Theory of Computation

Introduced By:

Prof. Safia Abbas

Text Books:

"***An Introduction to the theory of computation***" by Michael Sipser, 2nd Edition, PWS Publishing Company, Inc. 2006; ISBN: 0-534-95097-3

"***An Introduction to Formal Languages & Automata***" by Peter Linz, 5th Edition, Jones & Bartlett Publishers, Inc. January 2012; ISBN:978-1-4496-1552-9 .

COURSE OUTLINE

A- Methods Used	1. Midterm 2. Quizzes 3. Project 4. Assignments 5. Final Exam
B- Assessment Schedule	1. Midterm 2. Quizzes 3. Assignments 4. Final Exam
C- Weighting of Assessments	1. Midterm 15% 2. Quiz & Assignments (5,10) 15% 3. Practical Exam 20% 4. Final Exam 50% Total 100%

CSC420. Theory of Computation [3 CH]
Co-requisite: BSC221. Discrete Mathematics

- This module introduces the theory of computation through a set of abstract machines that serve as models for computation - finite automata, pushdown automata, and Turing machines – and examines the relationship between these automata and formal languages. Additional topics beyond the automata classes themselves include deterministic and nondeterministic machines, regular expressions, context free grammars, undecidability, and the P = NP question.

Theory of computation

Course Details

- Automata theory:
  - What is Formal language,
  - DFA,NFA,PDA,
  - regular and non regular languages, context free language and grammar,
  - regular expressions.
  - formal definition of computation and Turing machines
  - TM design, implementation levels and equivalence
- Computability theory:
  - Church's thesis: Grammars and Helbert's tenth Problem. Turing computability.
  - The un-computability: The halting problem, Turing enumerability, Turing acceptability, and Turing decidability and unsolvable problems about Turing machines.
- Complexity theory:
  - Complexity Theory , Time complexity.
  - P and NP. Polynomial-time reducibility. NP-Completeness. The Cook-Levin Theorem.
  - Example reductions among NP-hard sets.

Why we study theory of computation???

- We study this course in order to answer the following questions:
  - What are the fundamental capabilities and limitations of computers???
  - Can any problem be solved and computed by computers???

This question was asked by mathematicians in the 1930's, when they were trying to understand the meaning of a "computation". The question showed up, because they wanted to know whether all mathematical problems can be solved in a systematic way. The research that started in those days led to computers as we know them today.

Complexity Theory: *Classify problems according to their degree of "difficulty" Giving a proof that problems that seem to be "hard" are really "hard".*

Informally, a problem is called "easy", if it is efficiently solvable.

- **Examples of "easy" problems** are (i) sorting a sequence of, say, 1,000,000 numbers, (ii) searching for a name in a telephone directory, and (iii) computing the fastest way to drive from a place to another.

On the other hand, a problem is called "hard", if it cannot be solved efficiently, or if we don't know whether it can be solved efficiently.

- Examples of "hard" problems are (i) time table scheduling for all courses, (ii) factoring a 300-digit integer into its prime factors..

Computability theory: *Classify problems as being solvable or unsolvable.*

- In the 1930's, Godel, Turing, and Church discovered that some of the fundamental mathematical problems cannot be solved by a "computer".
- An example of such a problem is "Is an arbitrary mathematical statement true or false?" To attack such a problem, we need formal definitions of the notions of computer, algorithm, and computation.
- The theoretical models that were proposed in order to understand solvable and unsolvable problems led to the development of real computers.

What is Automata???

- ***Automaton*** is defined as self operating machine that considered as an abstract model for digital computers with limited memory. ***Automata Theory*** deals with definitions and properties of different types of "computation models" such as:

- ***Finite automatons*** (***FA***): which used in text processing, compilers, and hardware design. It is the heart of many electromechanical devices such as automatic doors, elevators, dishwashers, calculators,…etc.

- ***Context-Free Grammars:*** These are used to define programming languages in Artificial Intelligence (NLP).

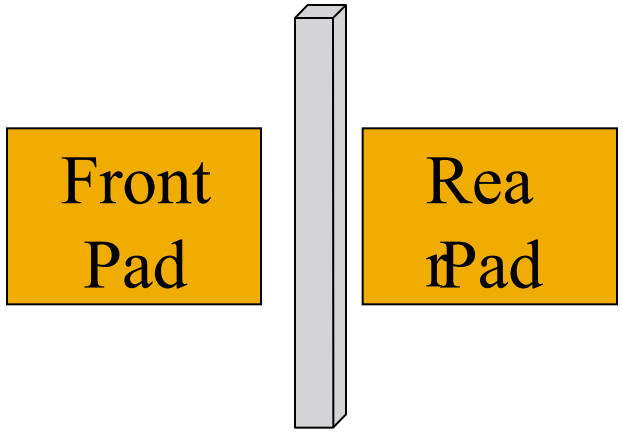
- ***Turing Machines:*** These form a simple abstract model of a "real" computer, such as your PC at home.

Some devices we will see

finite automata	Devices with a finite amount of memory. Used to model "small" computers.
push-down automata	Devices with infinite memory that can be accessed in a restricted way. Used to parse grammars
Turing Machines	Devices with infinite memory. Used to model any computer.
time-bounded Turing Machines	Infinite memory, but bounded running time. Used to model any computer program that runs in a reasonable amount of time.

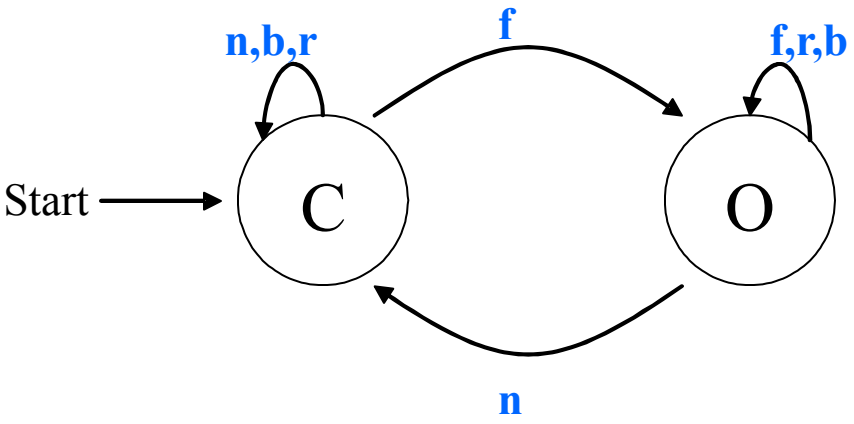
Example – 1 way door

- consider a one-way automatic door.
- The door has two pads that can sense when someone is standing on them, a front and rear pad.
- People must walk through the front and toward the rear, but not allowed to walk the other direction:



One Way Door

- Let's assign the following codes to our different input cases:
  - n - Nobody on either pad (neither)
  - f - Person on front pad
  - r - Person on rear pad
  - b - Person on front and rear pad (both)
- We can design the following automaton so that the door doesn't open if someone is still on the rear pad and hit them:

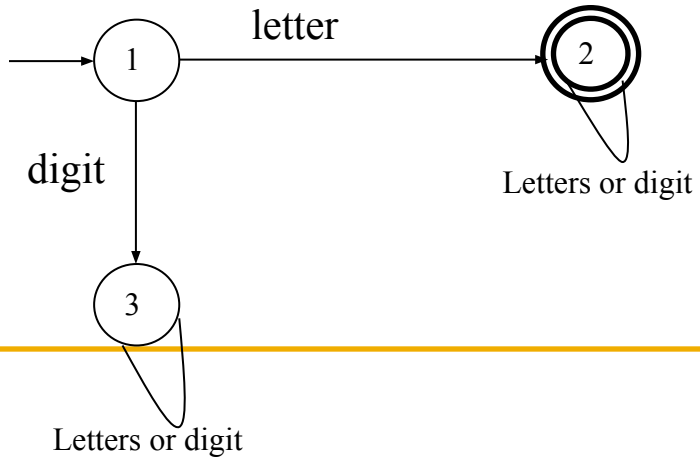


	n	f	r	b
Clos	C	O	C	O
Open	C	O	O	O

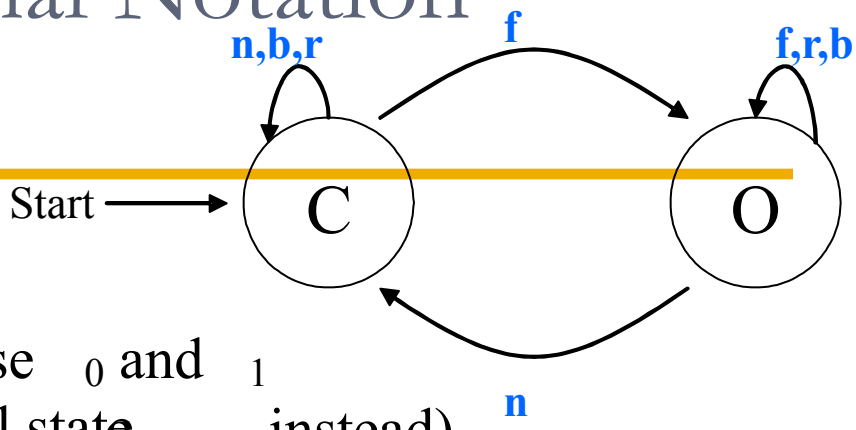
Deterministic Finite Automata (DFA)

- Formally, we can define **Deterministic Finite Automata** as the quintuple
  $M = (Q, \Sigma, \delta, q0, F)$, **where**

- **Q :** is a finite set of internal states.
- **Σ :** is a finite set of symbols, called input alphabet.
- **δ :** $Q \times \Sigma \rightarrow Q$ is a total function called the transition function, it specifies exactly one next state.
- **q0** $\in$ **Q** is the initial state.
- **F** $\subseteq$ **Q** is a set of final states. (ex. Identifier in compiler)



One Way Door – Formal Notation



Using our formal notation, we have:

$Q = \{C, O\}$	(usually we'll use q_0 and q_1 instead)
$F = \{\}$	There is no final state
$q_0 =$	This is the start state
$\sum = \{n, f, r, b\}$	

The transition function, δ, can be specified by the table:

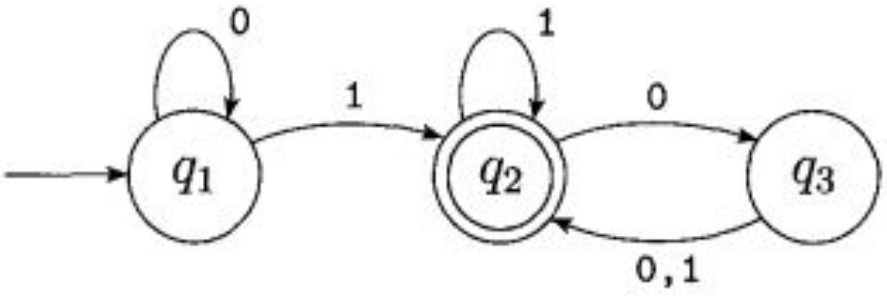
		n	f	r	b
→	**C**	C	O	C	C
	O	C	O	O	O

Write each δ(state,symbol)???

- The start state is indicated with the →
- If there are final accepting states, that is indicated with a* in the proper row.

Deterministic Finite Automaton (DFA)

- Identify the 5-tuples for the following state diagram ?



state diagram

- What is the output of the string 1101?
- What is the language of this machine??

Deterministic finite automata (DFA)

- Informally, a deterministic finite automaton over an alphabet "A" can be thought of as a finite directed graph with the property that each node emits one labeled edge for each distinct element of "A", the nodes are called states. There is one special state called the start or the initial state, and there is at least one final state.
- A DFA accepts a string w in A* if there is a path from the start state to some final states such that w is the concatenation of the labels on the edges of the path. Otherwise, the DFA rejects w.
- the set of all strings accepted by a DFA M is called the language of M and denoted by L(M).

Regular Language

- **Definition:** Let M = (Q, Σ, δ, q, F) be a **Deterministic finite automaton**. The language L(M) accepted by M is defined to be the set of all strings that are accepted by M:

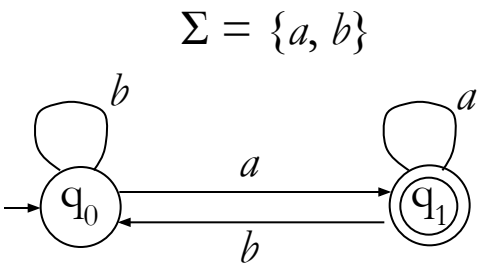
$$L(M) = \{s : s \text{ is a string over } \Sigma \text{ and M } \underline{\textbf{accepts}} \text{ s } \}.$$

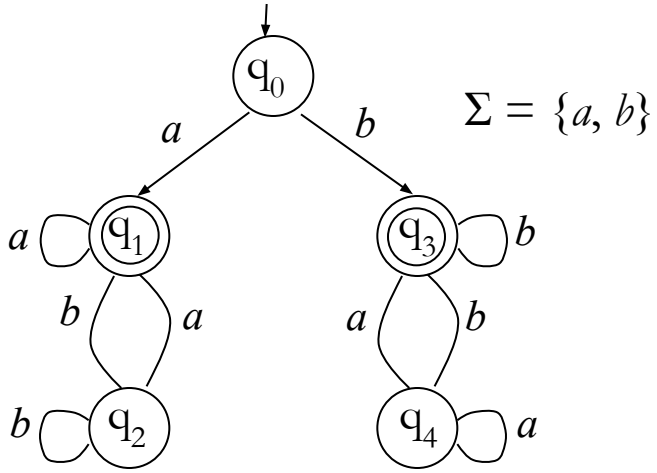
$L(M) = \{w \in \Sigma^* : \delta^*(q0, w) \in F\}$,

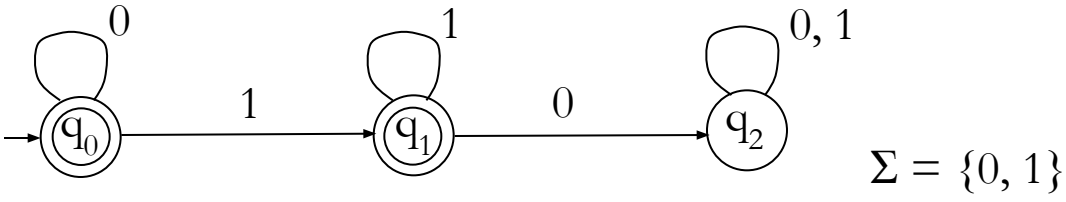
$L(M) = \{w \in \Sigma^* : \delta^*(q0, w) \notin F\}$ non accepted words

- **Definition:** A language L is called regular language if and only if there exit a finite automaton M such that L = L(M).
- A machine may accept several strings, but it always recognizes only one language.
- If the machine accepts no strings, it still recognizes one language namely, the empty language (λ, Ø)

Examples



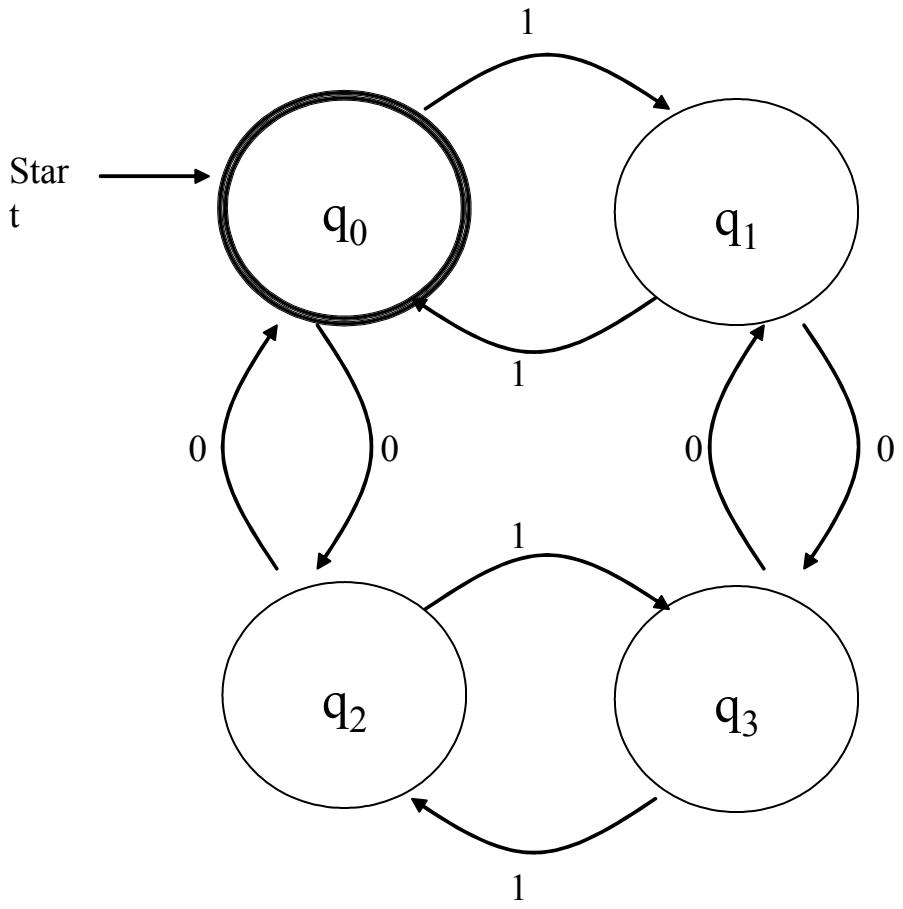




What are the languages of these automata?

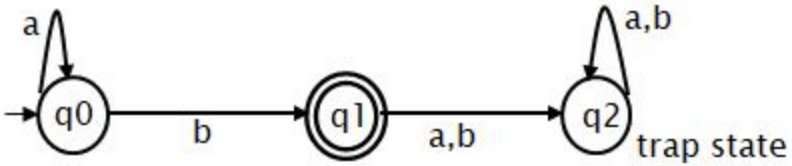
DFA Example

- Here is a DFA for the language that is the set of all strings of 0's and 1's whose numbers of 0's and 1's are both even.

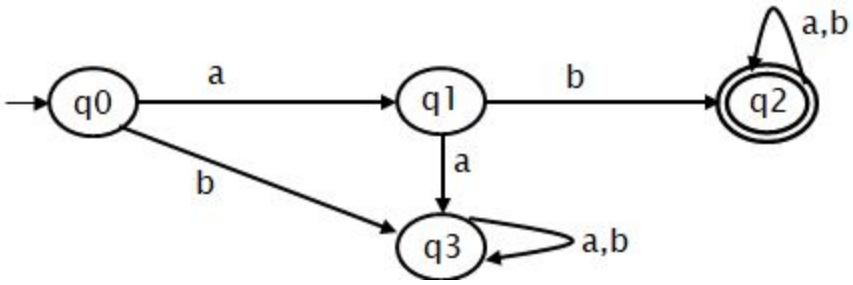


Draw DFA !!!

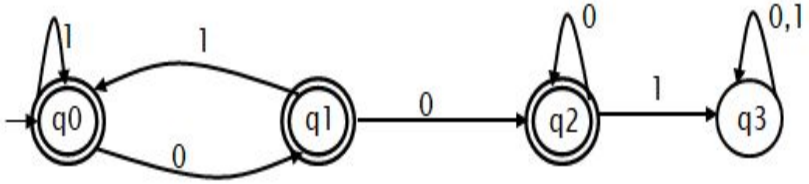
- **L(M) = { $a^n b$: n >= 0}**



- **L(M)={ab(a+b)*}**



- **Find FA that accepts all strings on {0, 1}, except those containing the substring 001.**

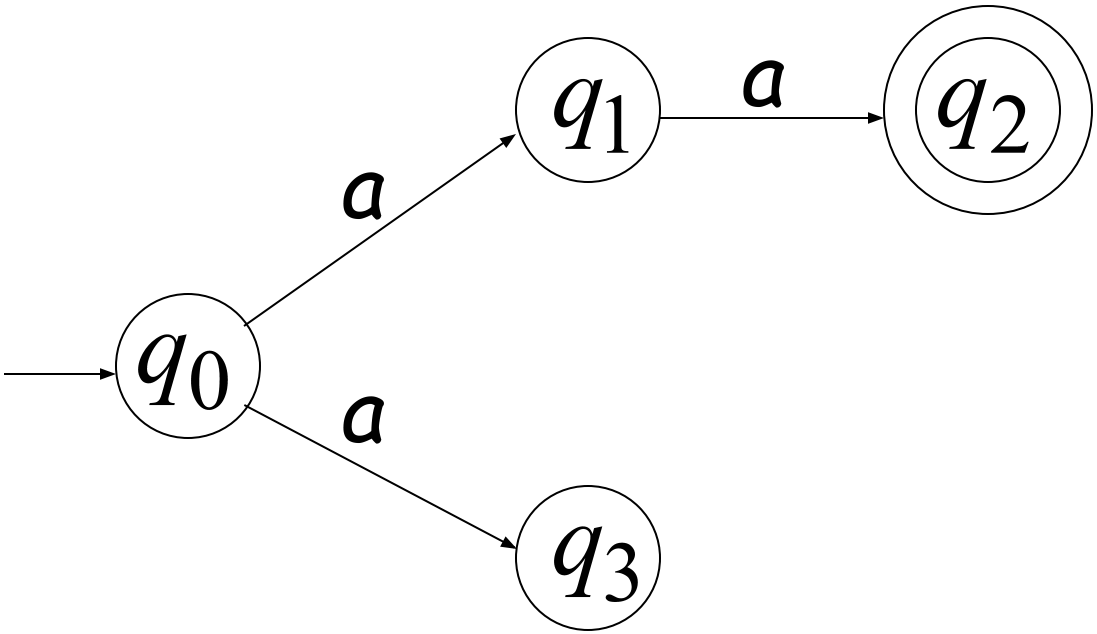


The non-Deterministic Finite Automata (NFA)

- **A nondeterministic finite automaton is defined as quintuple M = (Q, Σ, δ, q_0, F), where Q, Σ, q_0, F are defined as for dfa but δ : Q X (Σ ∪ λ) → 2^Q and δ is a relation.**
- **There are three major differences between dfa and nfa:**
  - **The range of δ is in the power set 2^Q, so that its value defines a set of all possible states that can be reach by the transition**
  - **We allow λ as the second argument, this means that nfa can make a transition without consuming any input symbol**
  - **The value of δ may be empty, meaning that there is no transition defined for this specific situation.**

Nondeterministic Finite Automata (NFA)

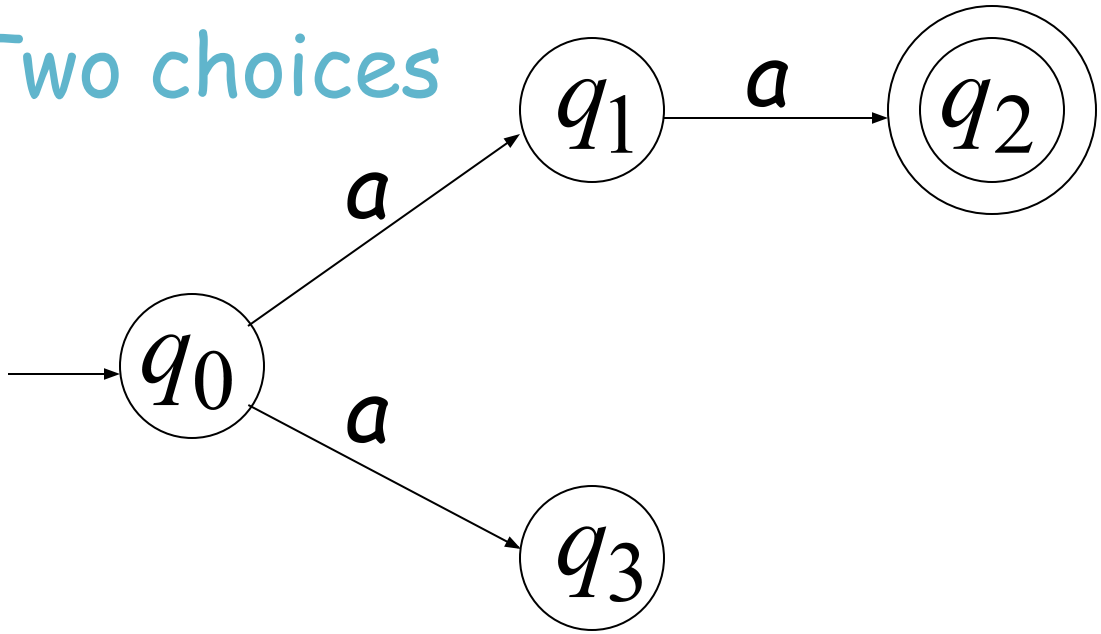
Alphabet = $\{a\}$



Nondeterministic Finite Automata (NFA)

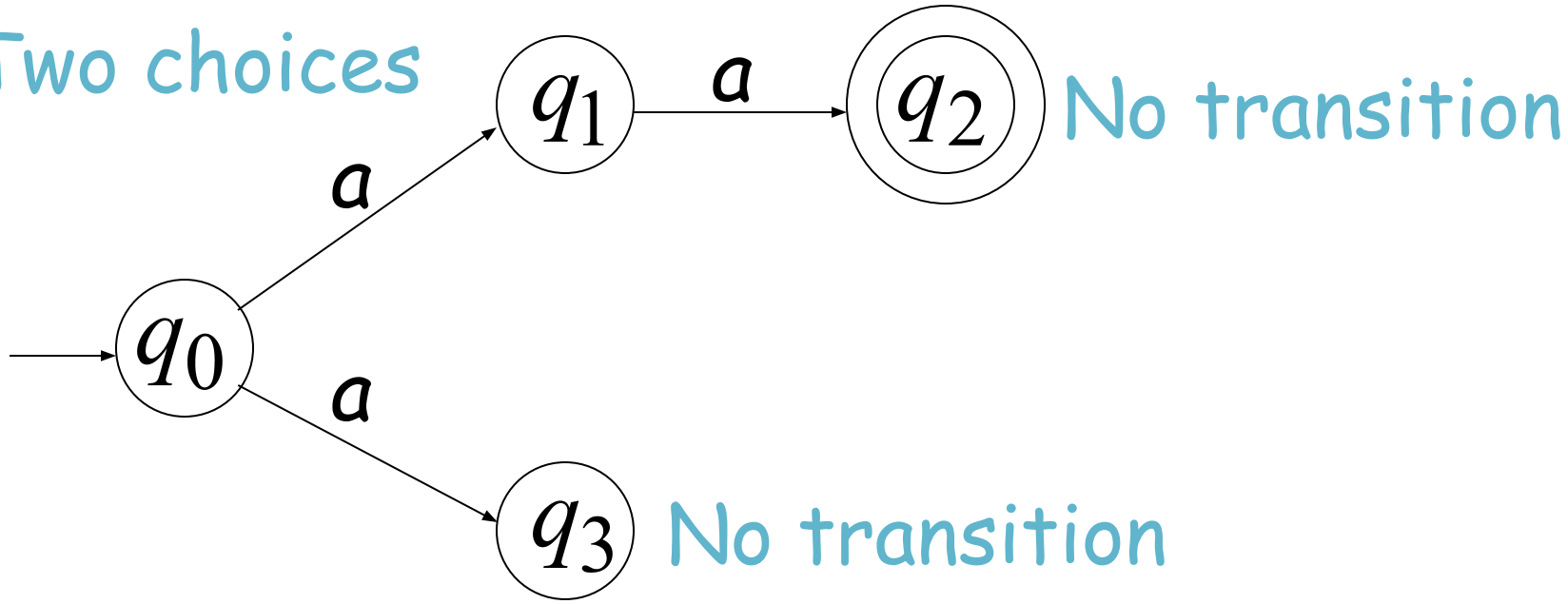
Alphabet = $\{a\}$

Two choices



Nondeterministic Finite Automata (NFA)

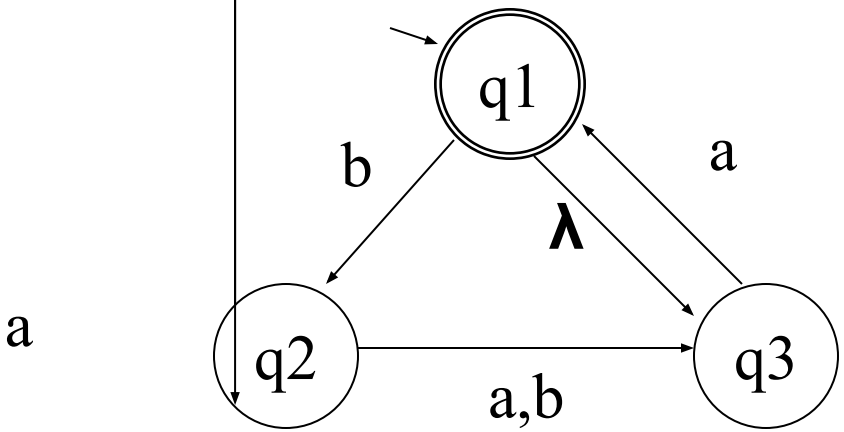
Alphabet = $\{a\}$



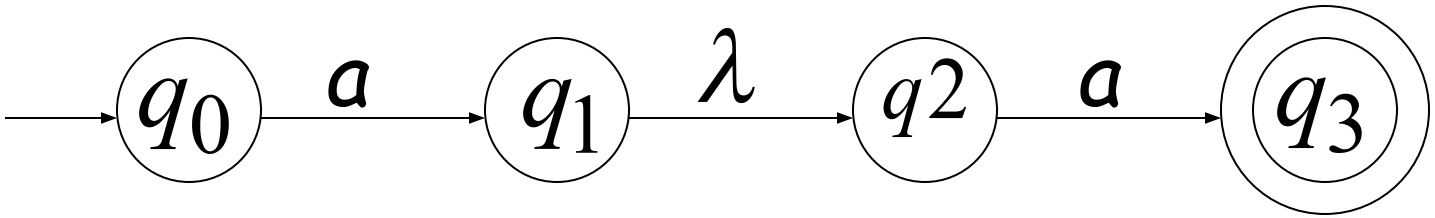
Nondeterministic Finite Automata

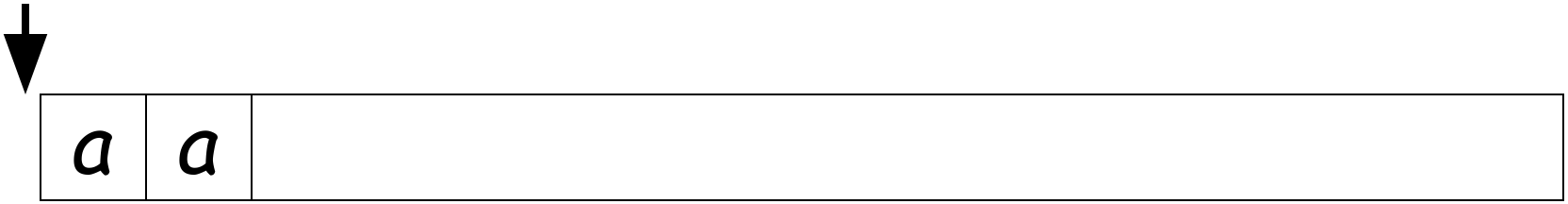
Example:

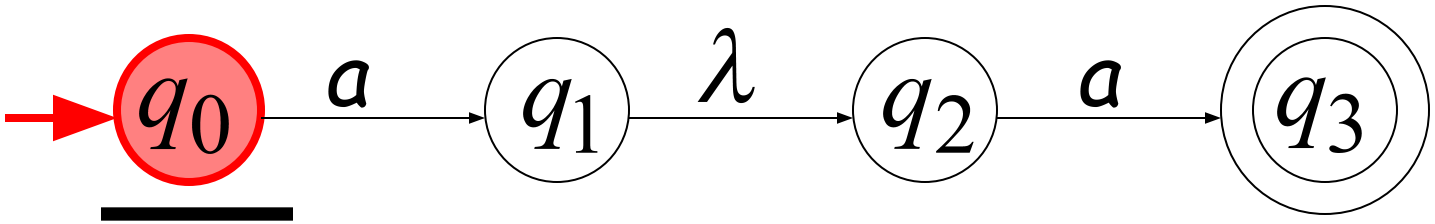
The following NFA accepts λ, a, baba, baa, and aa, but that it doesn't accept b, bb, and babba.

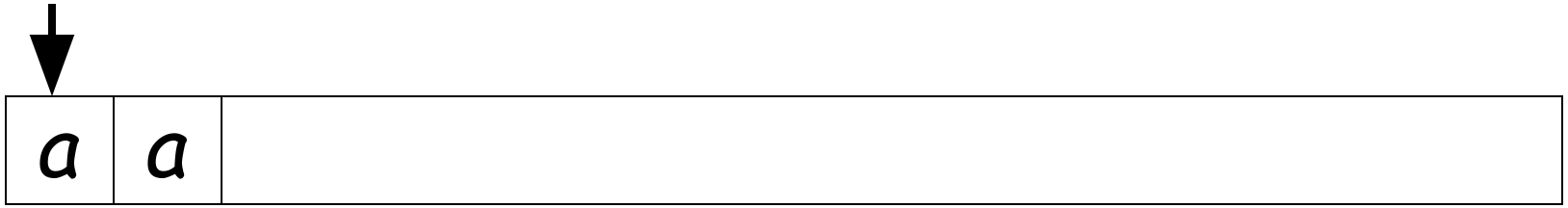


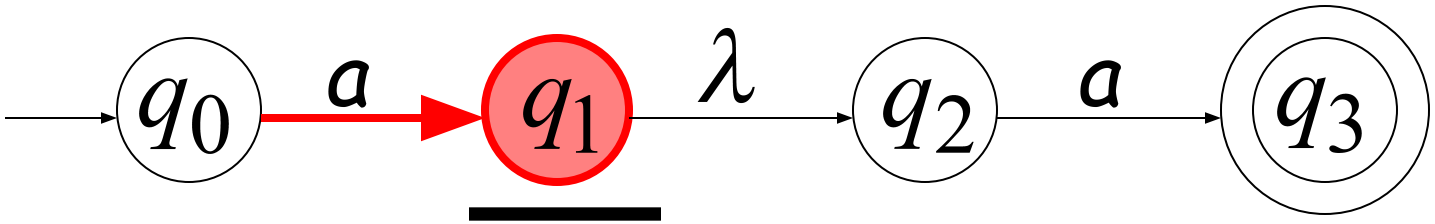
Lambda Transitions



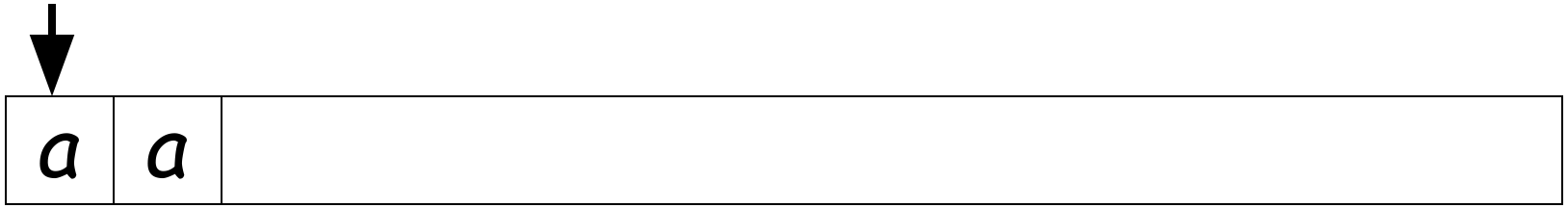

a
a

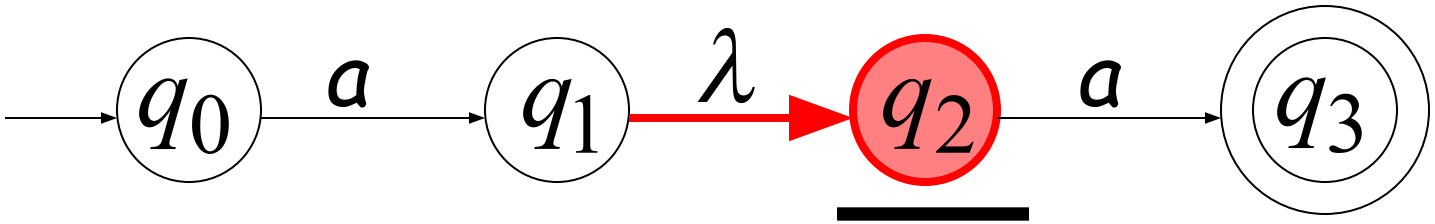

q_0
a
q_1
λ
q_2
a
q_3

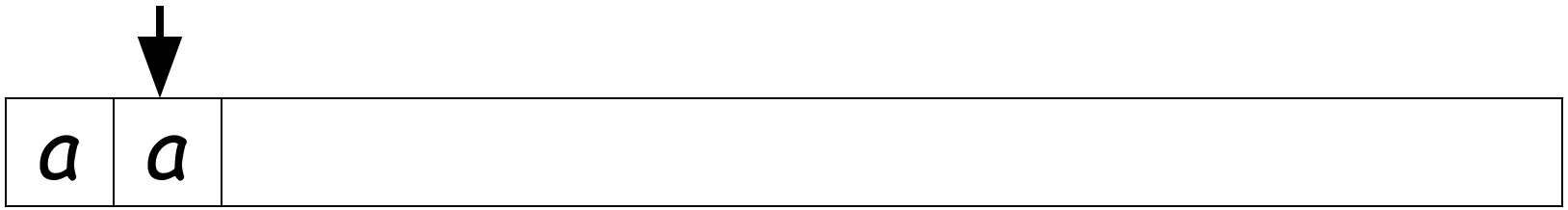

a
a

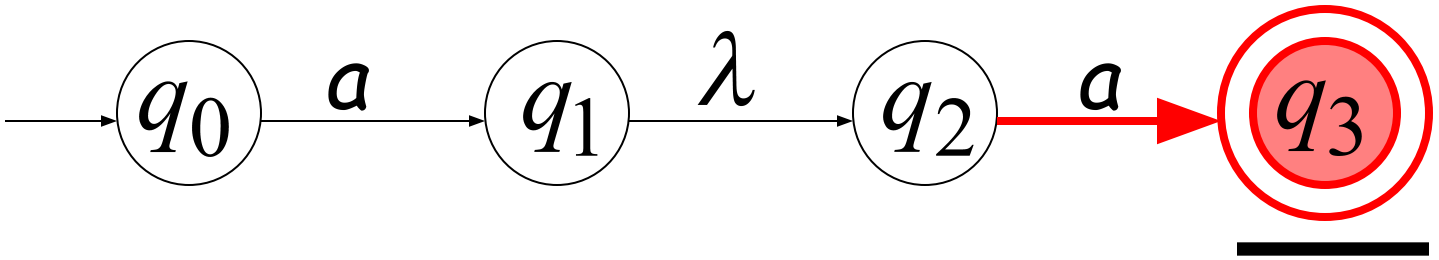

q_0
a
q_1
λ
q_2
a
q_3

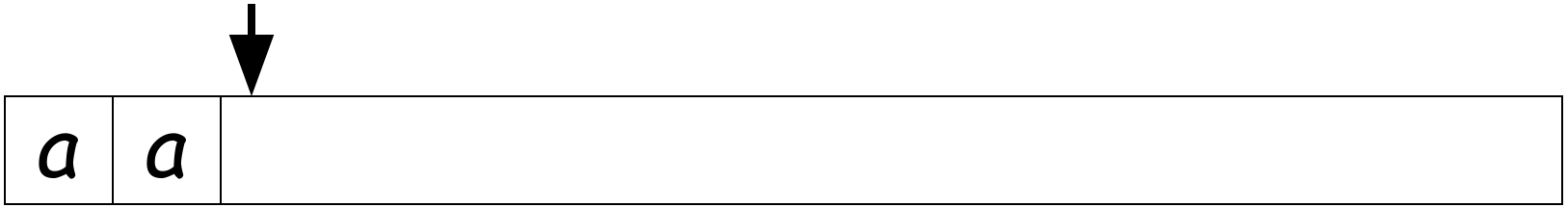
(read head doesn't move)



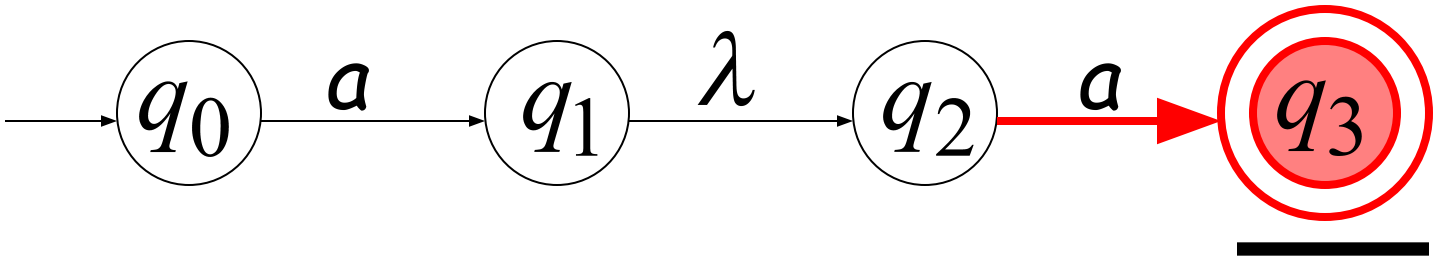



a
a


q_0
a
q_1
λ
q_2
a
q_3

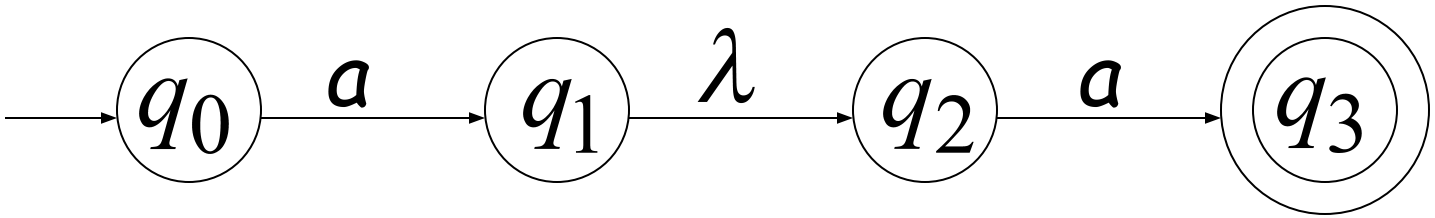


"accept"

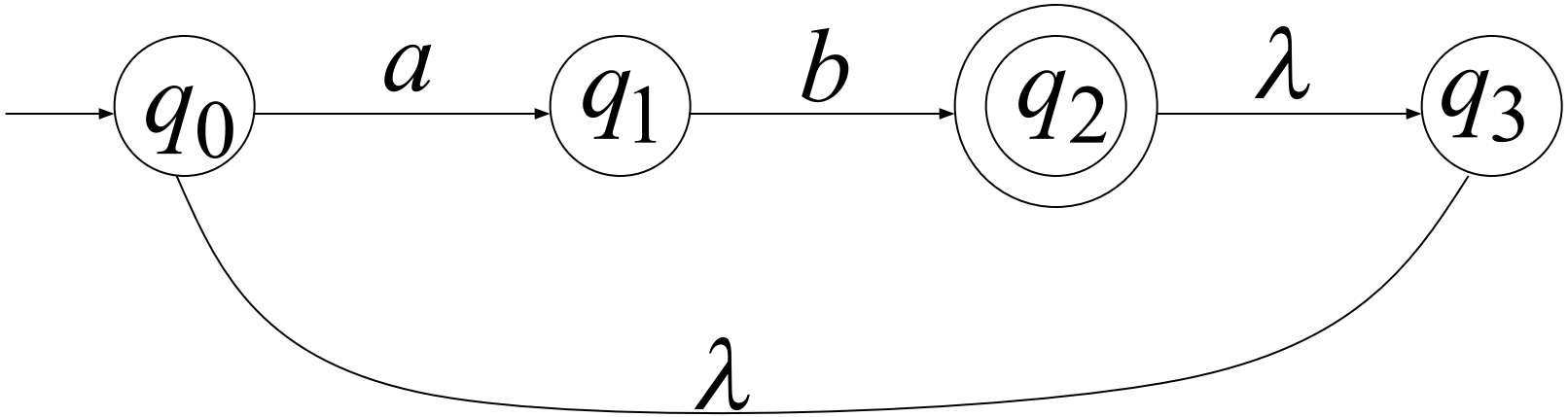


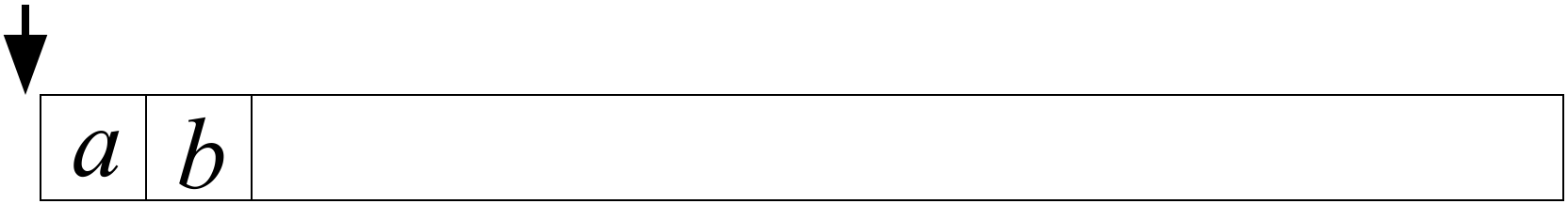
String aa is accepted

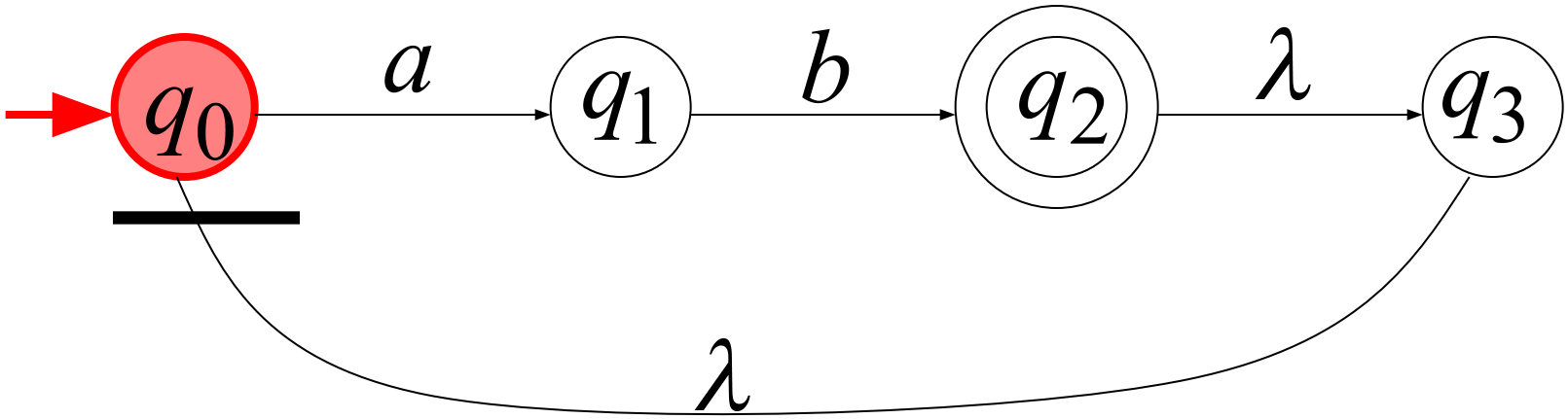
Language accepted: $L = \{aa\}$

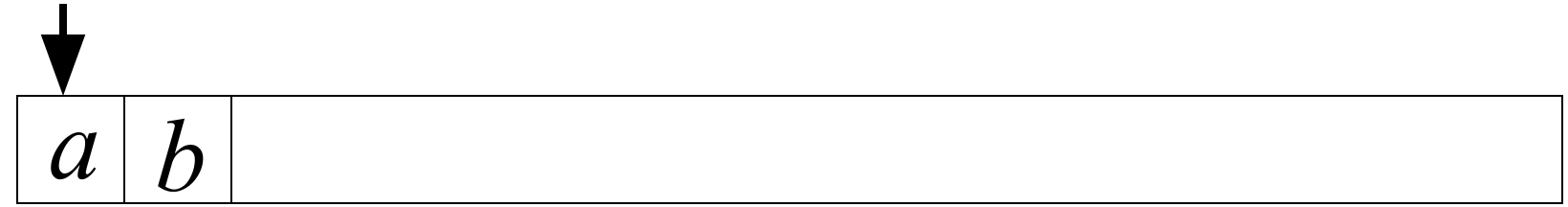


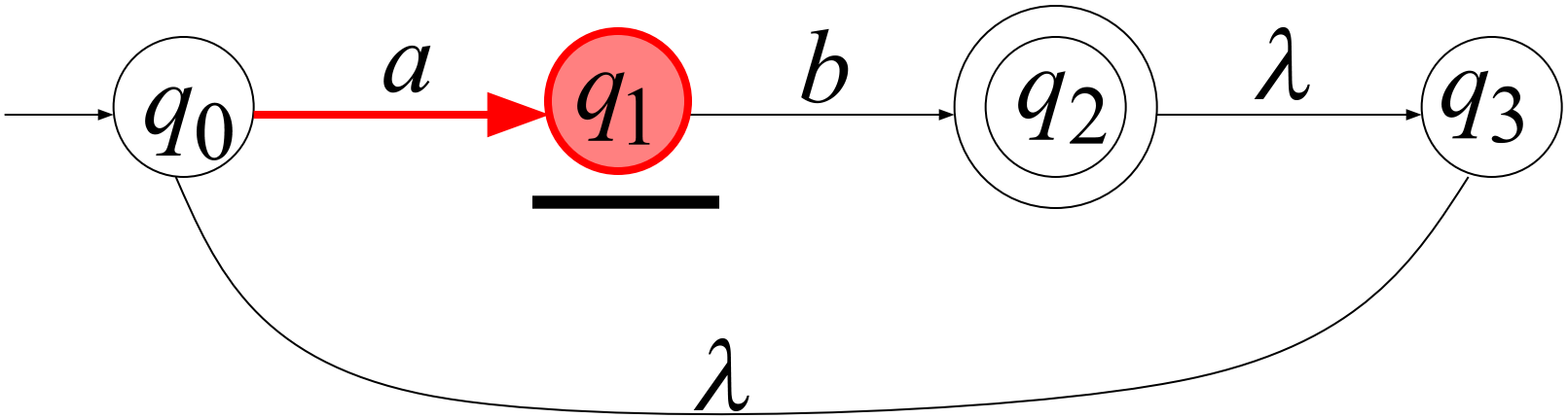
Another NFA Example

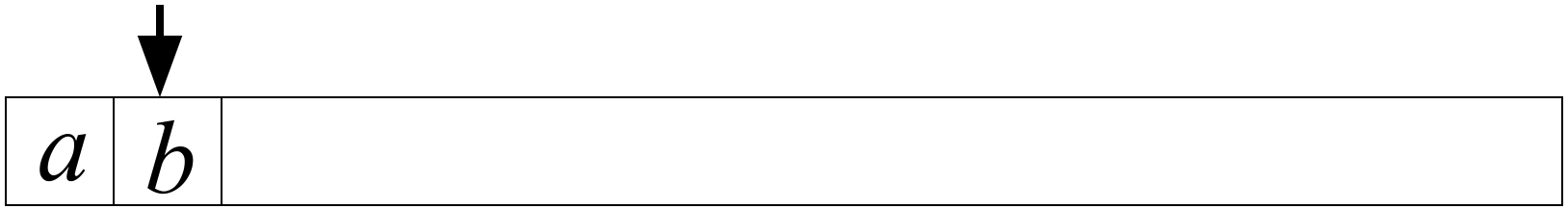


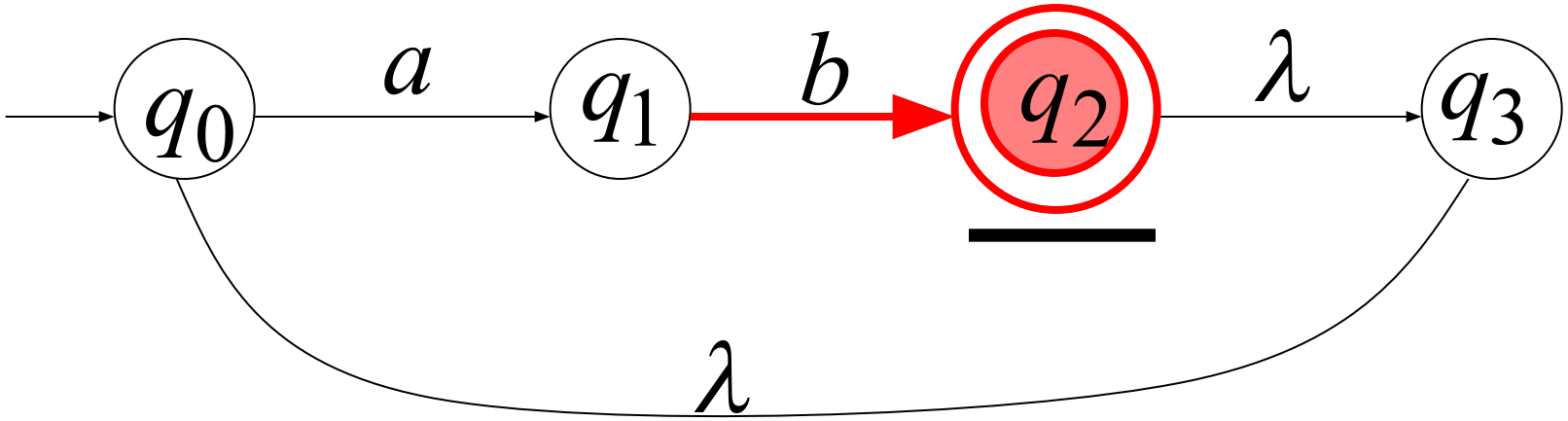

a
b

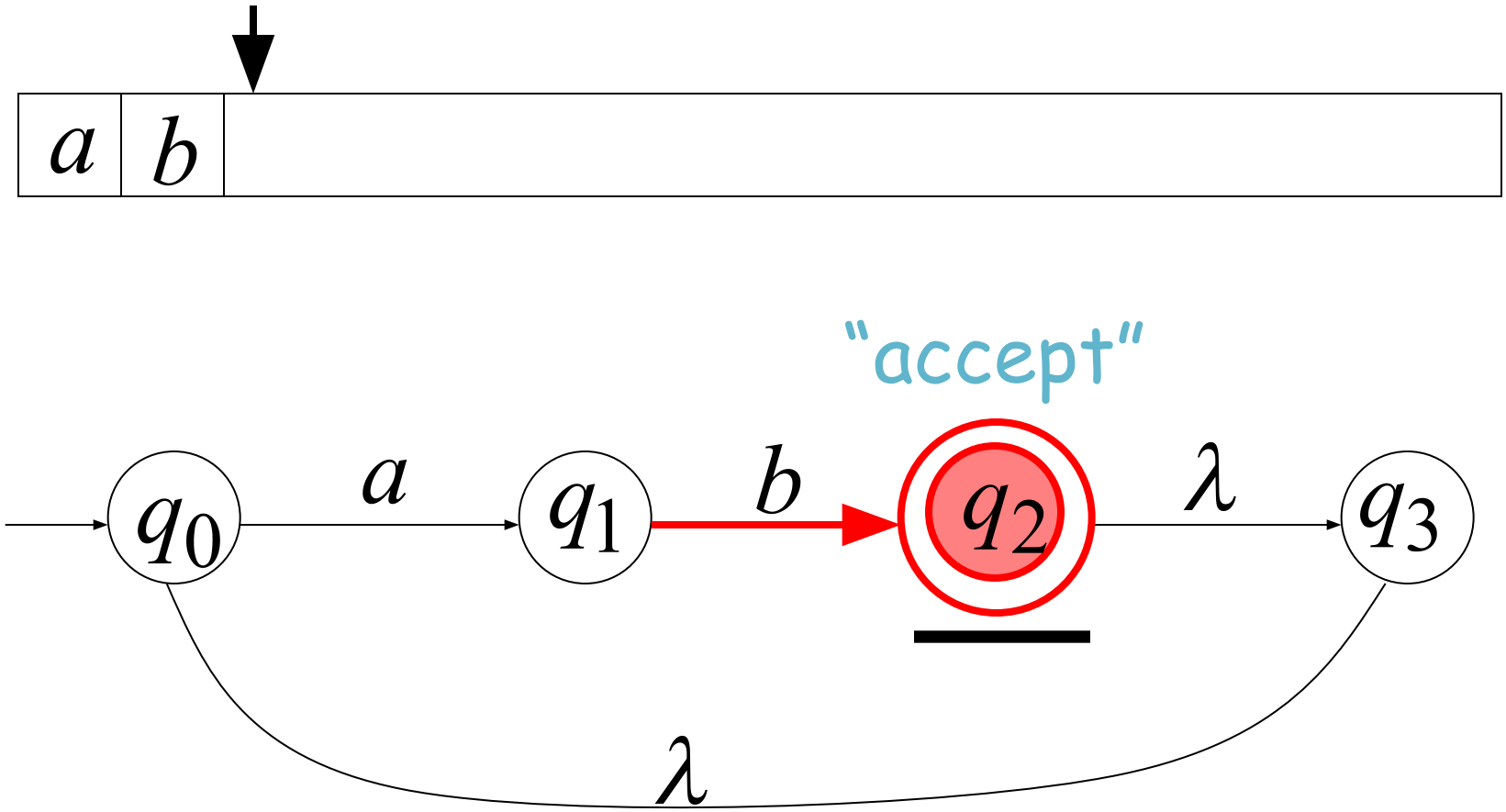

q_0
a
q_1
b
q_2
λ
q_3
λ


a
b

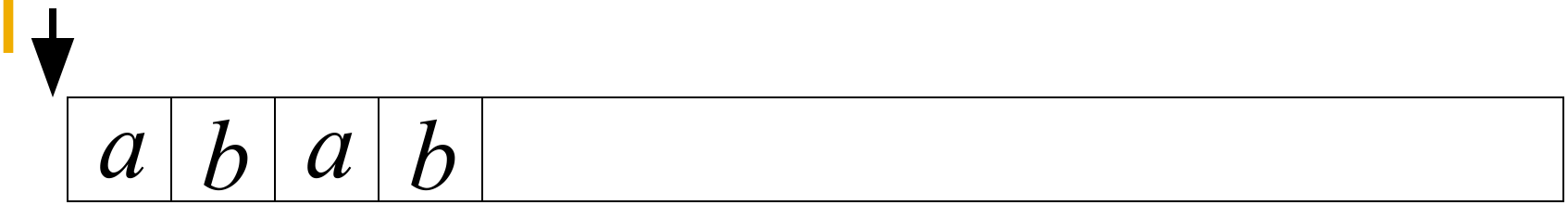

q_0
a
q_1
b
q_2
λ
q_3
λ

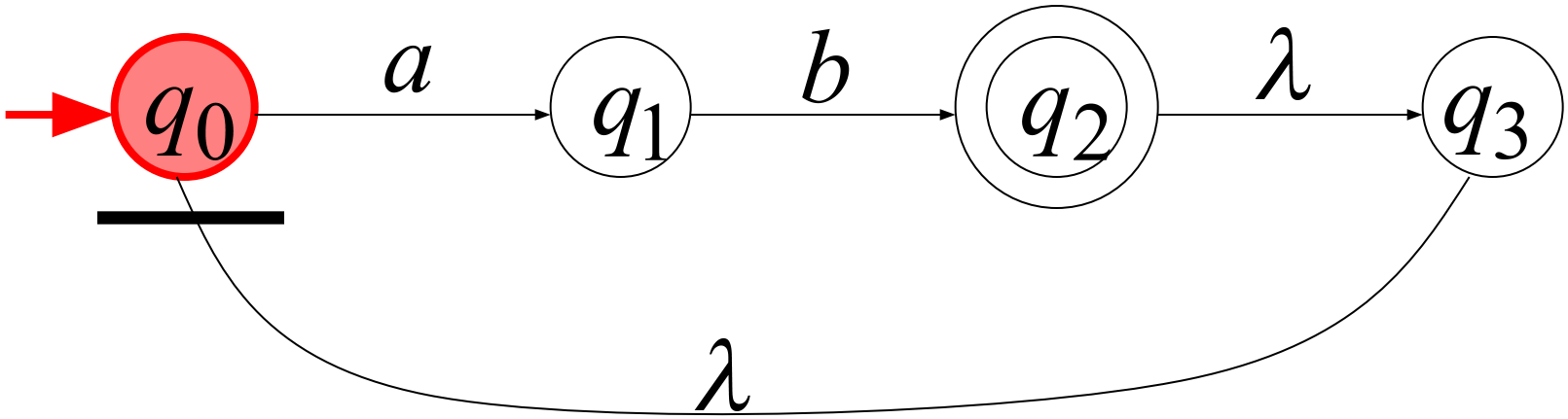

a
b

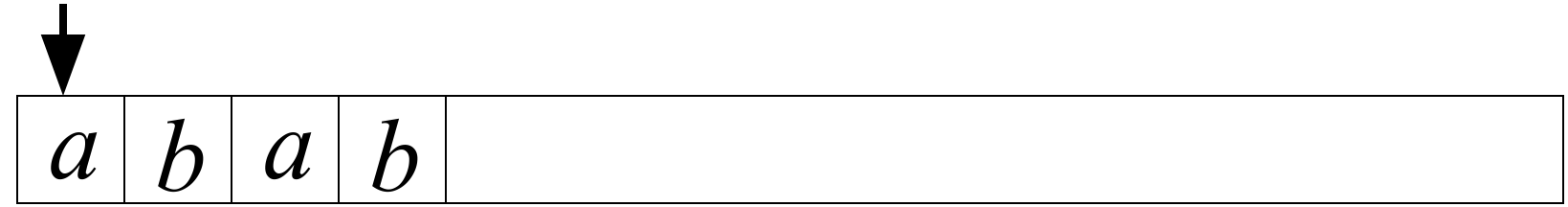

q_0
a
q_1
b
q_2
λ
q_3
λ

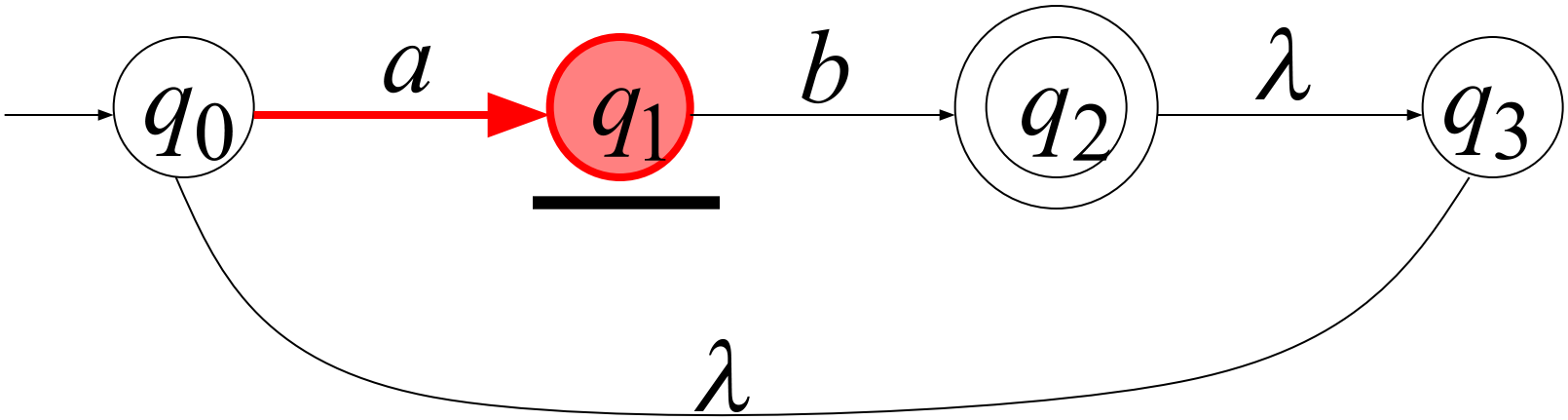

a
b
"accept"
q_0
a
q_1
b
q_2
λ
q_3
λ

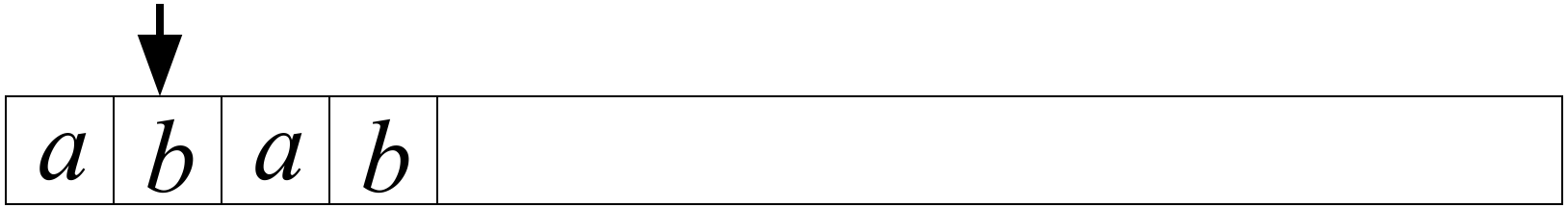
Another String

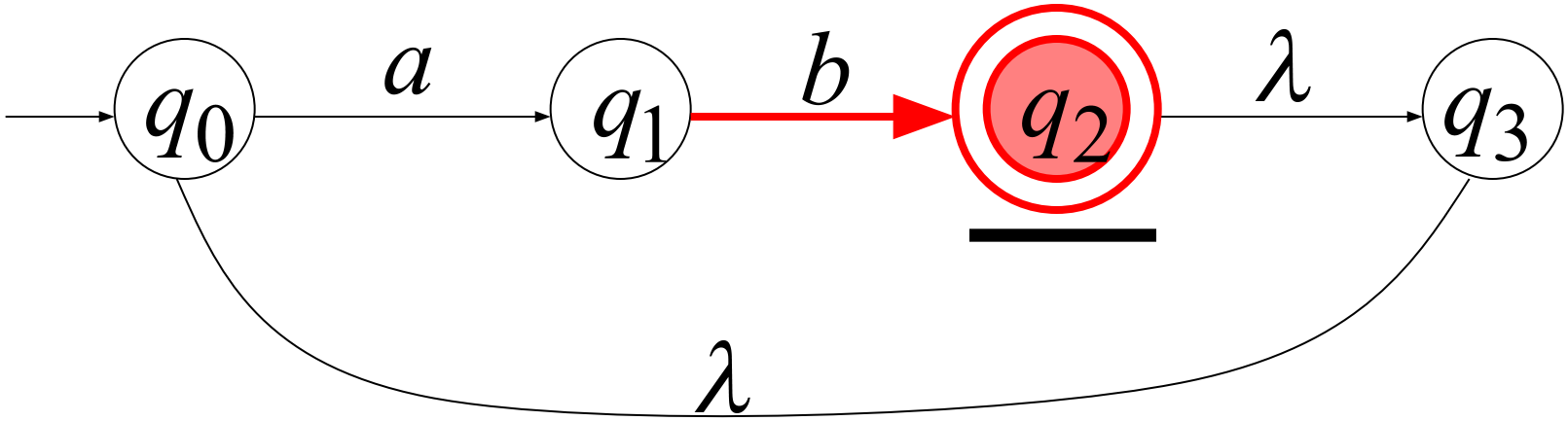


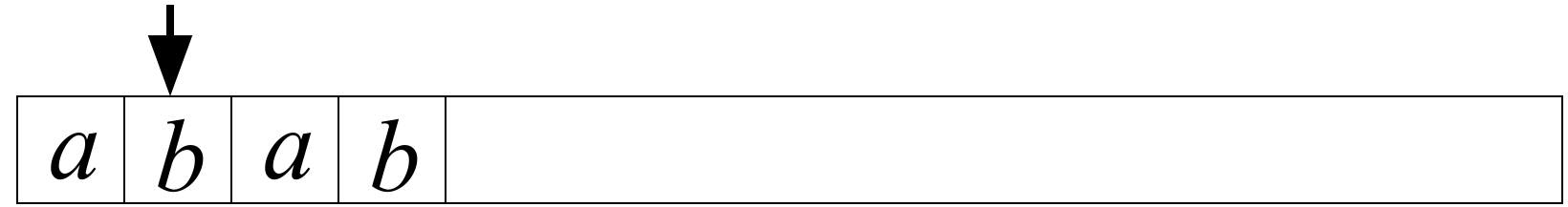


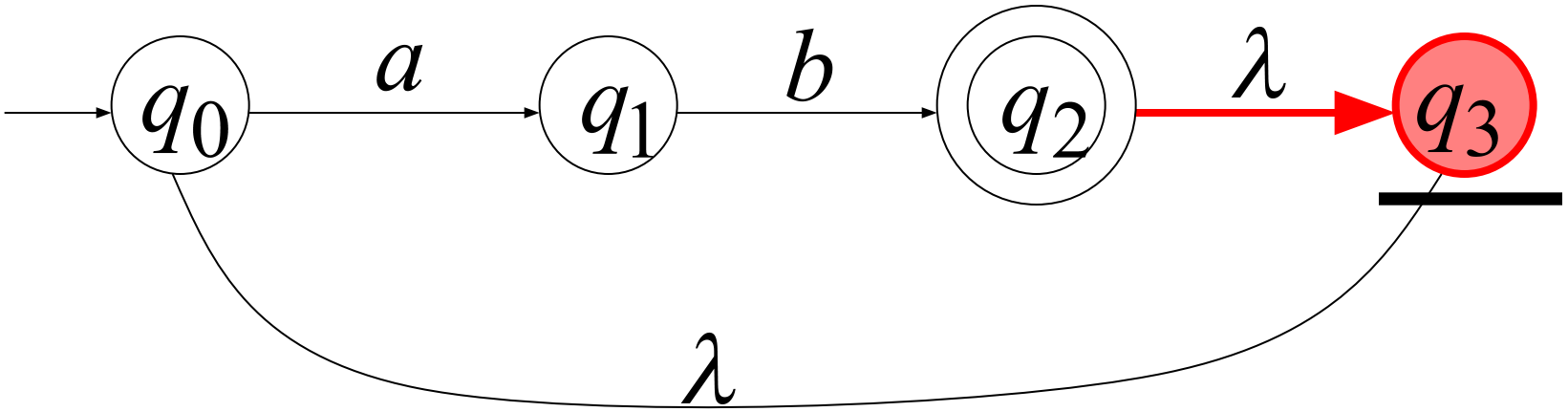

a
b
a
b

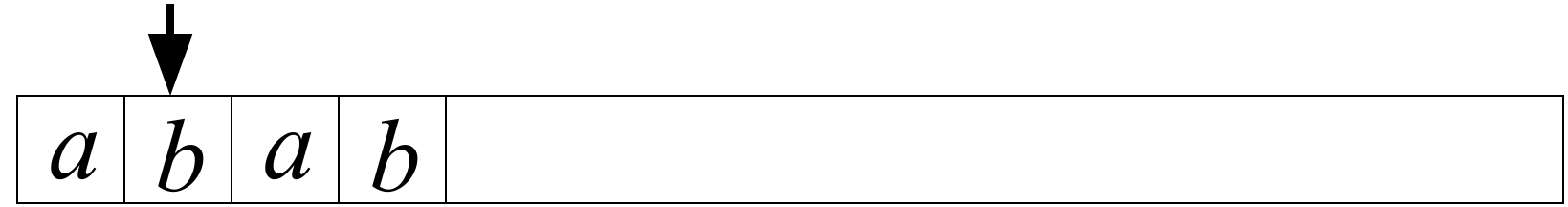

q_0
a
q_1
b
q_2
λ
q_3
λ

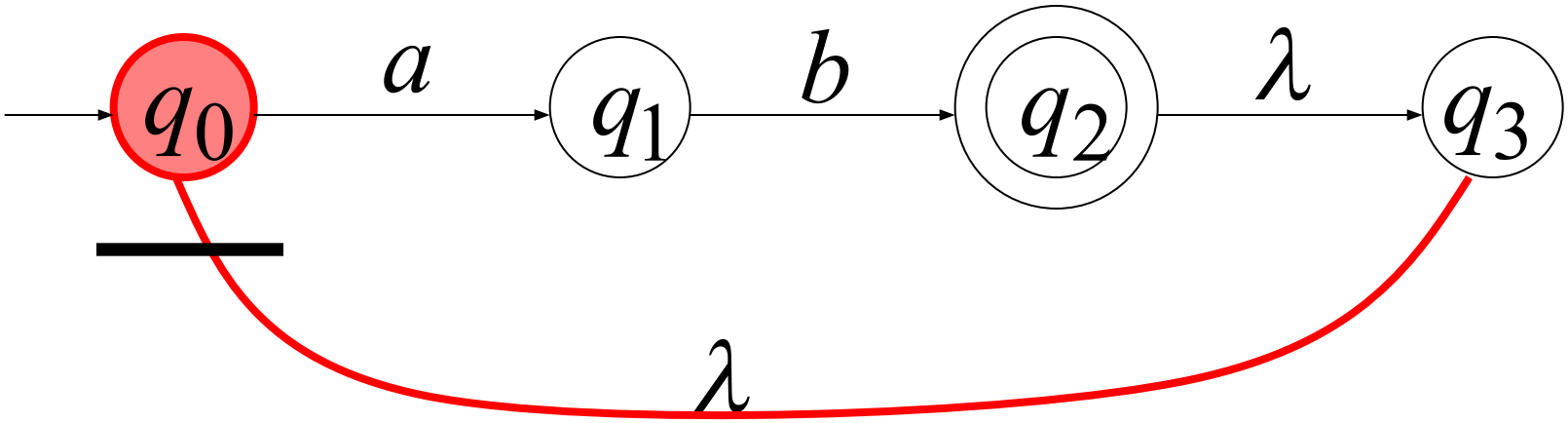

a
b
a
b

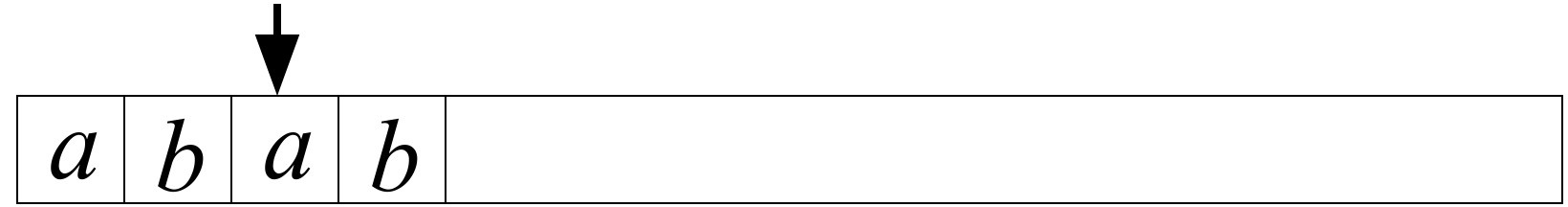

q_0
a
q_1
b
q_2
λ
q_3
λ

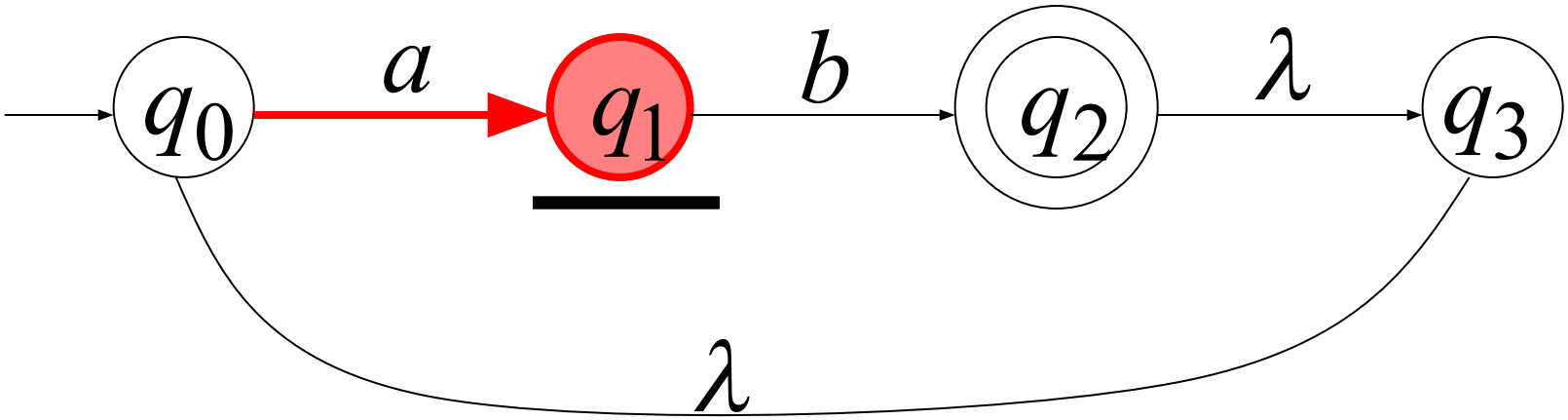

a
b
a
b

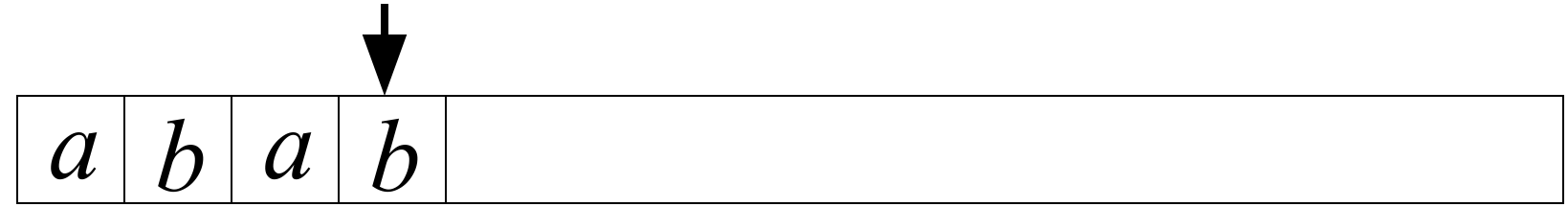

q_0
a
q_1
b
q_2
λ
q_3
λ

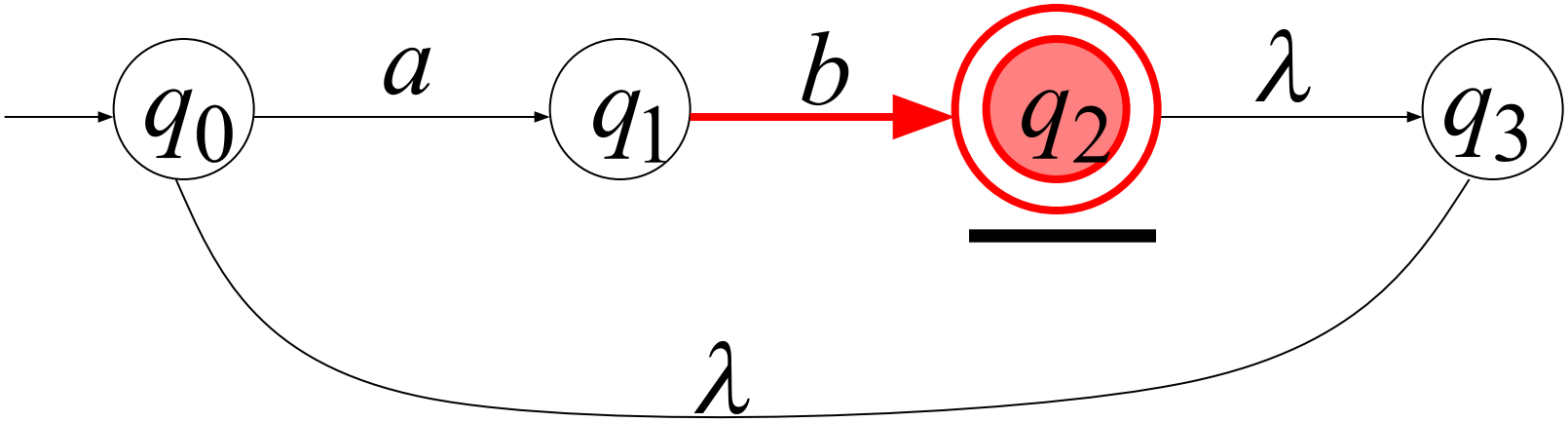

a
b
a
b

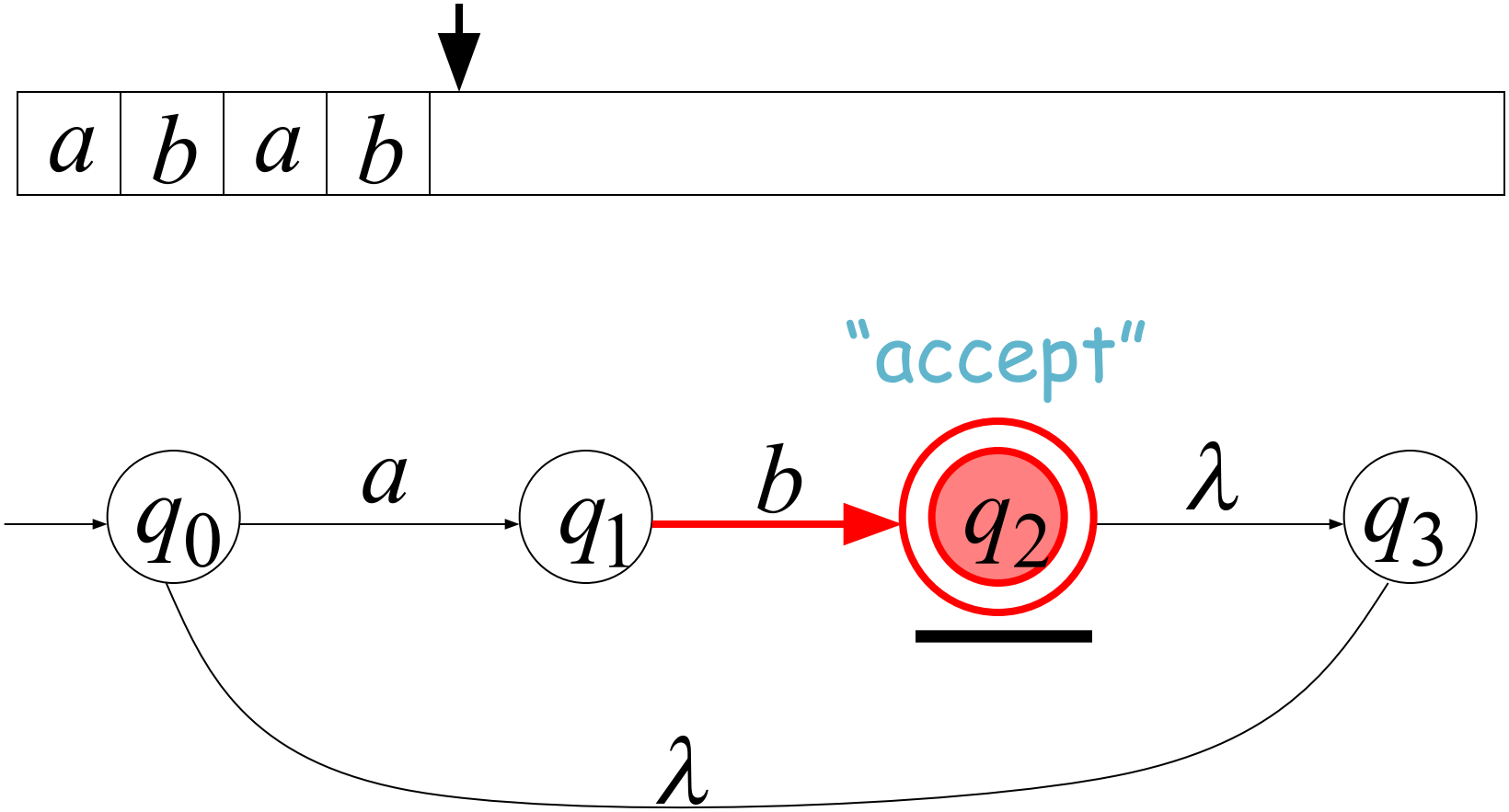

q_0
a
q_1
b
q_2
λ
q_3
λ


a
b
a
b


q_0
a
q_1
b
q_2
λ
q_3
λ


a
b
a
b

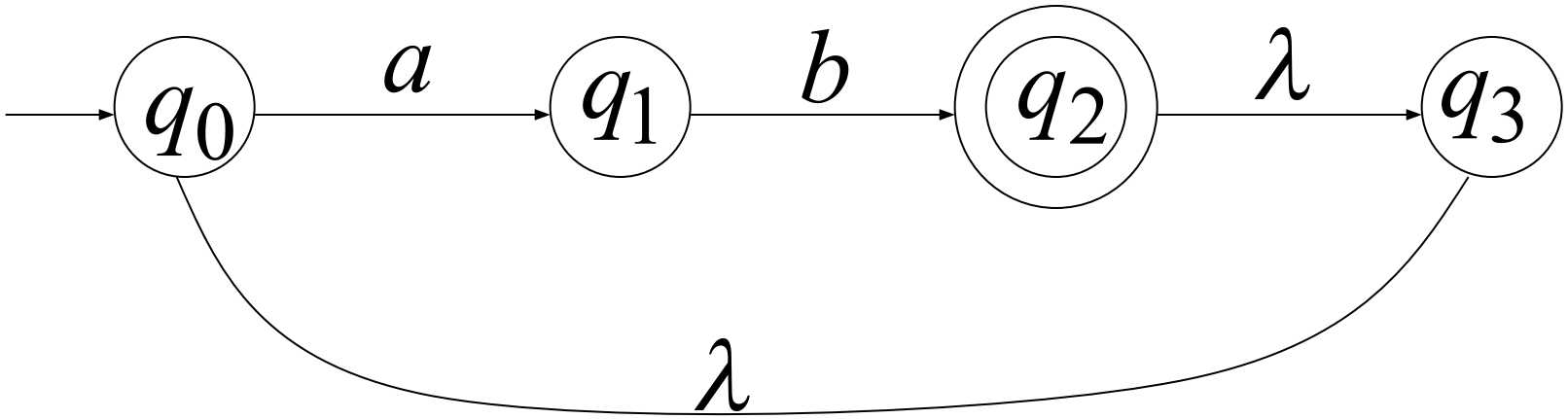

q_0
a
q_1
b
q_2
λ
q_3
λ


a
b
a
b
"accept"
q_0
q_1
q_2
q_3
a
b
λ
λ

Language accepted

$$L = \{ab,\ abab,\ ababab,\ ...\}$$

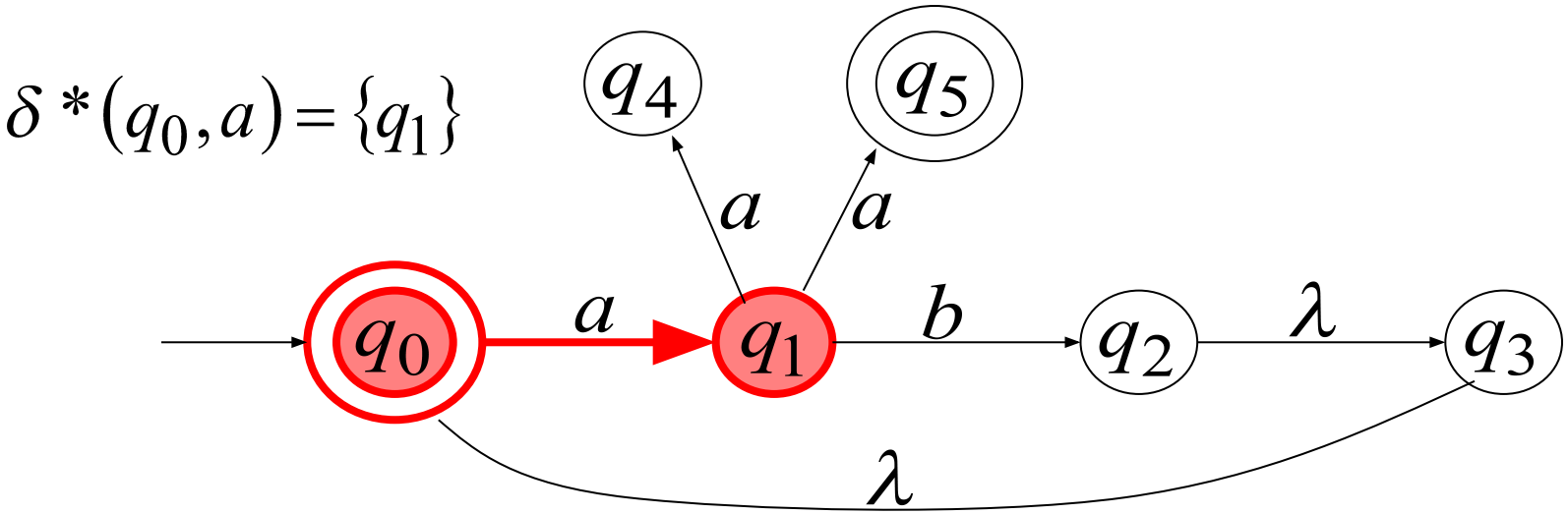
$$= \{ab\}^{+}$$



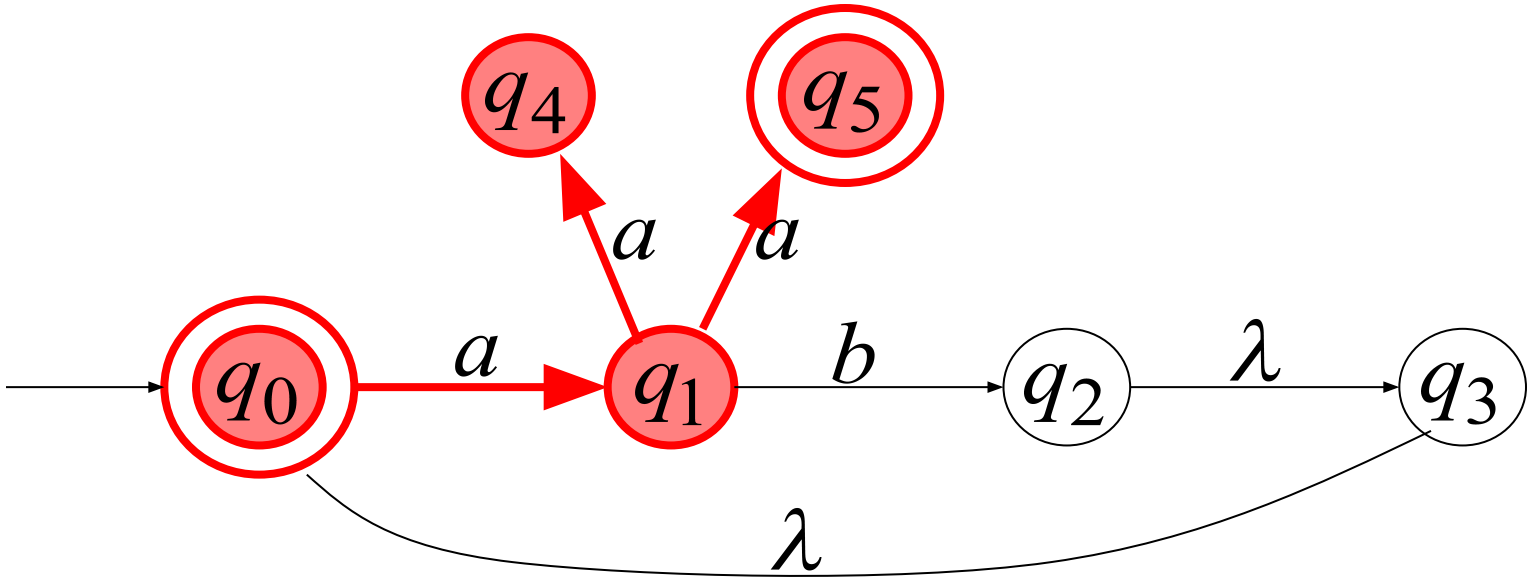
Extended Transition δ^*

- **Definition:** The language accepted by a Non Determentistis finite automaton (**NFA)** M = (Q, Σ, δ, q_0, F) is the set of all strings on Σ accepted by M. Formally,

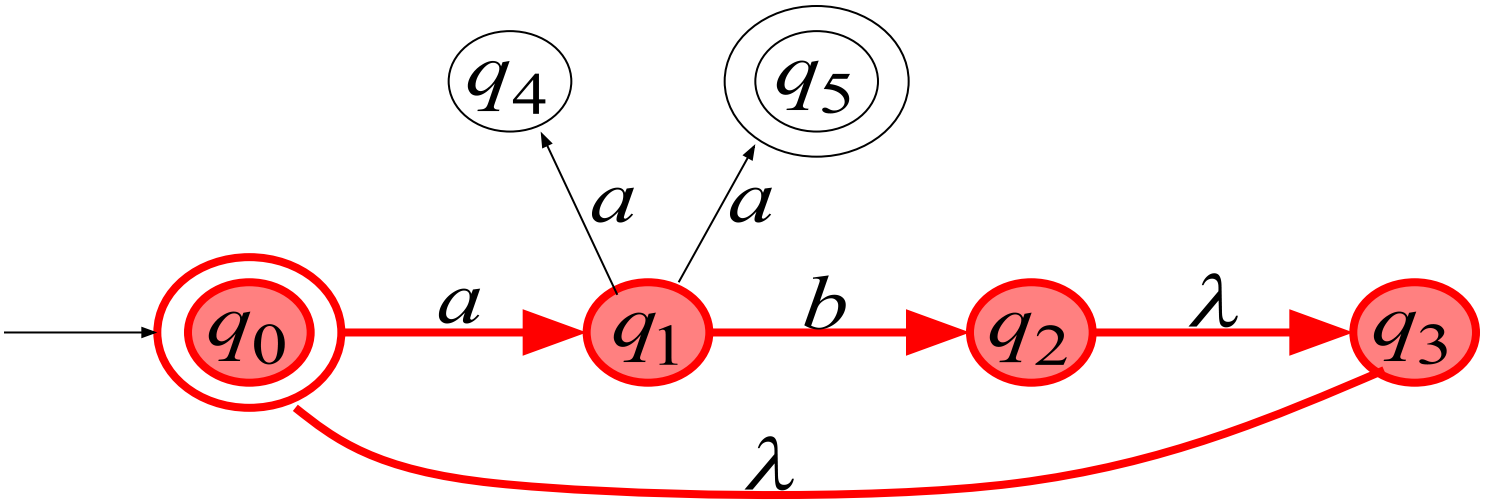
$$L(M) = \{w \in \Sigma^* : \delta^*(q_0, w) \cap F \neq \varnothing \}$$



$$\delta^*(q_0, aa) = \{q_4, q_5\}$$

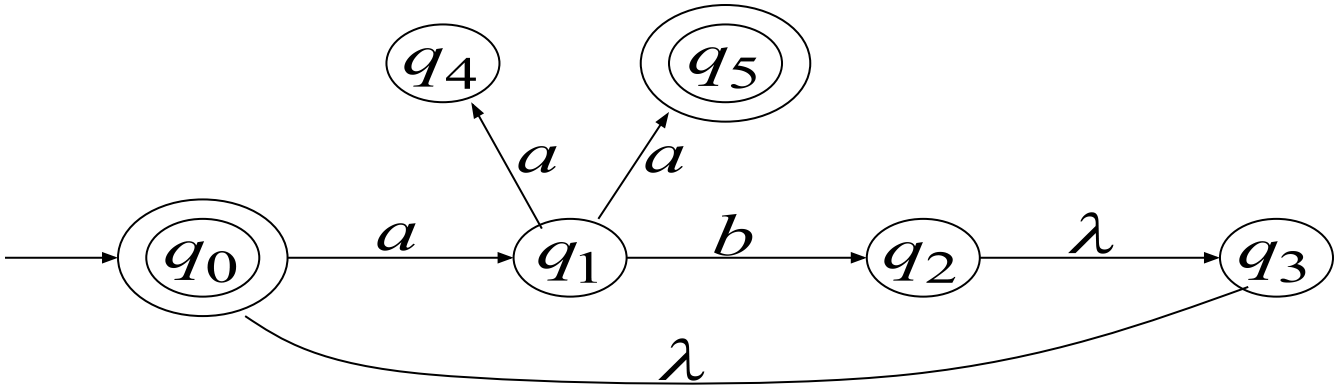


$$\delta^*(q_0, ab) = \{q_2, q_3, q_0\}$$



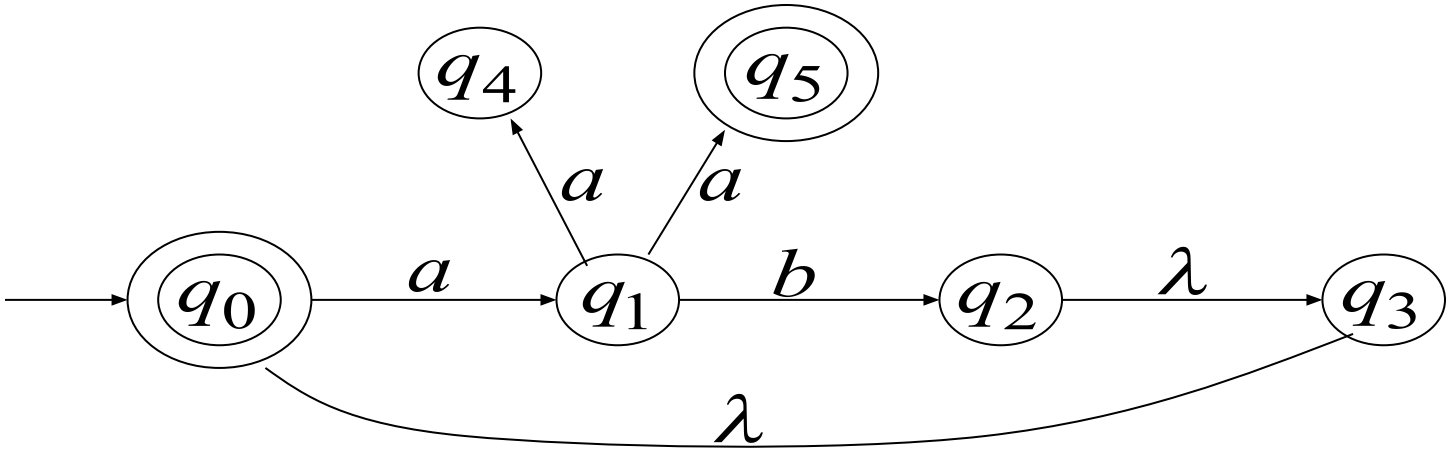
The Language of an NFA M

If $F = \{q_0, q_5\}$ $\delta^*(q_0, aa) = \{q_4, q_5\}$



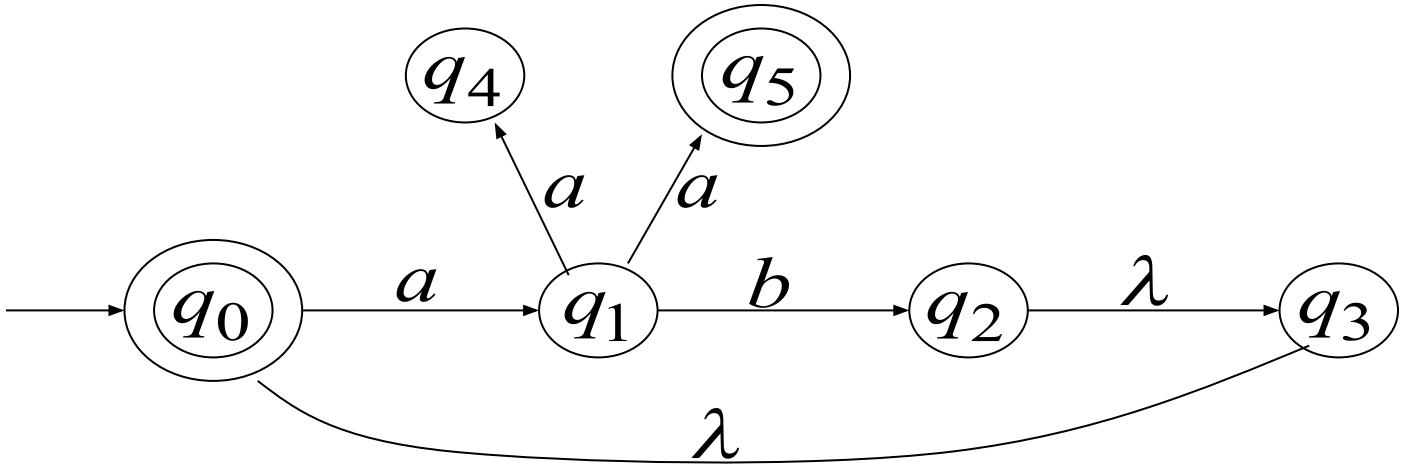
then $aa \in L(M)$

$F = \{q_0, q_5\}$ $\delta^*(q_0, ab) = \{q_2, q_3, \underline{q_0}\}$



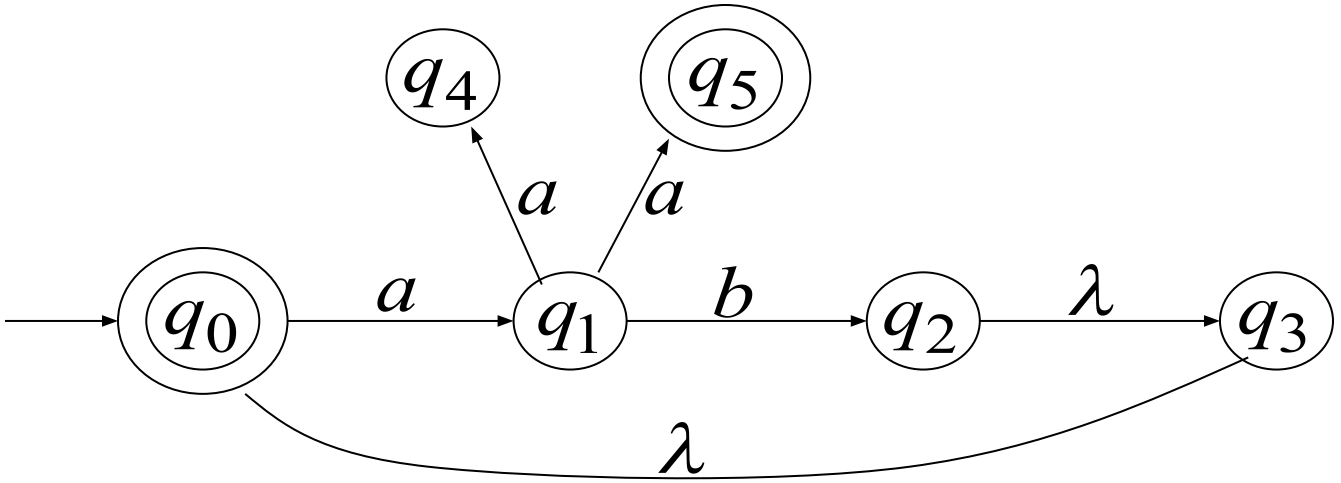
$ab \in L(M)$

- $F = \{q_0, q_5\}$ $\quad \delta^*(q_0, abaa) = \{q_4, \underline{q_5}\}$

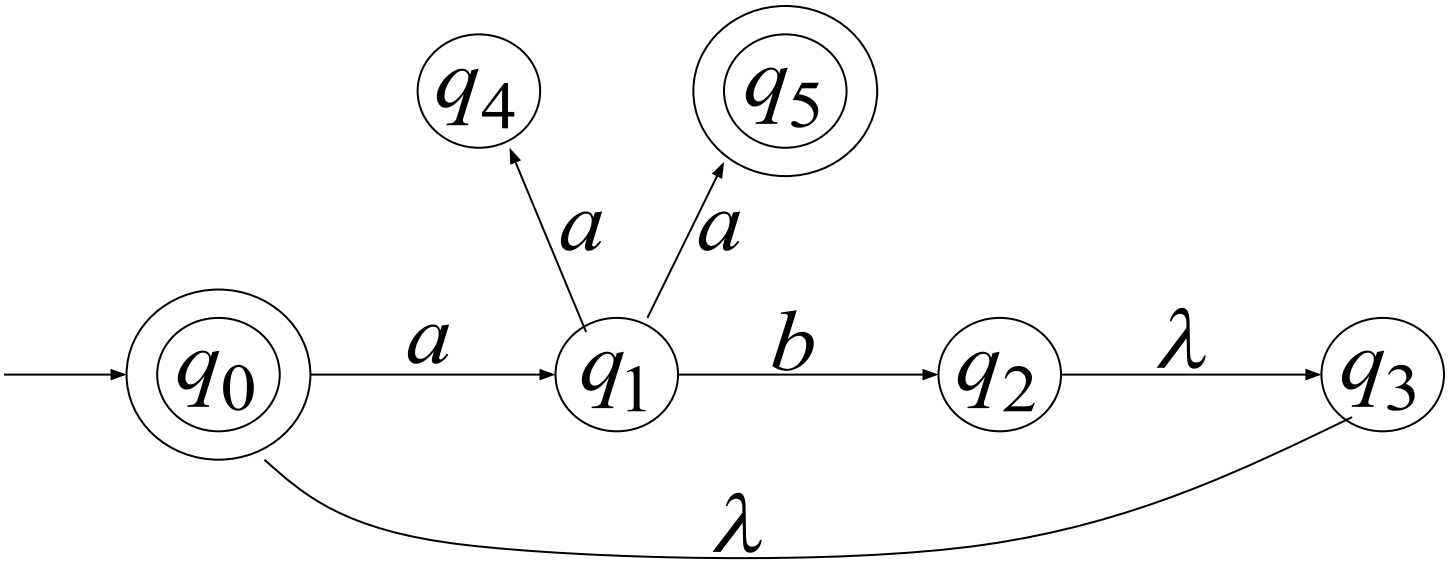


$$aaba \in L(M)$$

$F = \{q_0, q_5\}$ $\delta^*(q_0, aba) = \{q_1\}$

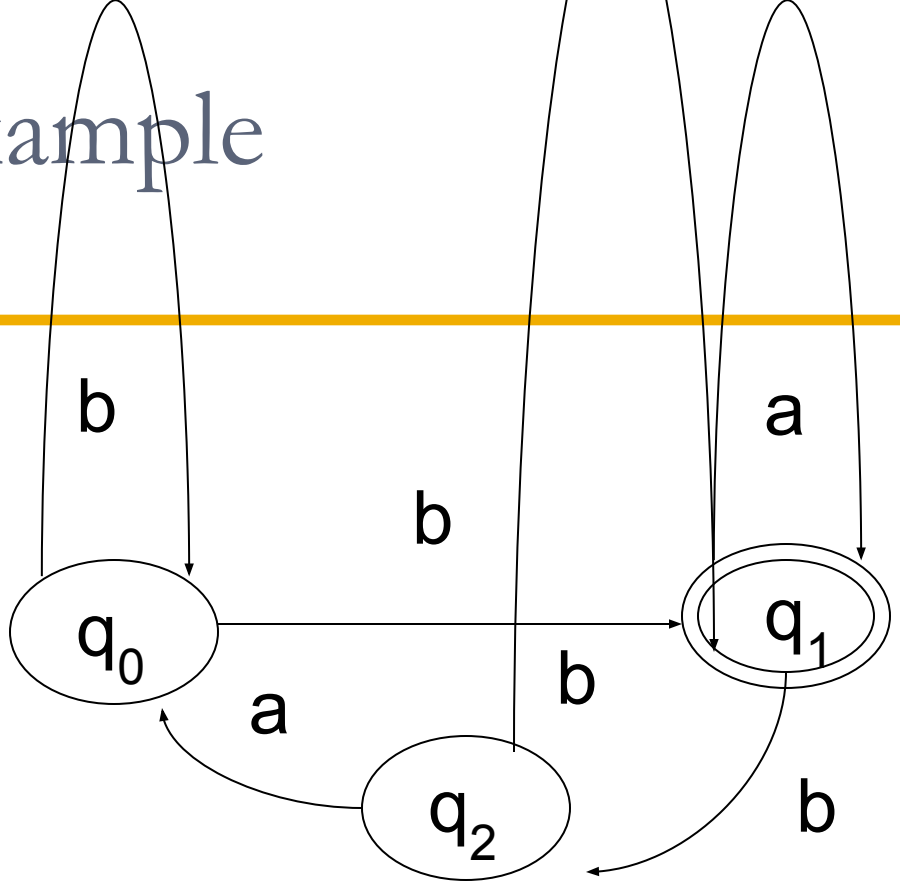


$aba \notin L(M)$



$$L(M) = \{ab\}^* \cup \{ab\}^* \{aa\}$$

NFA example



b	b	a	b	b

q_0 q_0 q_1 q_1 q_2 q_1

q_0 q_1 q_2 q_0 q_0 q_0

b	b	a	b	b

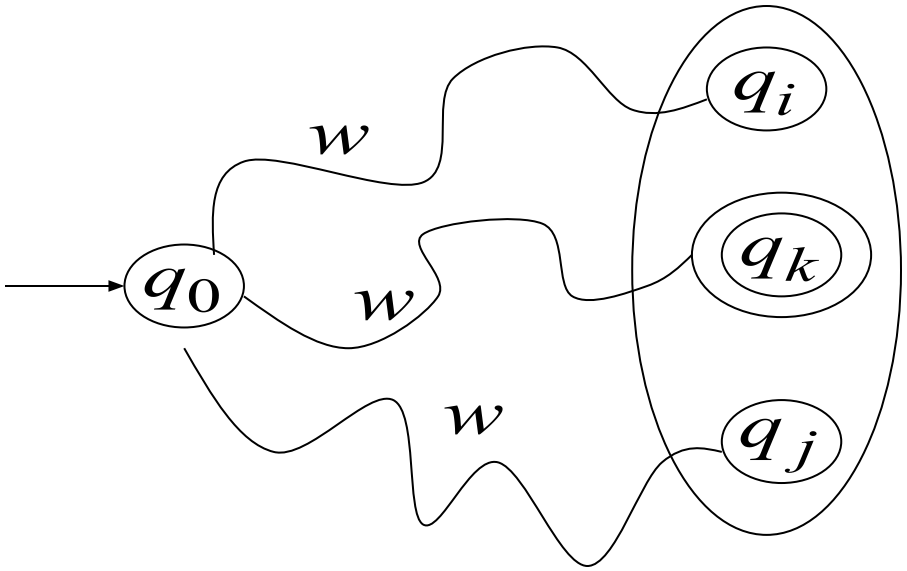
q_0 q_1 q_2 q_0 q_0 q_1

q_0 q_1 q_2 q_0 q_1 q_2

Nondeterministic Finite Automata

- **Definition:** The language accepted by a nfa M = (Q, Σ, δ, q_0, F) is the set of all strings on Σ accepted by M. Formally,

$$\mathbf{L(M) = \{w \in \Sigma^* : \delta^*(q_0, w) \cap F \neq \varnothing\}}$$

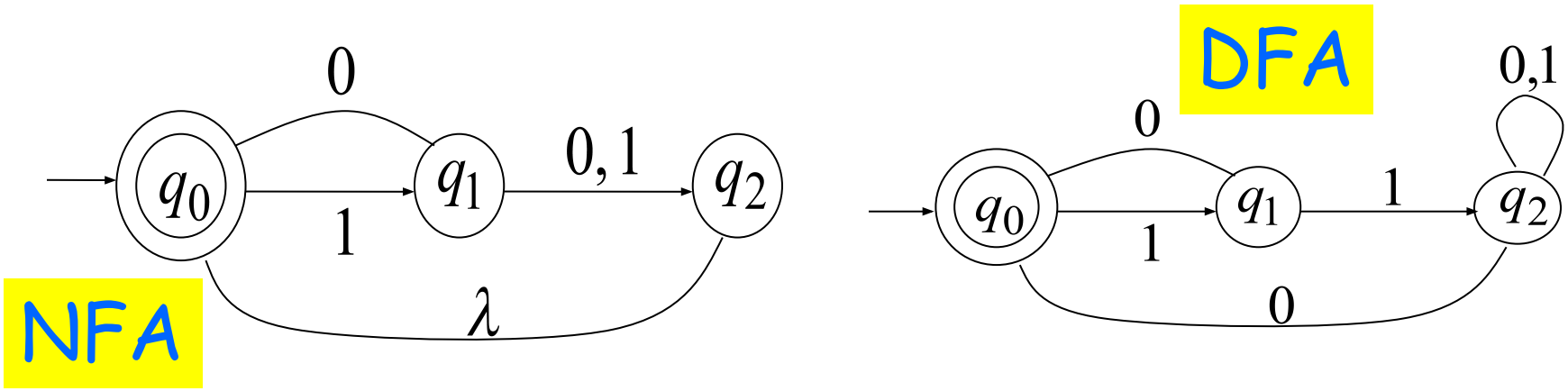


NFA and DFA

Example:

The following are NFA and the DFA that describes the language

$$L=\{(10)^n : n\geq 0\}.$$

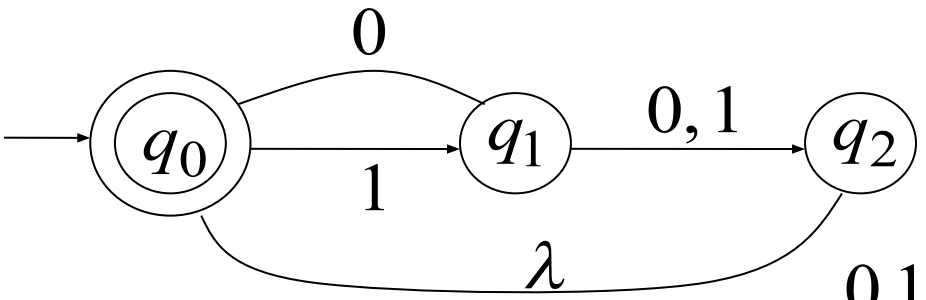


NFA and DFA

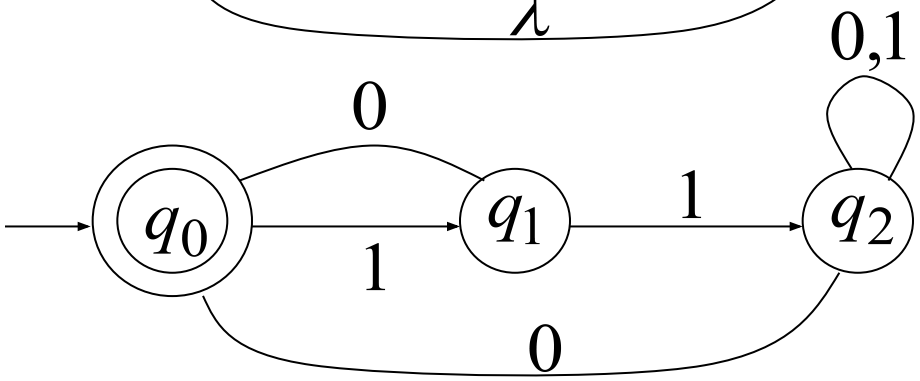
- Since $L(M_1) = L(M_2) = \{10\}^*$

Then machines M_1 and M_2 are equivalent

NFA M_1



DFA M_2



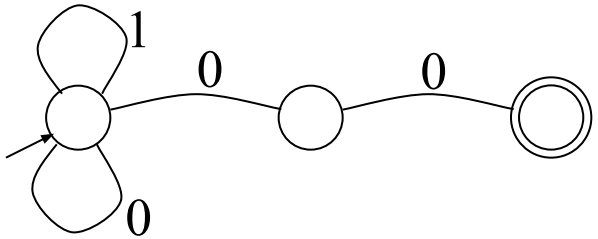
NFA vs. DFA (1)

- NFAs and DFAs recognize the same set of languages (regular languages)
- DFAs are easier to implement
  - There are no choices to consider
- However, the NFAs are usually simpler than DFAs
  - They do not need an edge out of each node for each letter of the alphabet
  - It is suitable for the game playing program. In which, the best move is unknown so the backtracking is necessary.

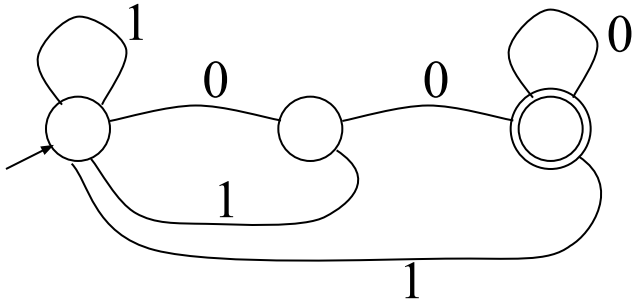
NFA vs. DFA (2)

- For a given language the NFA can be simpler than the DFA

NFA



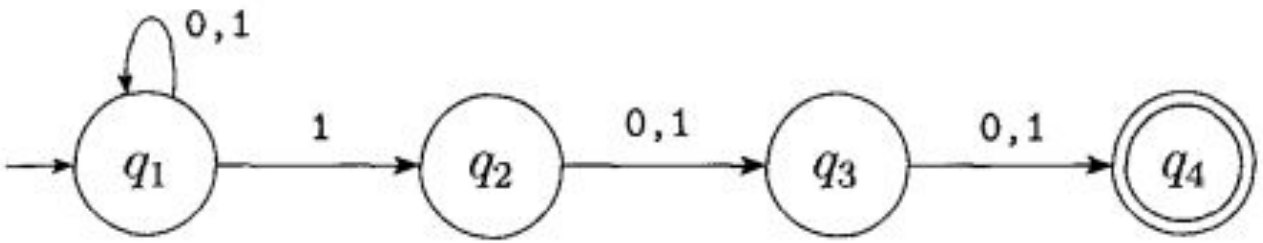
DFA



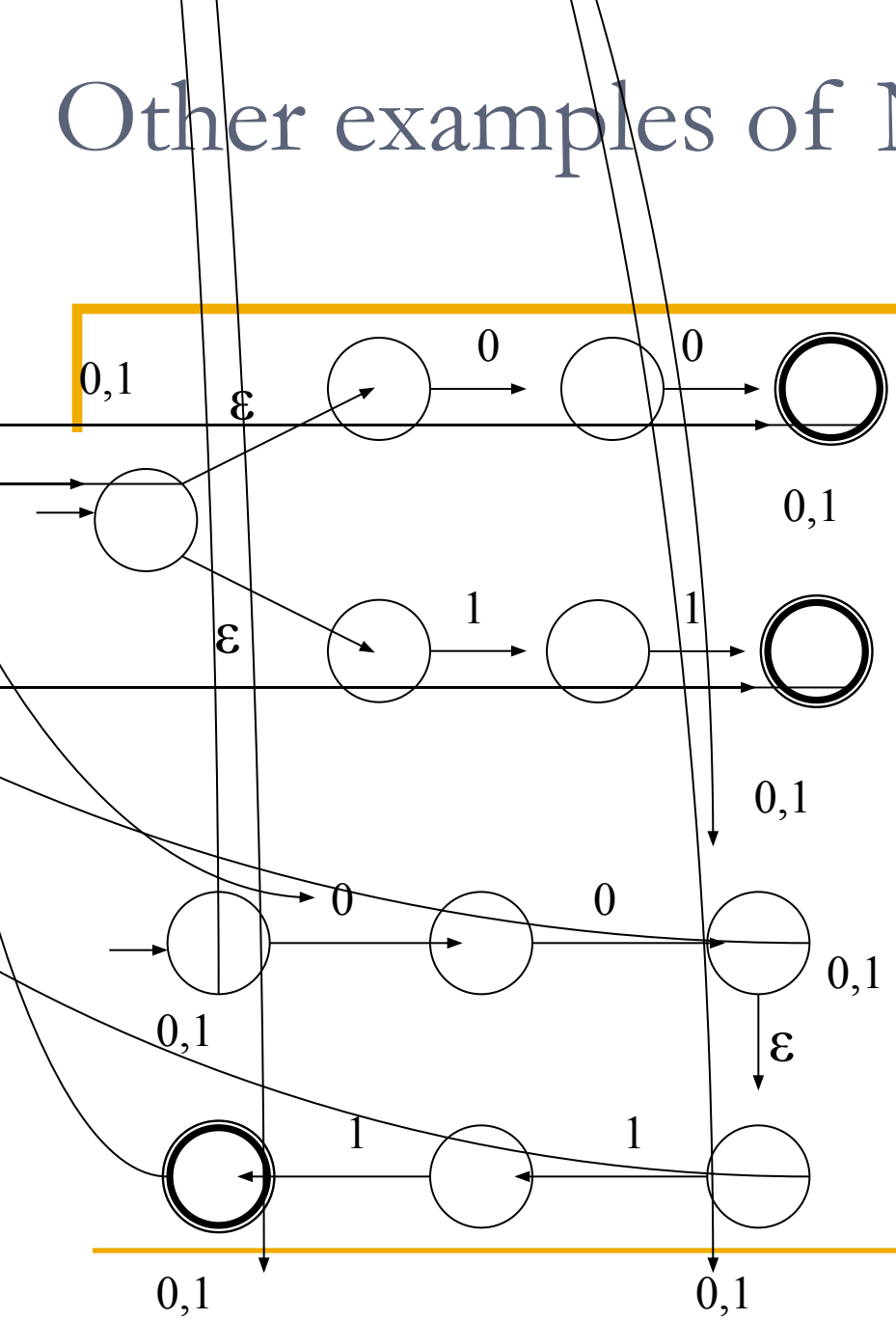
- DFA can be exponentially larger than NFA

Draw an NFA

- Draw an NFA that accepts the language L={all strings over {0,1} that containing a 1 from the third position from the end}



Other examples of NFA

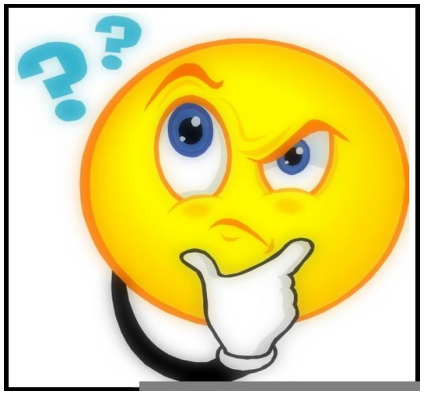


- $\{w \in \{0,1\}^* \mid w$ has either 00 or 11 as substring$\}$
- $\{w \in \{0,1\}^* \mid w$ has 00 and 11 as substrings and 00 comes before 11$\}$

Attempt

- Draw a nondeterministic FA that accepts $\{w \in \{a, b\}^* : w$ contains at least one instance of aaba, bbb or ababa$\}$.

IS NFA MODEL EQUIVALENT TO DFA MODEL???



CAN WE CONVERT NFA TO DFA???