23MT2014

THEORY OF COMPUTATION

Topic:

NFA AND ACCEPTANCE OF REGULAR LANGUAGE

Session - 4

AIM OF THE SESSION

The aim of the Non-Deterministic Finite Automata (NFA) course is to introduce students to the theory and applications of NFAs, enabling them to understand the fundamental concepts and techniques related to non-deterministic automata theory.

INSTRUCTIONAL OBJECTIVES

This Session is designed to:

1. To familiarize students with the concept of non-deterministic finite automata and their components, including states, alphabet, transitions, and accepting states.
2. To enable students to analyze and construct non-deterministic finite automata for various languages and regular expressions and also convert NFA equivalent DFA

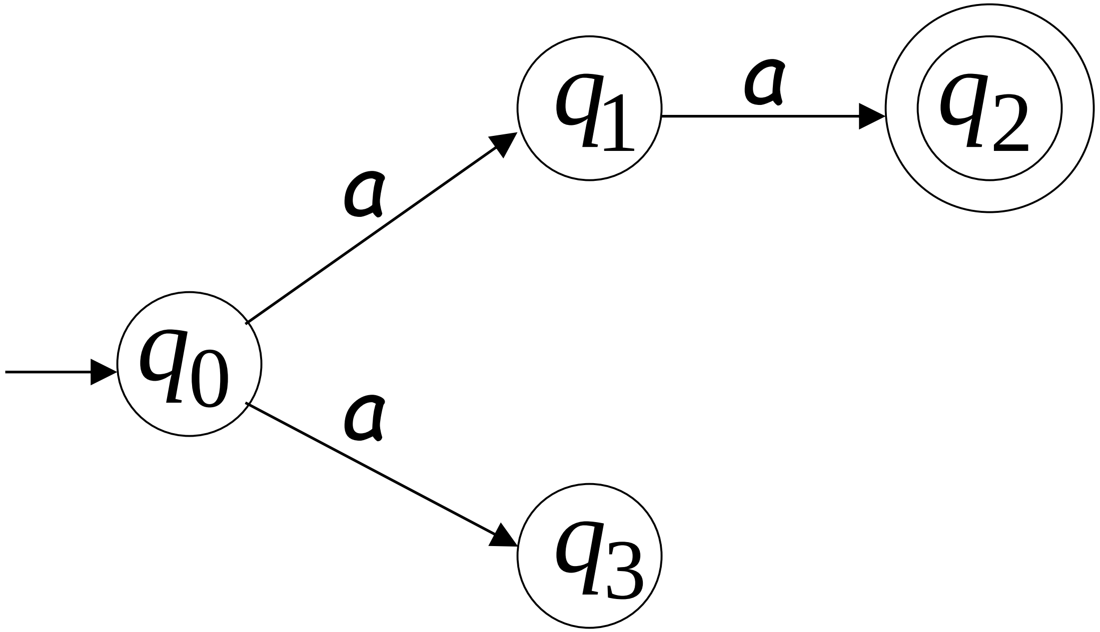
LEARNING OUTCOMES

At the end of this session, you should be able to:

1. Understand the concept of non-deterministic finite automata and its components, including states, alphabet, transitions, and accepting states.
2. Analyze and construct non-deterministic finite automata for given languages and regular expressions and also convert NFA equivalent DFA.

Nondeterministic Finite Accepter (NFA)

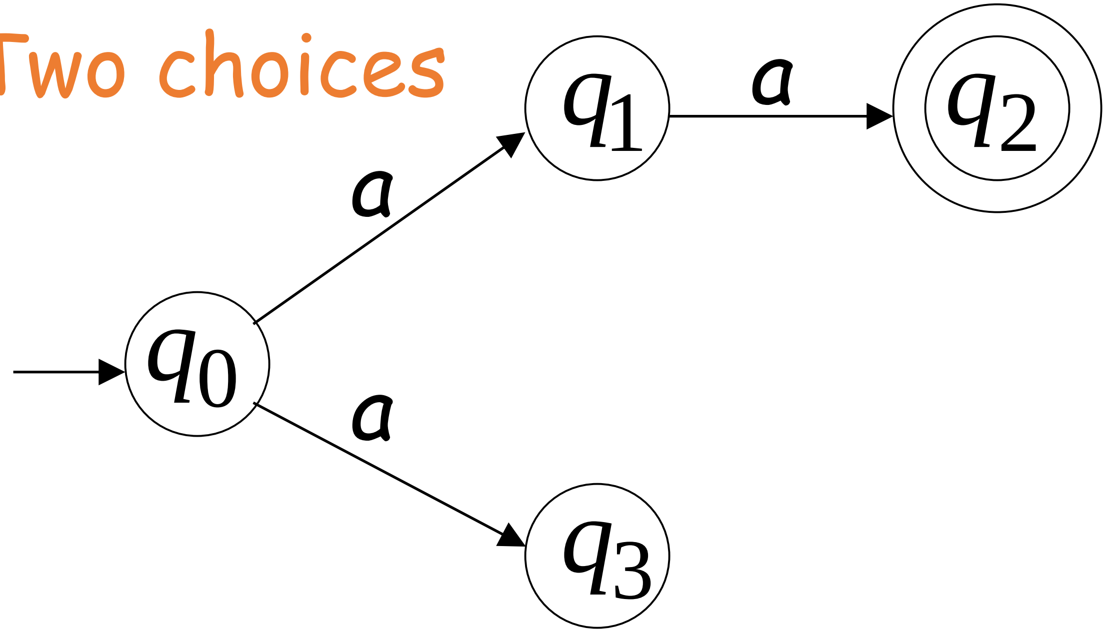
Alphabet = $\{a\}$



Nondeterministic Finite Accepter (NFA)

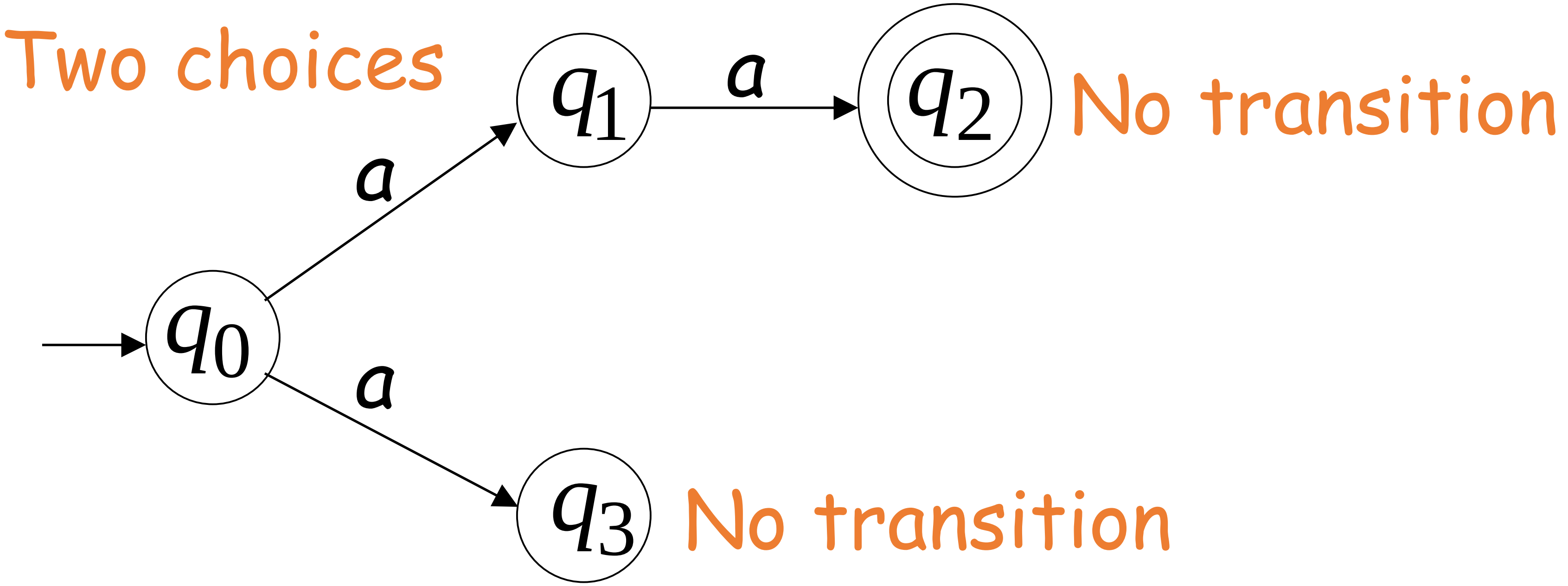
Alphabet = $\{a\}$

Two choices

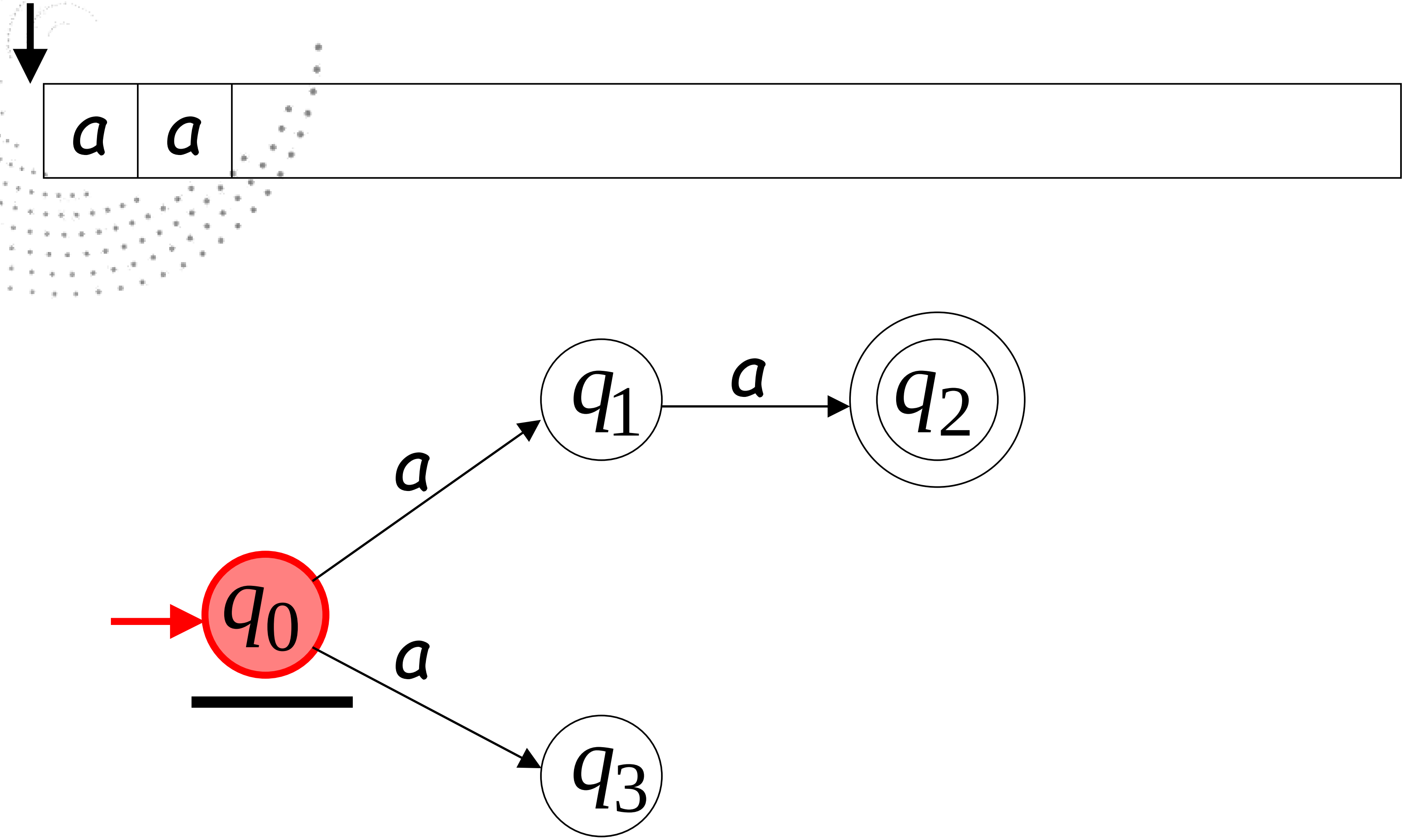


Nondeterministic Finite Accepter (NFA)

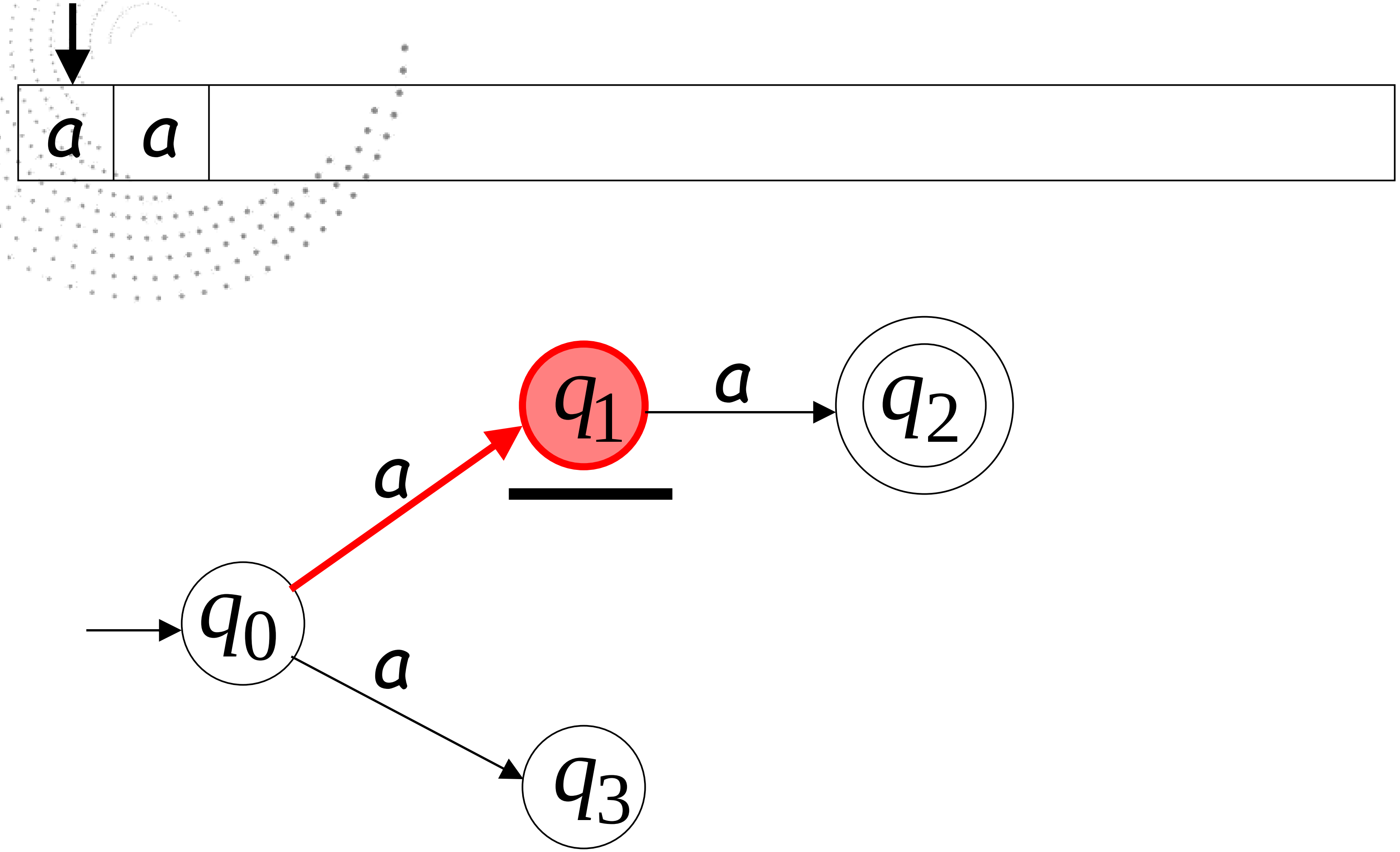
Alphabet = $\{a\}$



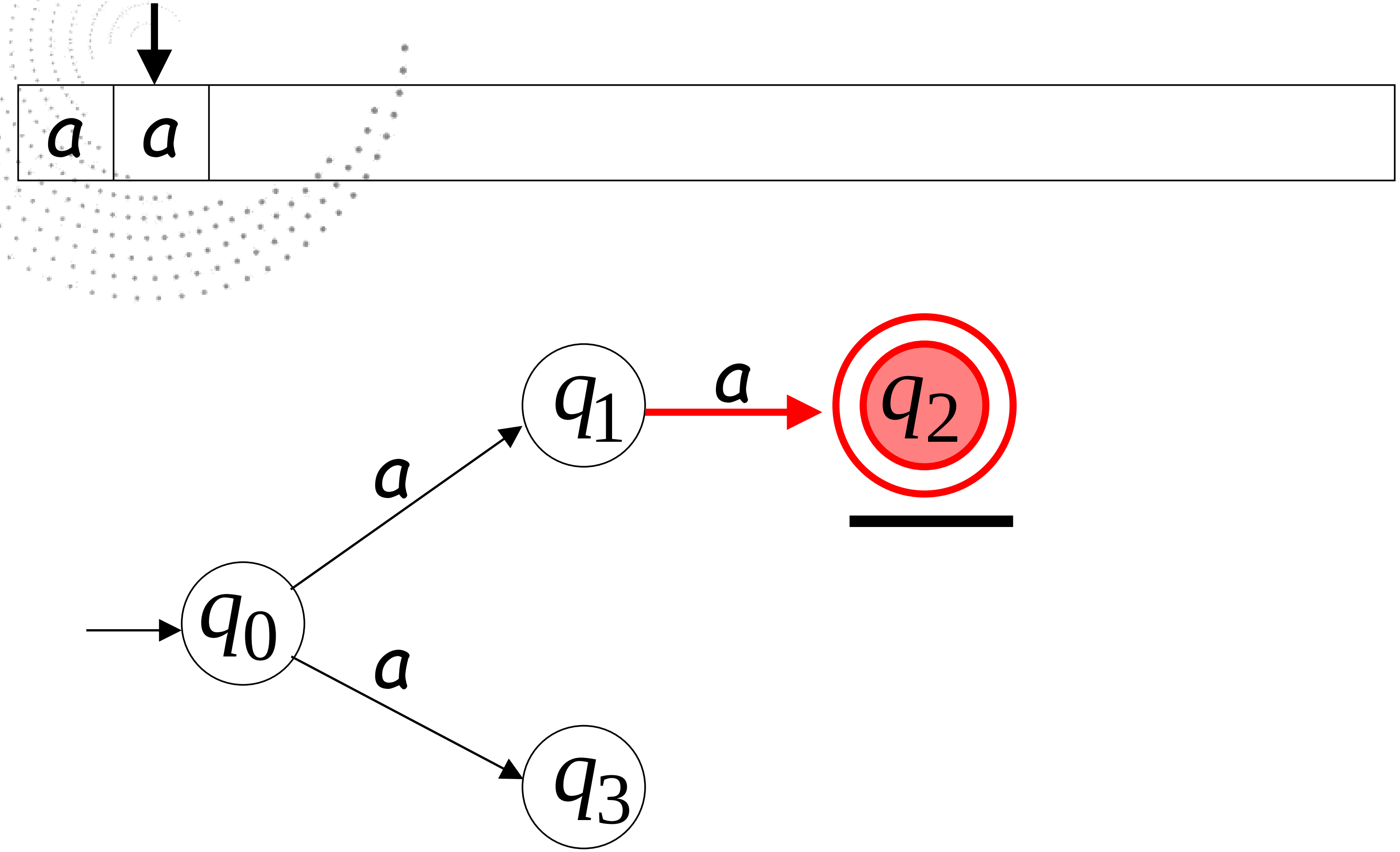
First Choice



First Choice

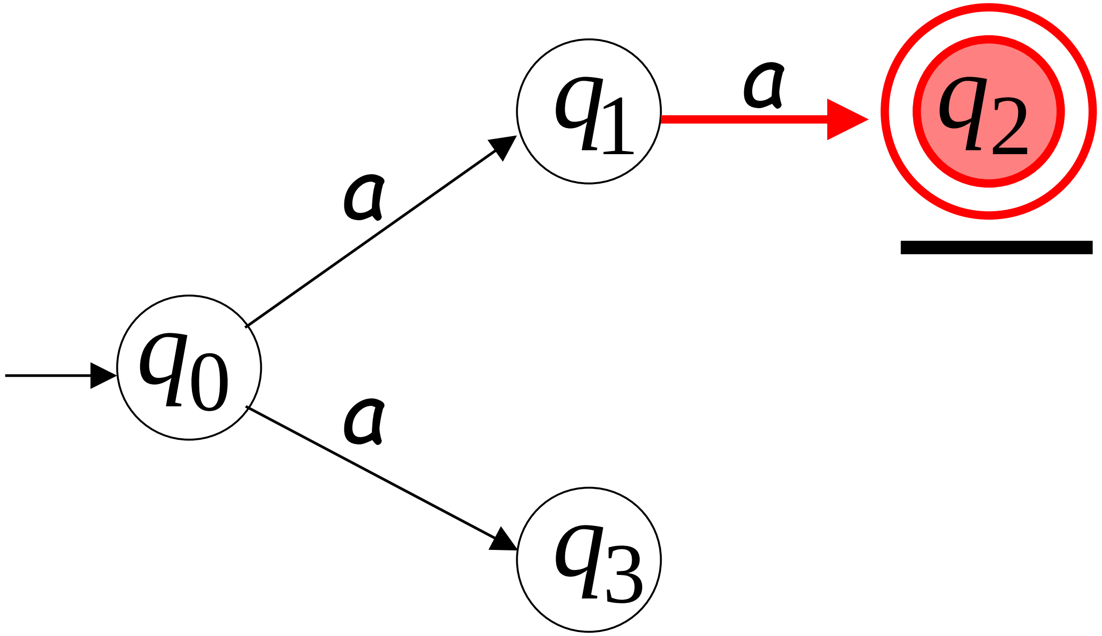


First Choice



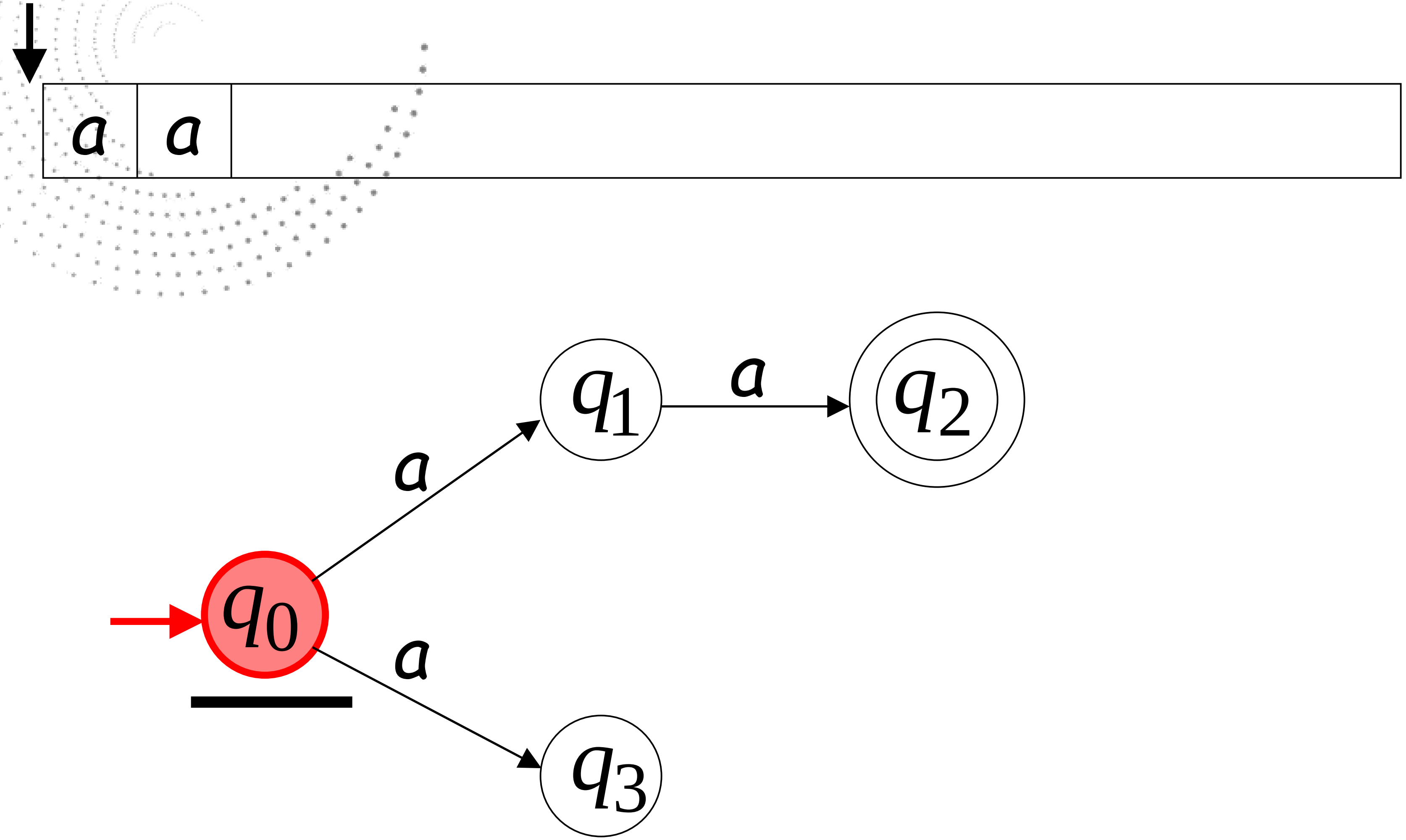
First Choice

All input is consumed

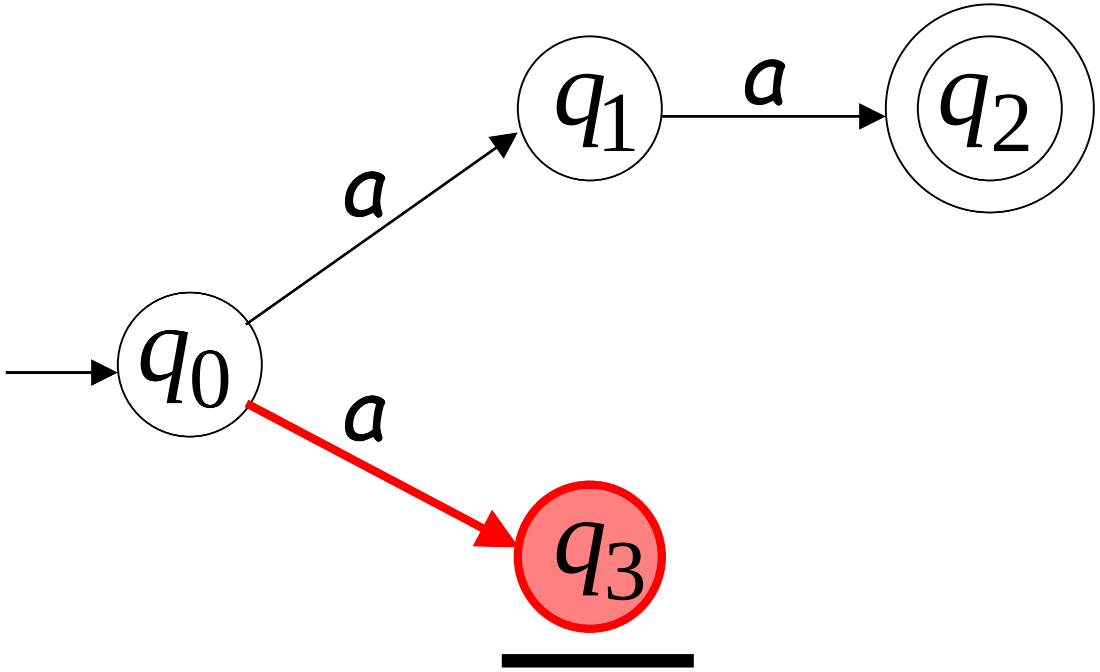


"accept"

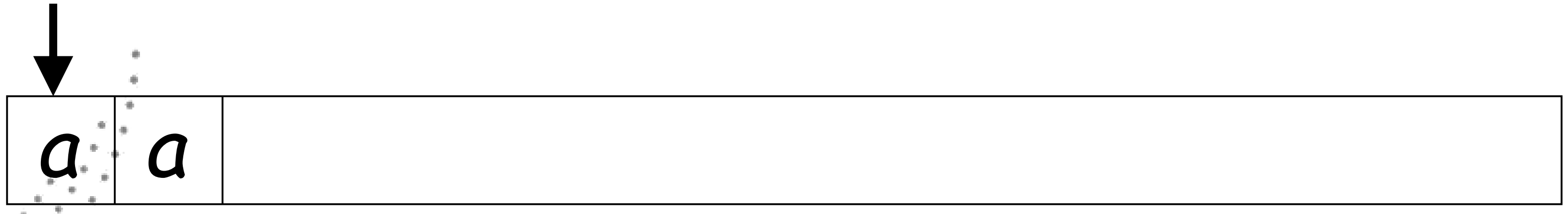
Second Choice

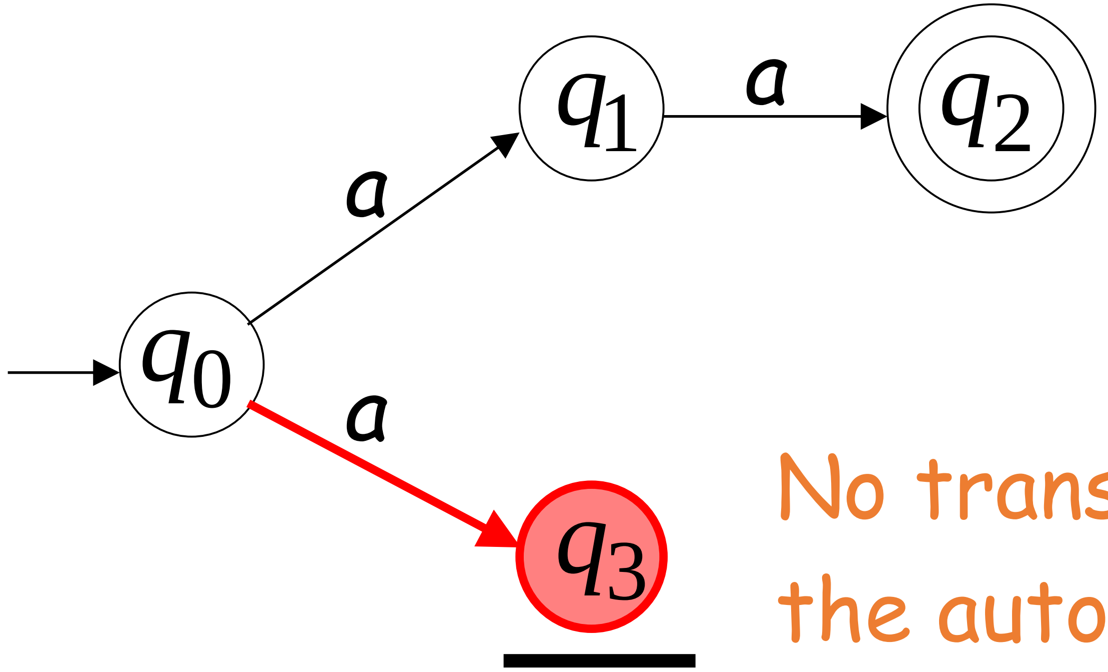


Second Choice



Second Choice

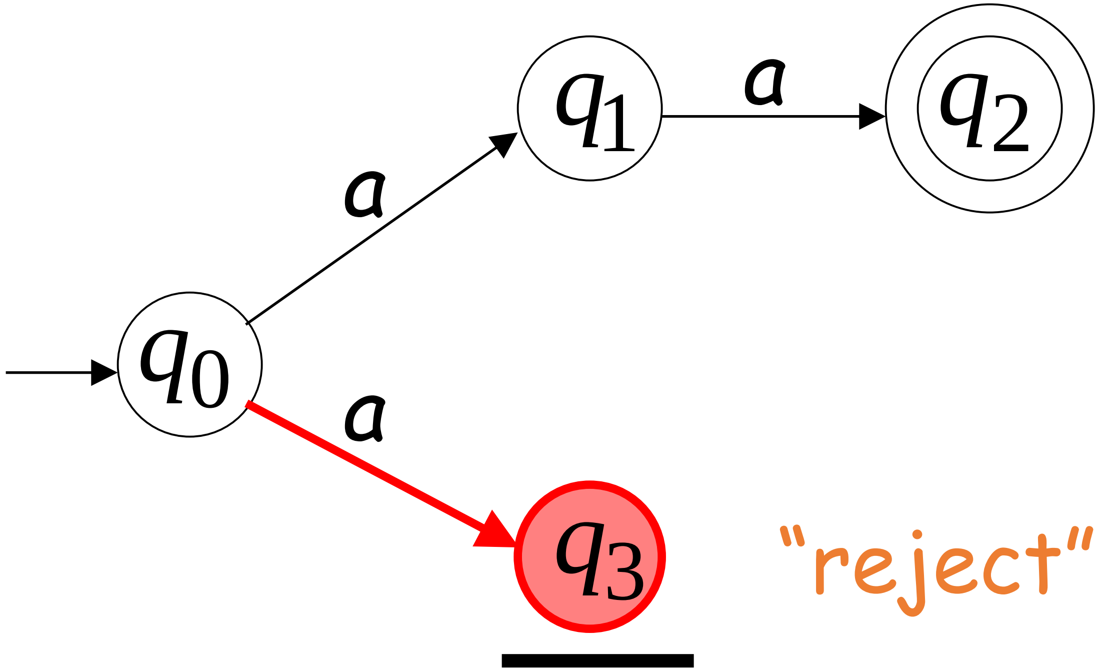




No transition:
the automaton hangs

Second Choice

Input cannot be consumed



An NFA accepts a string:

when there is a computation of the NFA that accepts the string

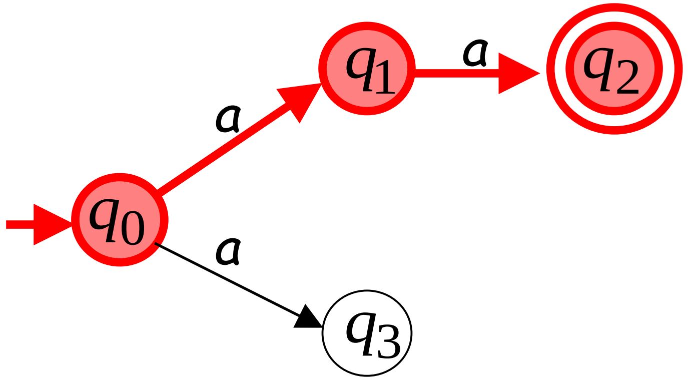
AND

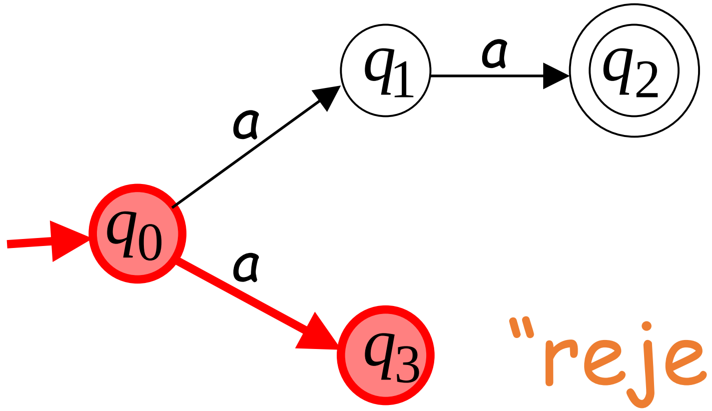
all the input is consumed and the automaton is in a final state

Example

aa is accepted by the NFA:

"accept"

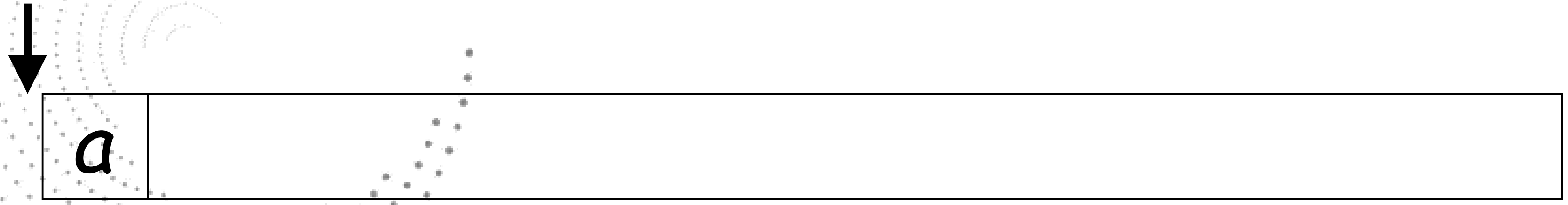


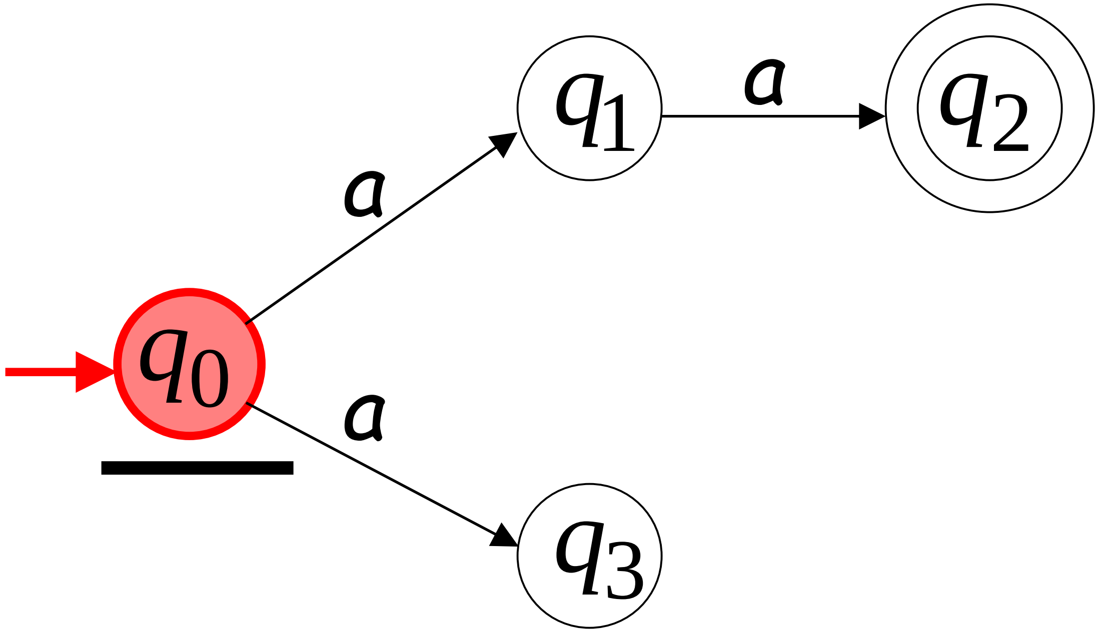


"reject"

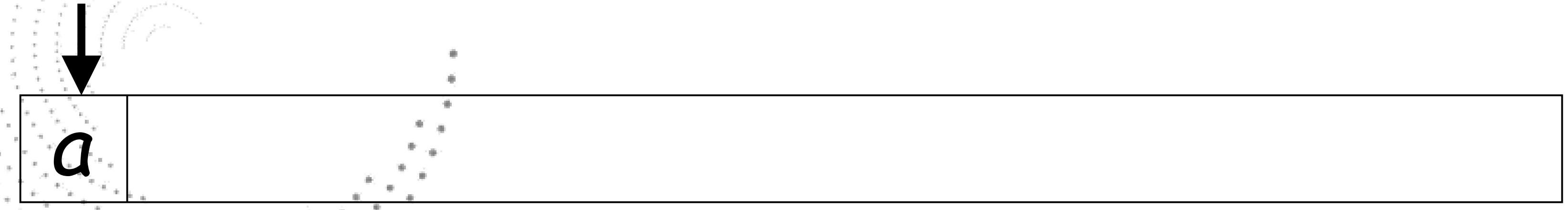
because this computation accepts aa

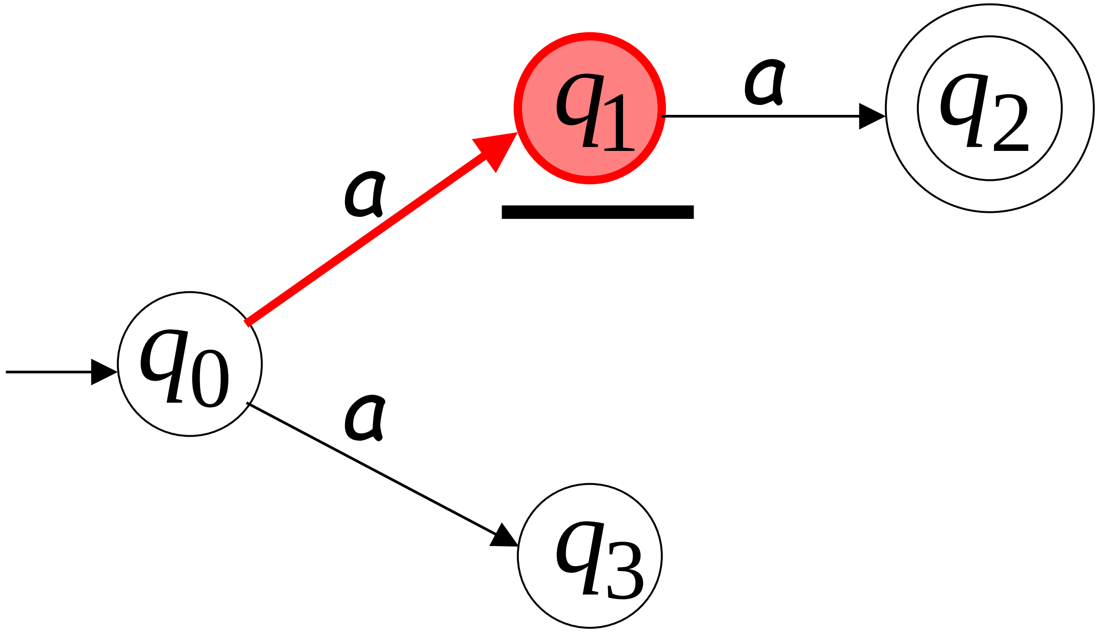
Rejection example



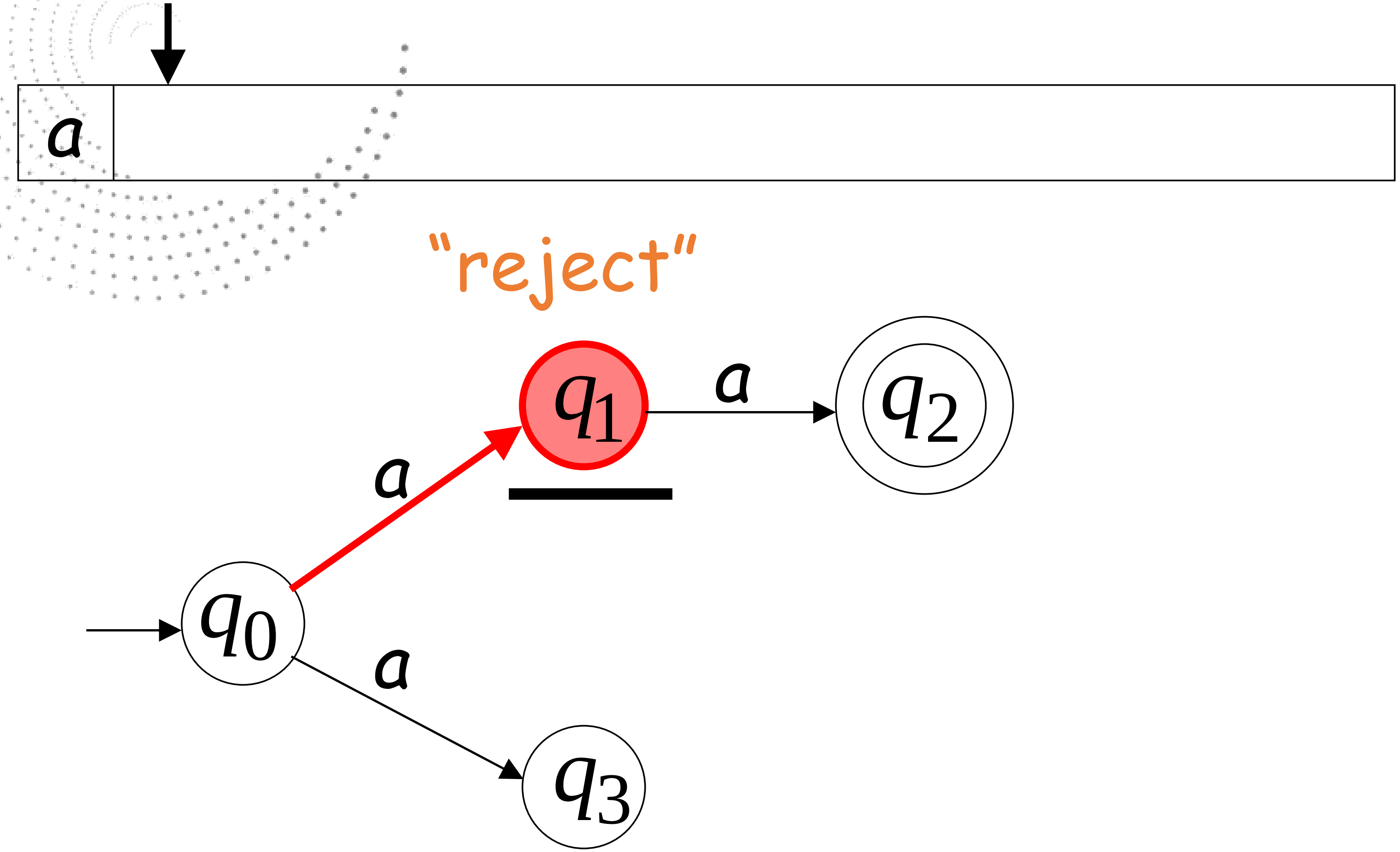


First Choice

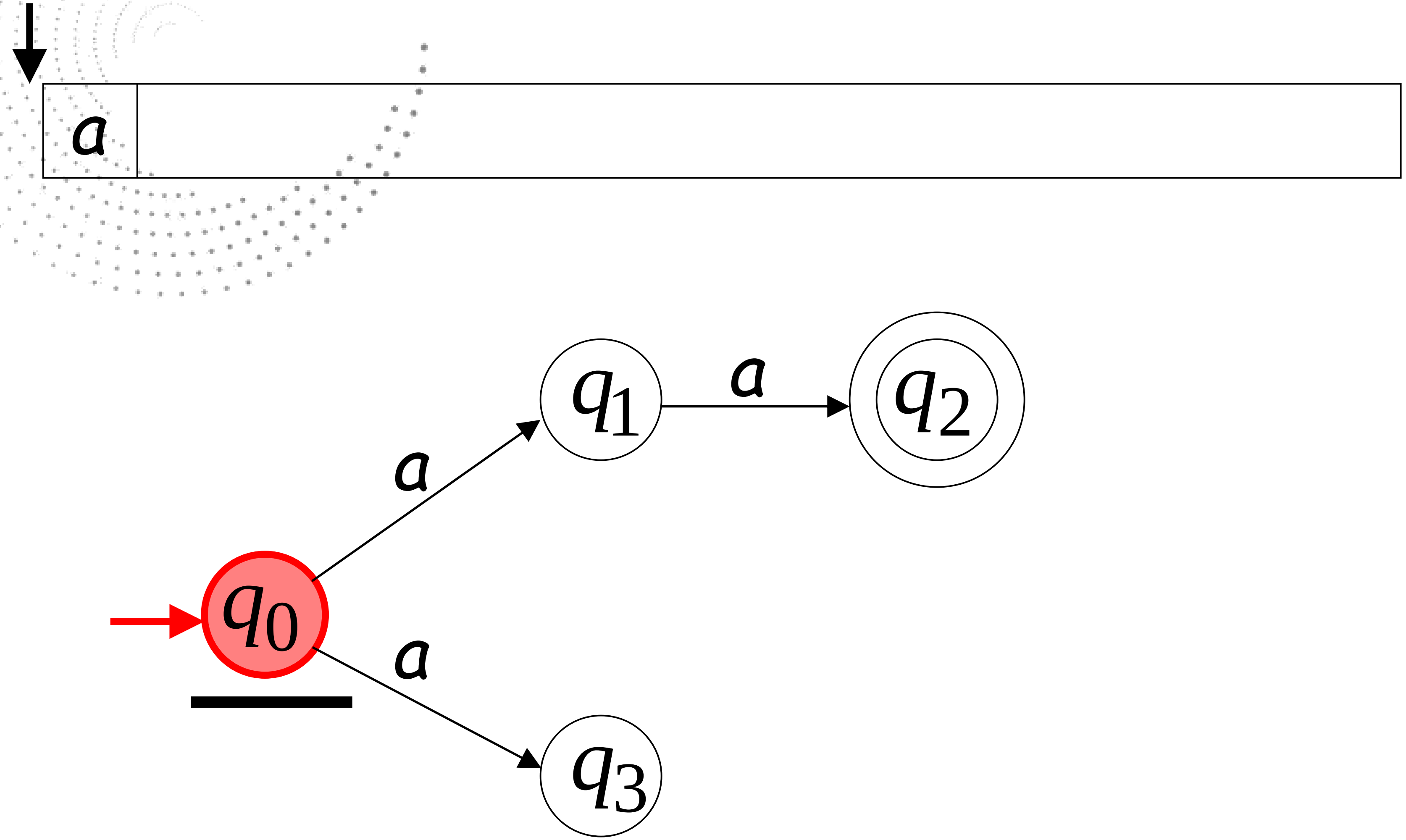




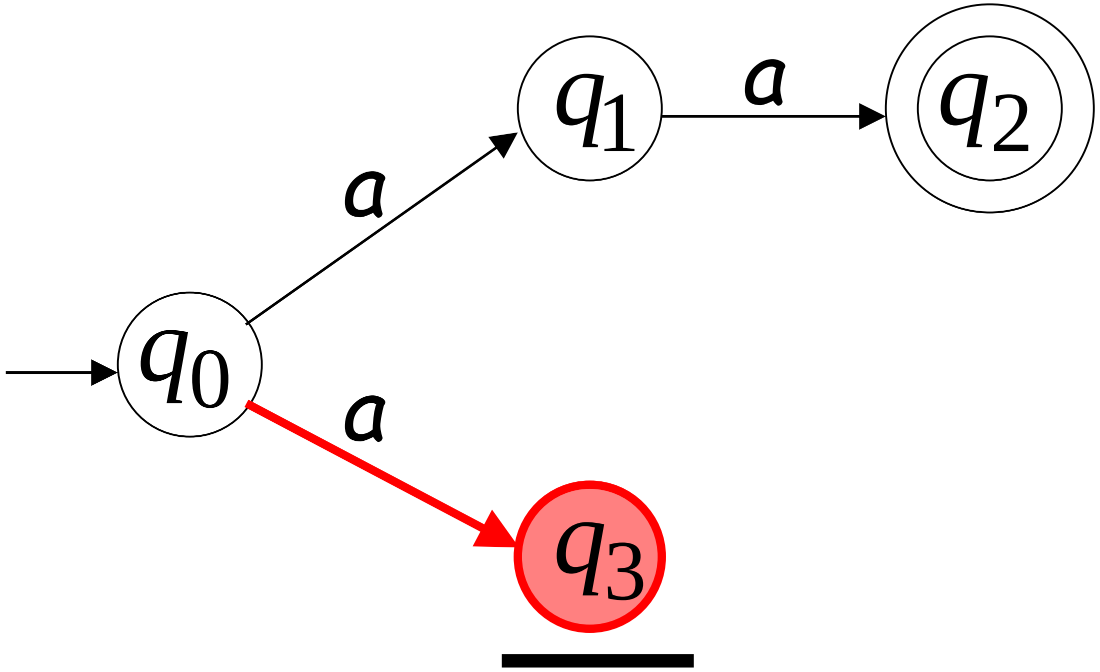
First Choice



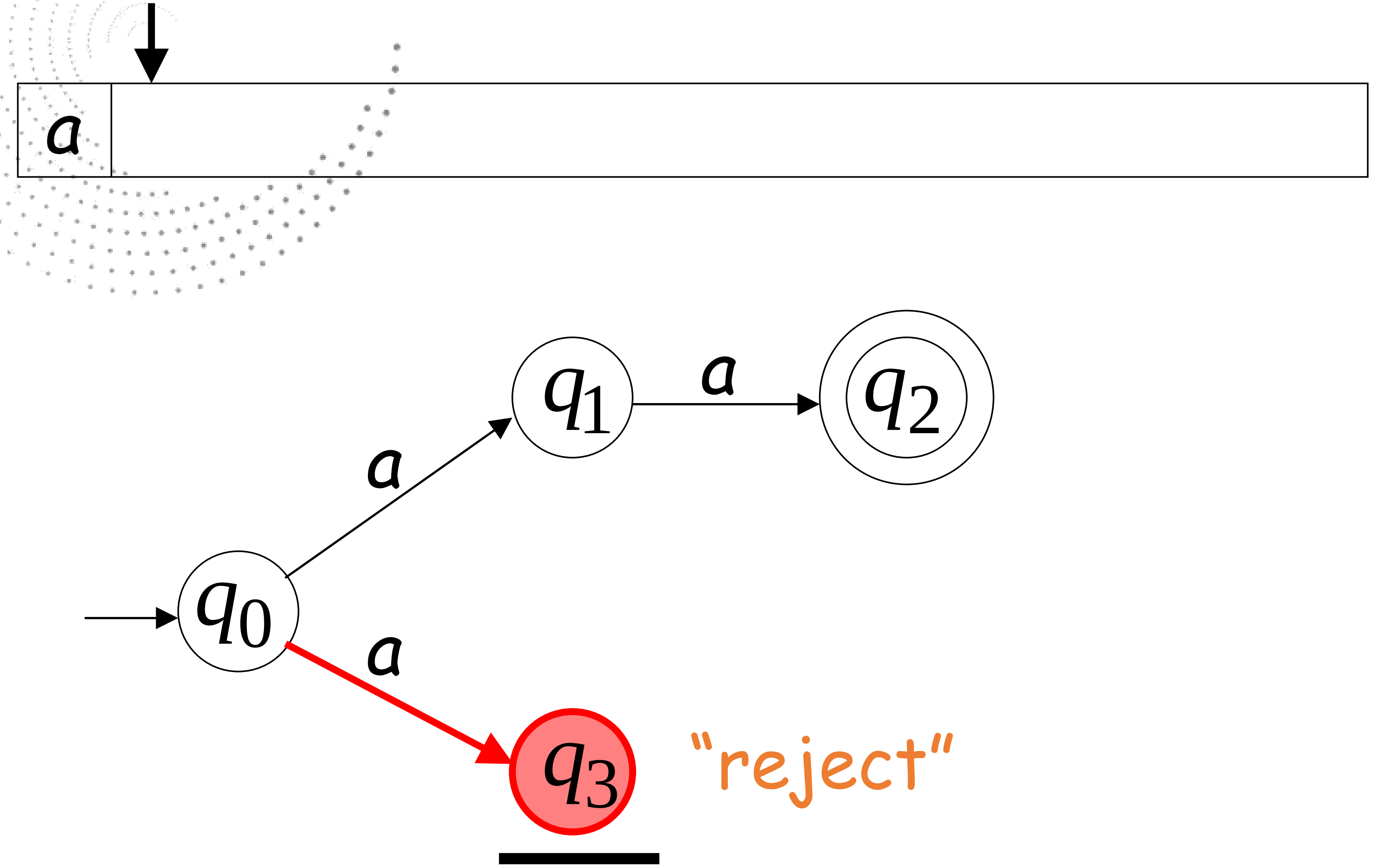
Second Choice



Second Choice



Second Choice



An NFA rejects a string:

when there is no computation of the NFA that accepts the string:

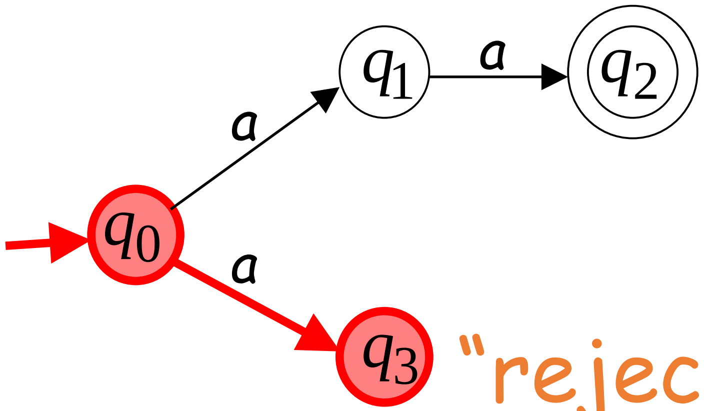
- All the input is consumed and the automaton is in a non final state

OR

- The input cannot be consumed

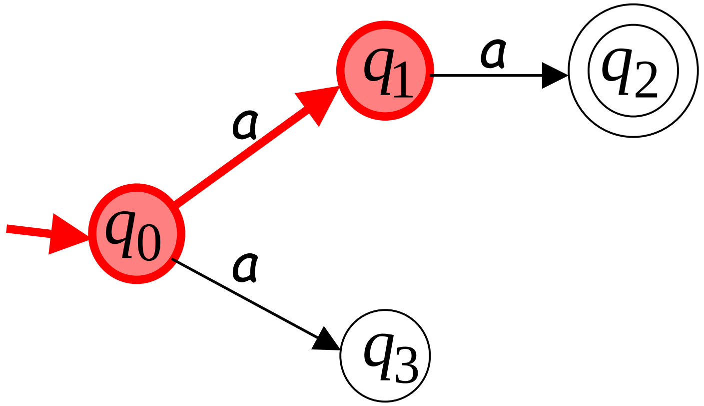
Example

a is rejected by the NFA:



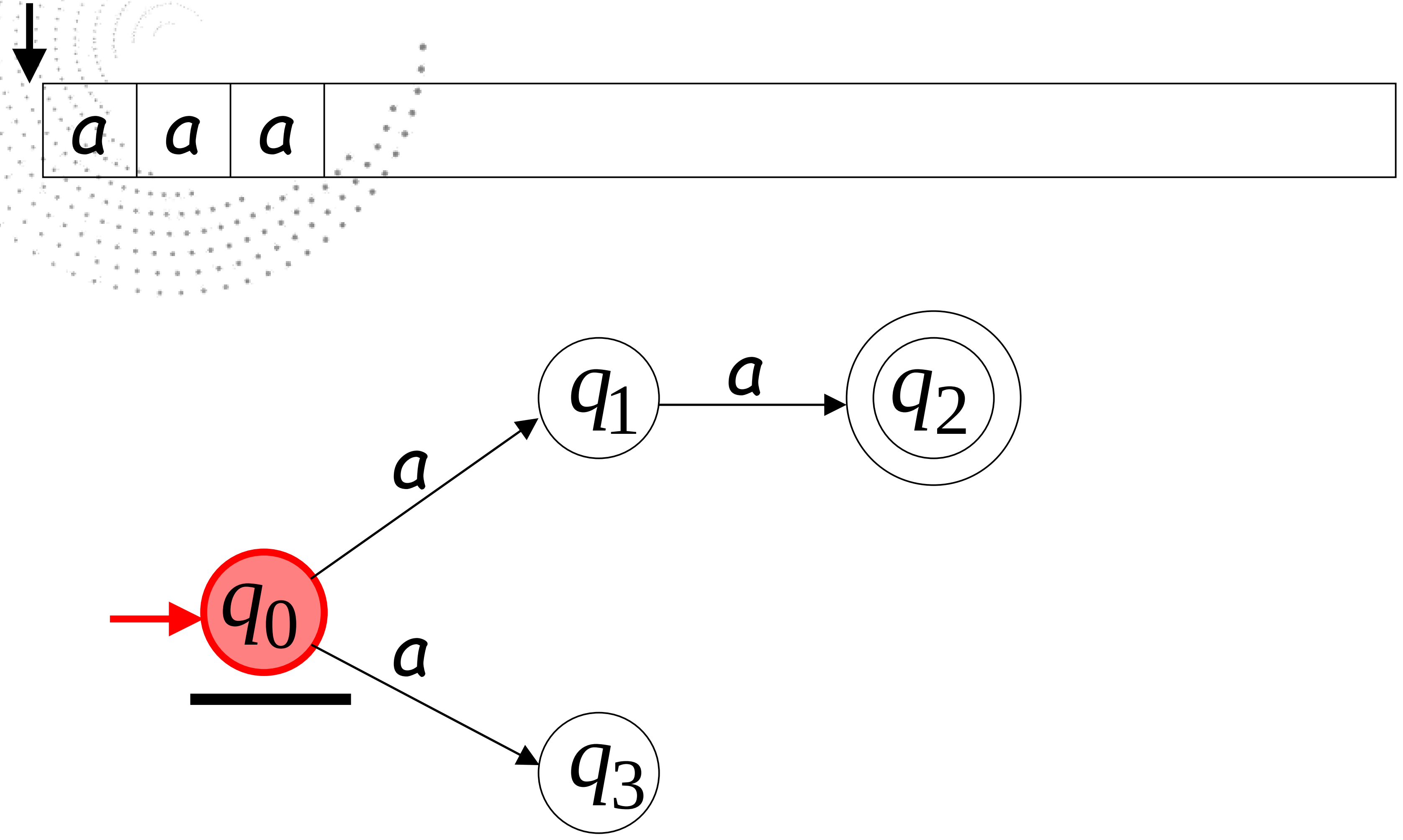
"reject"

"reject"

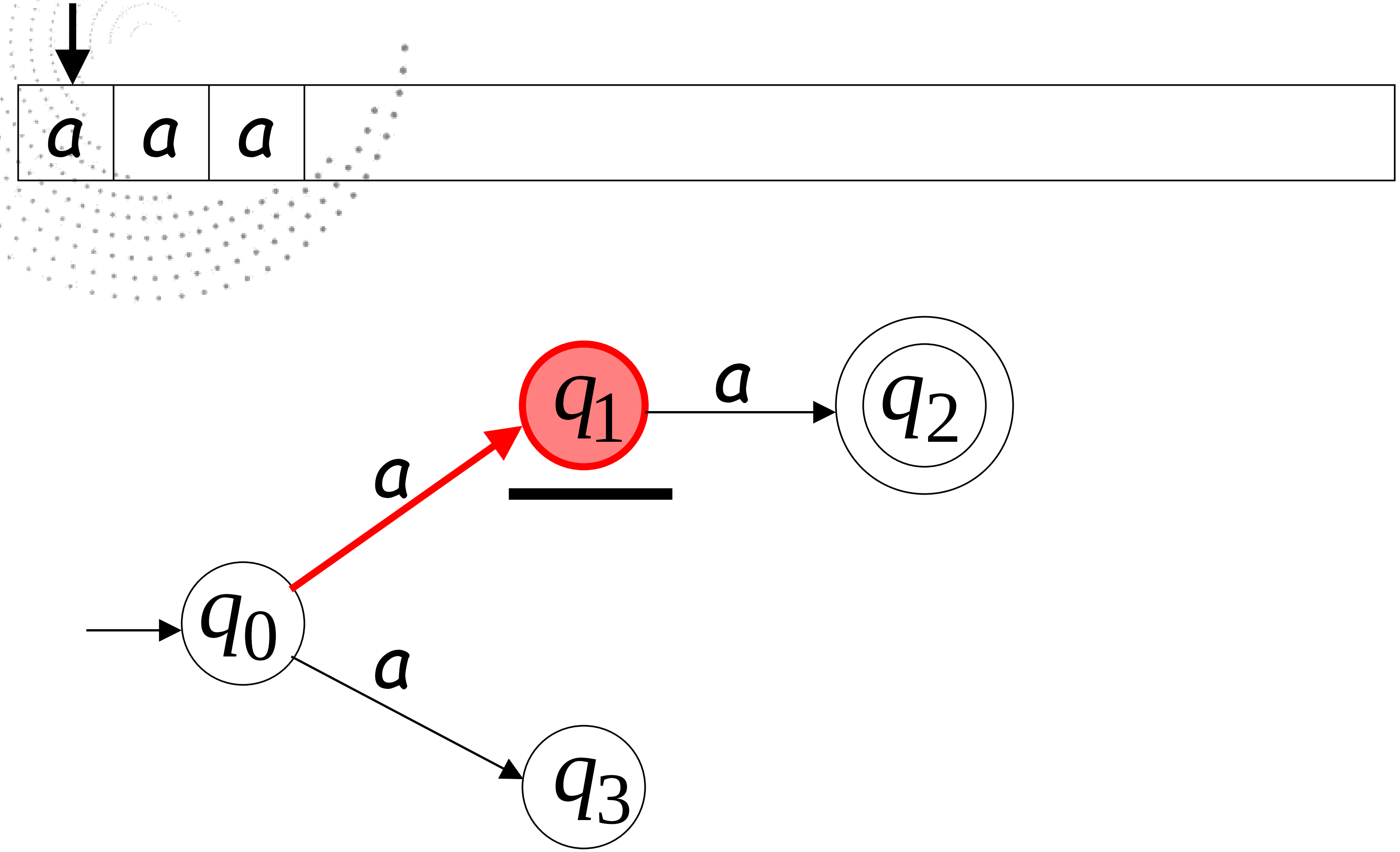


All possible computations lead to rejection

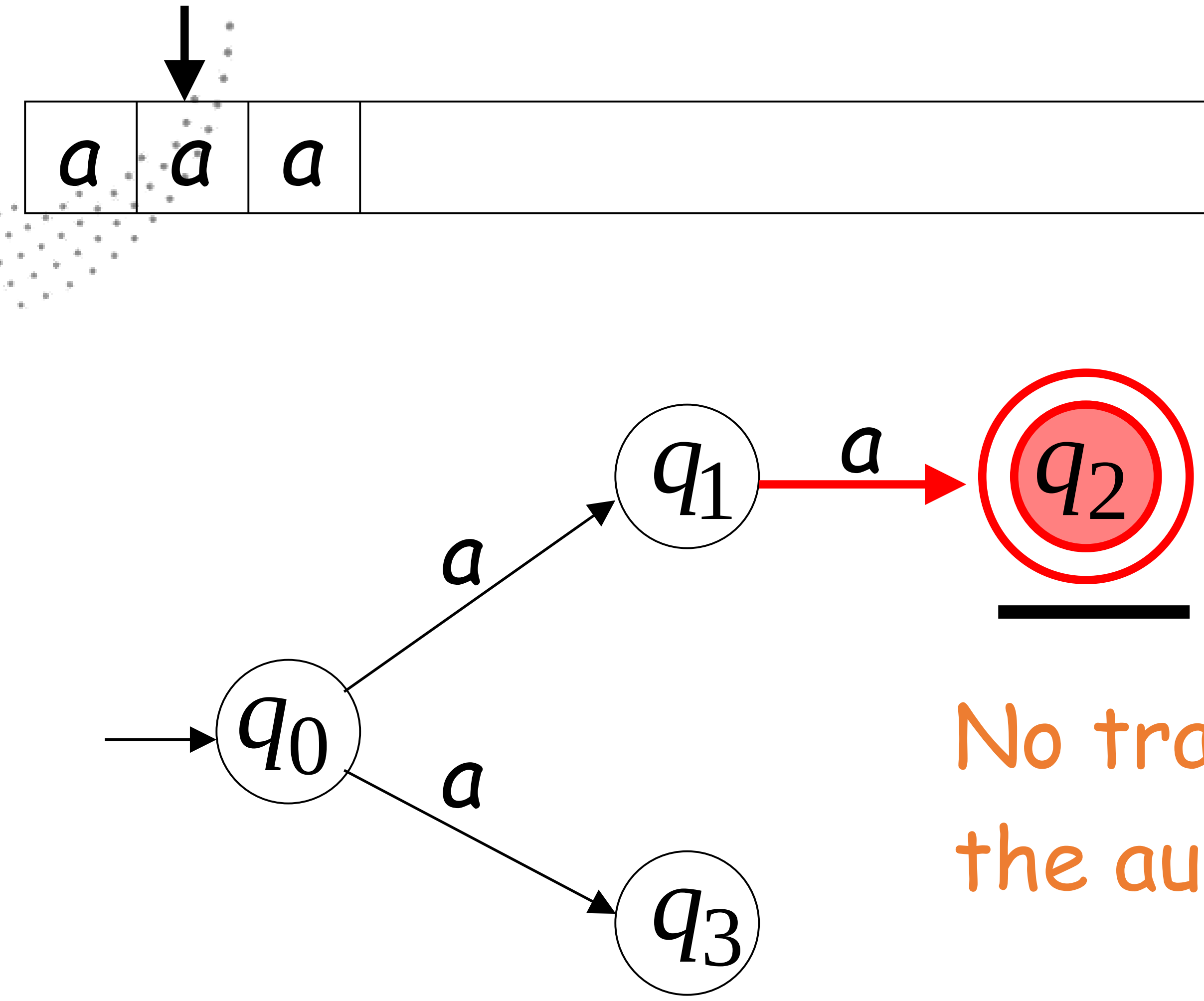
Rejection example



First Choice



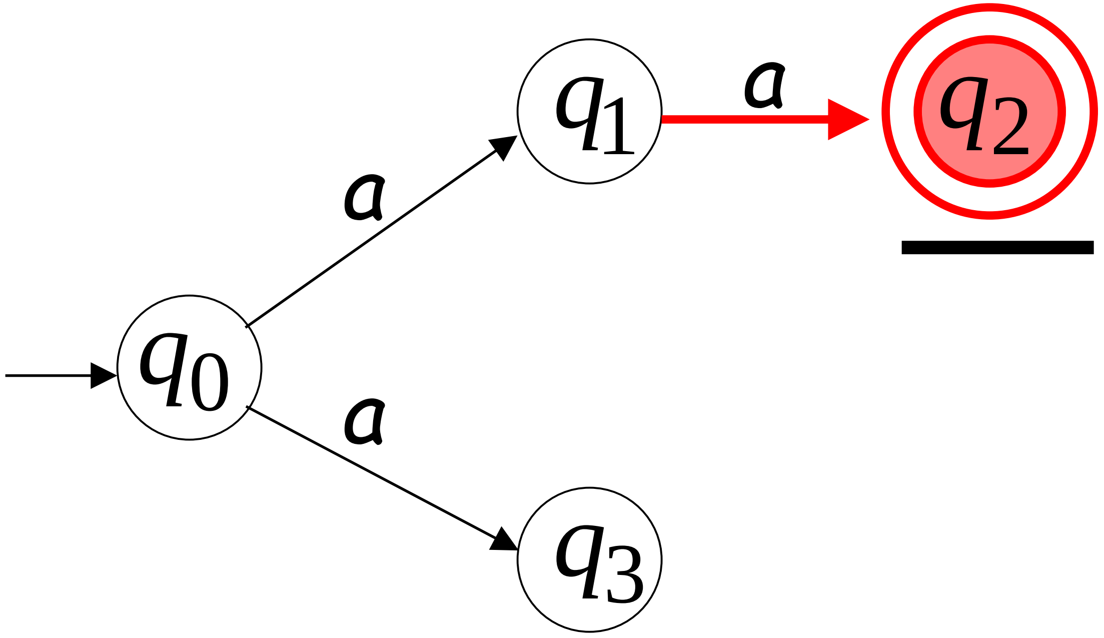
First Choice



No transition:
the automaton hangs

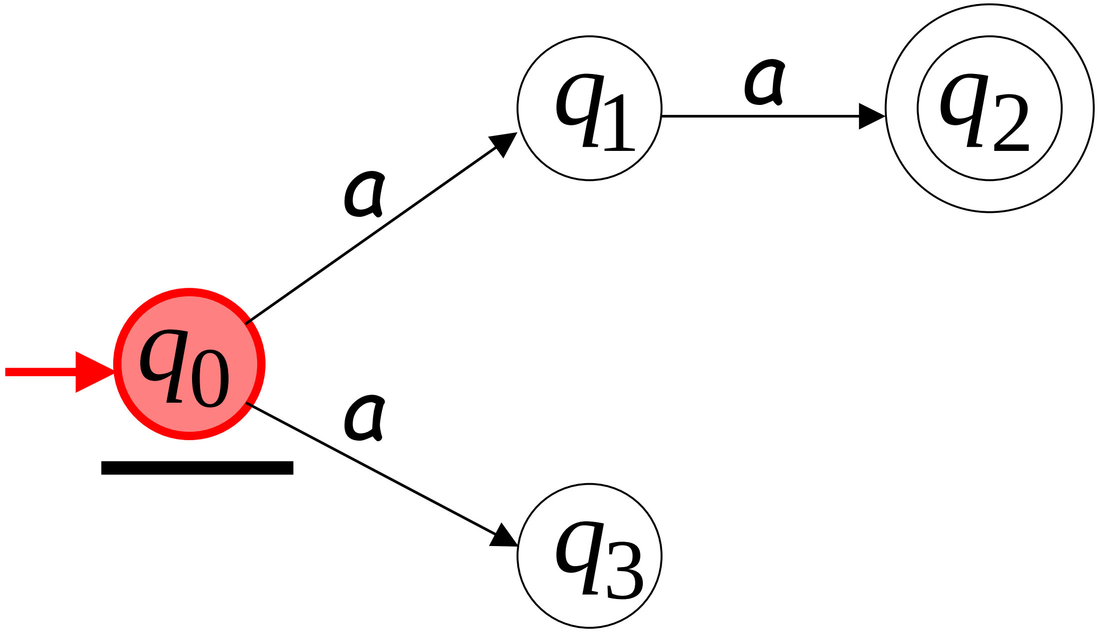
First Choice

Input cannot be consumed

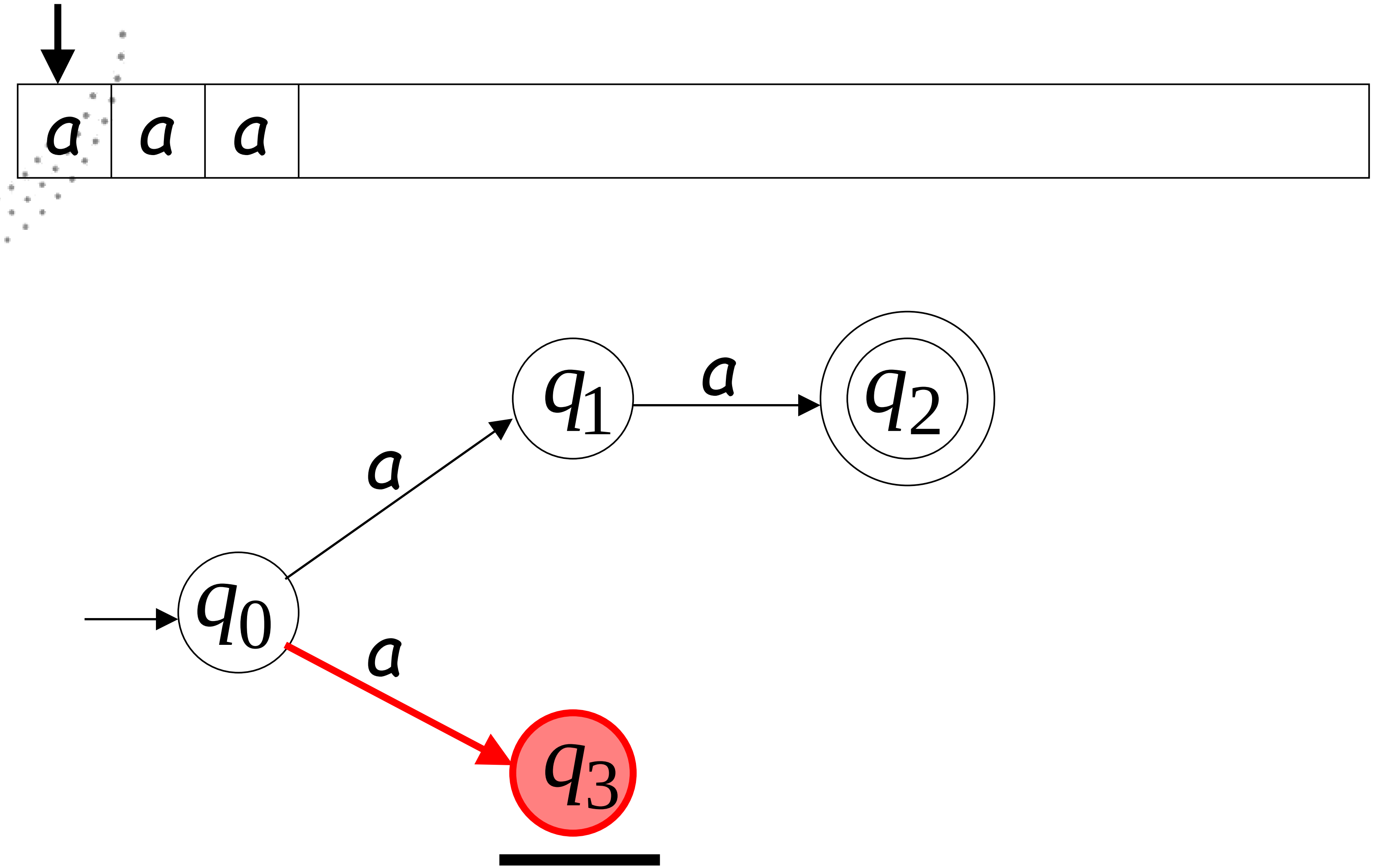


"reject"

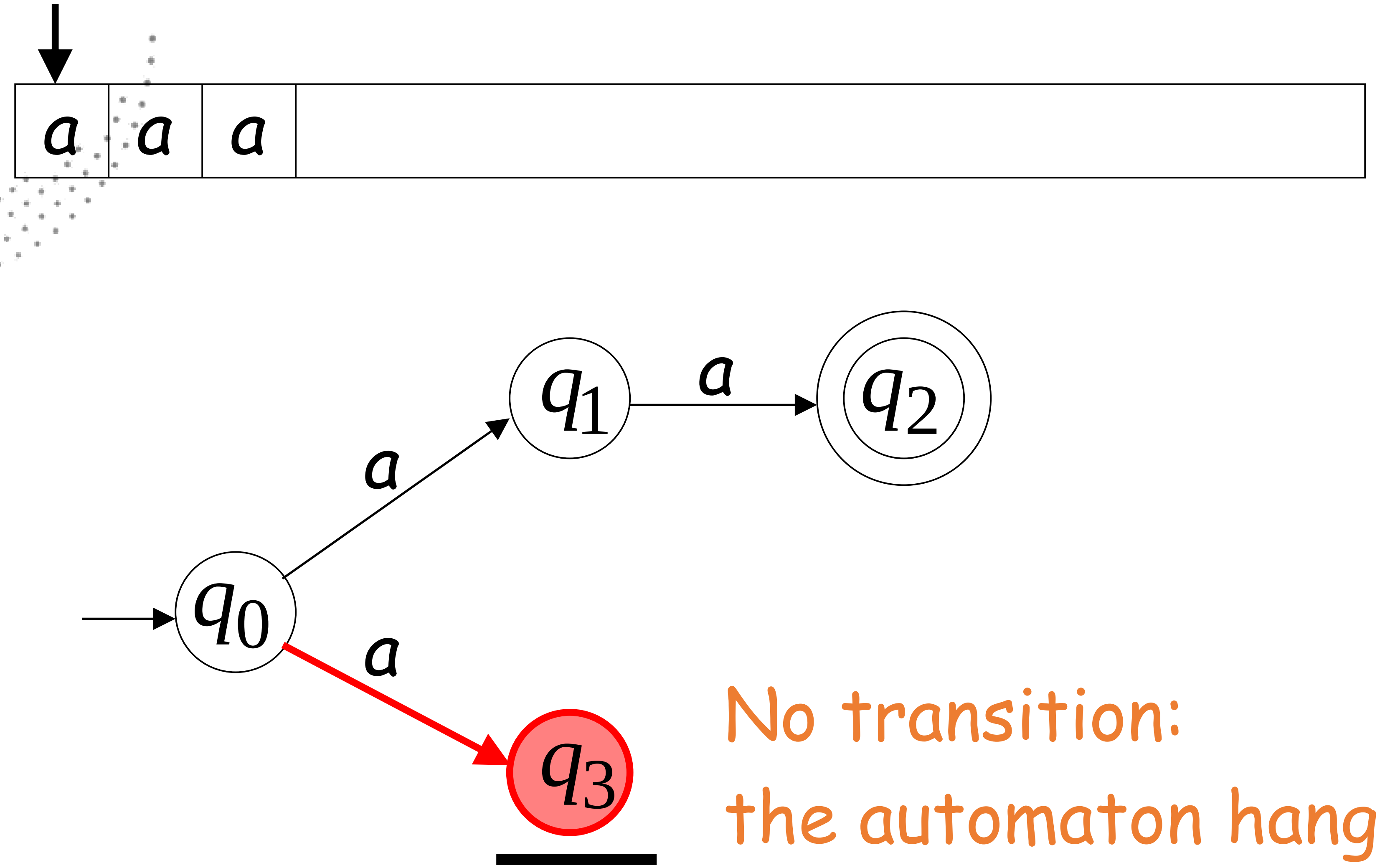
Second Choice



Second Choice



Second Choice

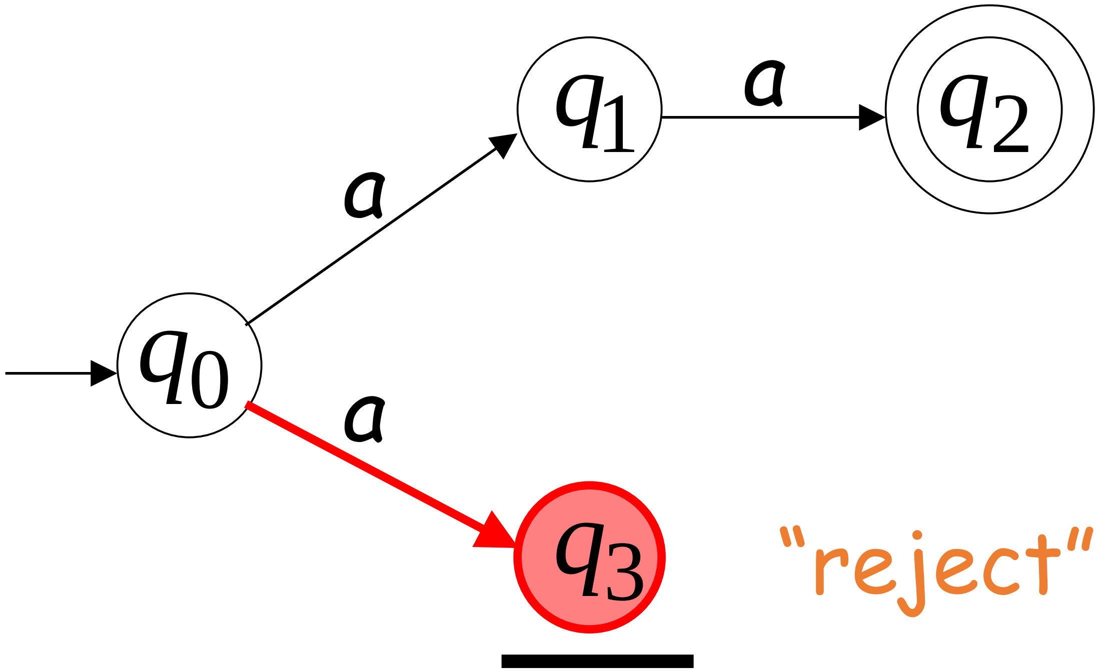


No transition:
the automaton hangs

Second Choice

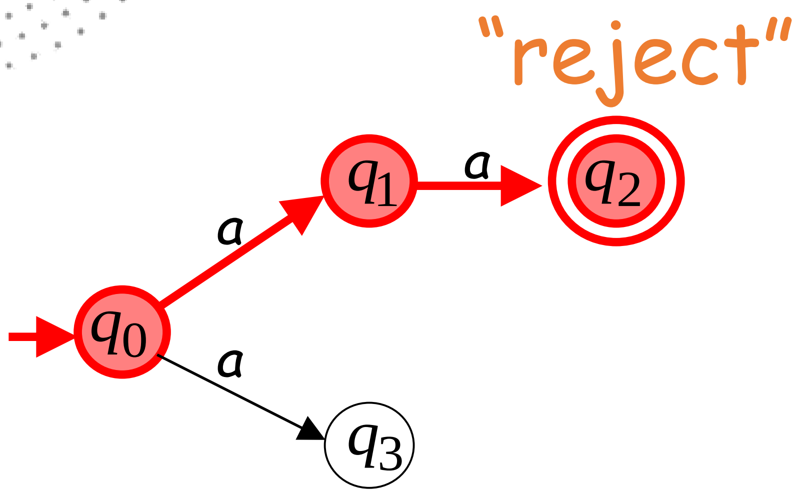
a	a	a	

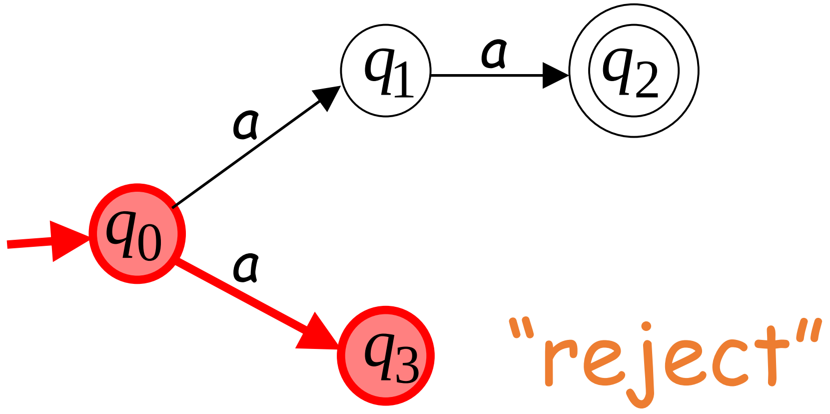
Input cannot be consumed



"reject"

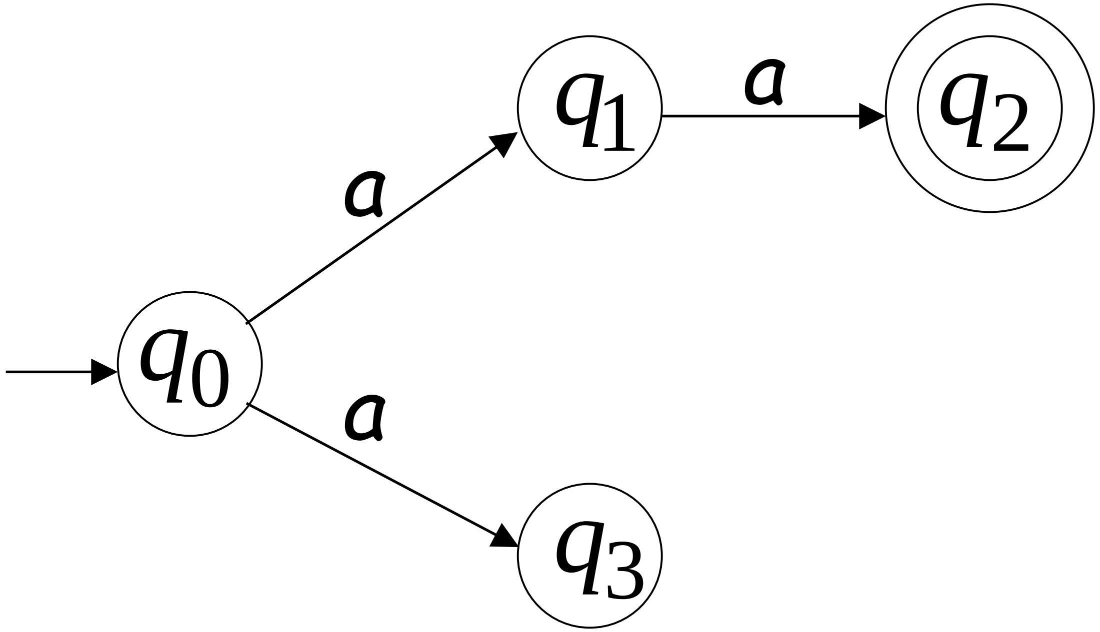
aaa is rejected by the NFA:



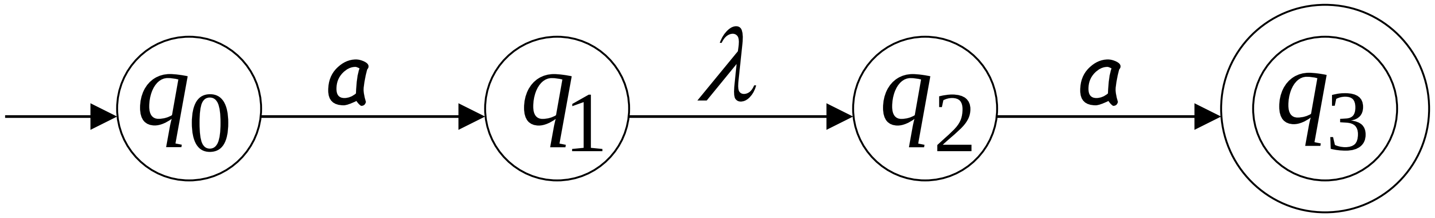


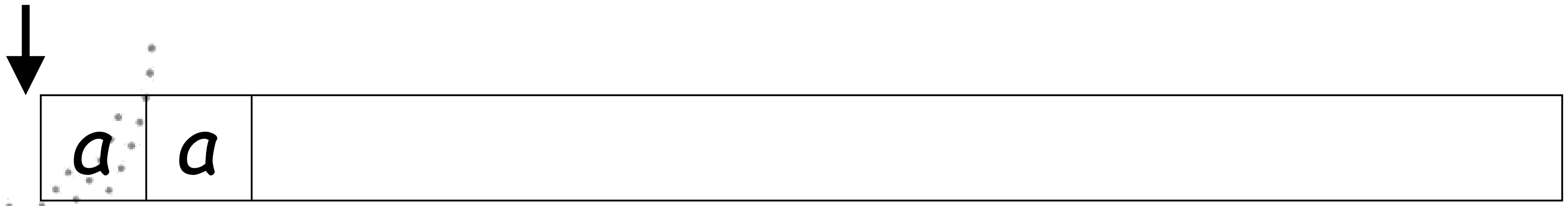
All possible computations lead to rejection

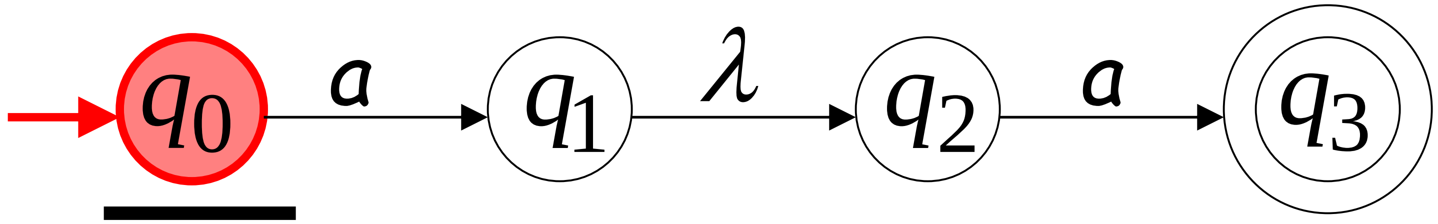
Language accepted: $L = \{aa\}$

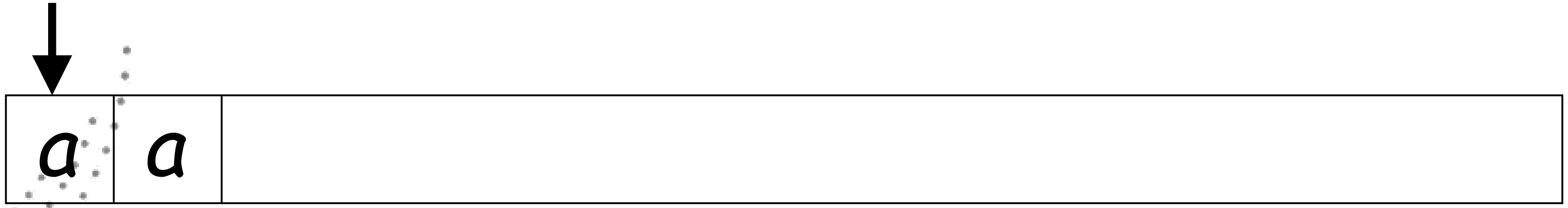


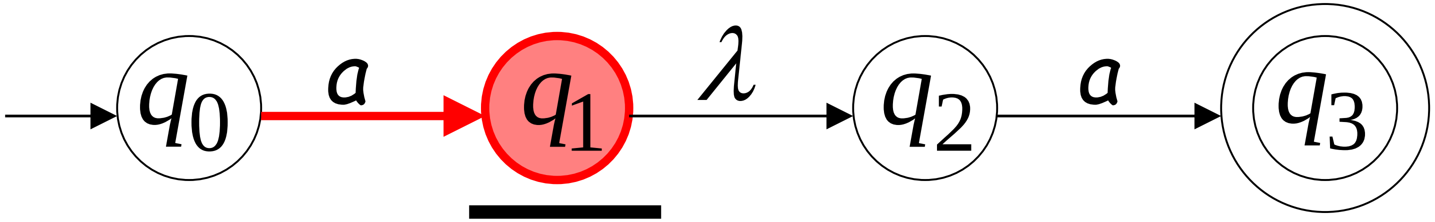
Lambda(λ) / Epsilon(ε) Transitions



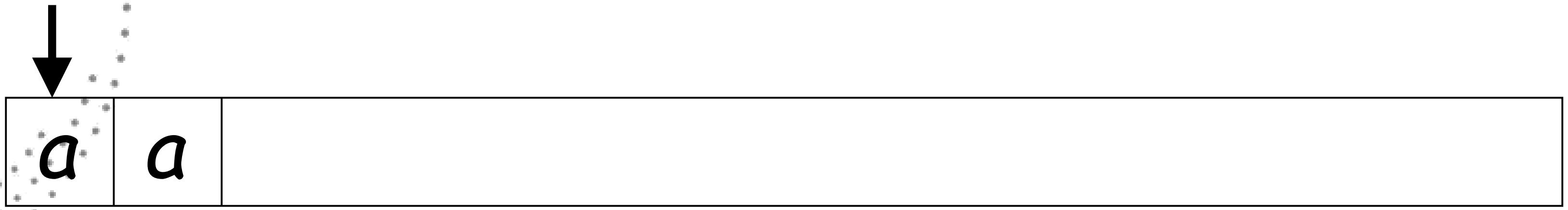

a
a

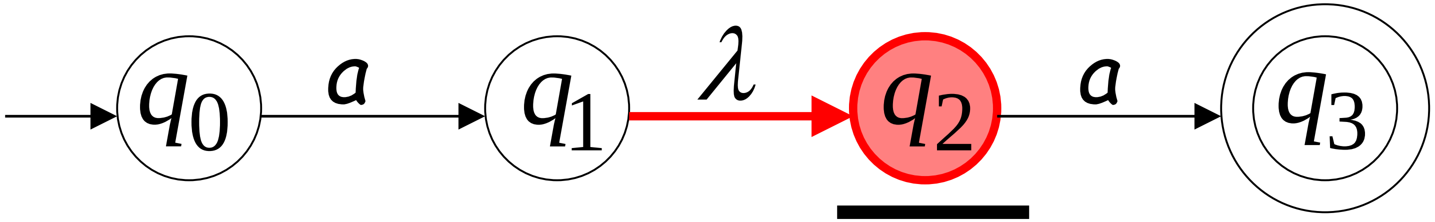

q_0
a
q_1
λ
q_2
a
q_3


a
a

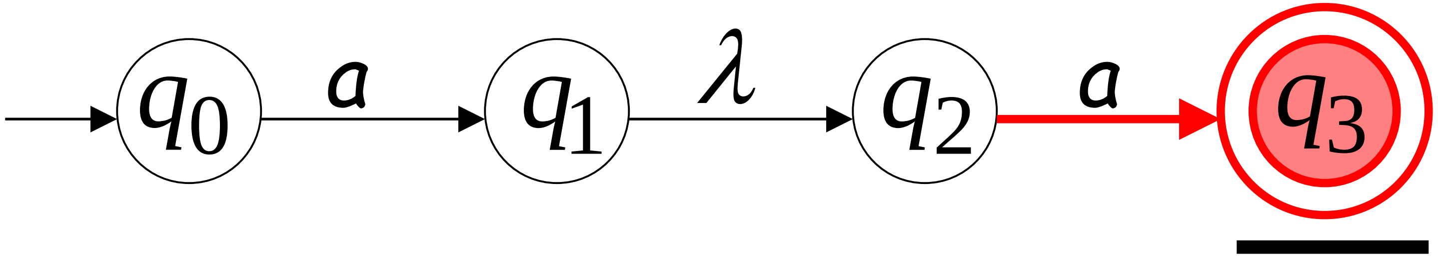

q_0
a
q_1
λ
q_2
a
q_3

(read head does not move)

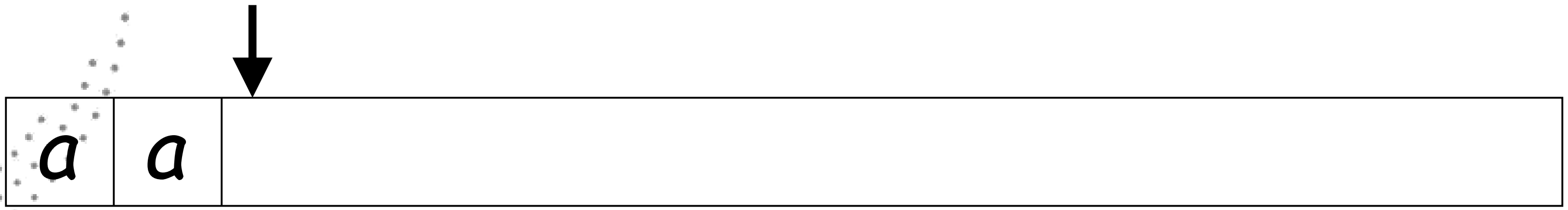




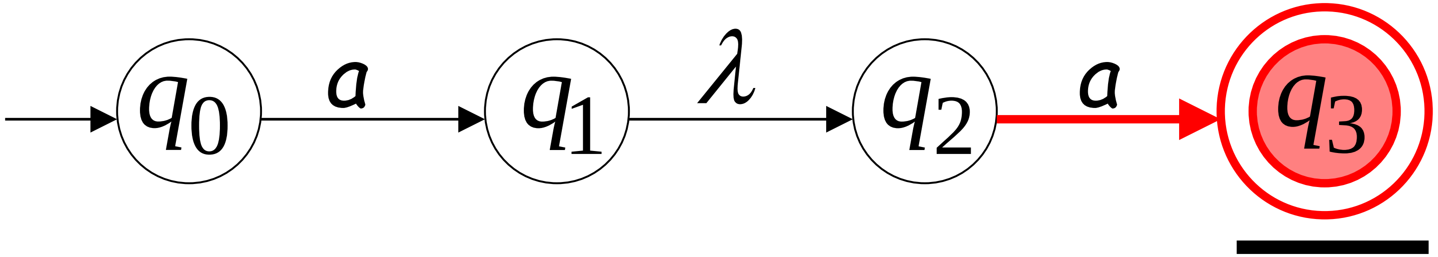

a
a


q_0
a
q_1
λ
q_2
a
q_3

all input is consumed

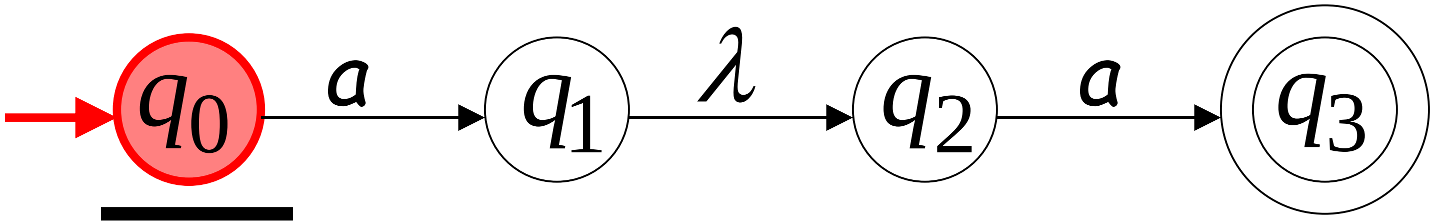


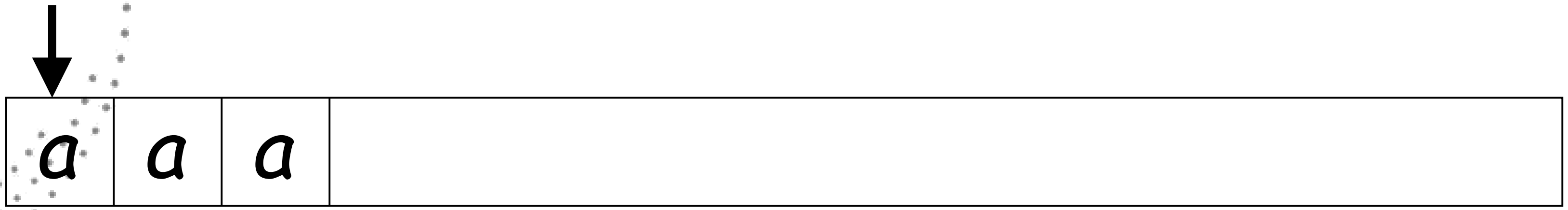
"accept"



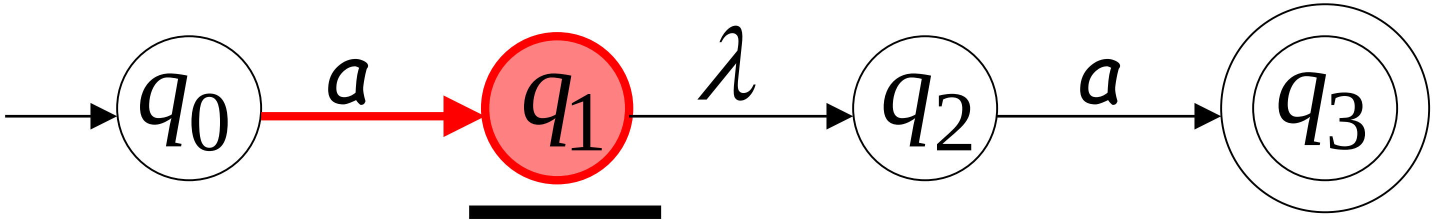
String aa is accepted

Rejection Example



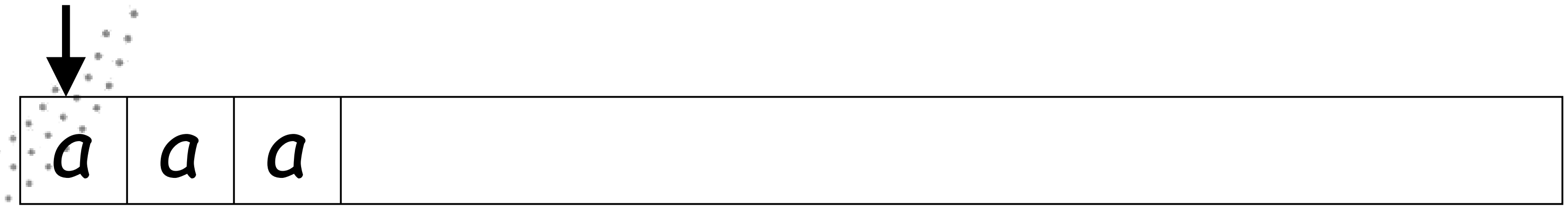


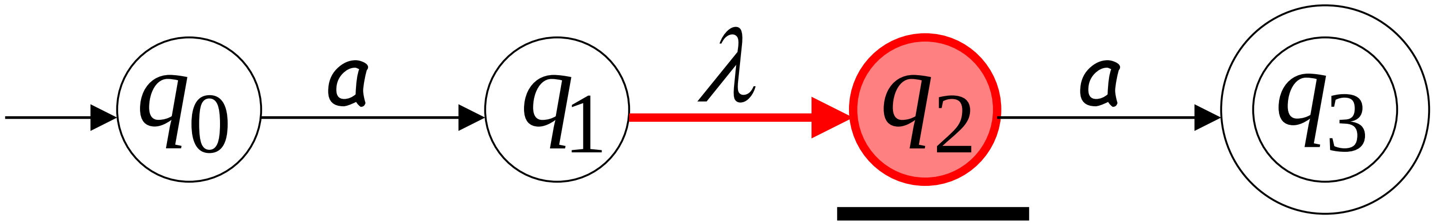
a a a

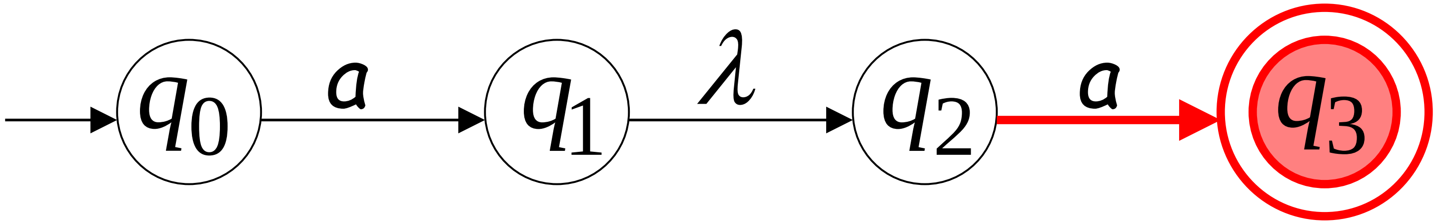


q_0
a
q_1
λ
q_2
a
q_3

(read head doesn't move)

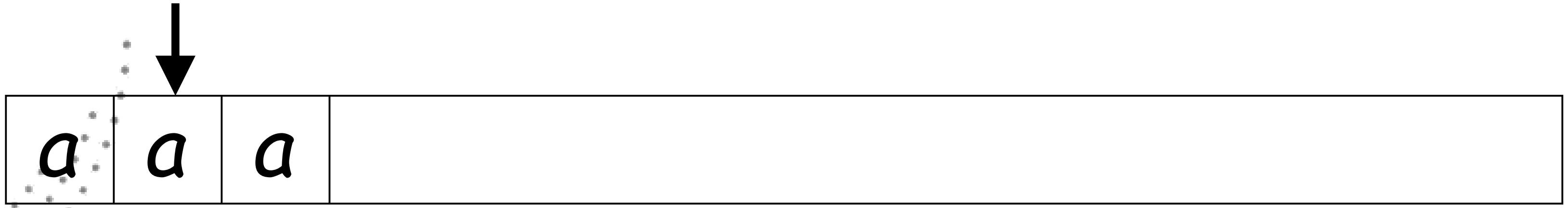


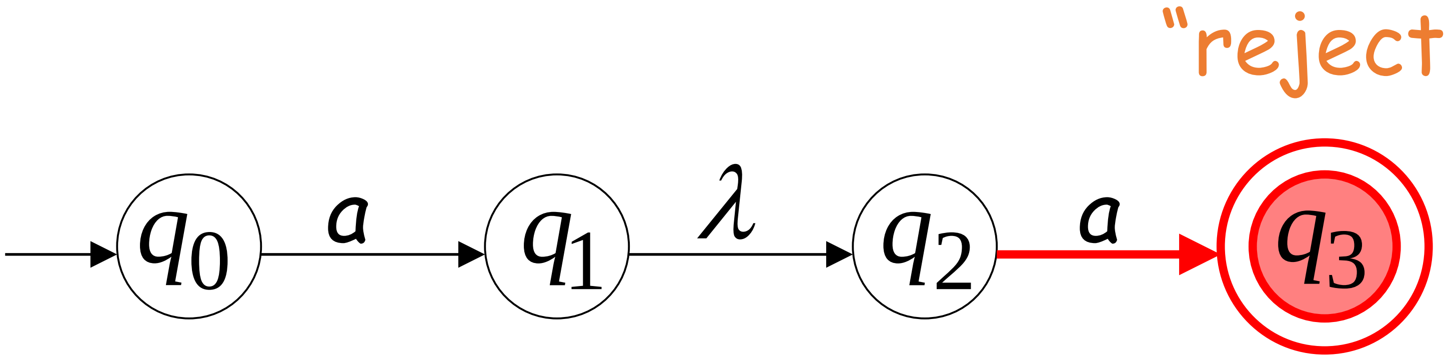




No transition:
the automaton hangs

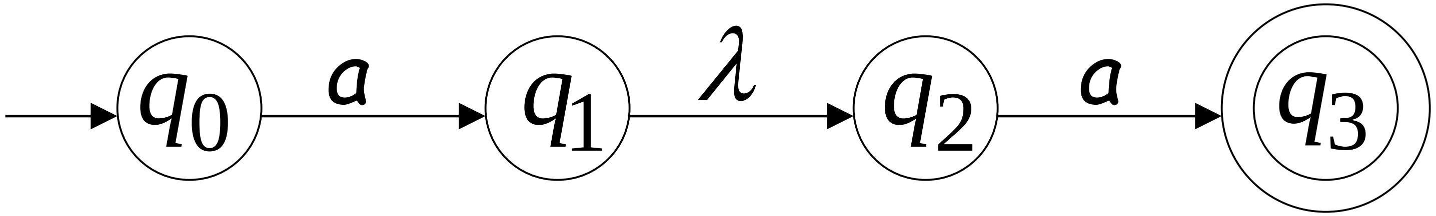
Input cannot be consumed



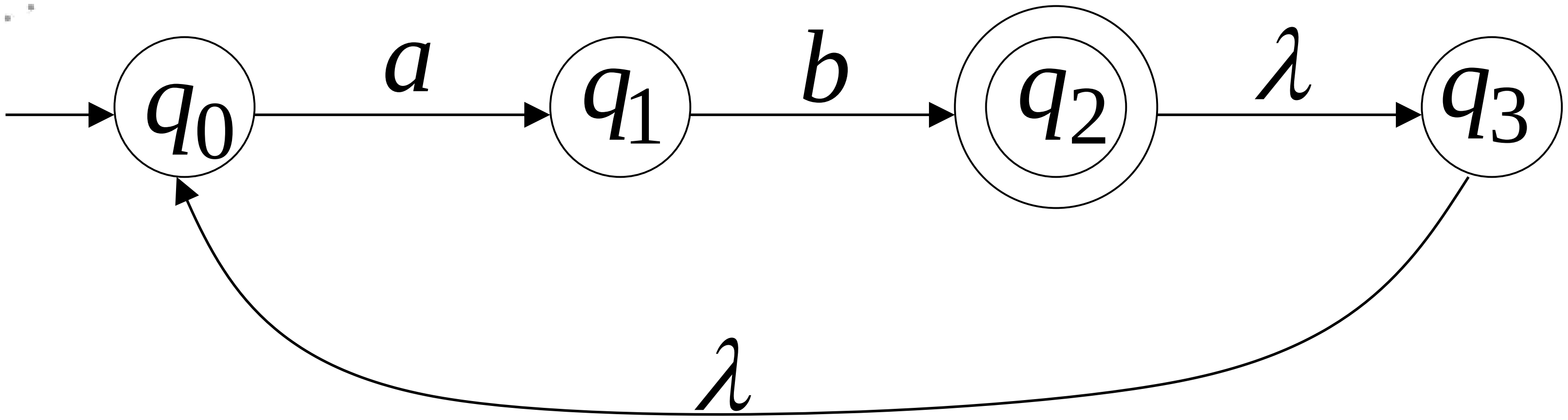


String aaa is rejected

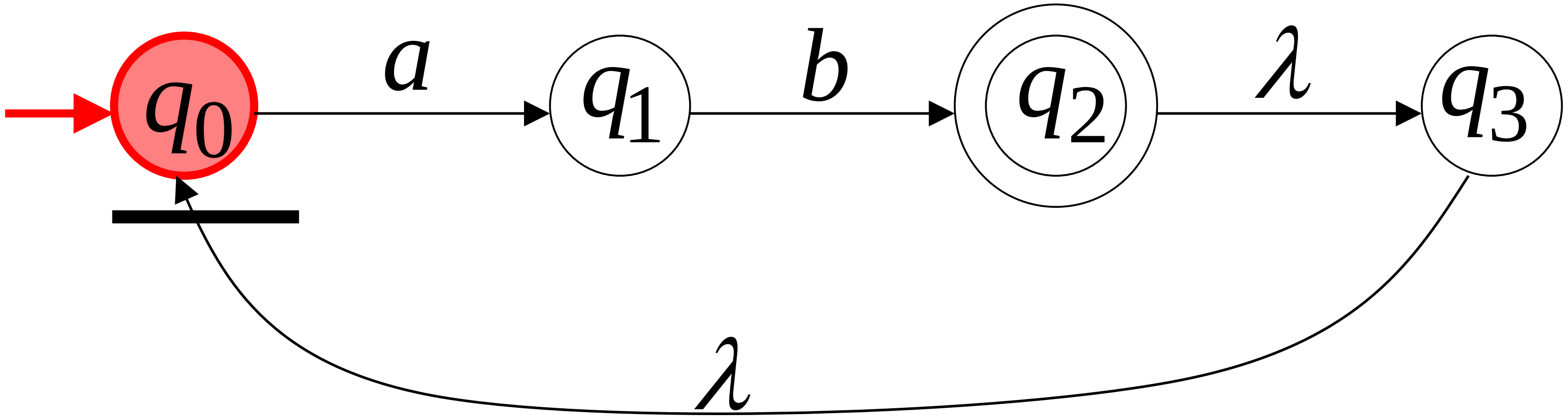
Language accepted: $L = \{aa\}$



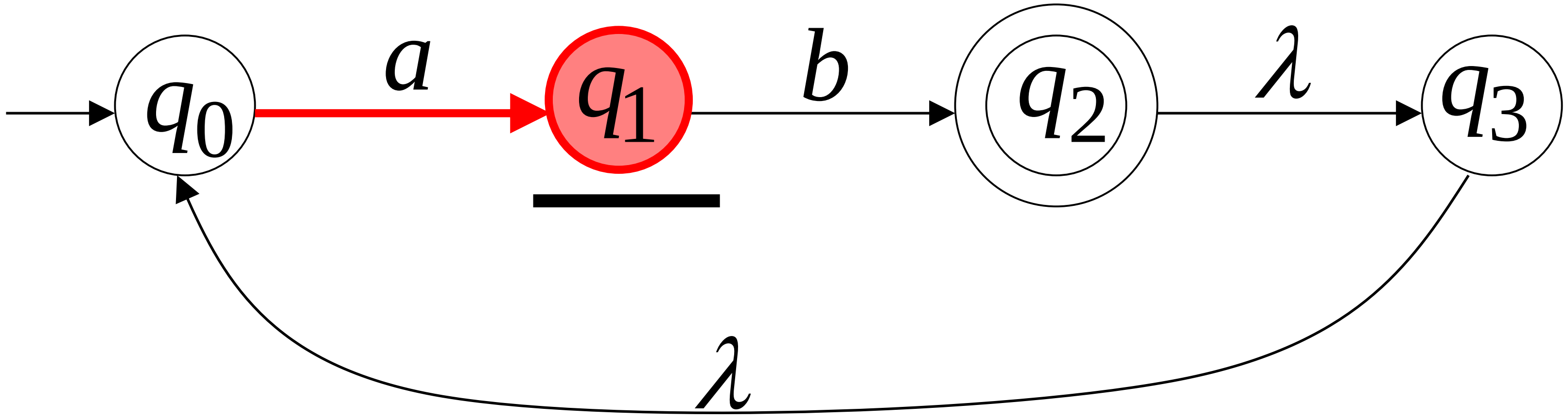
Another NFA Example

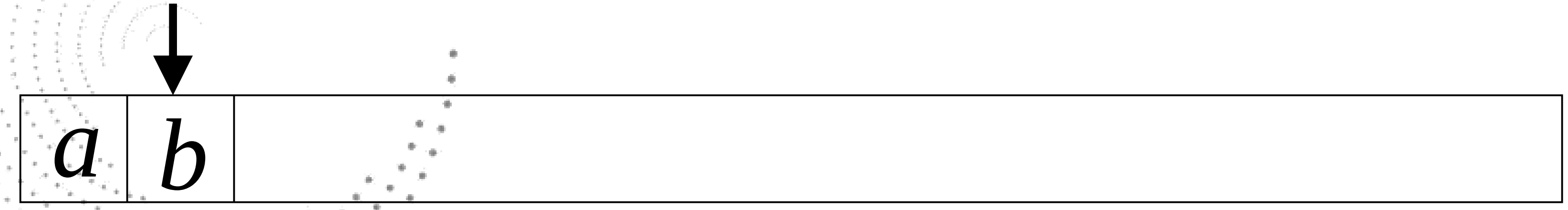


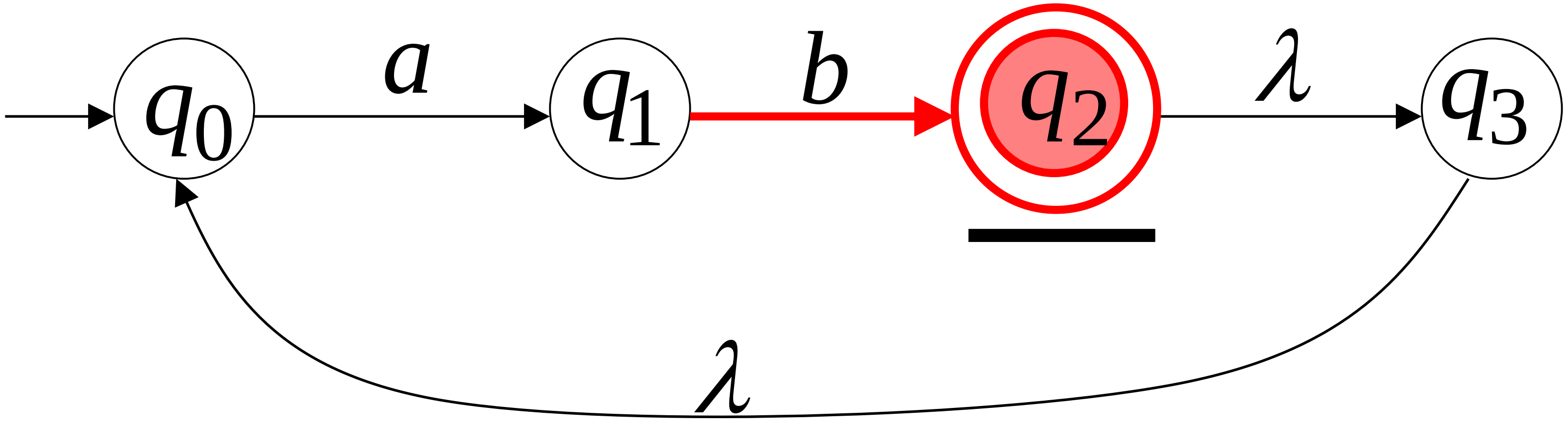

a
b


q_0
a
q_1
b
q_2
λ
q_3
λ

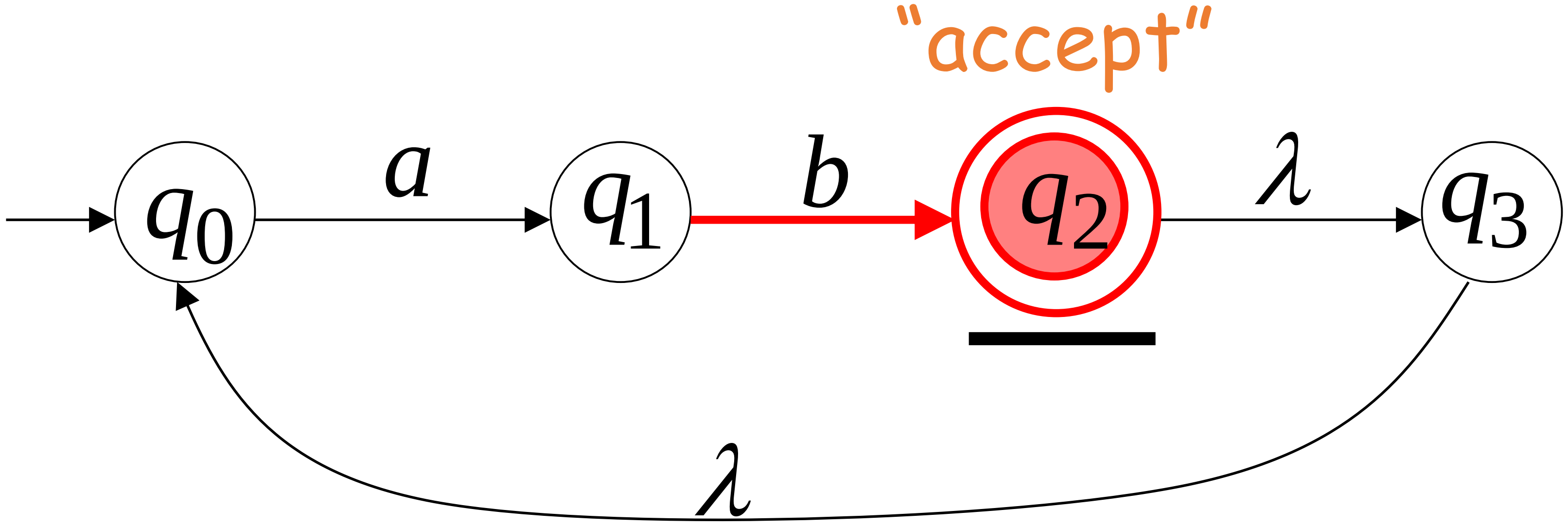

a
b


q_0
a
q_1
b
q_2
λ
q_3
λ

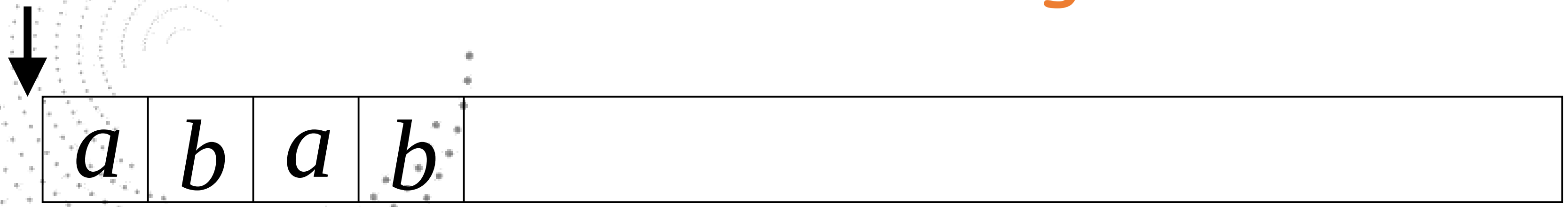

a
b

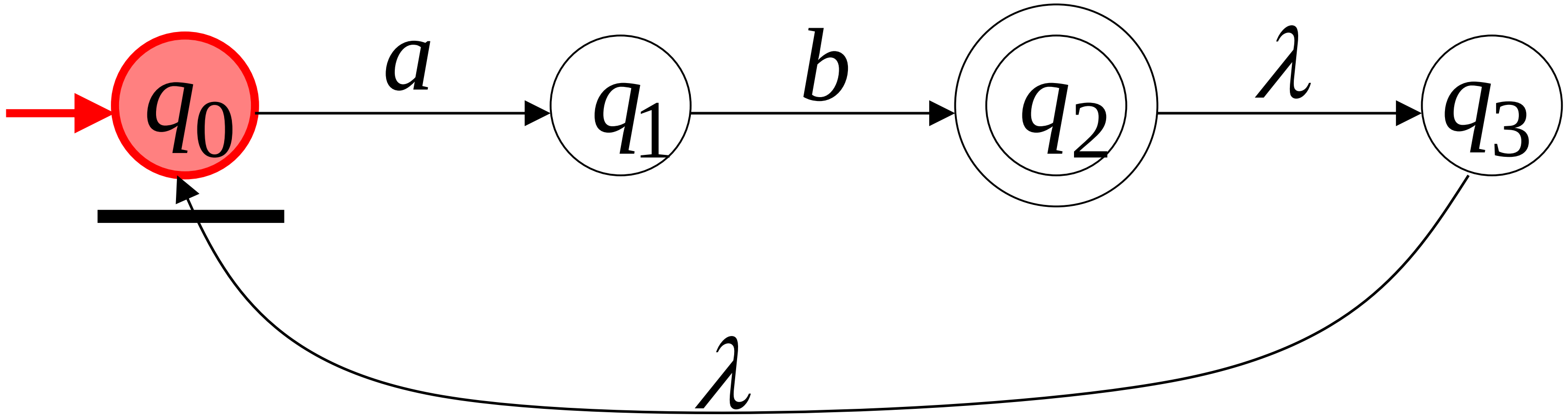

q_0
a
q_1
b
q_2
λ
q_3
λ


a
b

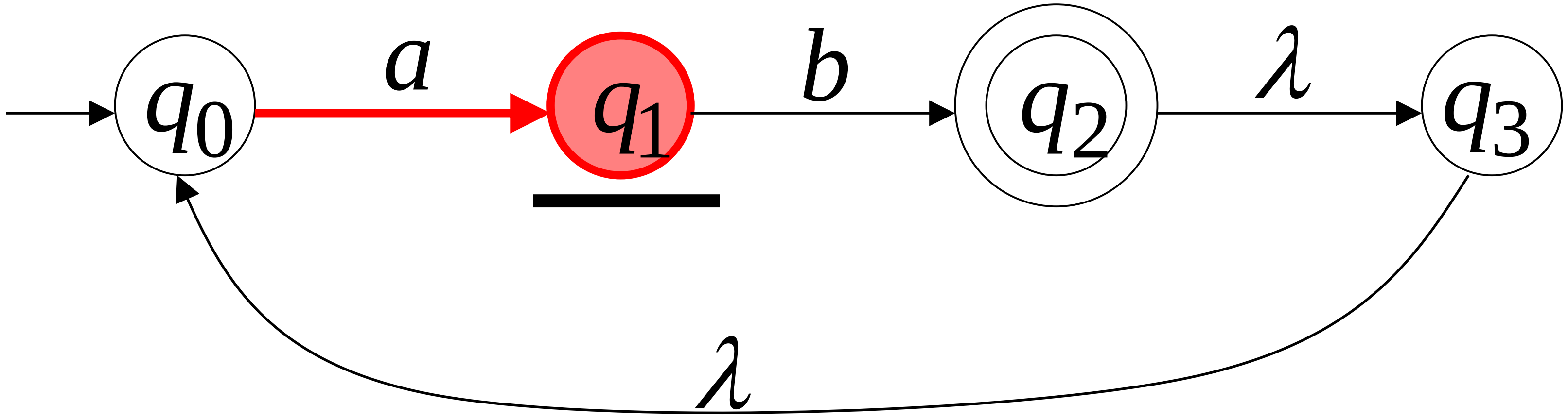

"accept"
a
b
λ
λ
q0
q1
q2
q3

Another String

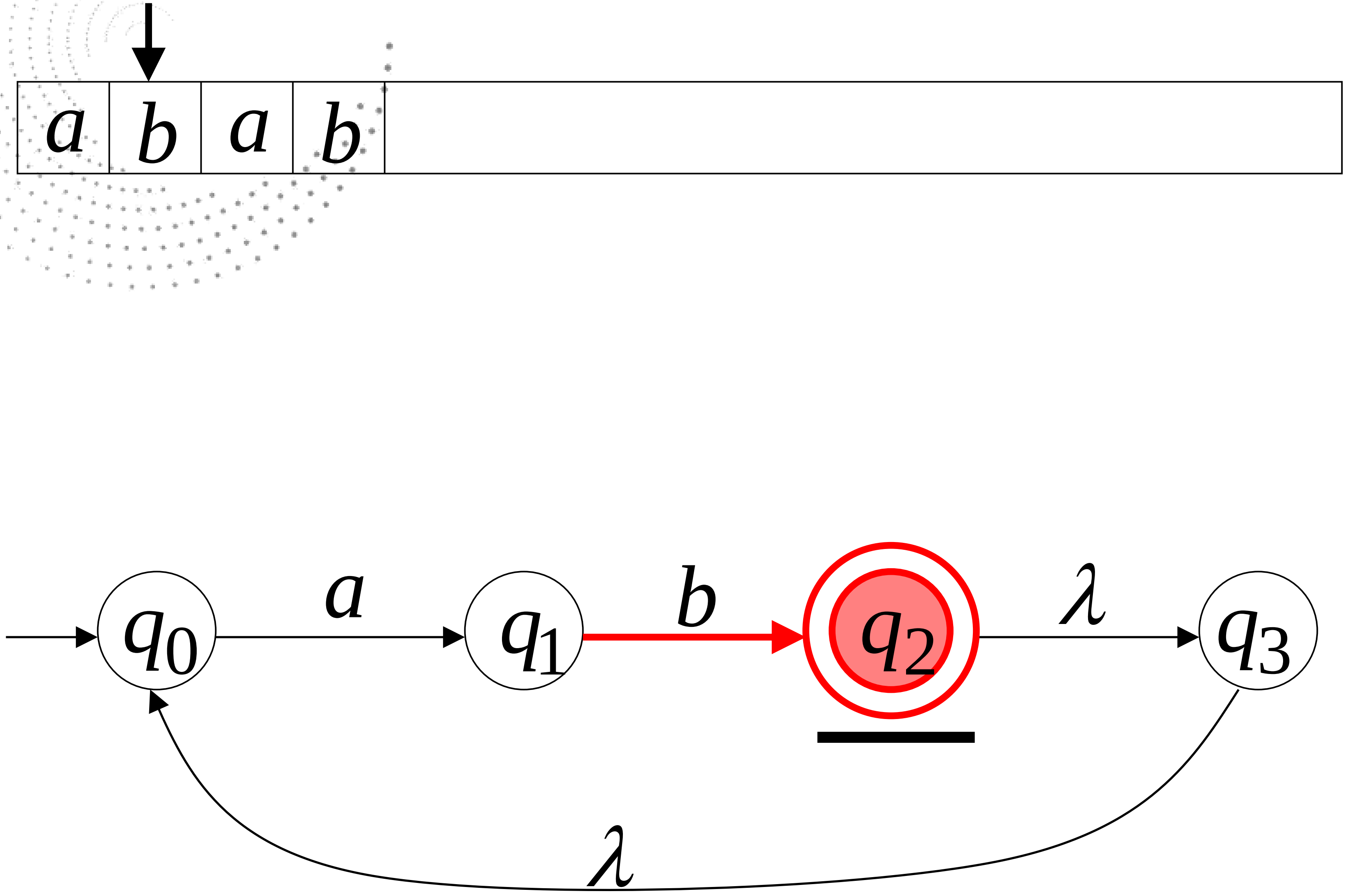




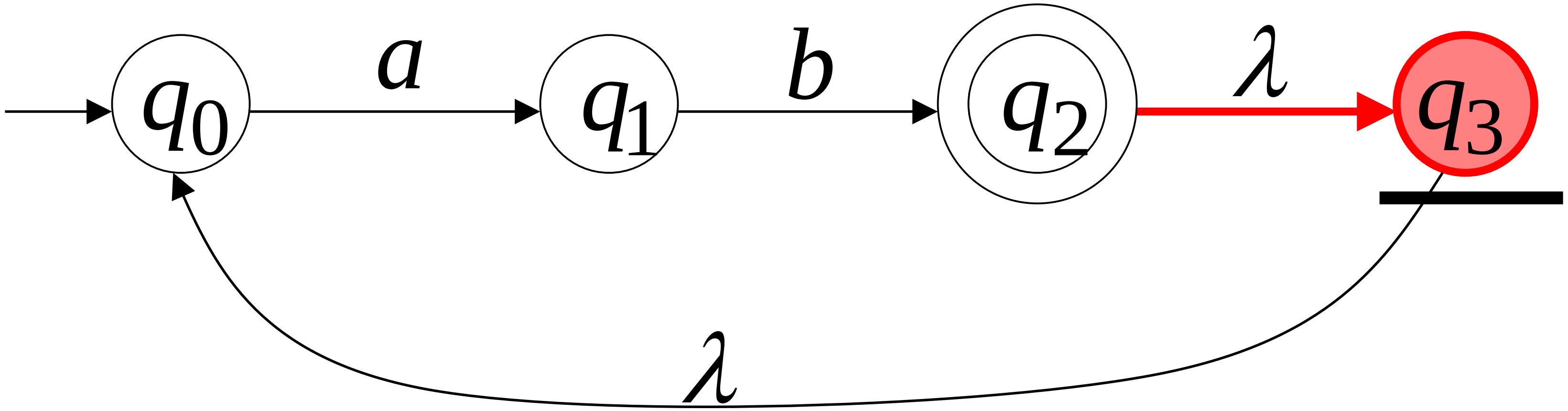
a b a b

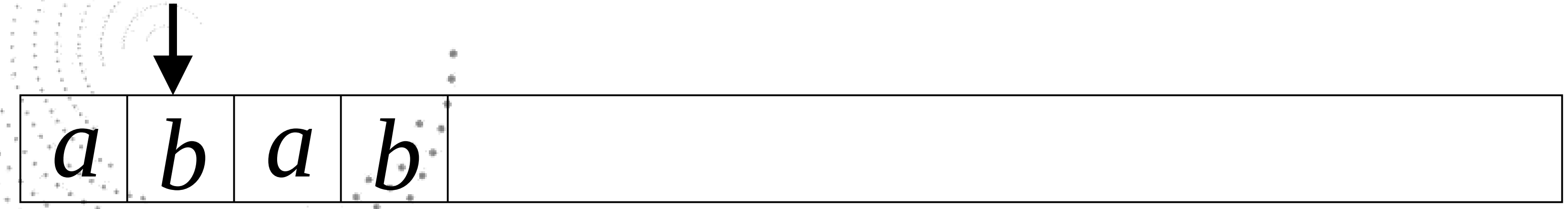


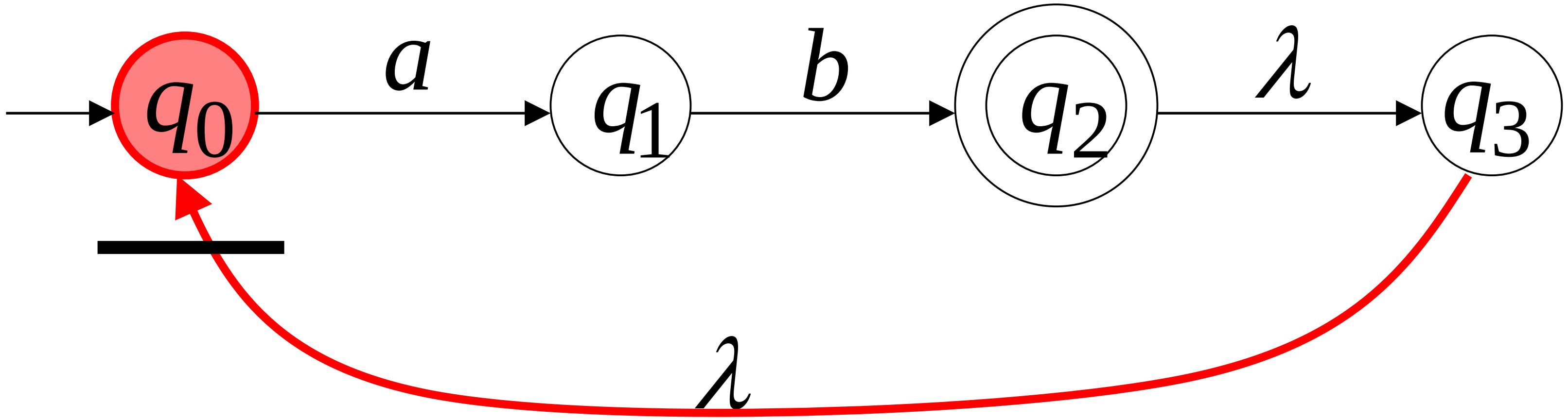
q_0
a
q_1
b
q_2
λ
q_3
λ

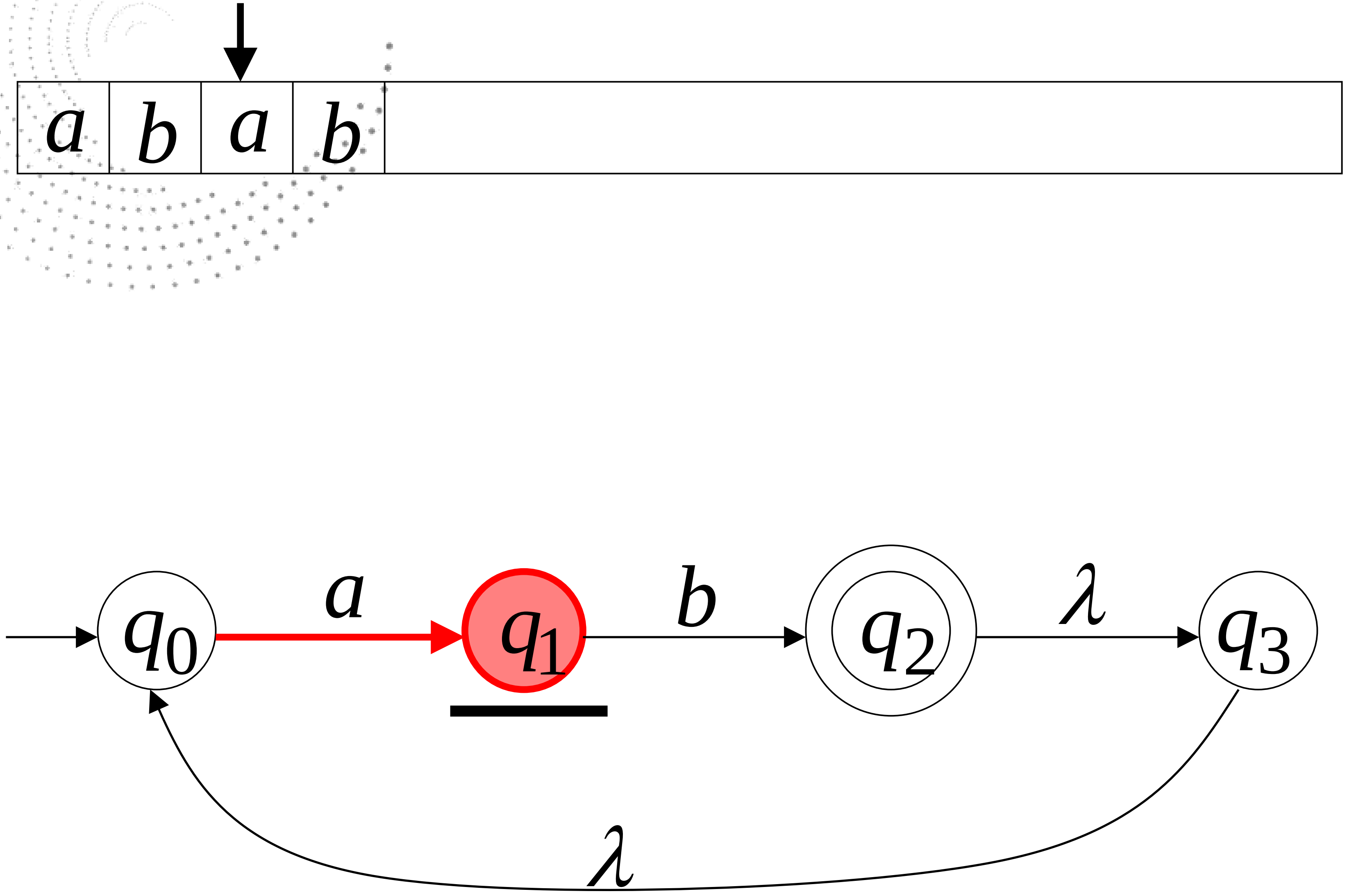

a
b
a
b
q_0
a
q_1
b
q_2
λ
q_3
λ

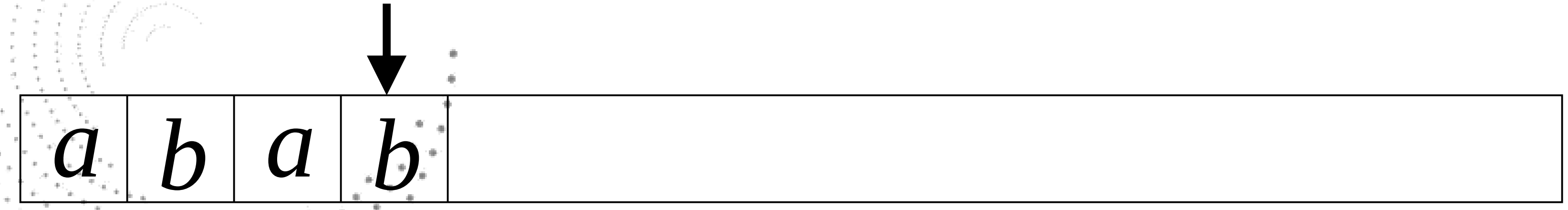

a
b
a
b


q_0
a
q_1
b
q_2
λ
q_3
λ

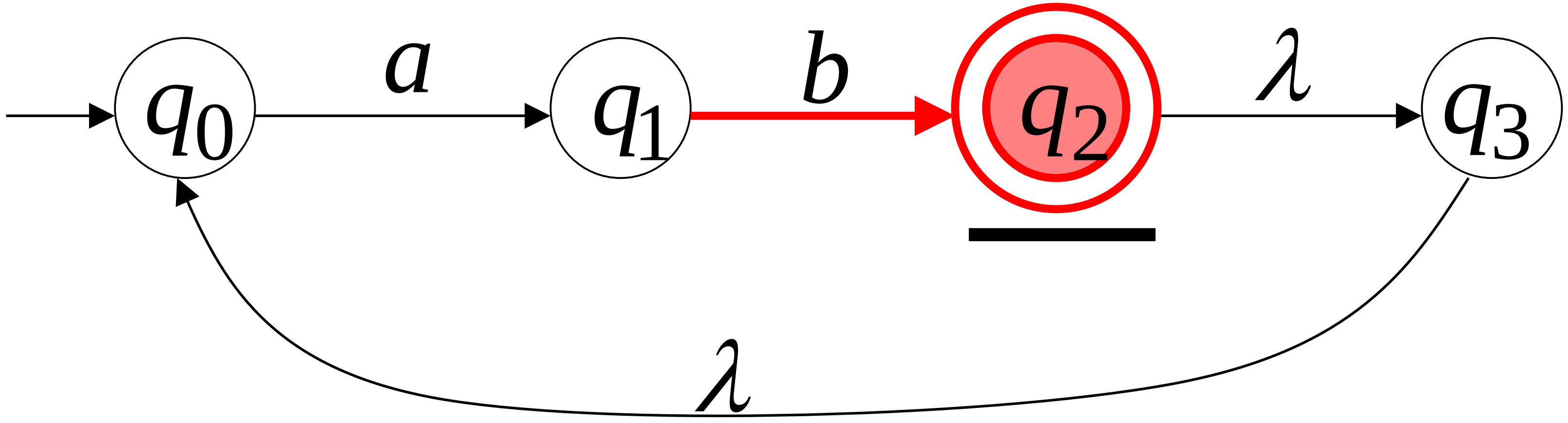

a
b
a
b


q_0
a
q_1
b
q_2
λ
q_3
λ

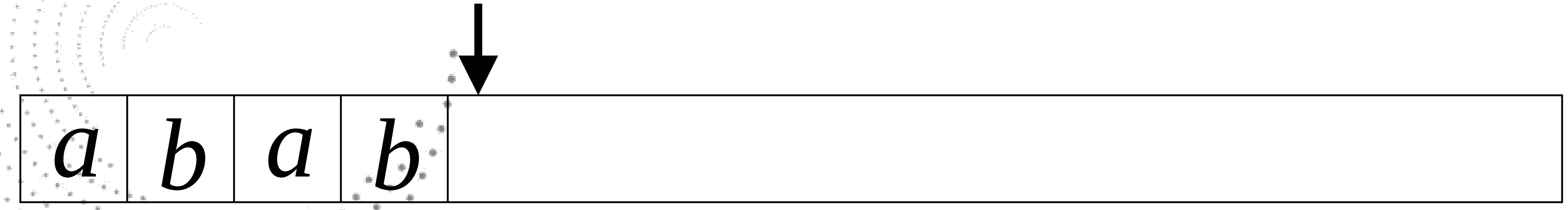

a
b
a
b
q_0
a
q_1
b
q_2
λ
q_3
λ

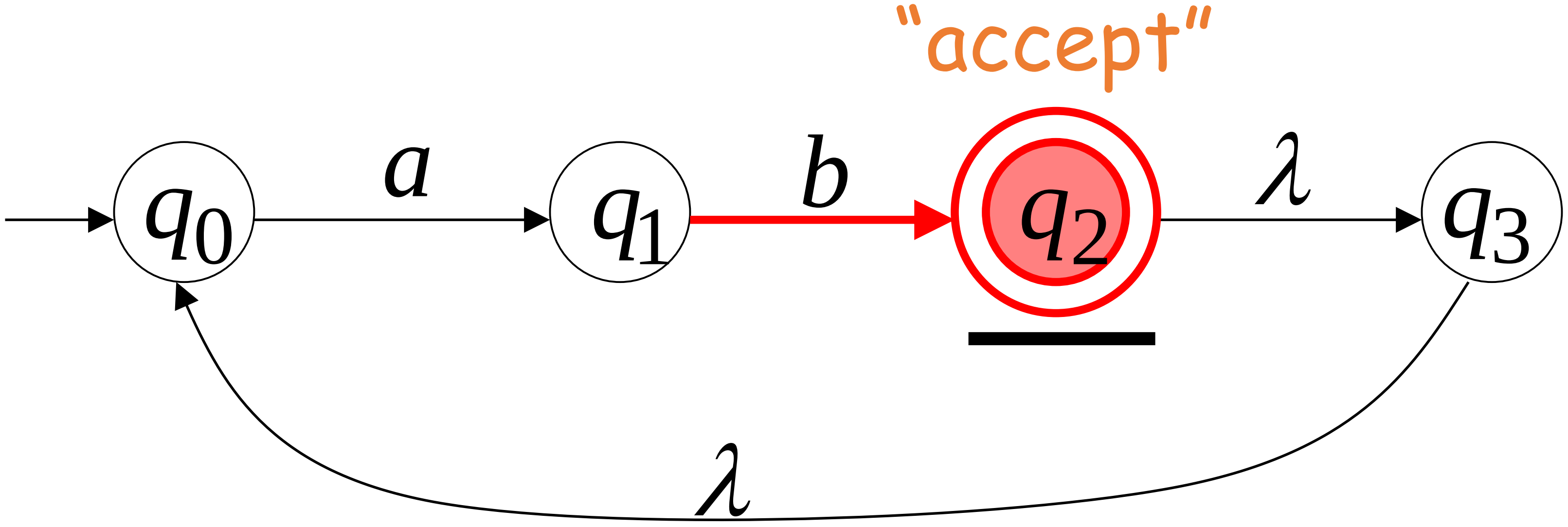


a
b
a
b



q_0
a
q_1
b
q_2
λ
q_3
λ

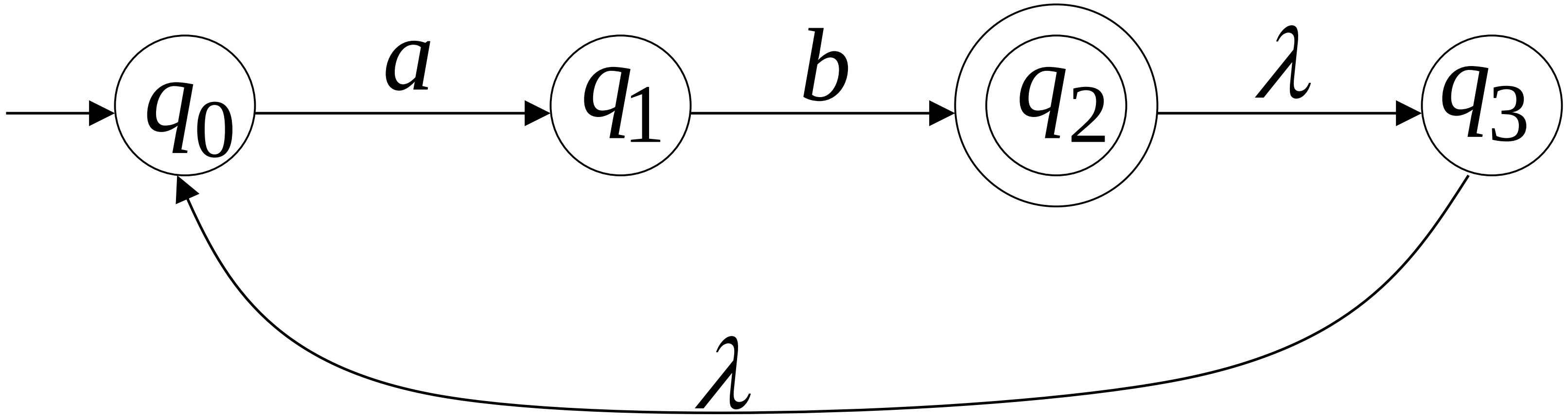

a
b
a
b


"accept"
q_0
a
q_1
b
q_2
λ
q_3
λ

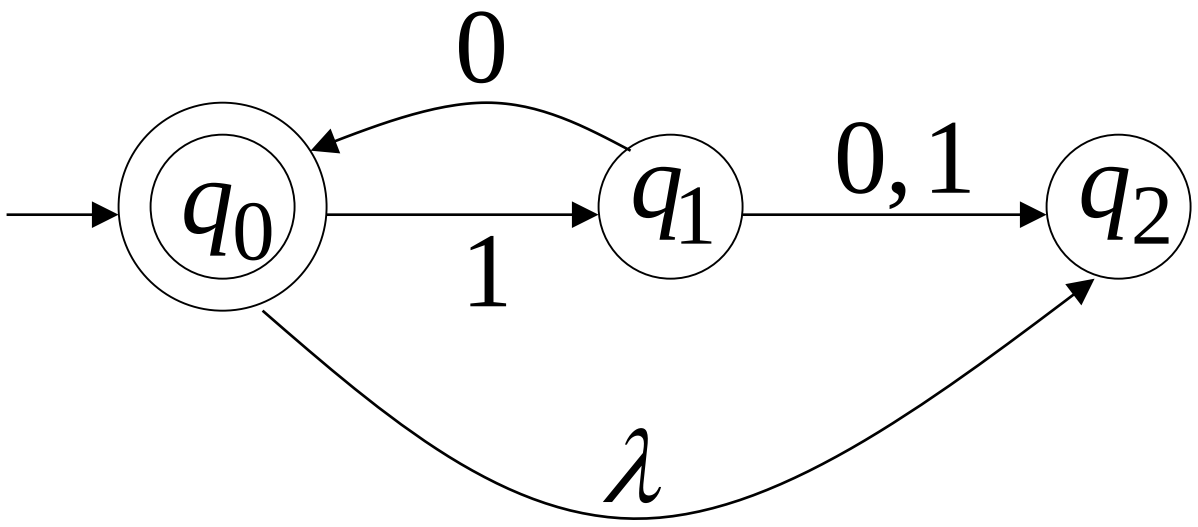
Language accepted

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

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

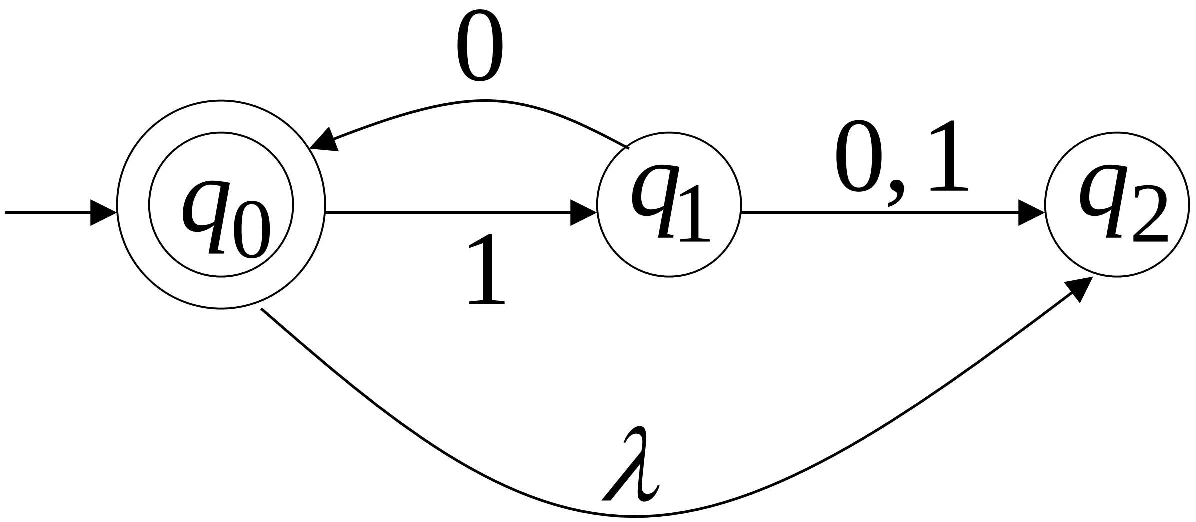


Another NFA Example



Language accepted

$$L(M) = \{\lambda,\ 10,\ 1010,\ 101010,\ \ldots\}$$
$$= \{10\}^*$$



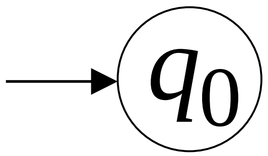
(redundant state)

Remarks:

- The λ symbol never appears on the input tape

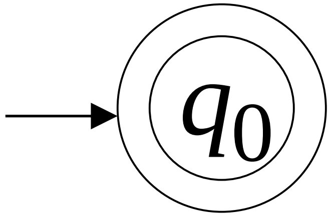
- Simple automata:

M_1



$$L(M_1) = \{\}$$

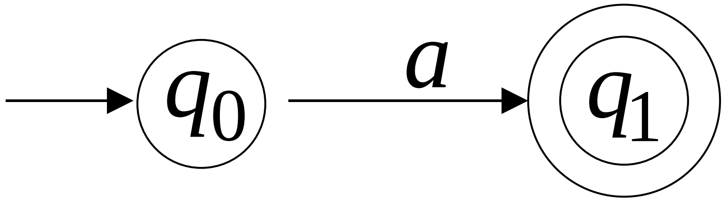
M_2



$$L(M_2) = \{\lambda\}$$

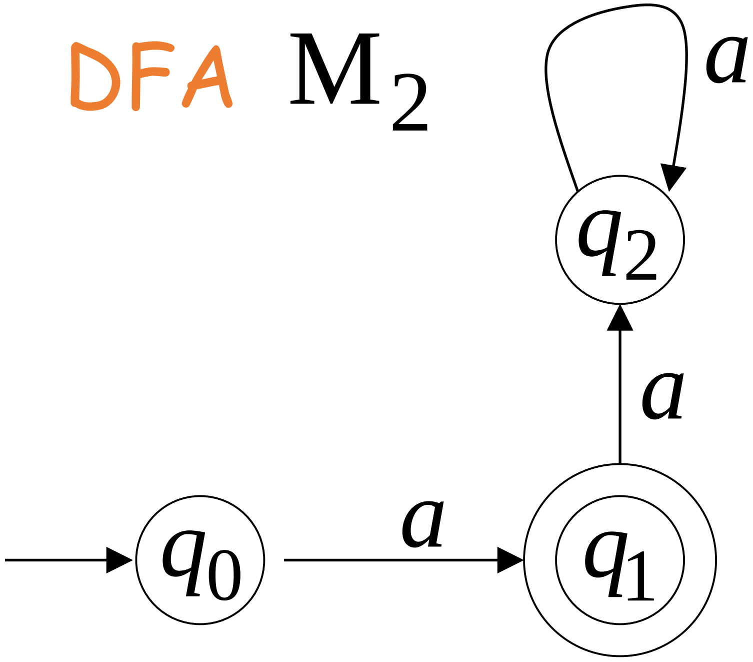
- NFAs are interesting because we can express languages easier than DFAs

NFA M_1



$$L(M_1) = \{a\}$$

DFA M_2



$$L(M_2) = \{a\}$$

Formal Definition of NFAs

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

$Q:$ Set of states, i.e. $\{q_0, q_1, q_2\}$

$\Sigma:$ Input aplhabet, i.e. $\{a, b\}$

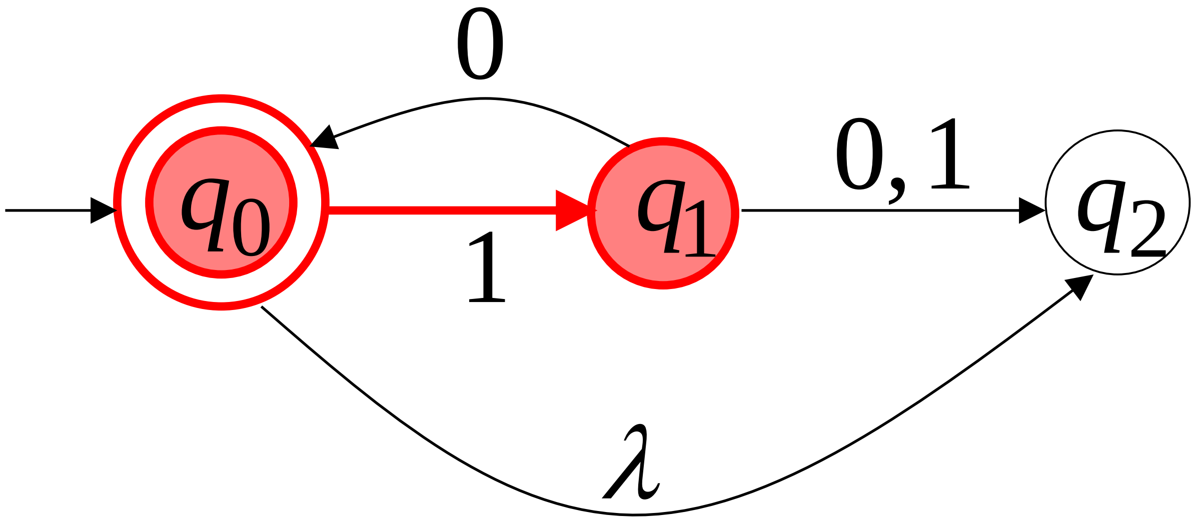
$\delta:$ Transition function

$q_0:$ Initial state

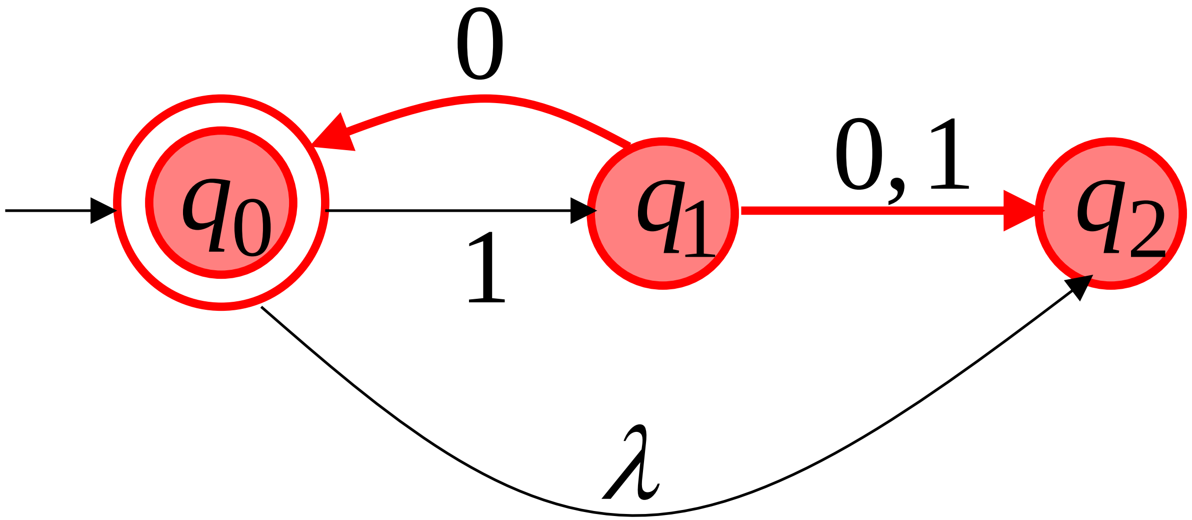
$F:$ Final states

Transition Function δ

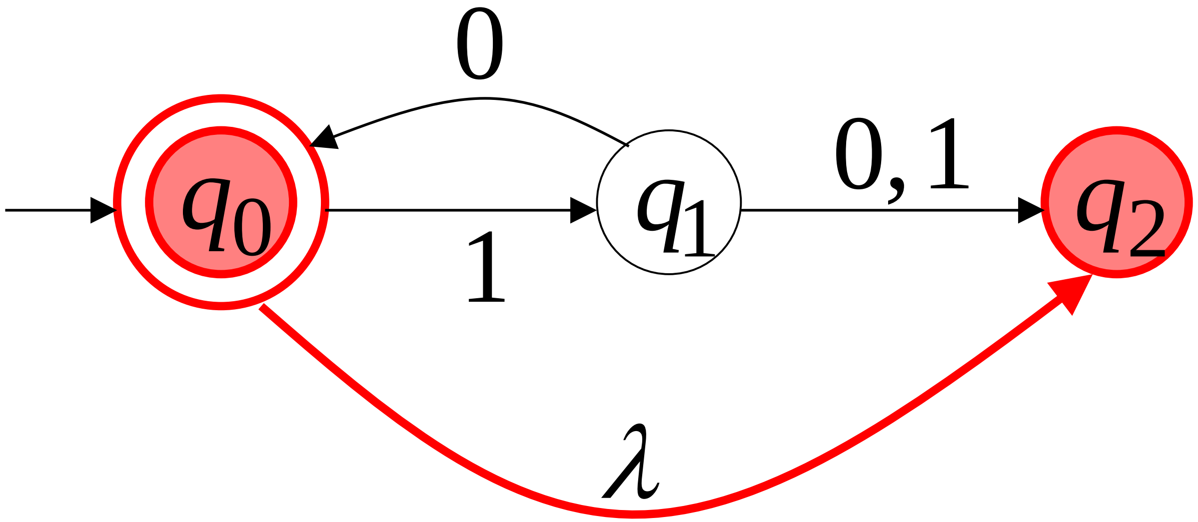
$$\delta(q_0, 1) = \{q_1\}$$



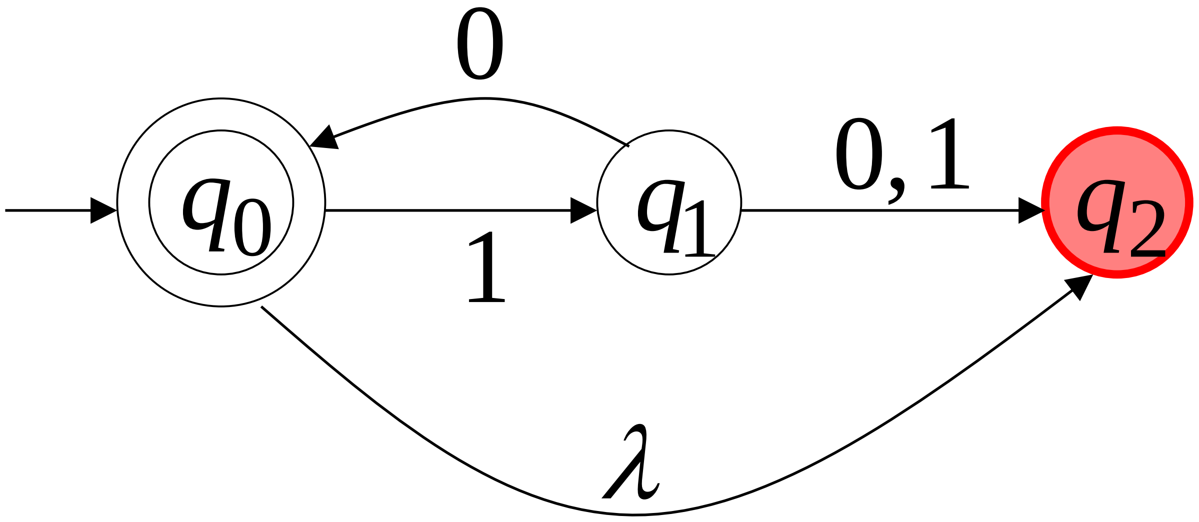
$$\delta(q_1, 0) = \{q_0, q_2\}$$



$$\delta(q_0, \lambda) = \{q_0, q_2\}$$

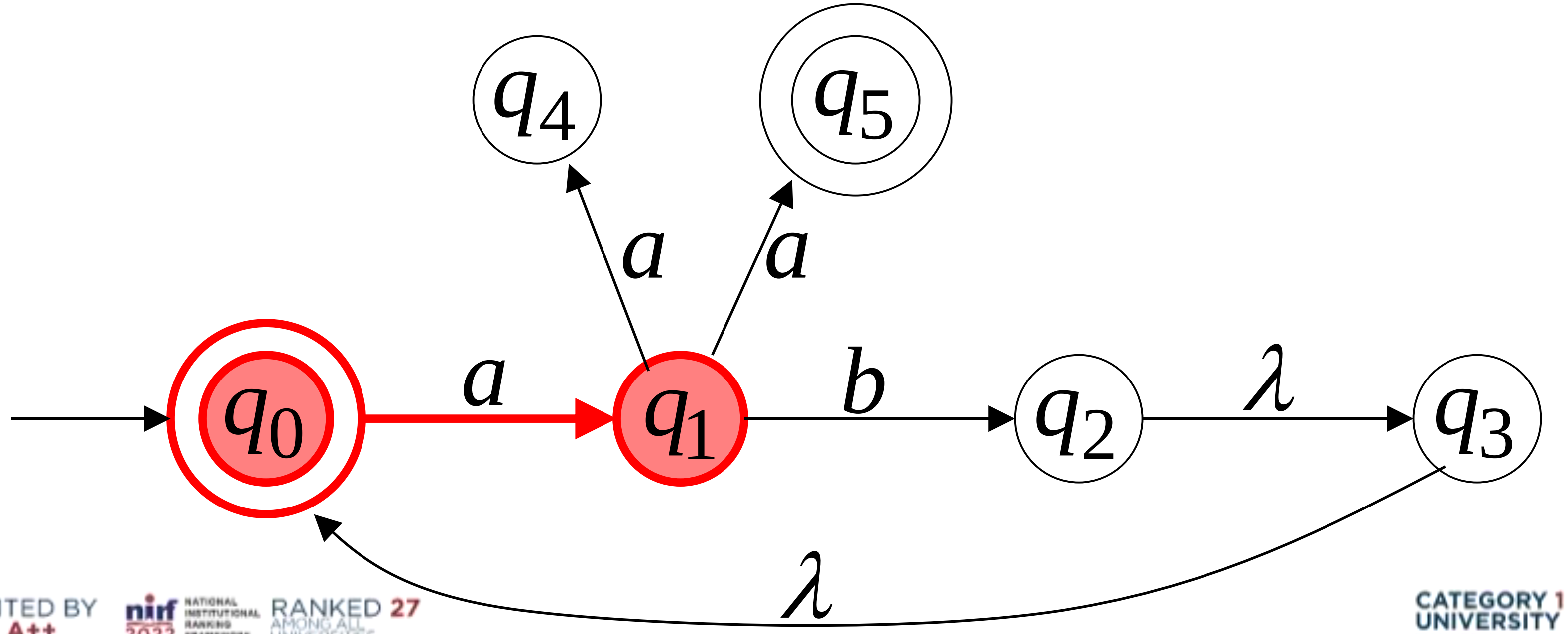


$$\delta(q_2, 1) = \varnothing$$

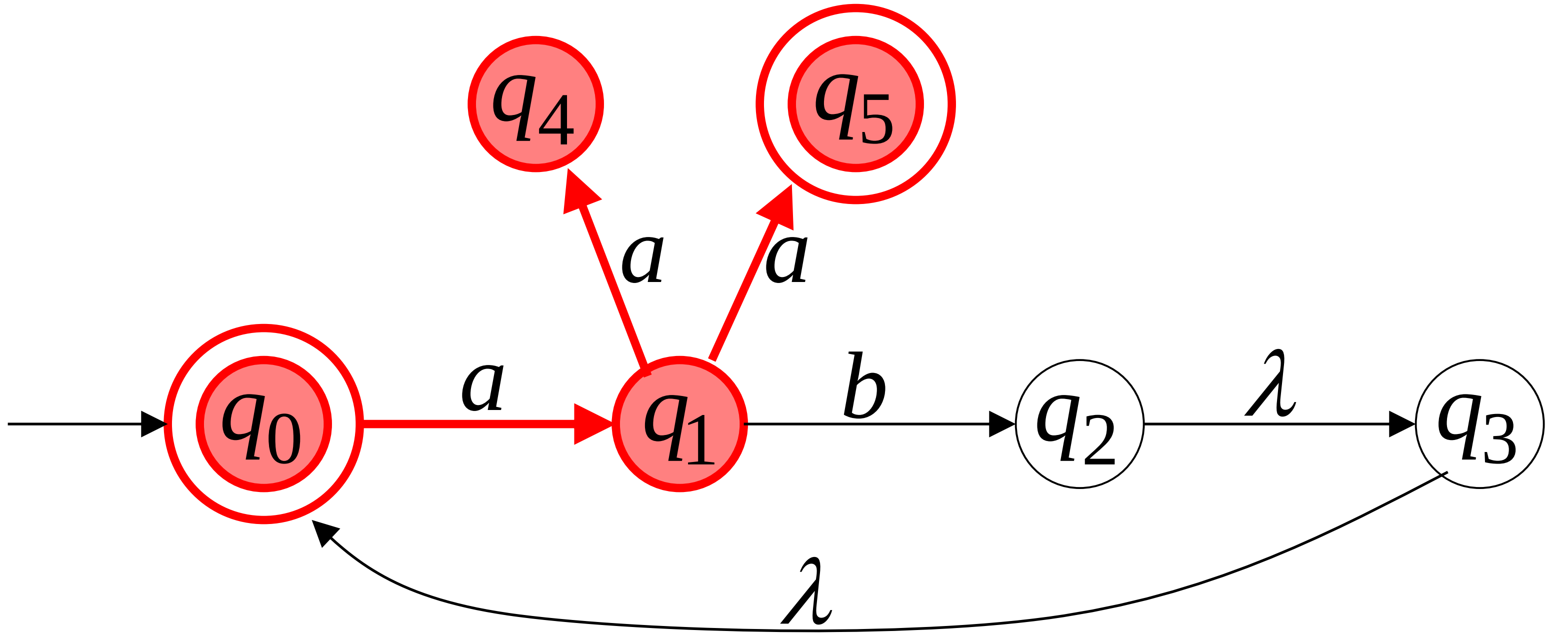


Extended Transition Function δ^*

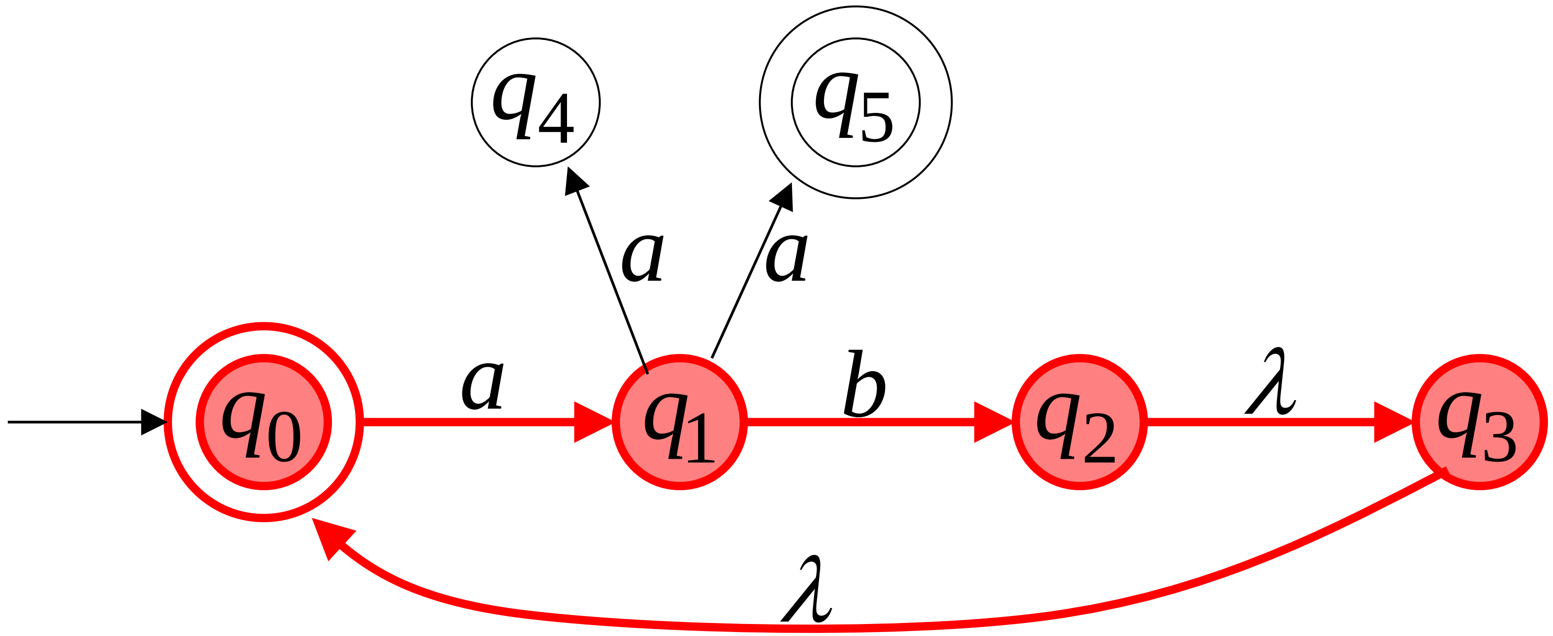
$$\delta^*(q_0, a) = \{q_1\}$$



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

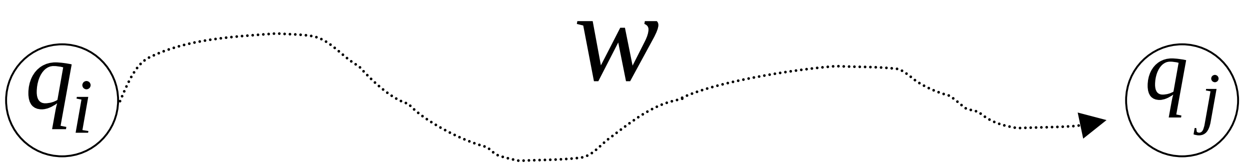


$$\delta^*(q_0, ab) = \{q_2, q_3, q_0\}$$

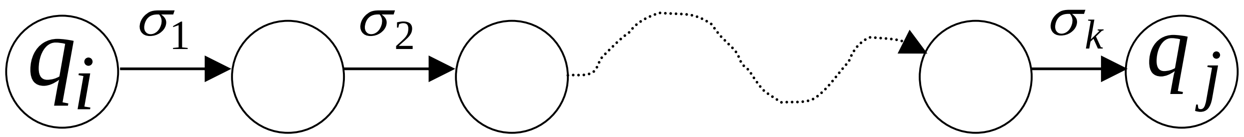


Formally

$$q_j \in \delta^*(q_i, w)$$: there is a walk from q_i to q_j with label w

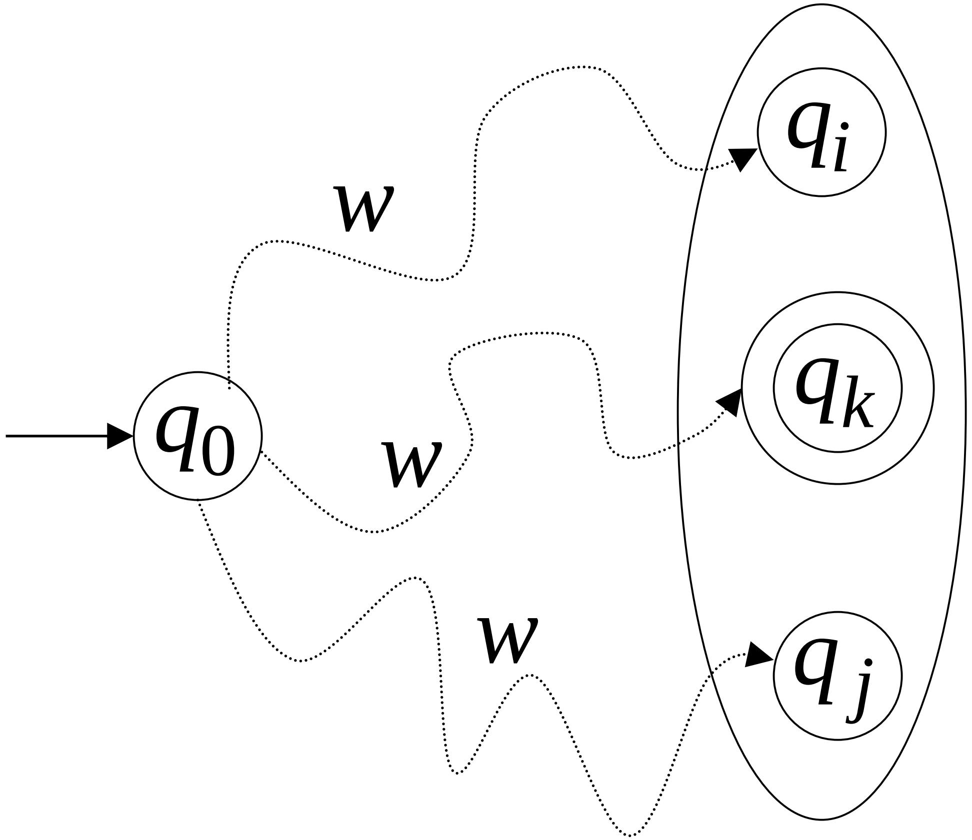


$$w = \sigma_1 \sigma_2 \cdots \sigma_k$$



A

$$q_j \in \delta^*(q_i, w) \qquad \delta^*(q_0, w)$$



Inductive Definition

- Basis: $\delta^*(q, \lambda) = \{q\}$

Induction: Suppose w = x a.

Also $\delta^*(q, x) = \{p_1, p_2, \cdots, p_k\}$

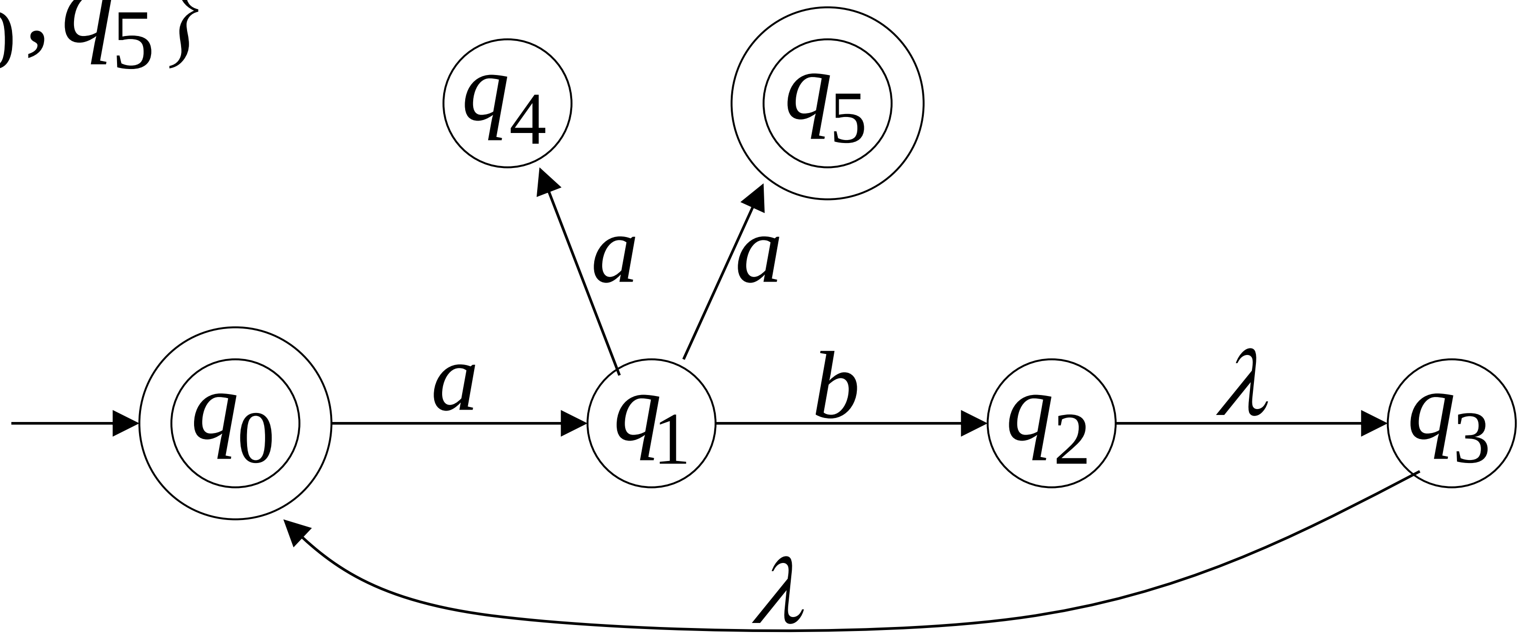
Let $\bigcup_{i=1}^{k} \delta(p_i, a) = \{r_1, r_2, \cdots, r_m\}$

Then $\delta^*(q, w) = \{r_1, r_2, \cdots, r_m\}$

The Language of an NFA

M

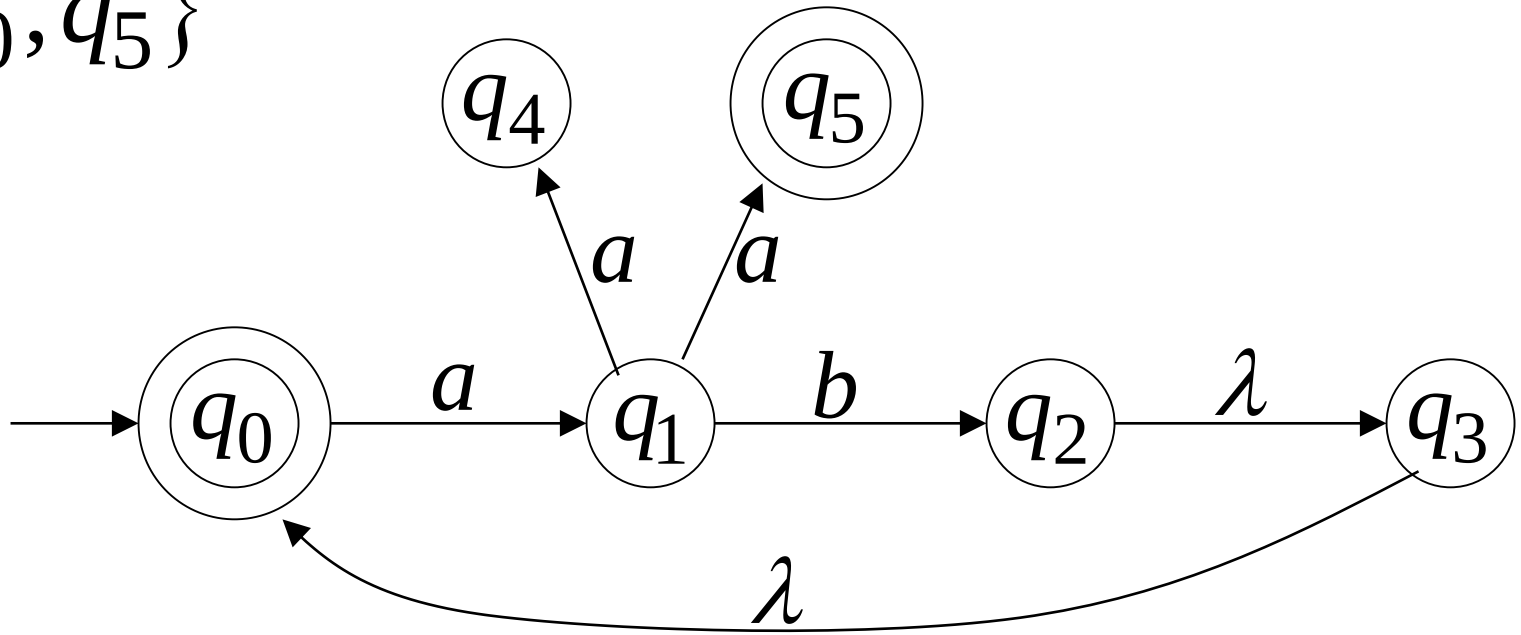
- $F = \{q_0, q_5\}$



$$\delta^*(q_0, aa) = \{q_4, \underline{q_5}\} \qquad aa \in L(M)$$

$$\searrow \in F$$

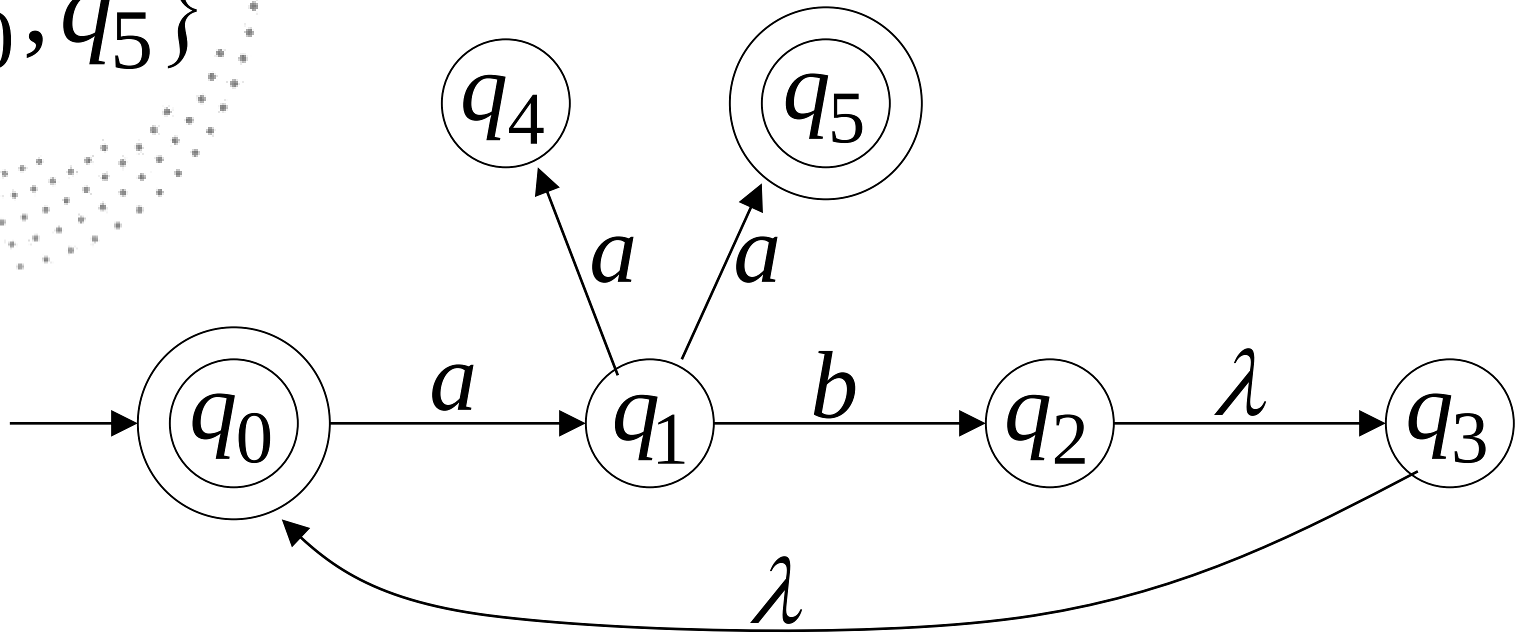
$$F = \{q_0, q_5\}$$



$$\delta^*(q_0, ab) = \{q_2, q_3, \underline{q_0}\} \qquad ab \in L(M)$$

$$\searrow \in F$$

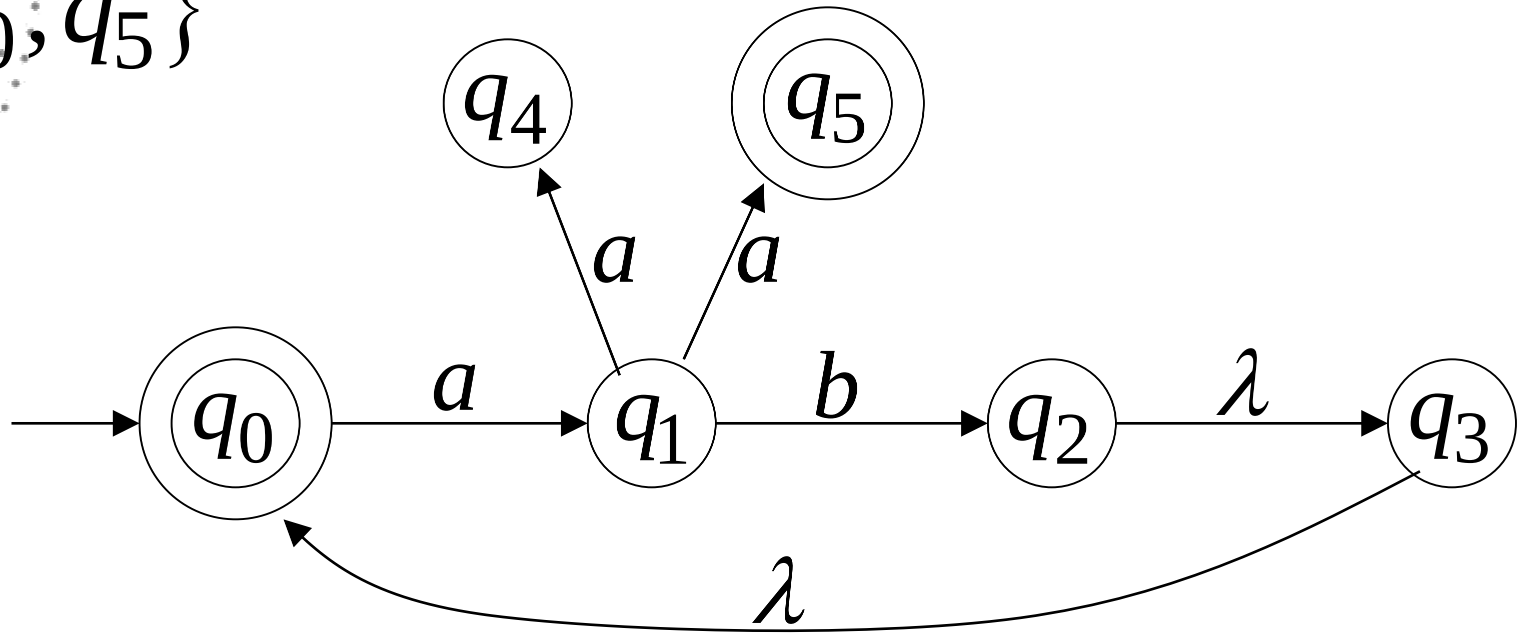
$$F = \{q_0, q_5\}$$



$$\delta^*(q_0, abaa) = \{q_4, \underline{q_5}\} \qquad abaa \in L(M)$$

$$q_5 \in F$$

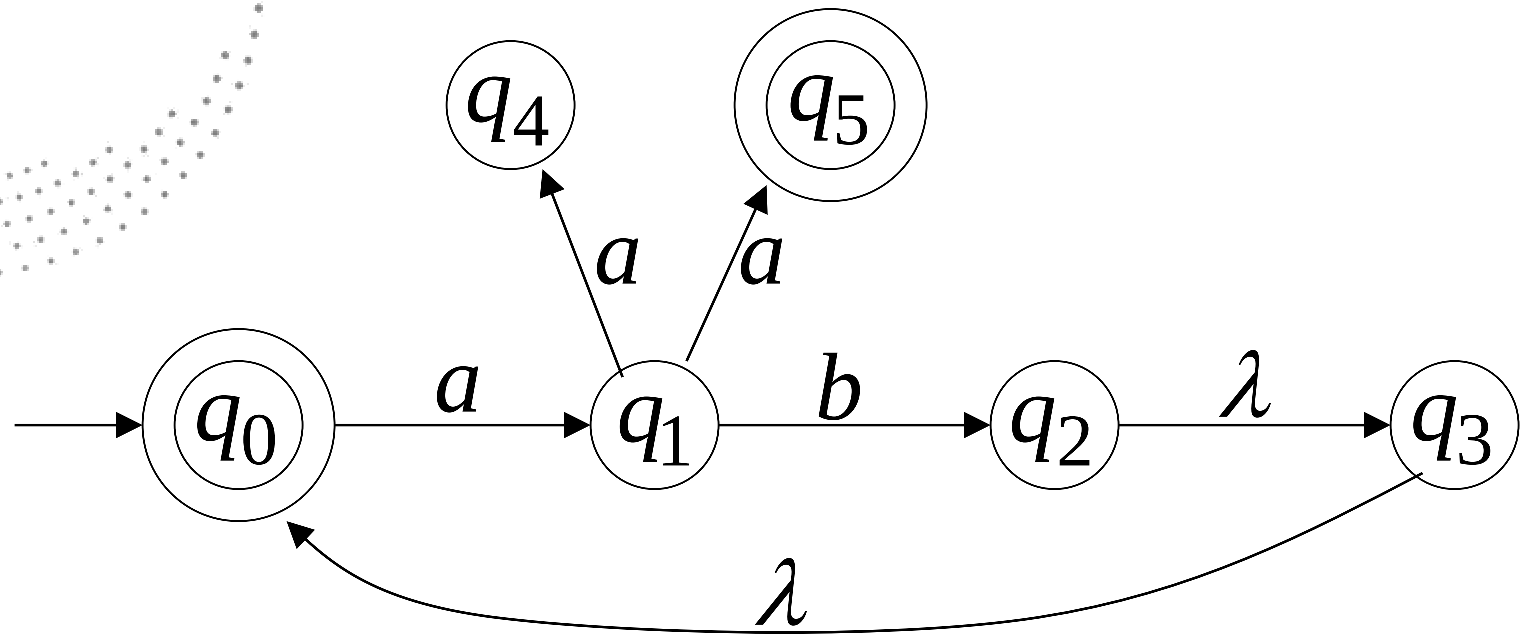
$$F = \{q_0, q_5\}$$



$$\delta^*(q_0, aba) = \{q_1\} \qquad aba \notin L(M)$$

$$\notin F$$

•



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

Formally

- The language accepted by NFA M is:

$$L(M) = \{w_1, w_2, w_3, \ldots\}$$

- where

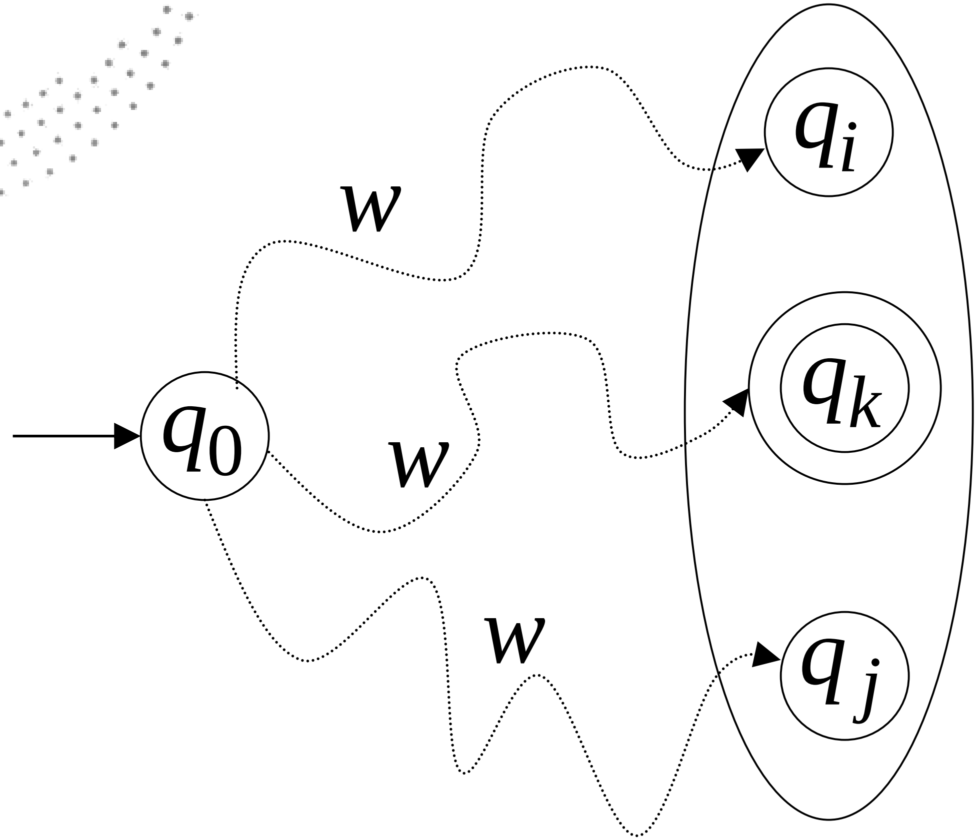
$$\delta^*(q_0, w_m) = \{q_i, q_j, \ldots, q_k, \ldots\}$$

- and there is some

$$q_k \in F$$ (final state)

- $w \in L(M)$

$\delta^*(q_0, w)$



$q_k \in F$

QUIZ TIME

Which of the following statements is true about non-deterministic finite automata (NFA)?

a) An NFA can only have one initial state.

b) An NFA can have multiple initial states.

c) An NFA does not have any final states.

d) An NFA can only recognize regular languages.

Answer: b) An NFA can have multiple initial states.

QUIZ TIME

What is the key difference between a deterministic finite automaton (DFA) and a non-deterministic finite automaton (NFA)?

a) An NFA can recognize non-regular languages, whereas a DFA cannot.

b) An NFA can recognize regular languages, whereas a DFA cannot.

c) An NFA can have multiple possible next states for a given input symbol, whereas a DFA has only one.

d) An NFA has a more complex state transition diagram compared to a DFA.

Answer: c) An NFA can have multiple possible next states for a given input symbol, whereas a DFA has only one.

QUIZ TIME

Which of the following operations is NOT closed under the class of regular languages?

a) Union

b) Intersection

c) Complementation

d) Concatenation

Answer: c) Complementation

QUIZ TIME

4: Which of the following is true regarding the acceptance of a string by an NFA?

a) An NFA accepts a string if there exists at least one accepting path.

b) An NFA accepts a string if there exists exactly one accepting path.

c) An NFA accepts a string only if all possible paths are accepting.

d) An NFA does not accept any strings.

Answer: a) An NFA accepts a string if there exists at least one accepting path.

QUIZ TIME

Which of the following algorithms can be used to convert a non-deterministic finite automaton (NFA) into an equivalent deterministic finite automaton (DFA)?

a) Thompson's construction algorithm

b) Subset construction algorithm

c) Hopcroft's algorithm

d) Pumping lemma algorithm

Answer: b) Subset construction algorithm

MCQ:

1. What is a non-deterministic finite automaton (NFA)? a) A finite automaton that can accept an infinite number of input strings b) A finite automaton that has multiple transitions for a state and input symbol c) A finite automaton that can recognize context-free languages d) A finite automaton that can recognize non-regular languages

- Answer: b) A finite automaton that has multiple transitions for a state and input symbol

2. What is the role of ε-transitions in a non-deterministic finite automaton (NFA)? a) They allow the automaton to accept an empty string as input b) They simplify the construction of a deterministic finite automaton (DFA) c) They provide a way to transition between states without consuming any input symbol d) They enable the automaton to recognize context-sensitive languages

Answer: c) They provide a way to transition between states without consuming any input symbol

MCQ

Which of the following statements is true about non-deterministic finite automata (NFAs) and deterministic finite automata (DFAs)?

a) NFAs can recognize more languages than DFAs

b) DFAs can recognize more languages than NFAs

c) NFAs and DFAs can recognize the same set of languages

d) NFAs and DFAs have different input alphabets

Answer: a) NFAs can recognize more languages than DFAs

Which of the following is an example of a language that can be recognized by a non-deterministic finite automaton (NFA) but not by a deterministic finite automaton (DFA)?

a) The set of all strings containing an equal number of '0's and '1's

b) The set of all strings that start and end with the same symbol

c) The set of all strings that have a prime number of characters

d) The set of all strings that consist of alternating '0's and '1's

Answer: a) The set of all strings containing an equal number of '0's and '1's

Terminal Questions

1. Explain the process of converting a deterministic finite automaton (DFA) to a regular expression.
2. Describe the steps involved in converting a non-deterministic finite automaton with ε-transitions (NFA-ε) to a regular expression.
3. What is the significance of the state elimination method in converting a finite automaton to a regular expression?
4. Can every finite automaton be converted to an equivalent regular expression? Explain.
5. Discuss the limitations of the state elimination method for converting a DFA to a regular expression.
6. Explain the role of the subset construction algorithm in converting an NFA to a regular expression.
7. Can a regular expression be directly derived from a non-deterministic finite automaton (NFA)? If not, what additional steps are required?
8. Discuss the advantages and disadvantages of converting a finite automaton to a regular expression.
9. Explain how closure properties of regular languages can be utilized in the conversion of a finite automaton to a regular expression.
10. Discuss the relationship between the size of a finite automaton and the resulting regular expression after conversion.

THANK YOU

Team – TOC