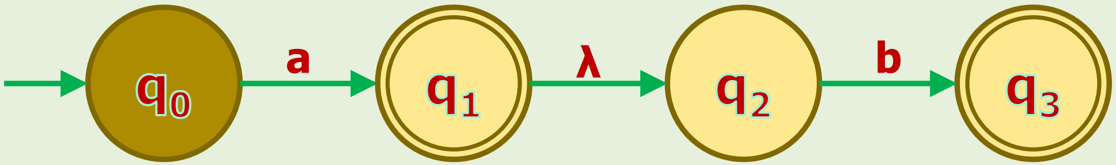
- Input tape reads 'a' and sends it to the control unit.
- The control unit makes a decision based on $\delta(q_0, a) = \{q_1, q_2\}$

λ-Transitions in Action

Example 13 (cont'd)

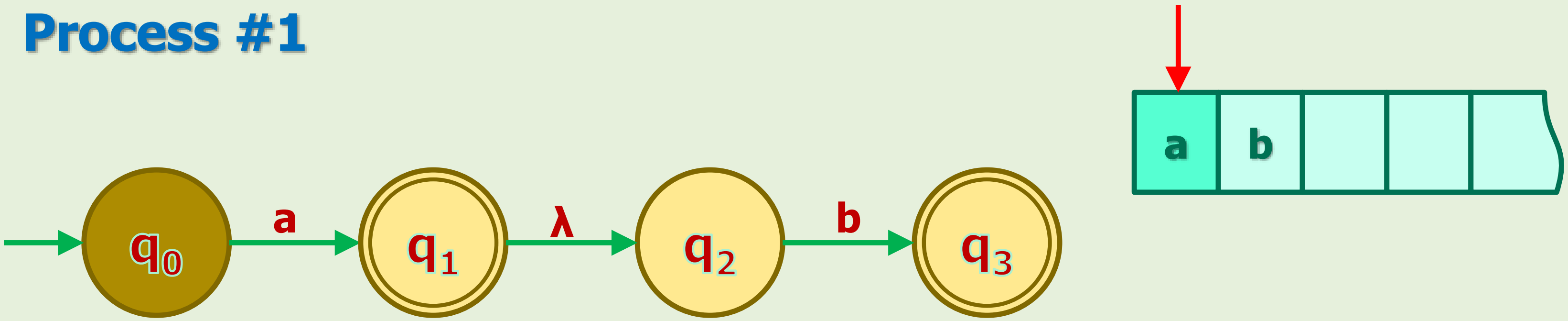


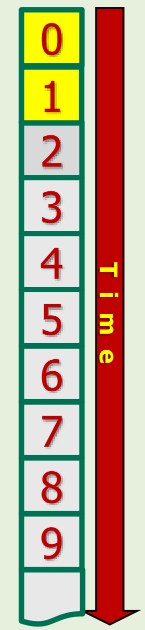
$\delta(q_0, a) = \{q_1, q_2\}$



- It encounters two possibilities: transition to q_1 or q_2.
- So, parallel processing starts!

Process #1

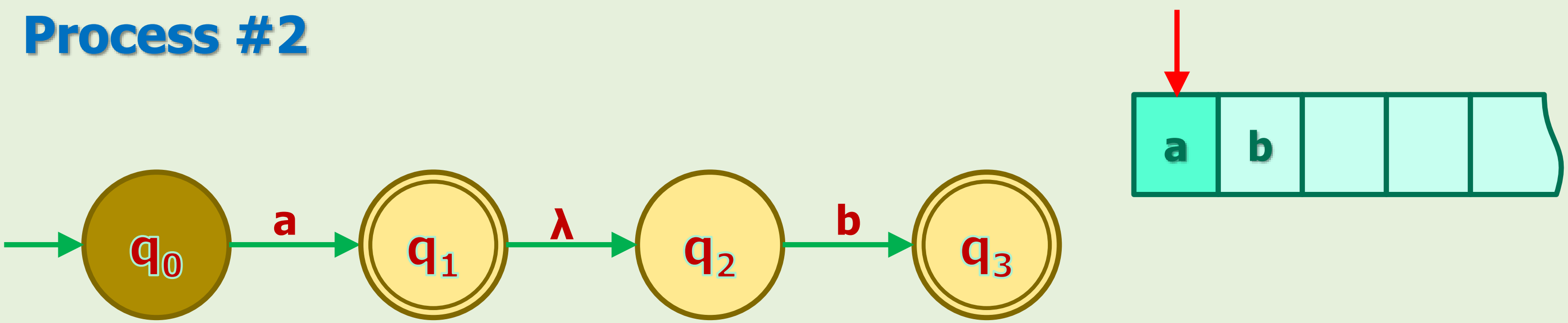




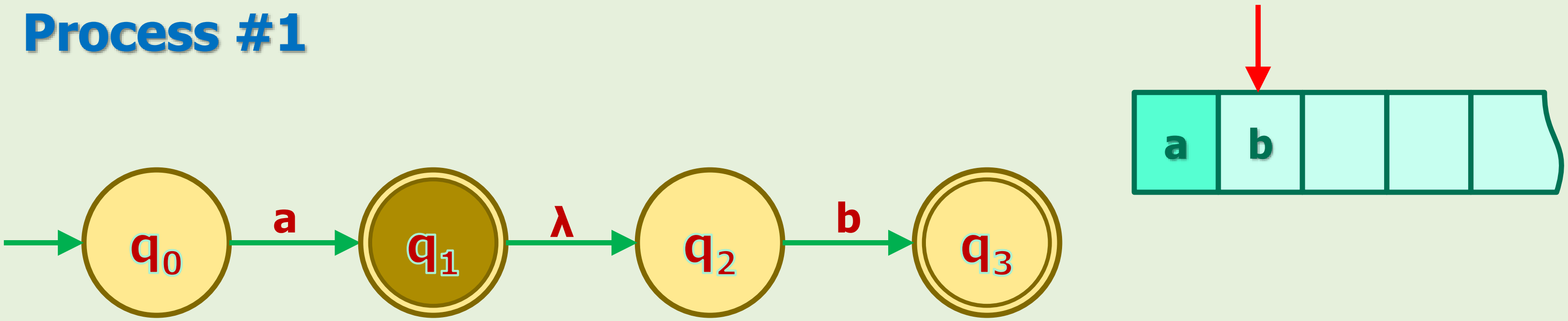
$\delta\ (q_0\ ,\ a) = \{q_1\ ,\ q_2\}$

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

Process #2



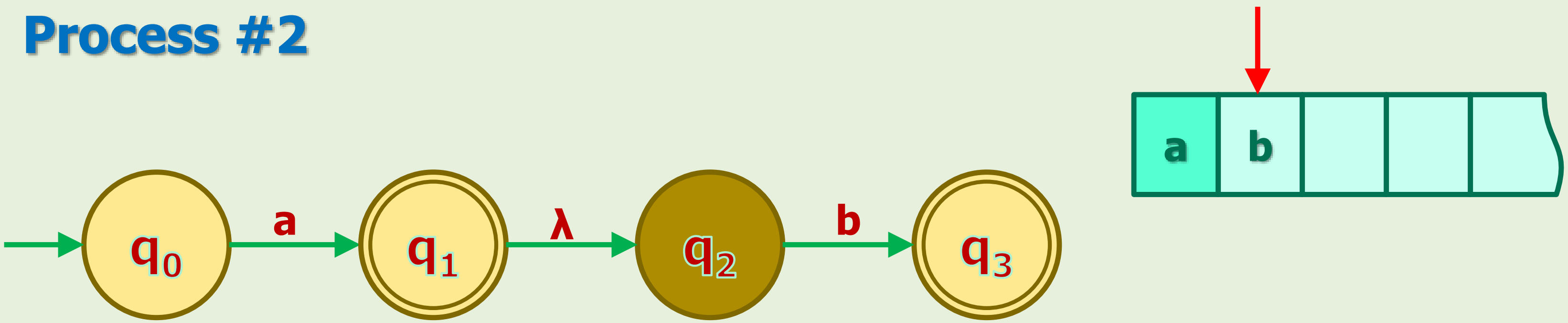
Process #1



$\delta(q_0, a) = \{q_1, q_2\}$

Process #1 consumes 'a' and transits to q_1.

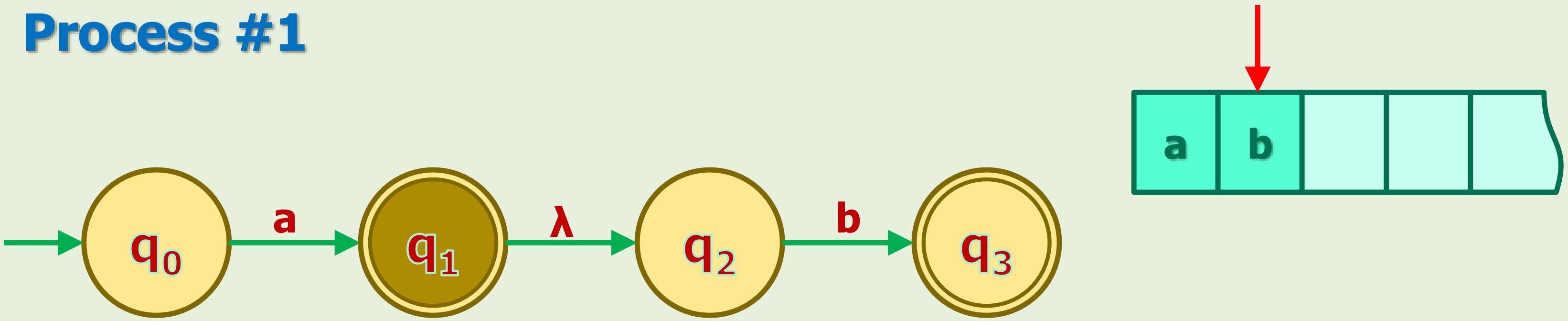
Process #2



Process #2 consumes 'a' and transits to q_2.

This is the end of timeframe 1.

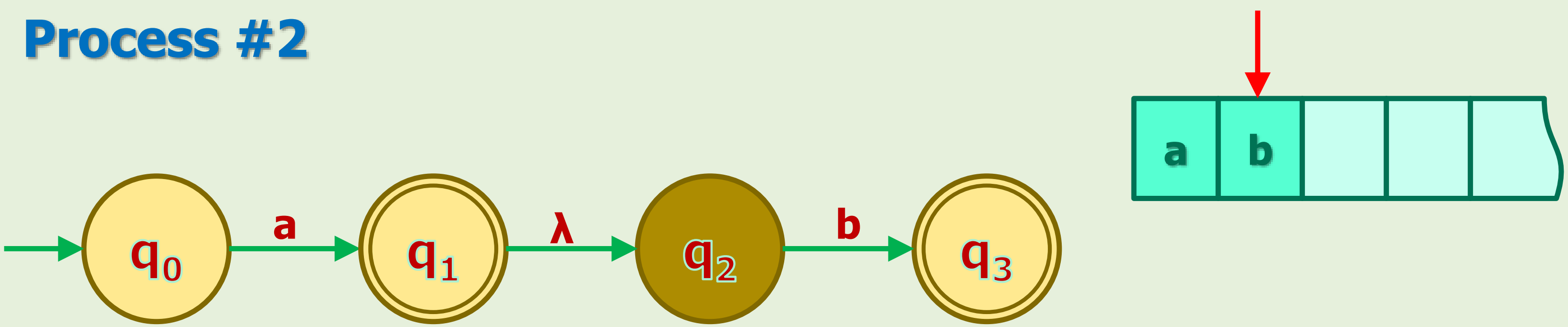
Process #1



The symbol 'b' is read and sent to the control unit.

Process #1 calculates $\delta(q_1, b) = \{q_3\}$.

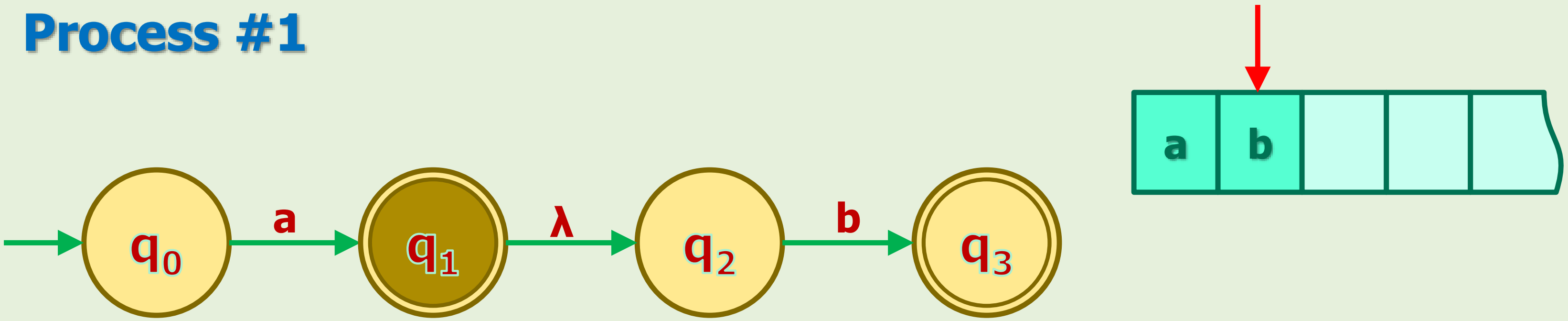
Process #2



The symbol 'b' is read and sent to the control unit.

Process #2 calculates $\delta(q_2, b) = \{q_3\}$.

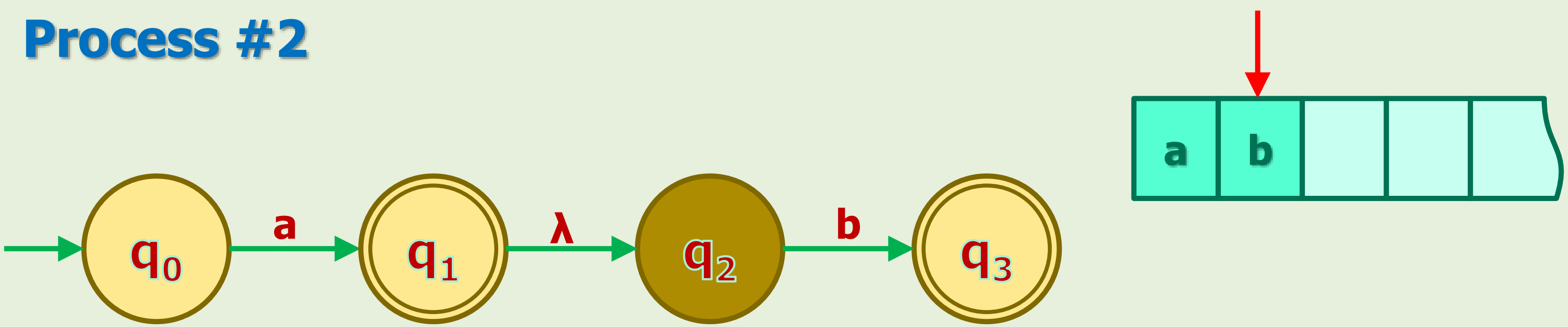
Process #1



$\delta(q_1, b) = \{q_3\}$

Process #1 consumes 'b' and transits to q_3.

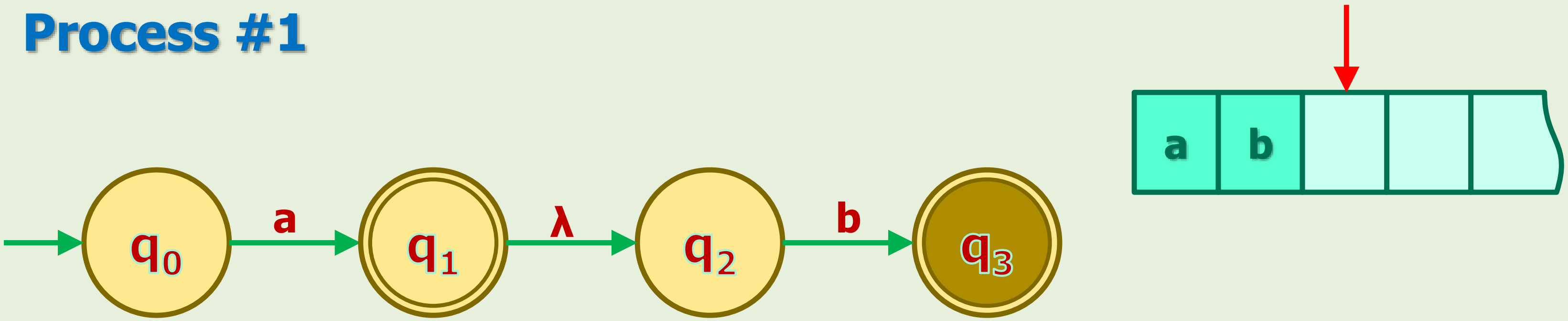
Process #2



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

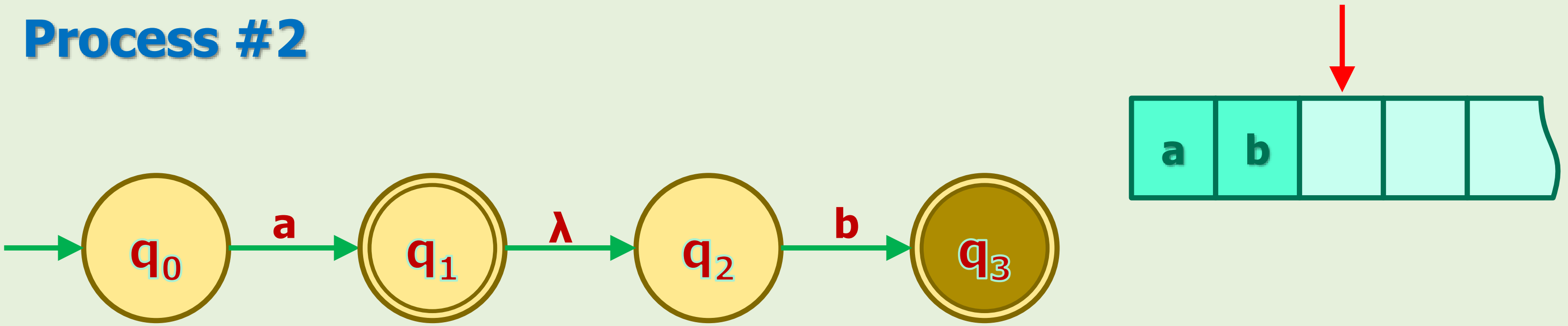
Process #2 consumes 'b' and transits to q_3.

Process #1



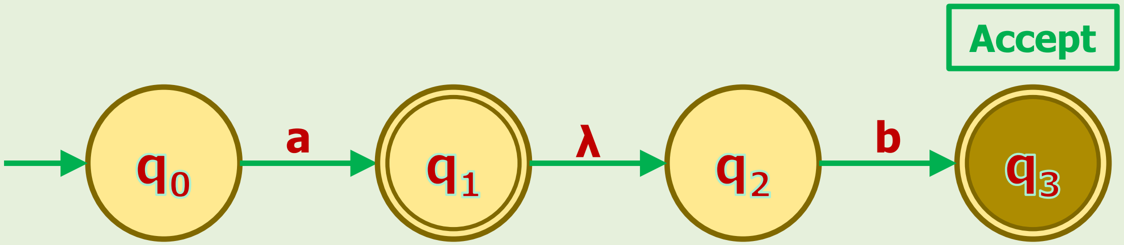
Process #1 is out of symbol and has to halt.

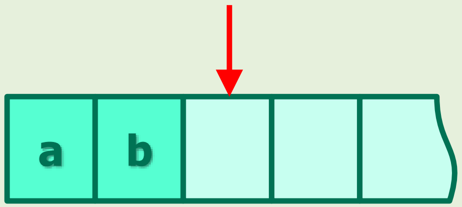
Process #2



Process #2 is out of symbol and hast to halt.

Process #1



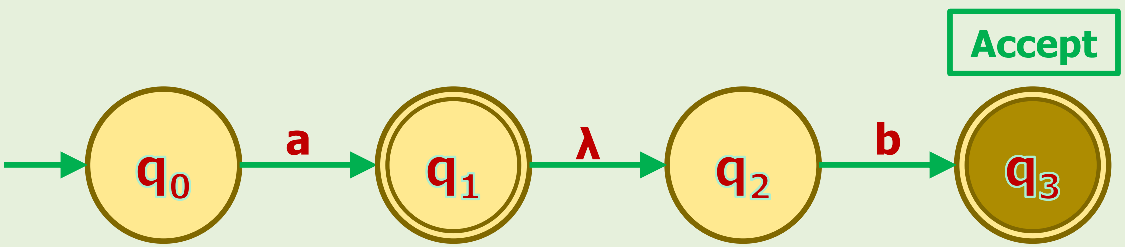


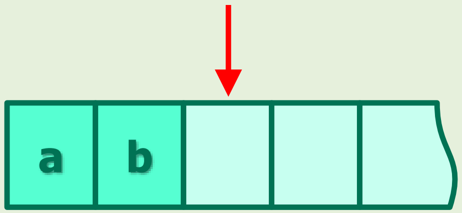
Overall Accepted

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

So, process #1 accepts w.

Process #2



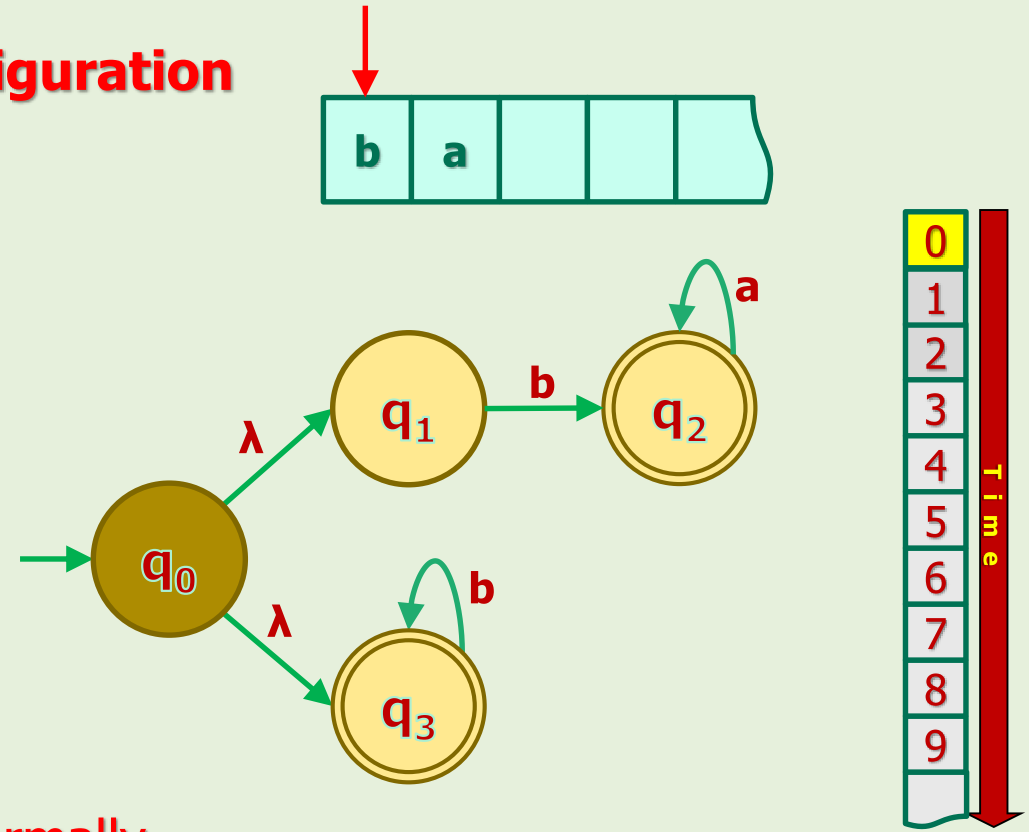


Process #2 halts in an accepting state AND all symbols are consumed.

So, process #1 accepts w.

λ-Transitions in Action

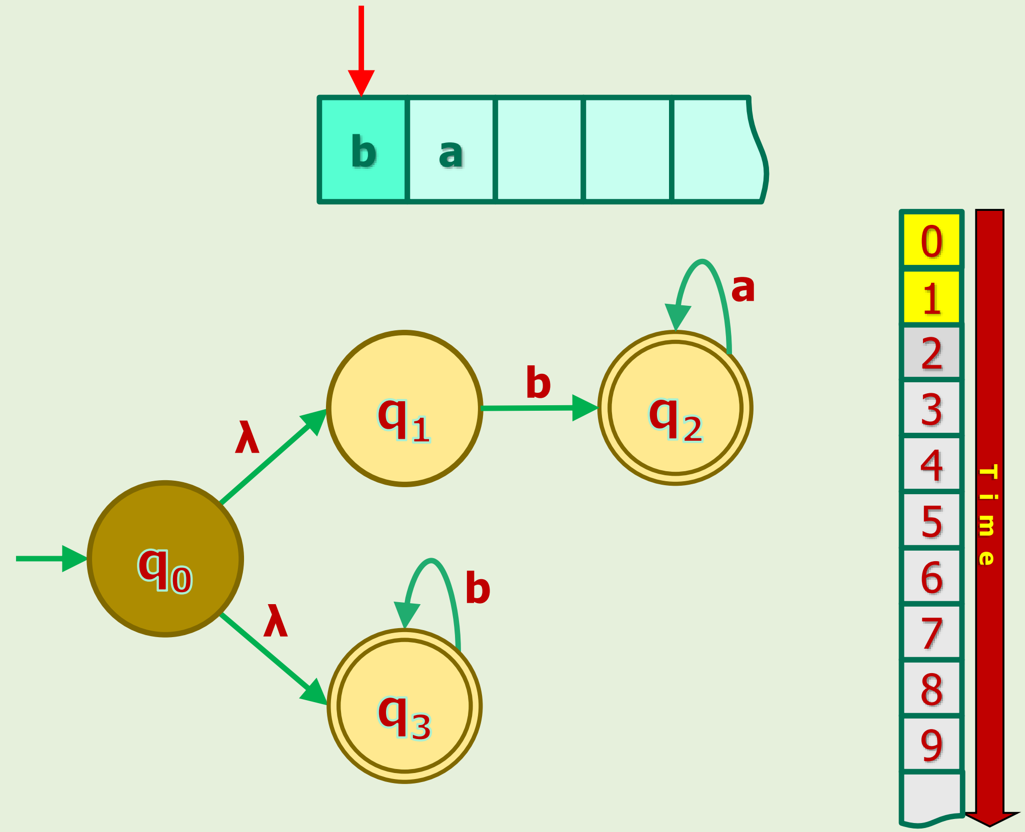
Example 14: Starting Configuration



- Process #1 (main) starts normally.

λ-Transitions in Action

Example 14 (cont'd)

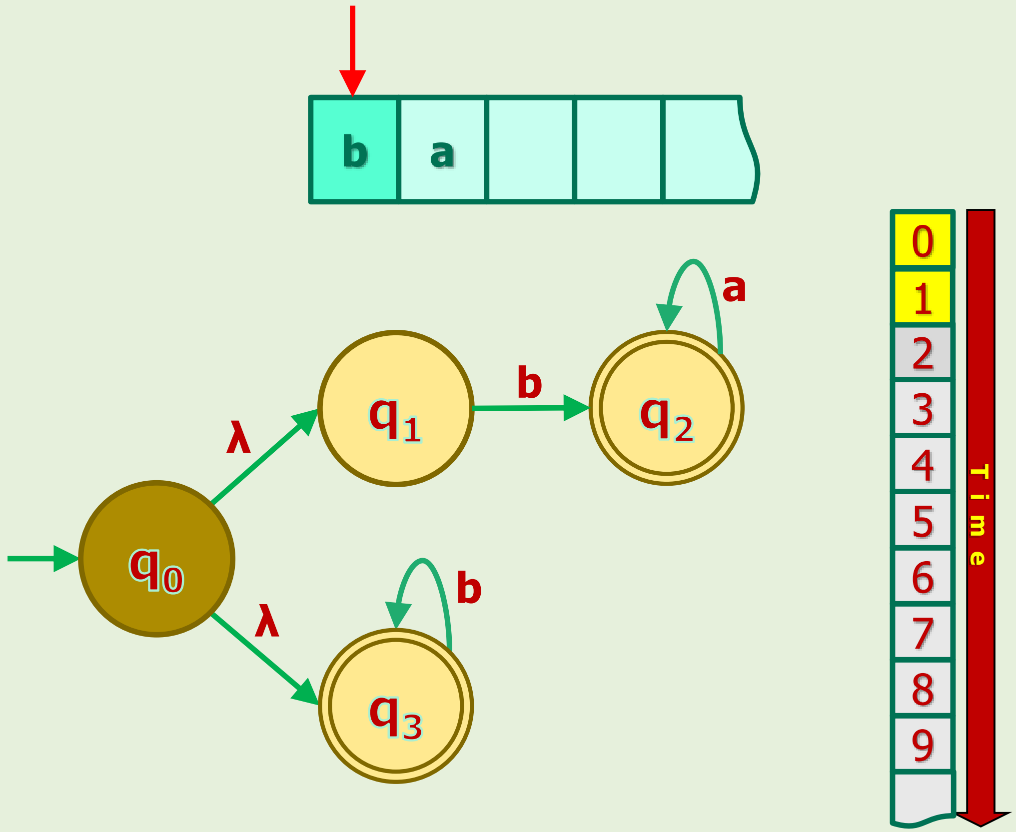


- Input tape reads 'b' and sends it to the control unit.
- The control unit makes a decision based on $\delta(q_0, b) = \{q_2, q_3\}$

λ-Transitions in Action

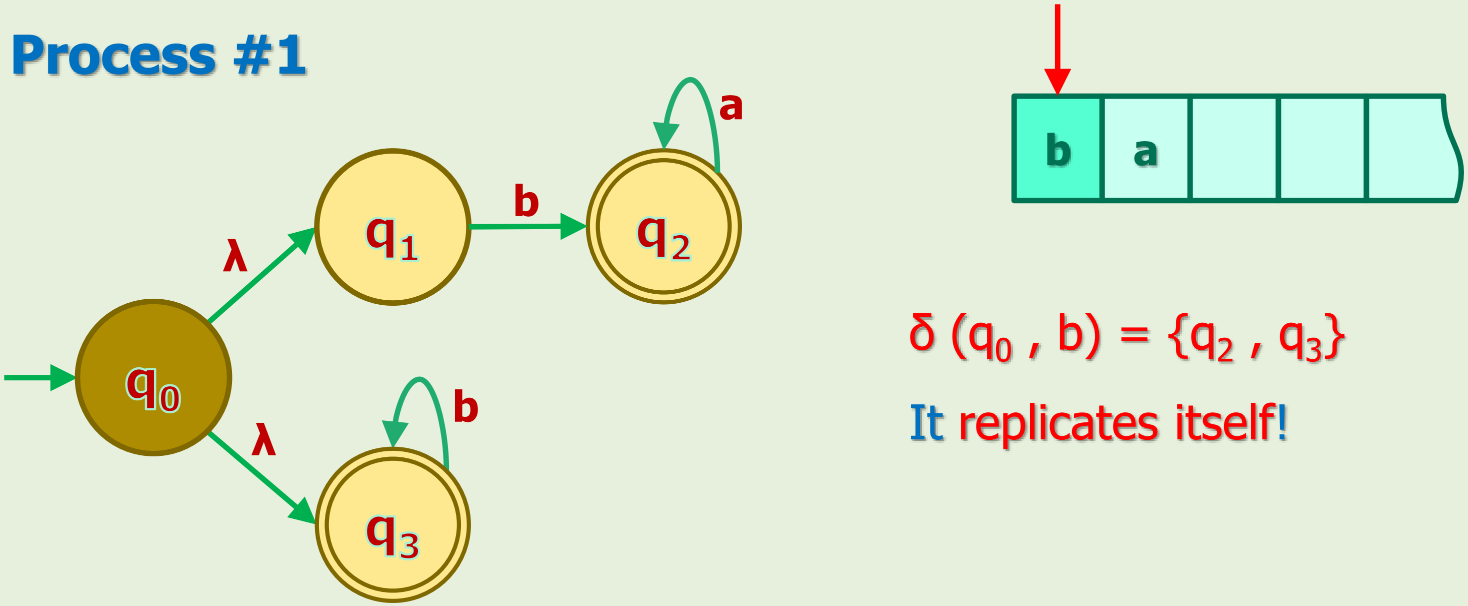
Example 14 (cont'd)

$\delta(q_0, b) = \{q_2, q_3\}$



- It encounters two possibilities: transition to q_2 or q_3.
- So, parallel processing starts!

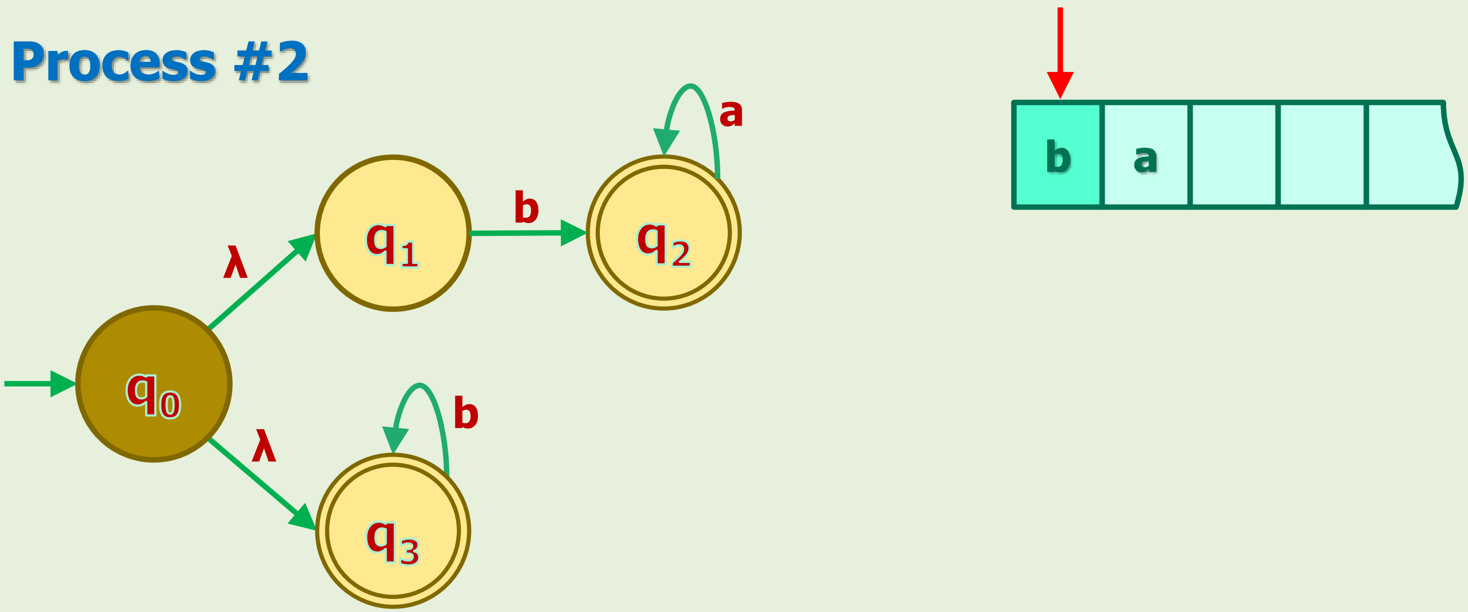
Process #1



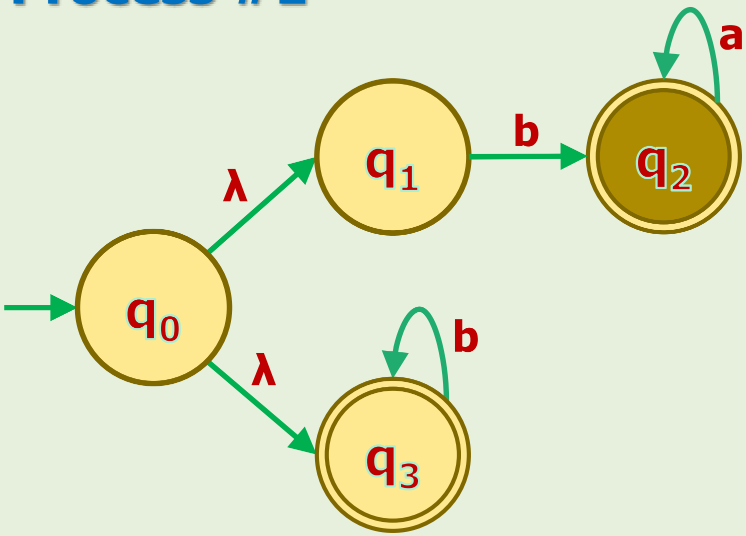
$\delta\ (q_0\ ,\ b) = \{q_2\ ,\ q_3\}$

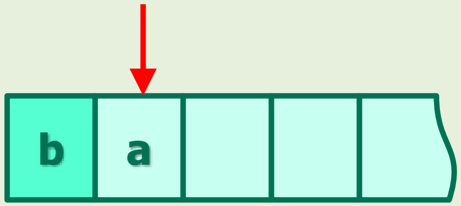
It replicates itself!

Process #2



Process #1

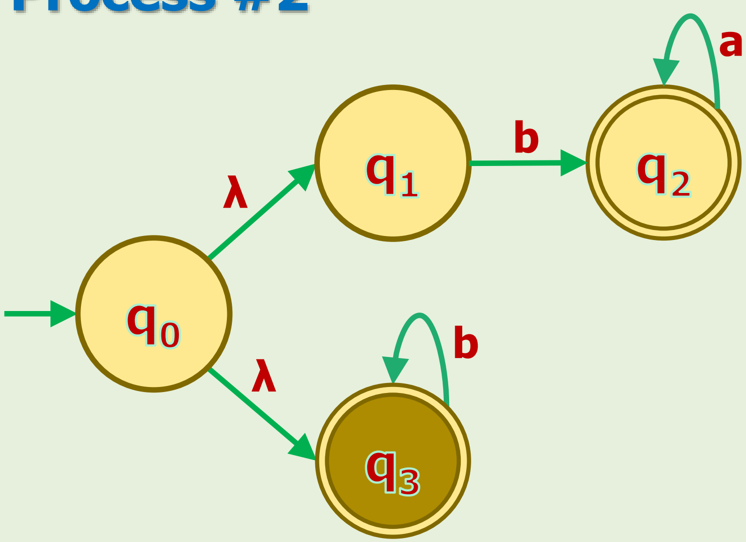


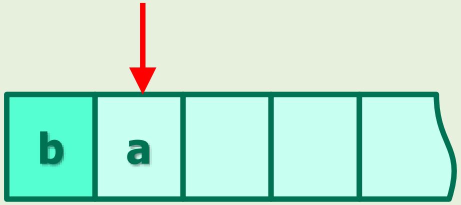


$\delta(q_0, b) = \{q_2, q_3\}$

Process #1 consumes 'b' and transits to q_2.

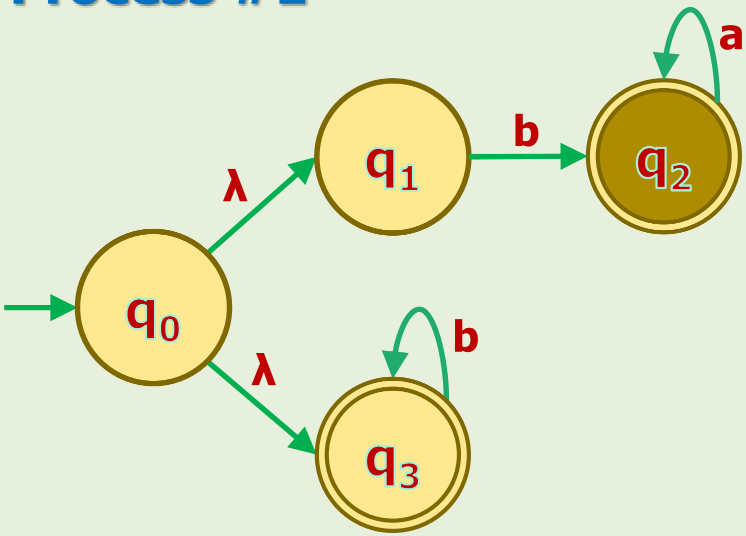
Process #2

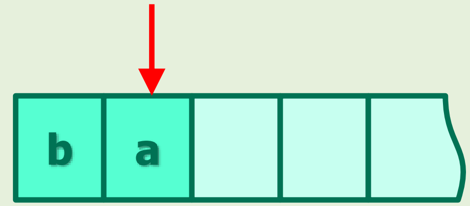




Process #2 consumes 'b' and transits to q_3.

Process #1

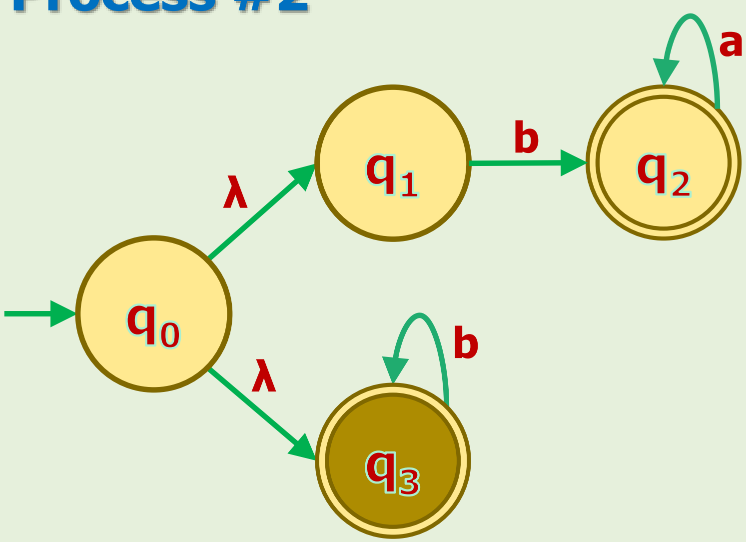


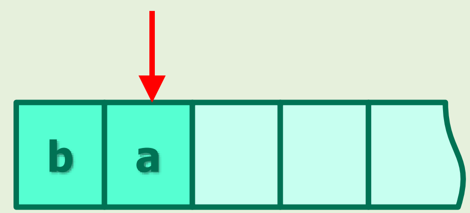


The symbol 'a' is read and sent to the control unit.

It calculates $\delta(q_2\ , a) = \{q_2\}$

Process #2

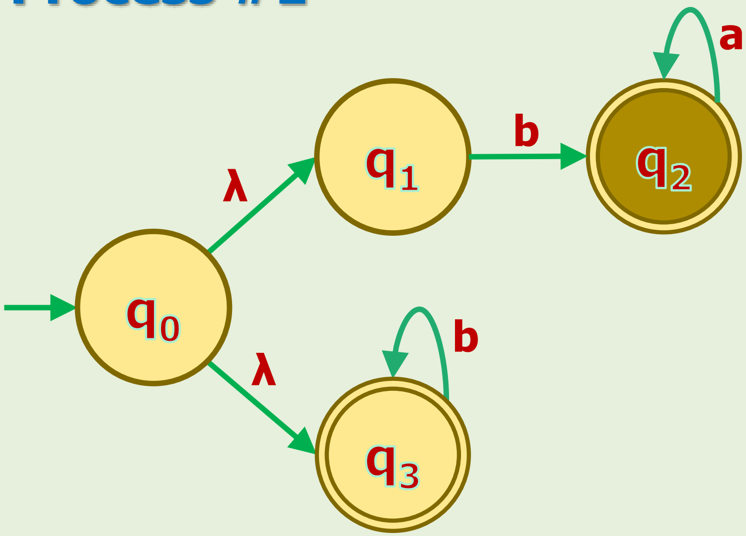


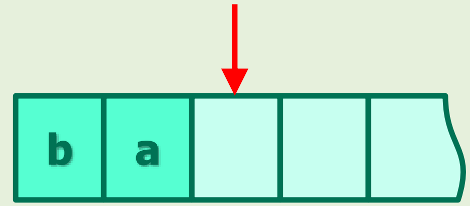


The symbol 'a' is read and sent to the control unit.

It calculates $\delta(q_3\ , a) = \{\ \}$

Process #1

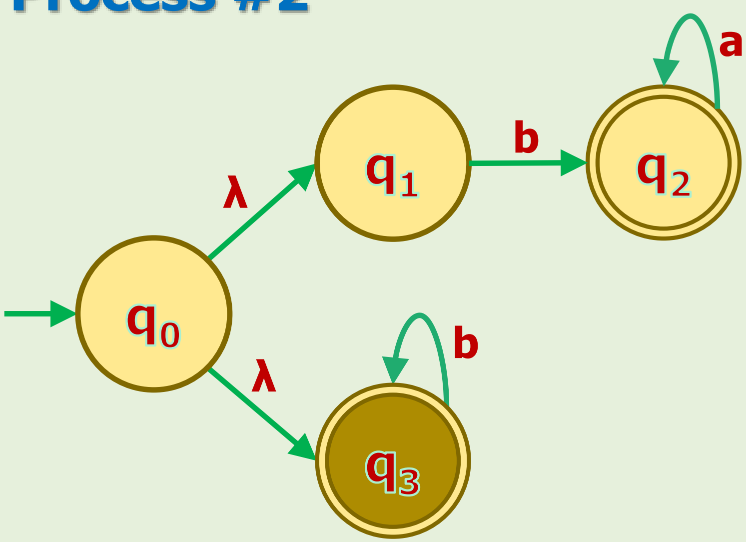


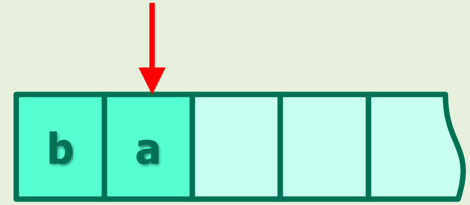


Process #1 consumes 'a' and transits to q_2.

It is out of symbol.

Process #2

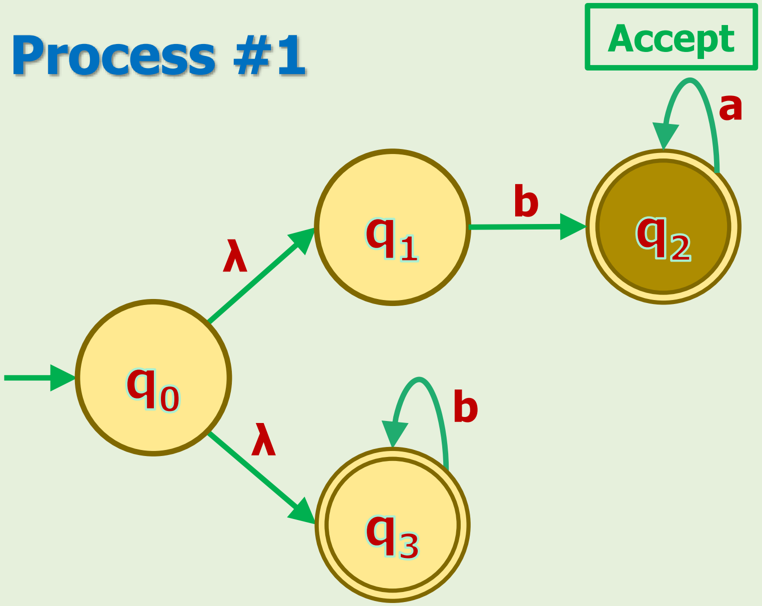


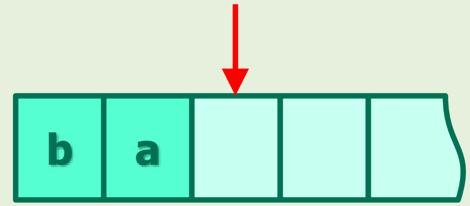


Process #2 has no choice for 'a'.

It has to halt.

Process #1

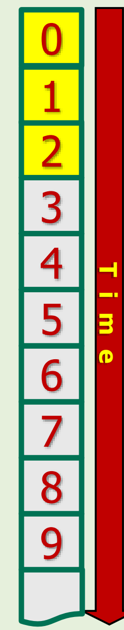




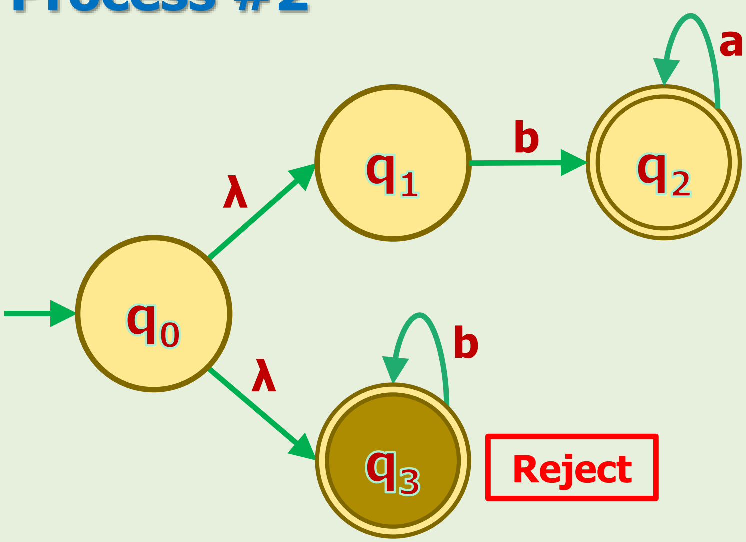
Overall Accepted

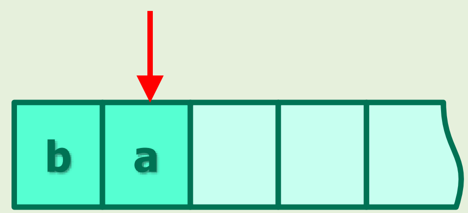
Process #1 halts in an accepting state AND
all symbols are consumed.

So, process #1 accepts w.



Process #2





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

So, process #2 rejects w.

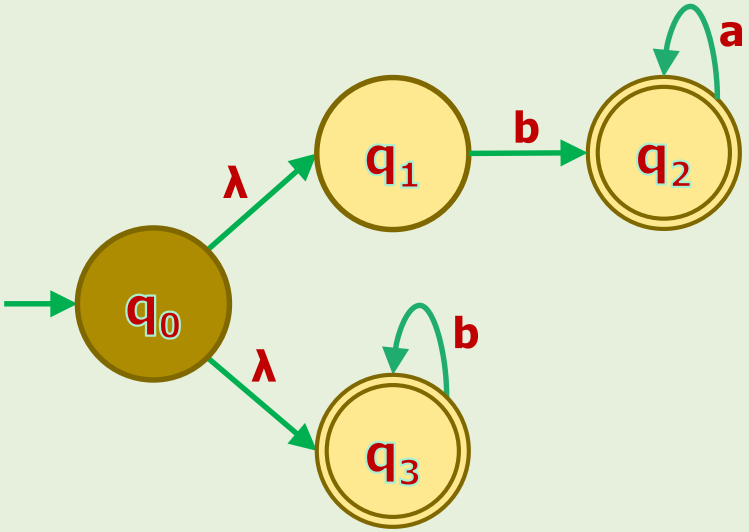
Homework

- Which of the following strings are accepted by this NFA over $\Sigma = \{a , b\}$?
- Draw all processes.

  $w = b$

  $w = bb$

  $w = baa$



6. Definitions

NFAs Transition Function

- Recall that DFAs' transition function is defined as:

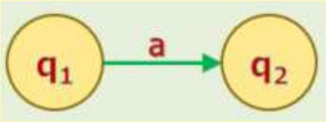
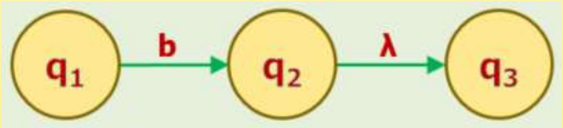
$$\delta: Q \times \Sigma \rightarrow Q$$

δ is total function.

- To accommodate those two violations,
  we change the RANGE of the function to a set.
- In this way, the range can have zero, one, or more states.
- In other words, the range of this function is a set of Qs.
- We already know that 2^Q is the power set of Q and it contains all subsets of Q.
- Therefore, we change the range from Q to 2^Q.

$$\delta: Q \times \Sigma \rightarrow 2^Q$$

Transition Function: DFAs vs NFAs

Class	Transition	Sub-Rule Example Transition Function
DFAs		$\delta\ (q_1, a) = q_2$ $\delta : Q \times \Sigma \rightarrow Q$
NFAs		$\delta\ (q_1, b) = \{q_2, q_3\}$ $\delta\ (q_2, a) = \{\ \}$ $\delta : Q \times \Sigma \rightarrow 2^Q$

6. Formal Definition of NFAs

- An NFA M is defined by the quintuple (5-tuple):

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

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

$$\delta : Q \times \Sigma \rightarrow 2^Q$$

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.

- Except δ, the rest items are the same as DFAs'.

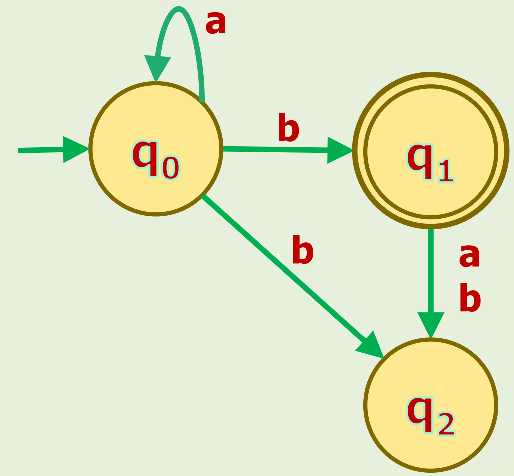
Why NFAs' δ is Total Function?

- Recall that if it is partial function, then at least one domain member is undefined.
- In that case, the machine does not know what to do!
- In other words, all domain elements must be defined, otherwise, in some situations, the machine won't know what to do.
- Let's take an example.

Why δ is Total Function?

Example 15

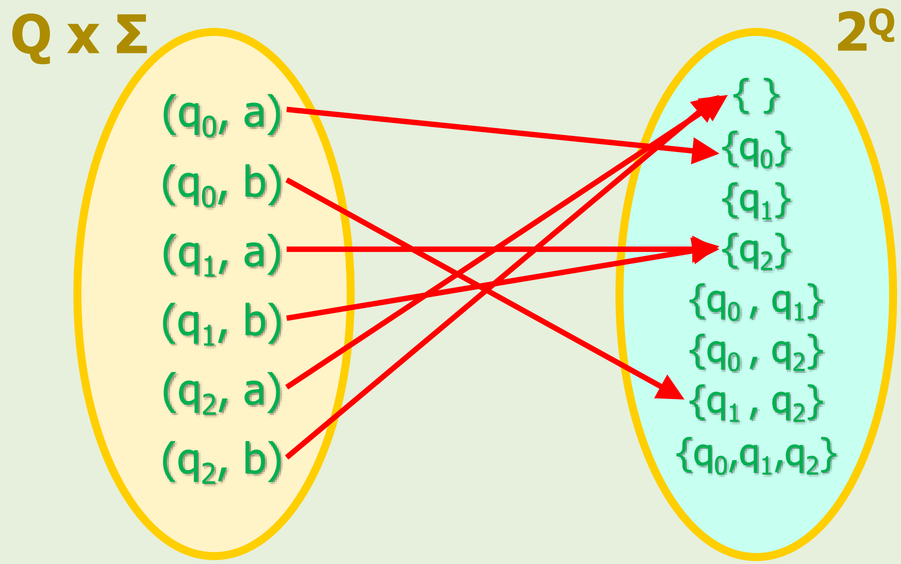
- Write the algebraic notation of the NFA's δ.



Solution

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

- Draw the diagram of δ.
- Isn't it total function?



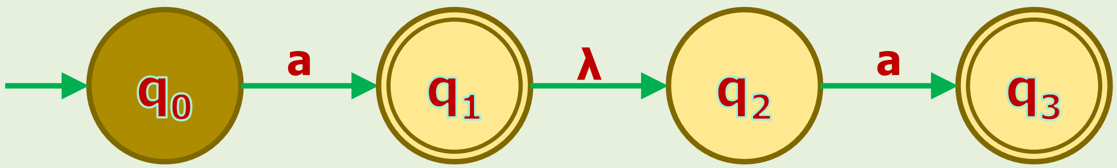
6. Formal Definitions: NFAs vs DFAs

	NFAs	DFAs
Transition function	$\delta : Q \times \Sigma \rightarrow 2^Q$	$\delta : Q \times \Sigma \rightarrow Q$
Examples	$\delta(q_1, a) = \{q_2, q_5, q_3\}$ $\delta(q_1, b) = \{q_1, q_3\}$ $\delta(q_2, a) = \{\ \}$	$\delta(q_1, a) = q_2$
Type of function	Total	Total
Type of processing	Parallel processing	Single processing

Associated Language to NFAs Examples

Example 16

- What is the associated language to the following NFA?



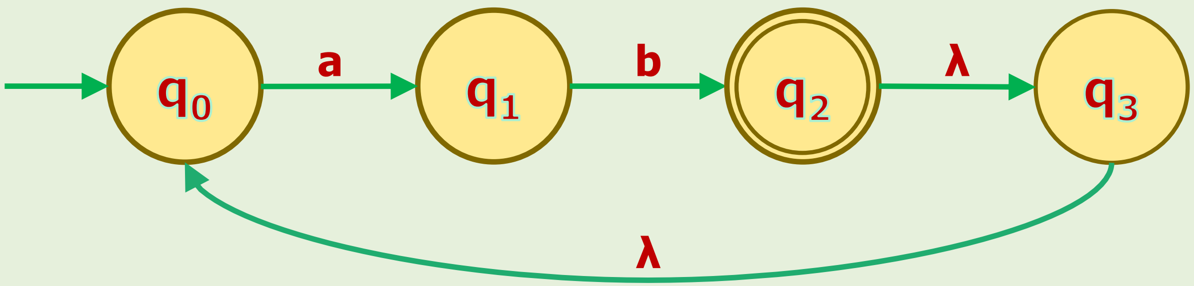
Solution

- $L(M) = \{a, aa\}$

Associated Language to NFAs Examples

Example 17

- What is the associated language to the following NFA?



Solution

- $L = \{ab, abab, ababab, \ldots\}$

  $= \{(ab)^n : n \geq 1\}$

NFA Design Example

Example 18

- Design a DFA and an NFA with 3 states for the following language over $\Sigma = \{a , b\}$.

"The set of all strings that ends with aa."

Homework

1. Let $L = \{a^n b : n \geq 0\}$, and $L' = L\,(L \cup \{\lambda\})$ over $\Sigma = \{a , b\}$.

   Design an NFA with 3 states for accepting L'.

2. Design an NFA for each of the following languages.
   a. $L = \{a^n b^m a^k : n, m \geq 0, k \geq 1\}$ with 3 states over $\Sigma = \{a , b\}$
   b. $L = \{(ab)^n (abc)^m : n \geq 0, m \geq 0\}$ over $\Sigma = \{a , b , c\}$

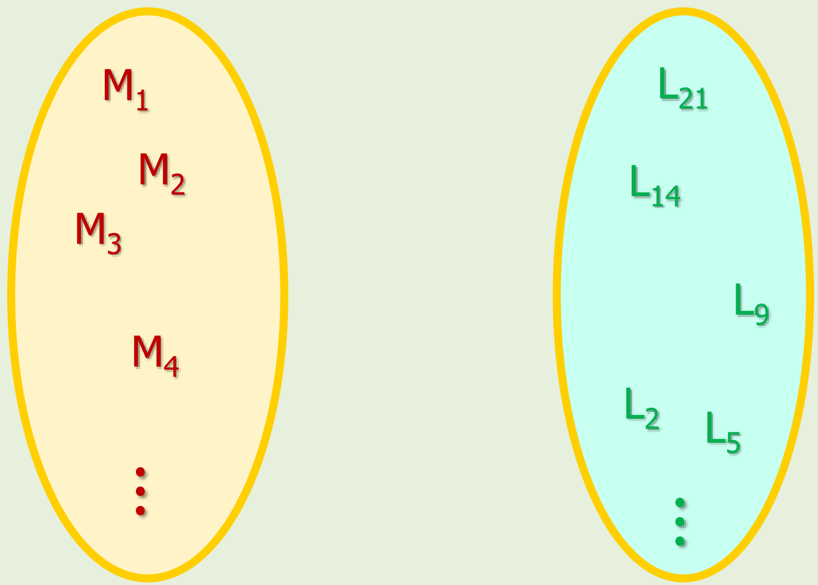
Machines and Languages Association

Machines and Languages Association

- What is the relationship between the set of all automata machines and the set of all formal languages?

All Automata Machines

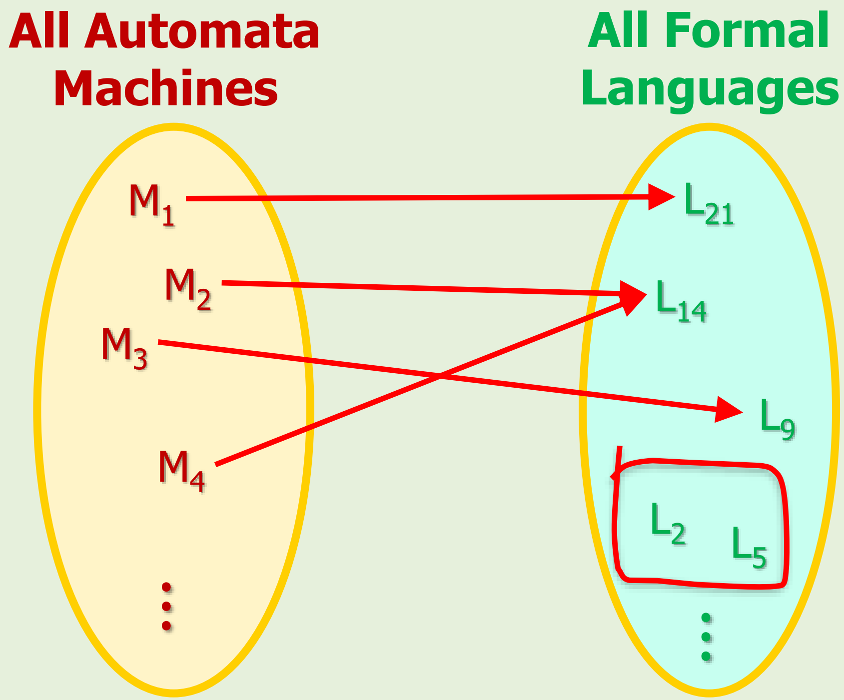
All Formal Languages



One of the most interesting topics of computer science

Machines and Languages Association

- So far, we learned that "every machine has an associated language".
- BUT we don't know yet whether or not for every language, we can construct a machine!
  - Our knowledge is not enough yet.



A Side Note: Computer Scientists Mission

- Why should we be interested in the relationship between machines and languages?
- Recall that we can encode all problems into formal languages.

Formal Language ≡ Problem

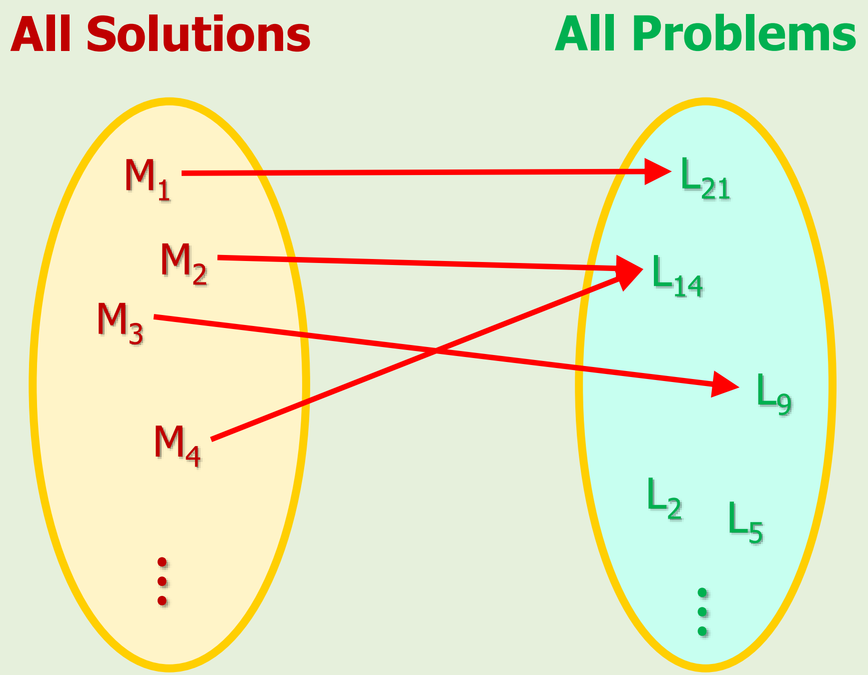
Accepting (understanding, recognizing) a language
≡ Solving the problem

- So, as computer scientists, our mission is:

To find a machine for every language ≡ To solve the problems

Machines and Languages Association

- Now, with this background, let's look at the association again.
- Let's rename them to "Solutions" and "Problems".
- Obviously, it's true that every solution is related to a problem!
- But, is this the case that for every problem, there is a solution?



7. DFAs vs NFAs

Objective

- The goal of this section is to compare two classes DFAs and NFAs.
- To compare two classes of automata, we'd need some "metrics".
- We'll use the concept of "power" as the metrics for comparison.
- So, first we need to define "power".

Power of Automata Classes

- Let's assume we have two classes of automata:
  - Class A (e.g. NFAs)
  - Class B (e.g. DFAs)

Question

- What is the best criteria to claim that:

Class A is "more powerful" than class B?

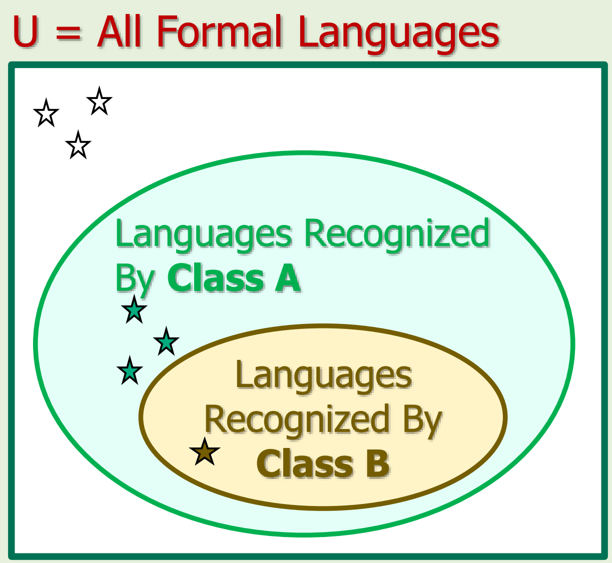
Answer

- If class A can solve more problems, then it is more powerful.
- Equivalently, if class A recognizes more languages, then it is more powerful.

Power of Automata Classes

Definition

- The (automata) class A is "more powerful" than class B iff the set of languages recognized by class B is a proper subset of the set of the languages recognized by class A.

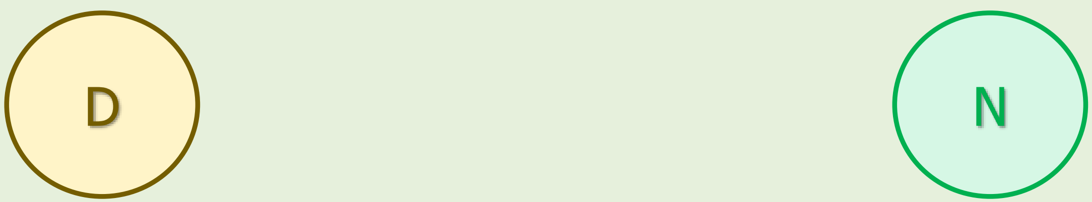


DFAs and NFAs Relationship

- Let's get back to our topic: DFAs vs. NFAs
- If the universal set is the set of all formal languages:
  1. What portion of the formal languages can be recognized by DFAs?
  2. What portion can be recognized by NFAs?
- Let's use the following definitions:

$$U = \{x : x \text{ is a formal language}\}$$

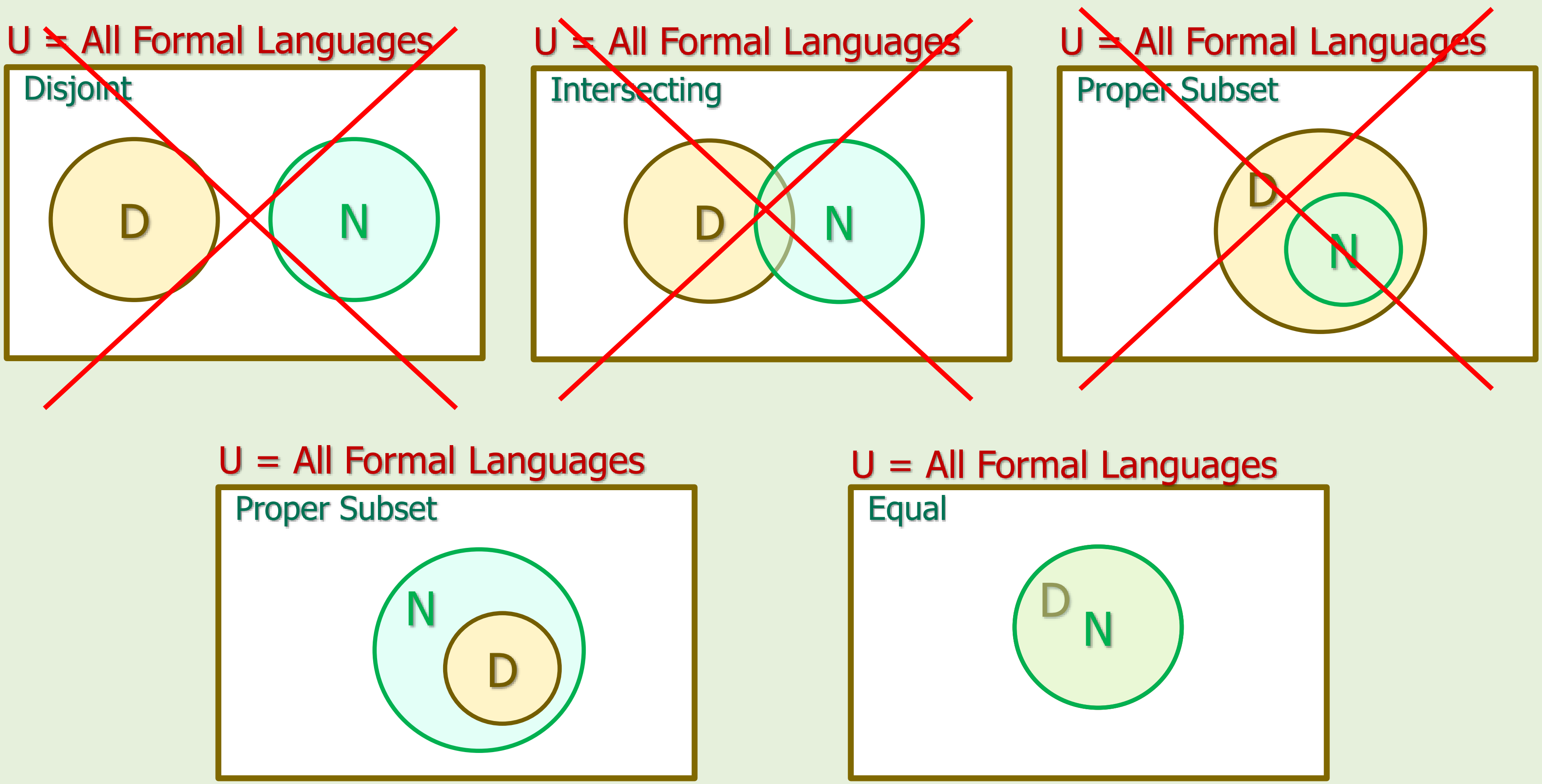
$$D = \{d : d \text{ is recognized by a DFA}\} \qquad N = \{n : n \text{ is recognized by a NFA}\}$$



- What is the relationship between the sets D and N?

DFAs and NFAs Relationship

- Which one is reasonable relationship between D and N?



Can NFAs Do Whatever DFAs Can Do?

- Let's assume that we've constructed a DFA for an arbitrary language L.
- Can we always construct an NFA for L?
- Yes! How?
- Mathematically speaking, the only difference between the definition of NFAs and DFAs is their transition function.
- So, we should prove that we can always convert a DFA's definition to an NFA's definition.
- Let's show this through an example.

Can NFAs Do Whatever DFAs Can Do?

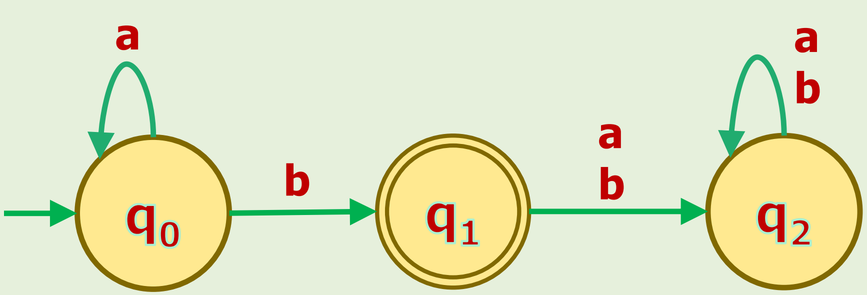
Example 19

- Convert the following DFA's definition to an NFA's.
- q_0 is the initial state, and q_1 is the final state.

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

DFA NFA

- Just convert the δ.
- The rest items are the same.



DFAs Can be Converted to NFAs

	DFA	NFA
States	$Q = \{q_0, q_1, q_2\}$	$Q = \{q_0, q_1, q_2\}$
Alphabet	$\Sigma = \{a, b\}$	$\Sigma = \{a, b\}$
Sub-rule	$\delta\ (q_i, a) = q_j$	$\delta\ (q_i, a) = \{q_j\}$
Initial state	q_0	q_0
Final states	$F = \{q_1\}$	$F = \{q_1\}$

Can NFAs Do Whatever DFAs Can Do?

- As the previous example showed, there is a simple algorithm to convert a DFA to an NFA.

Algorithm: Converting DFAs' Formal Definition to NFAs'

- Change all DFAs' sub-rules to NFAs format by enclosing the range element with a pair of curly brackets. i.e.:

$$\delta\ (q_i\ , x) = q_j$$

changes to

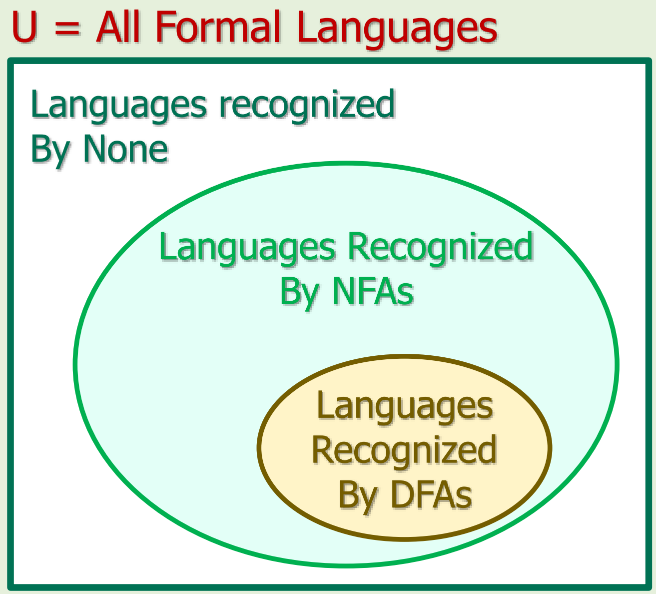
$$\delta\ (q_i\ , x) = \{q_j\}$$

- The rest of the definitions, (i.e. Q, Σ, q_0, F) are the same.

Can NFAs Do Whatever DFAs Can Do?

Conclusion

- Can NFAs do whatever DFAs can do?
- Yes, the set of all languages recognized by DFAs can be recognized by NFAs too.



- Now, let's ask another question ...

Can DFAs Do Whatever NFAs Can Do?

- Let's assume that we've constructed an NFA for an arbitrary language L.
- Can we always construct a DFA for L?
- The answer of this question is not so obvious.
- Let's take an example to make it clear.

Can DFAs Do Whatever NFAs Can Do?

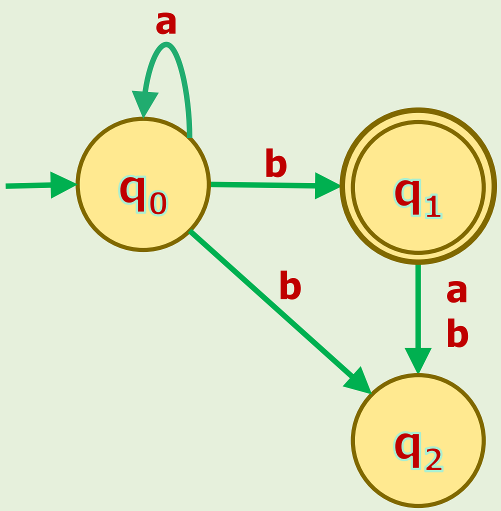
Example 20

- Can we convert the following NFA to a DFA?
- q_0 is the initial state, and q_1 is the final state.

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

NFA

DFA



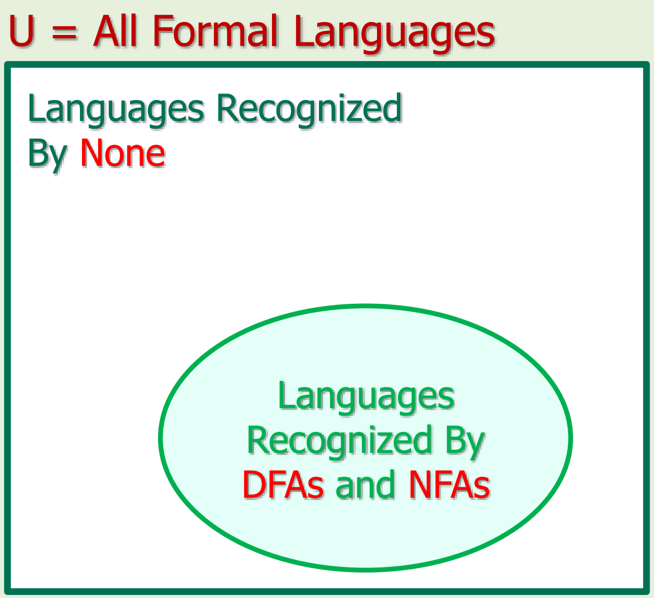
- Yes, but it needs a special technique to convert an NFA to DFA.
- We might cover it later if we have time!

DFAs Class and NFAs Class are Equivalent

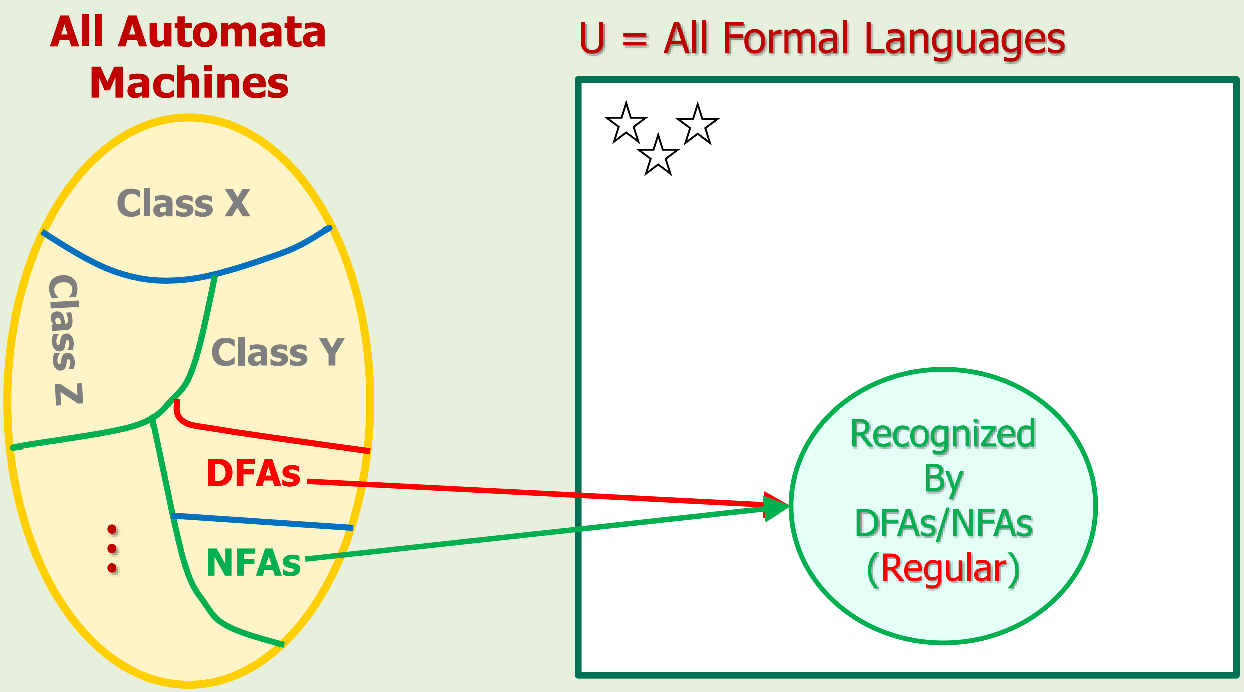
- DFAs and NFAs are equivalent as the following theorem states.

Theorem

- The set of languages recognized by NFAs are equal to the set of languages recognized by DFAs.



Machines and Languages Association

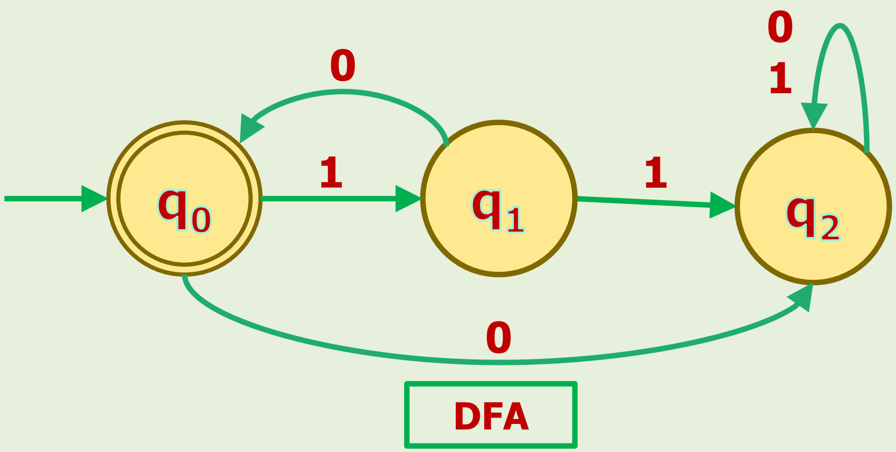


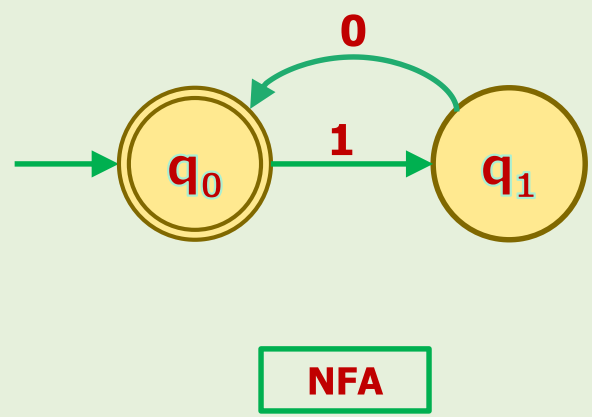
- DFAs and NFAs have the same power because both recognize the same portion of languages.
- Later we'd define other classes of machines (i.e. Class X, Y, Z, etc.) and the languages they are associated with.

Equivalency of DFAs and NFAs Example

Example 21

- What are the associated languages to the following machines?





$L_1 = \{(10)^n : n \geq 0\}$ $L_2 = \{(10)^n : n \geq 0\}$

- They are equivalent because both have the same associated languages.

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