COM3064
Automata Theory

Week 3: Nondeterministic Finite Automata

Lecturer: Dr. Sevgi YİĞİT SERT
Spring 2023

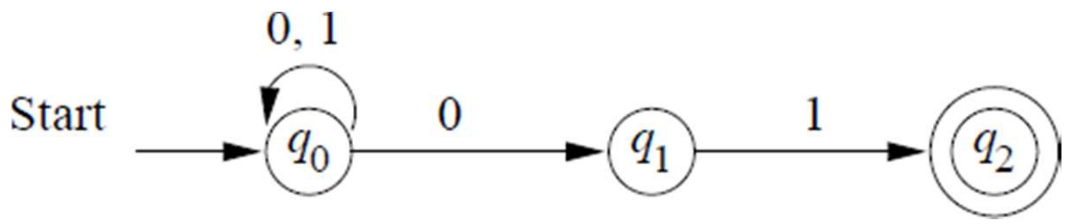
Resources: Introduction to The Theory of Computation, M. Sipser,
Introduction to Automata Theory, Languages, and Computation, J.E. Hopcroft, R. Motwani, and J.D. Ullman
BBM401 Automata Theory and Formal Languages, İlyas Çiçekli

Non-Deterministic Finite Automata (NFA)

- A **nondeterministic finite automata (NFA)** can be in several states at once.
- A NFA state can have more than one arc leaving from that state with the same symbol.
- A NFA can allow state-to-state transitions on ε input.
  - These transitions are done spontaneously, without looking at the input string.
- A NFA starts in the start state and it accepts if any sequence of choices for the string leads to a final state.
- Nondeterminism may be viewed as a kind of parallel computation wherein multiple independent “processes” or “threads” can be running concurrently.
  - When the NFA splits to follow several choices, that corresponds to a process “forking” into several children, each proceeding separately. If at least one of these processes accepts, then the entire computation accepts.

NFA – Example

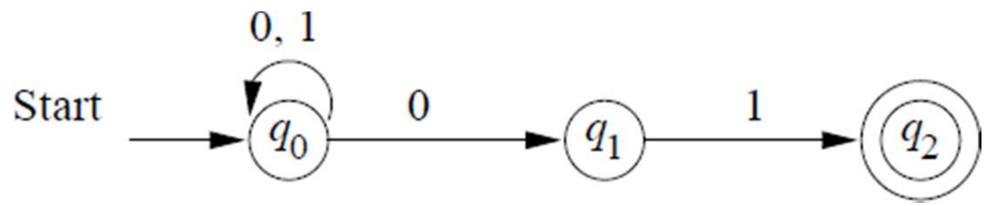
- An automata that accepts all and only strings ending in 01.



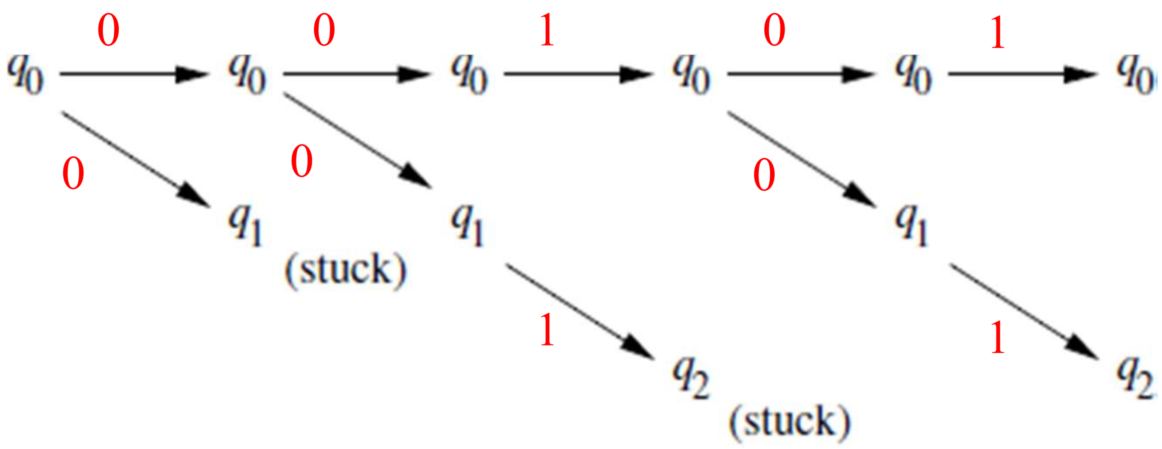
- State q_0 can go to q_0 or q_1 with the symbol 0. (non-determinism)

- NFA accepts a string w if there is a path accepts that string.
  - There can be other paths that do not accept that string.

NFA – Example

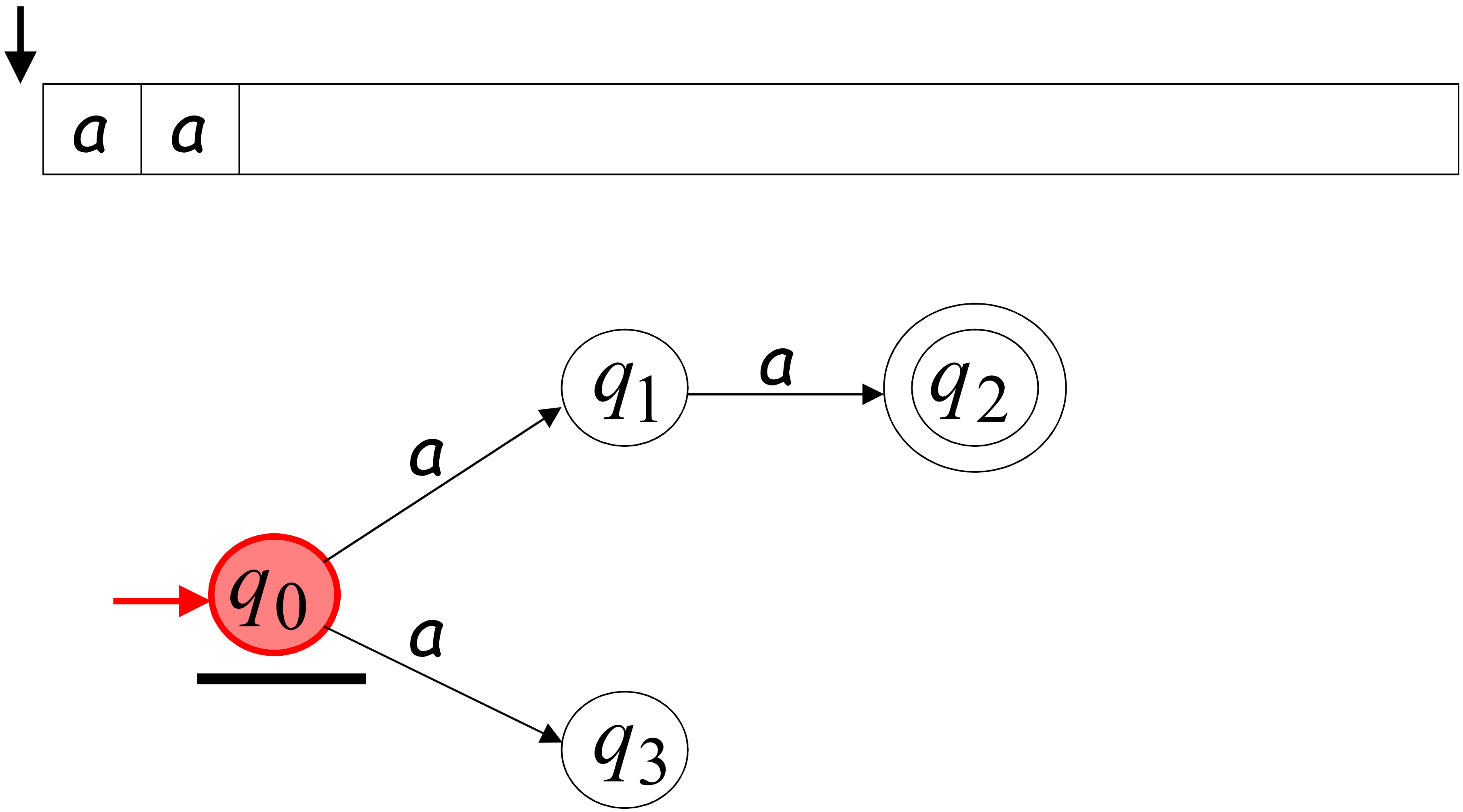


- What happens when the NFA processes the input 00101

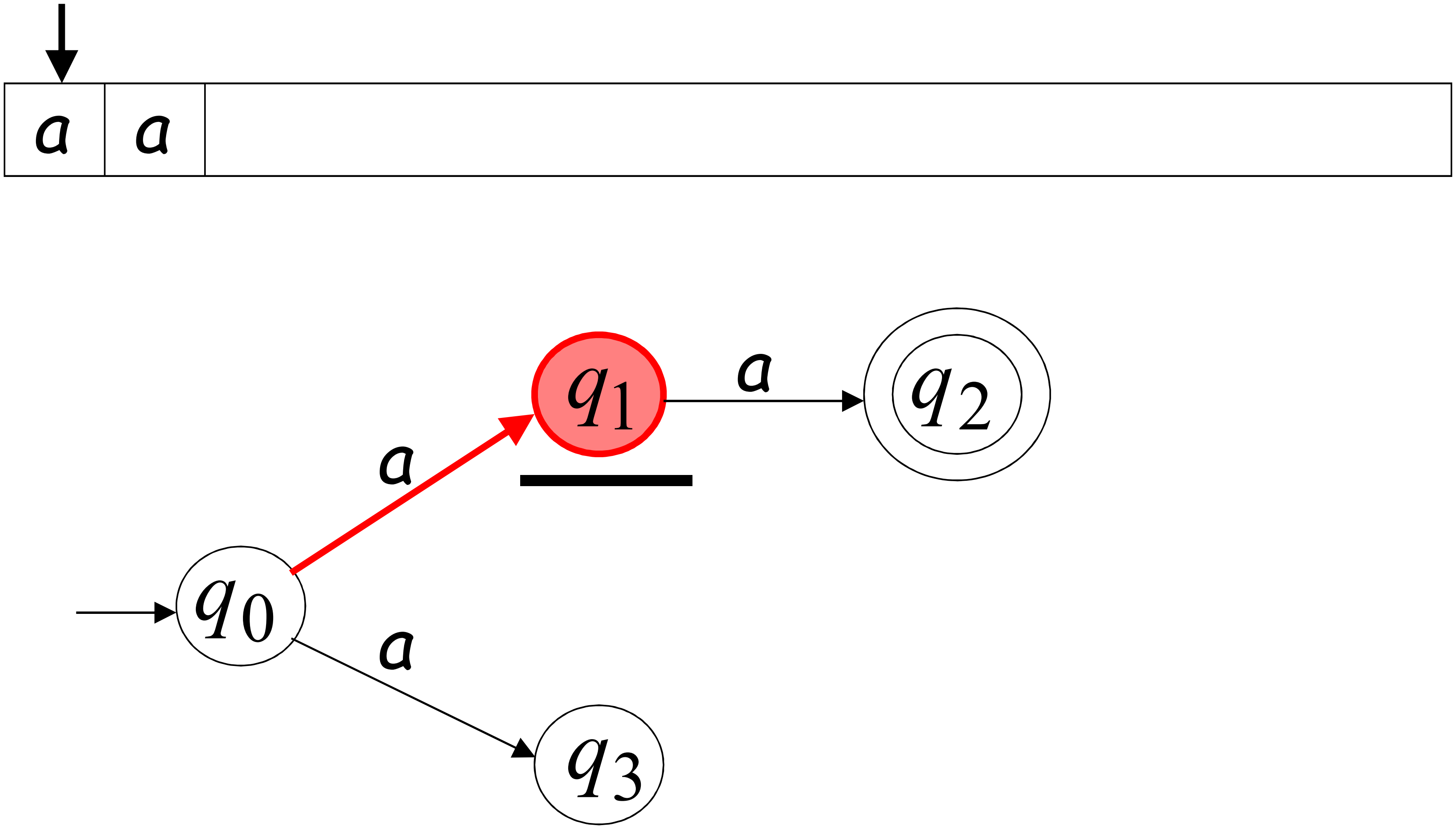


- All missing arcs go to a death state, the death state goes to itself for all symbols, and the death state is a non-accepting state.

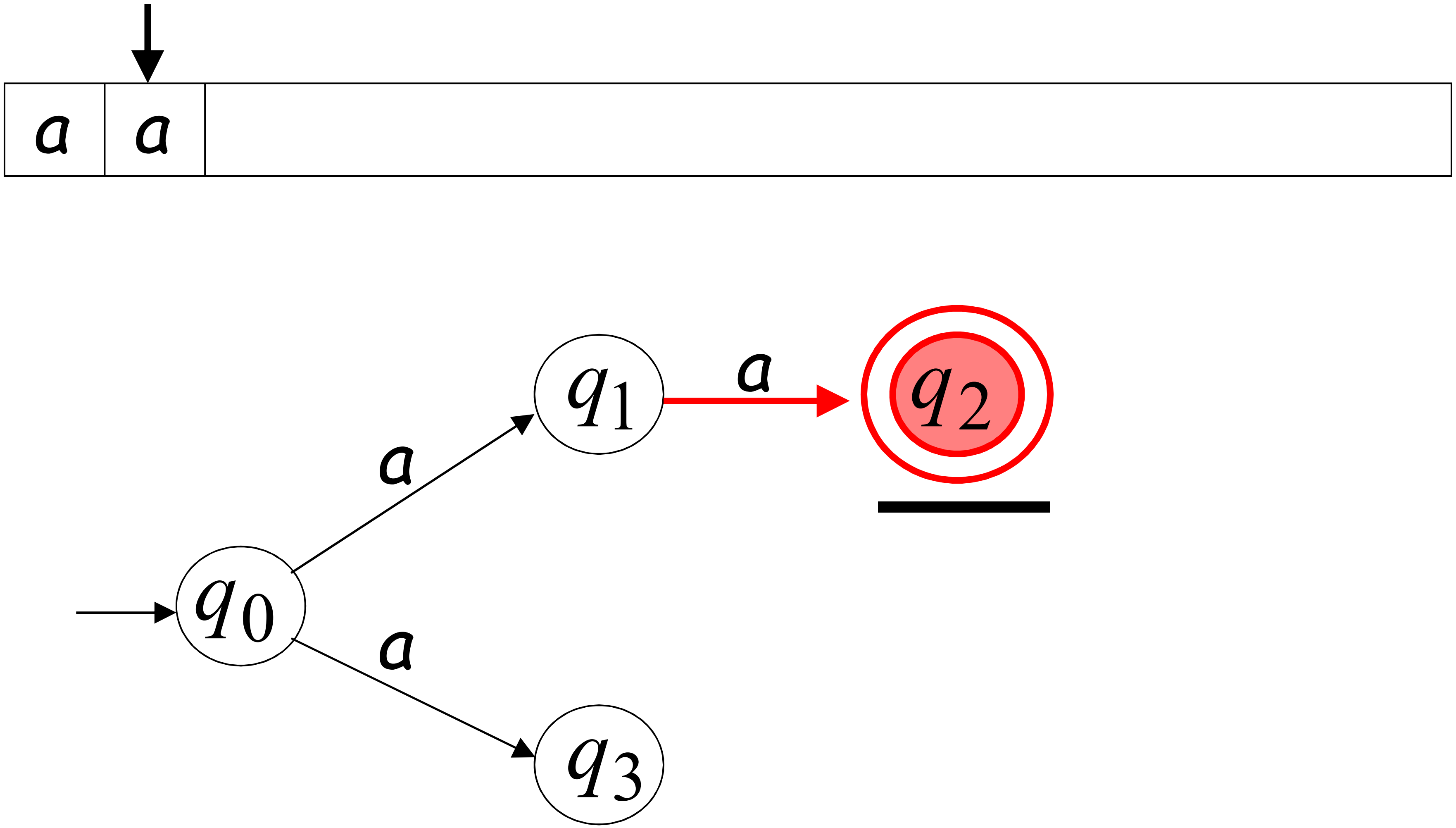
First Choice



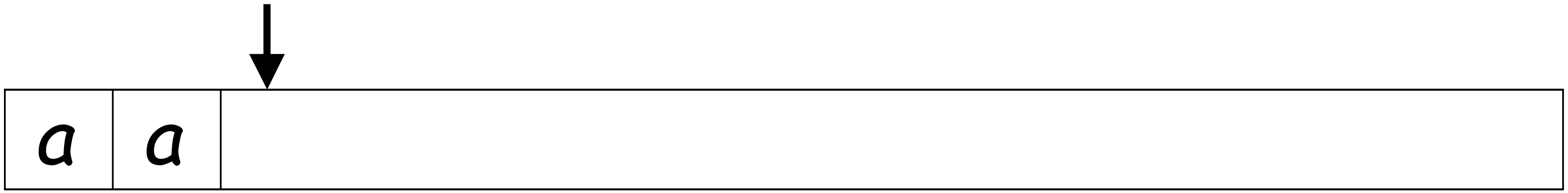
First Choice



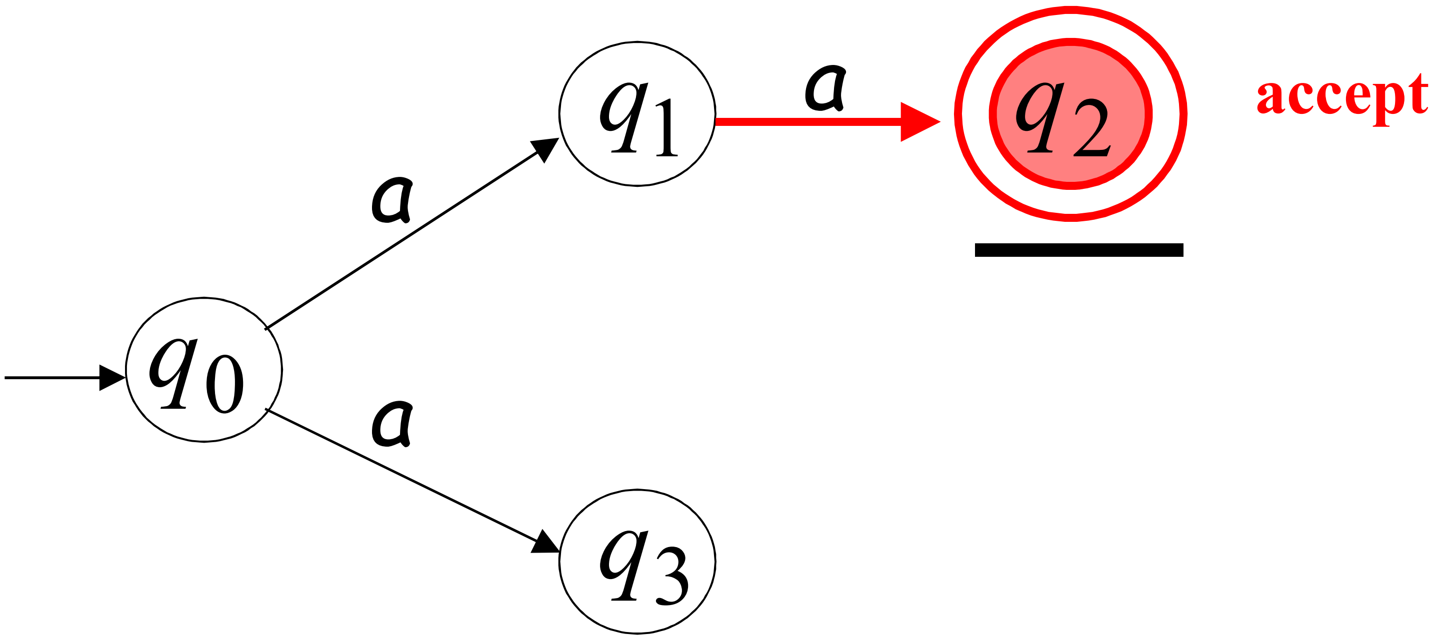
First Choice



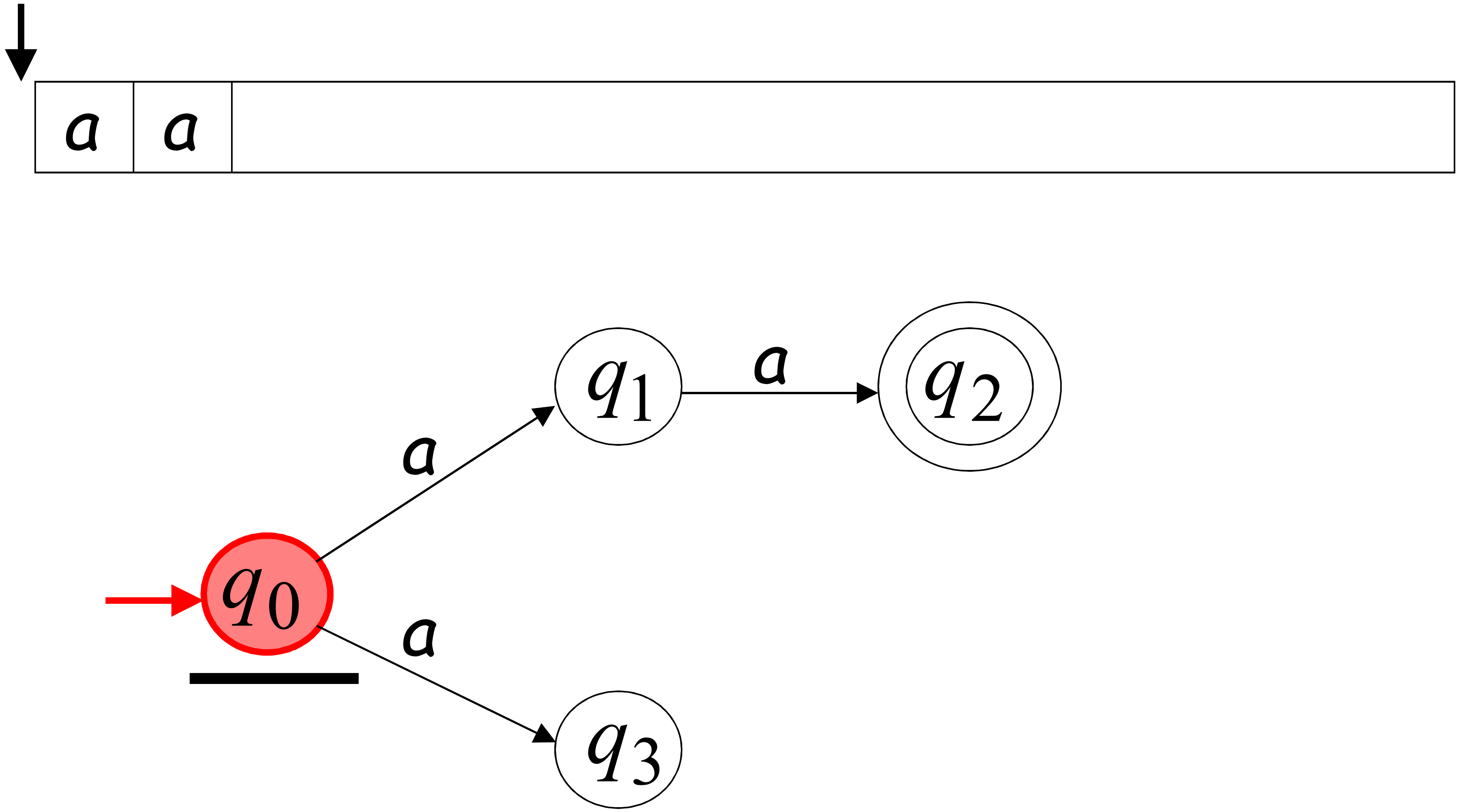
First Choice



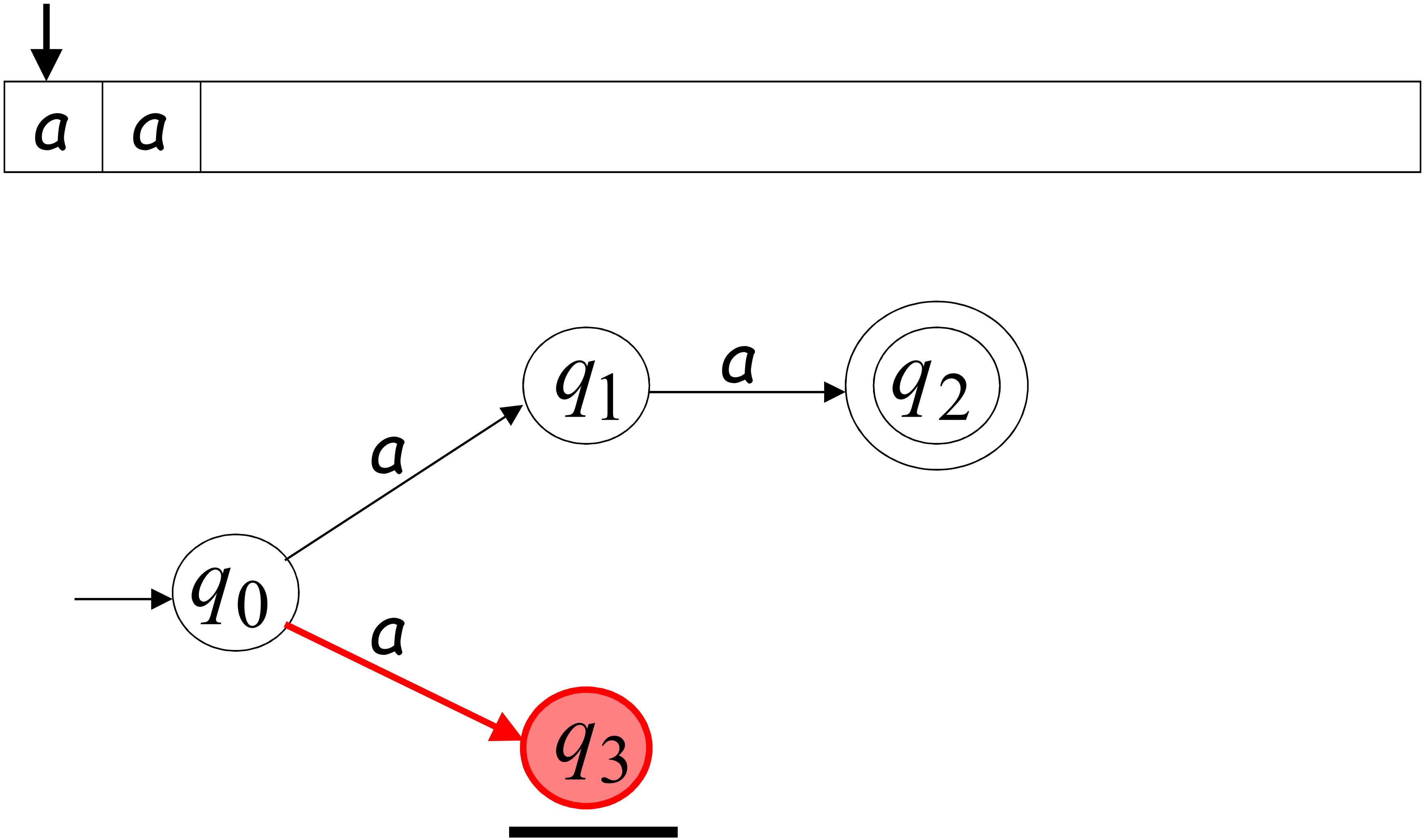
All input is consumed



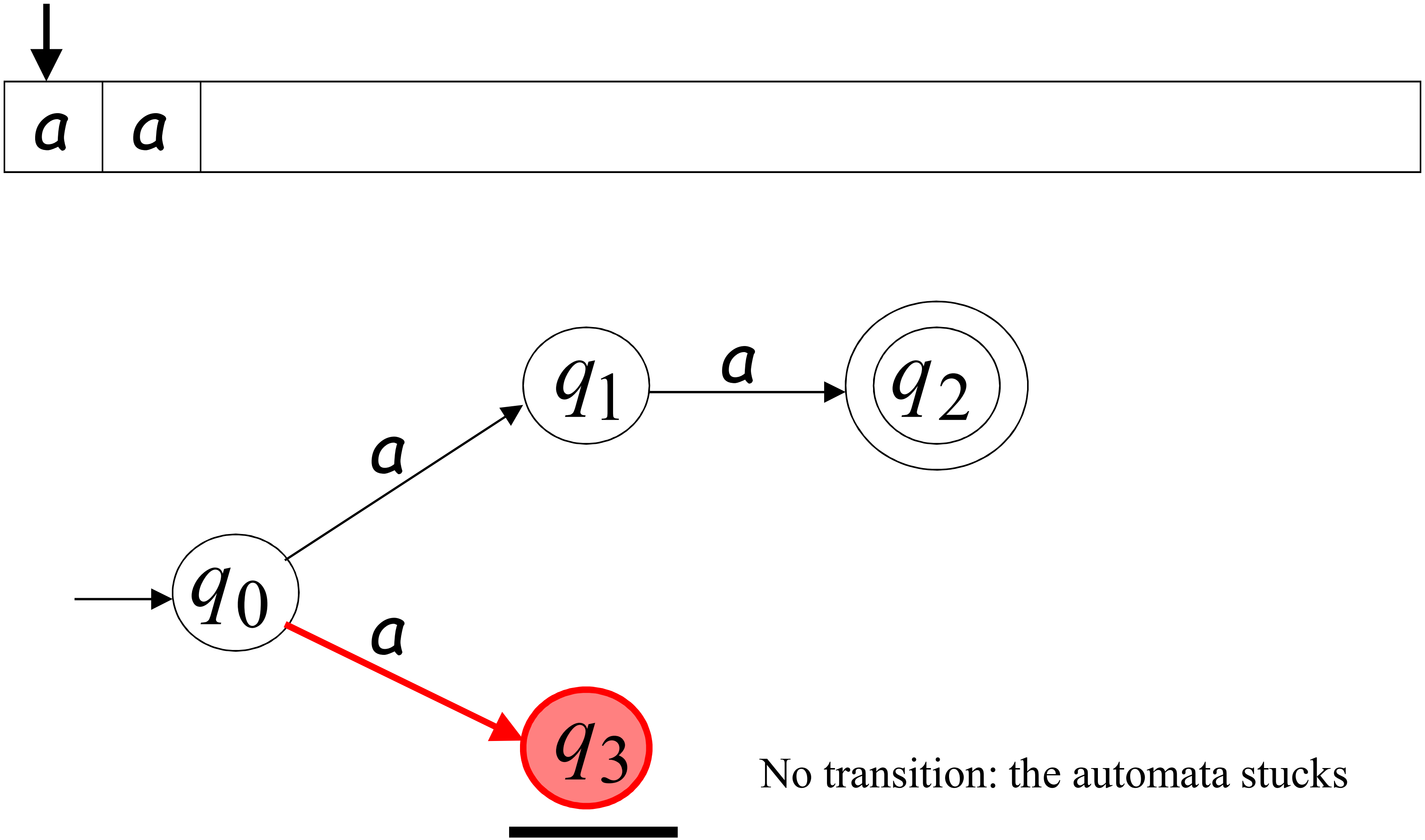
Second Choice



Second Choice

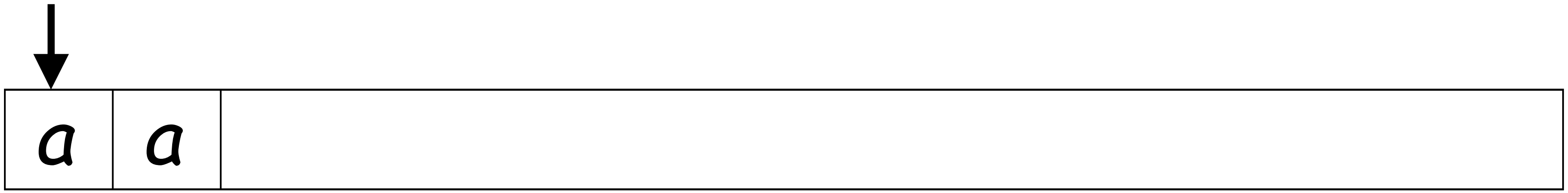


Second Choice

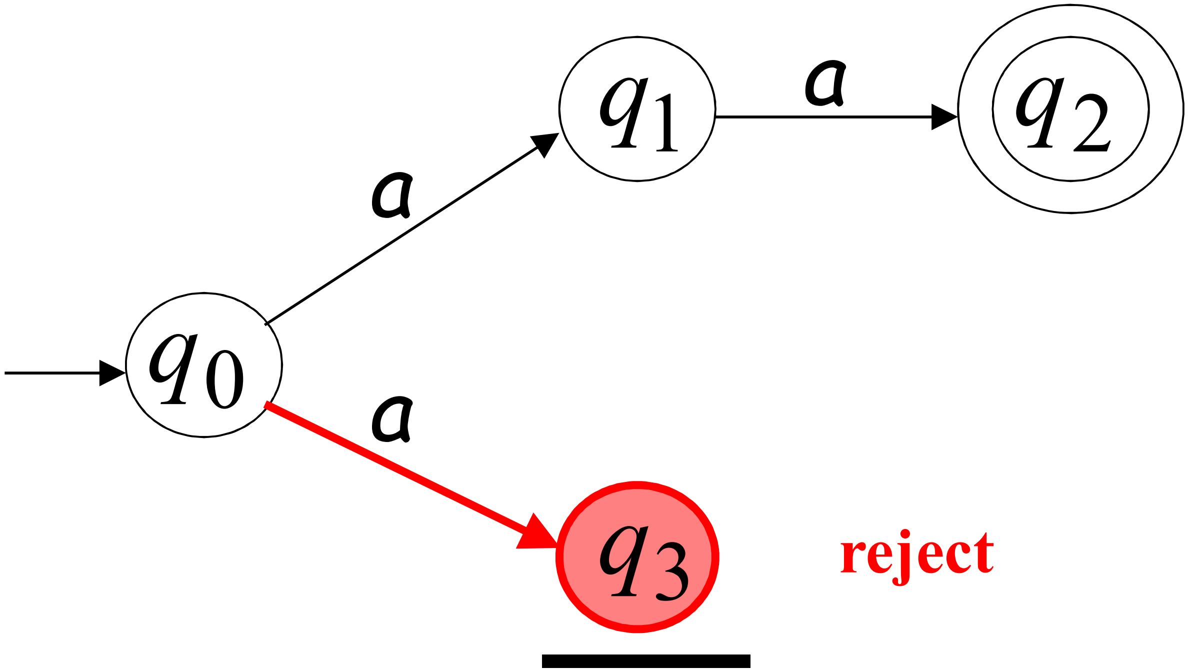


No transition: the automata stucks

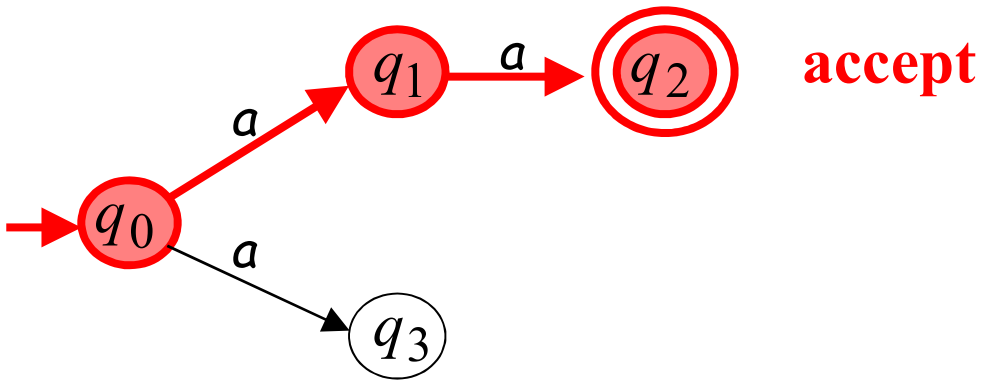
Second Choice

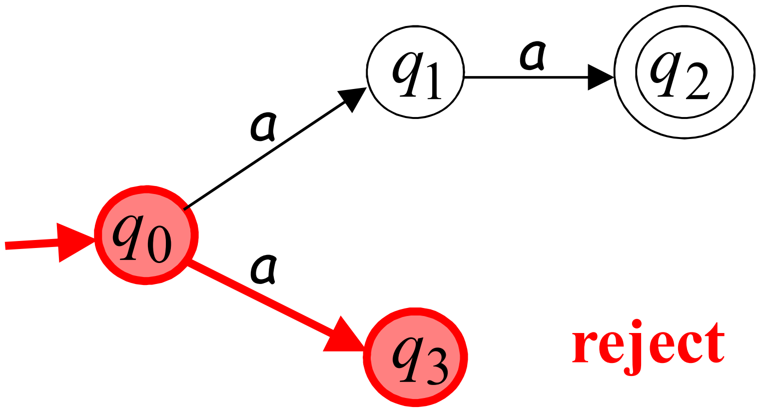


Input cannot be consumed



aa is accepted by the NFA:

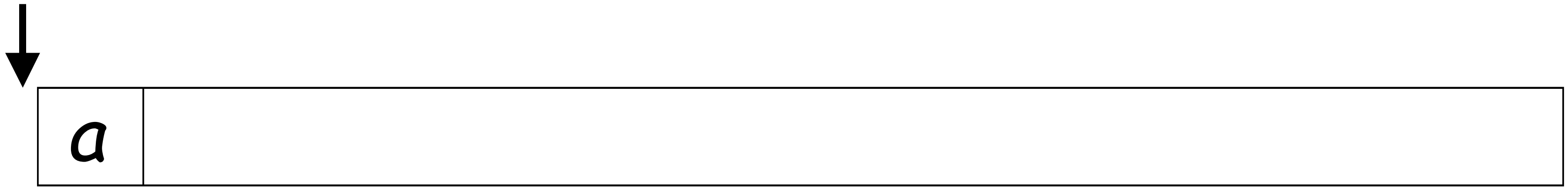


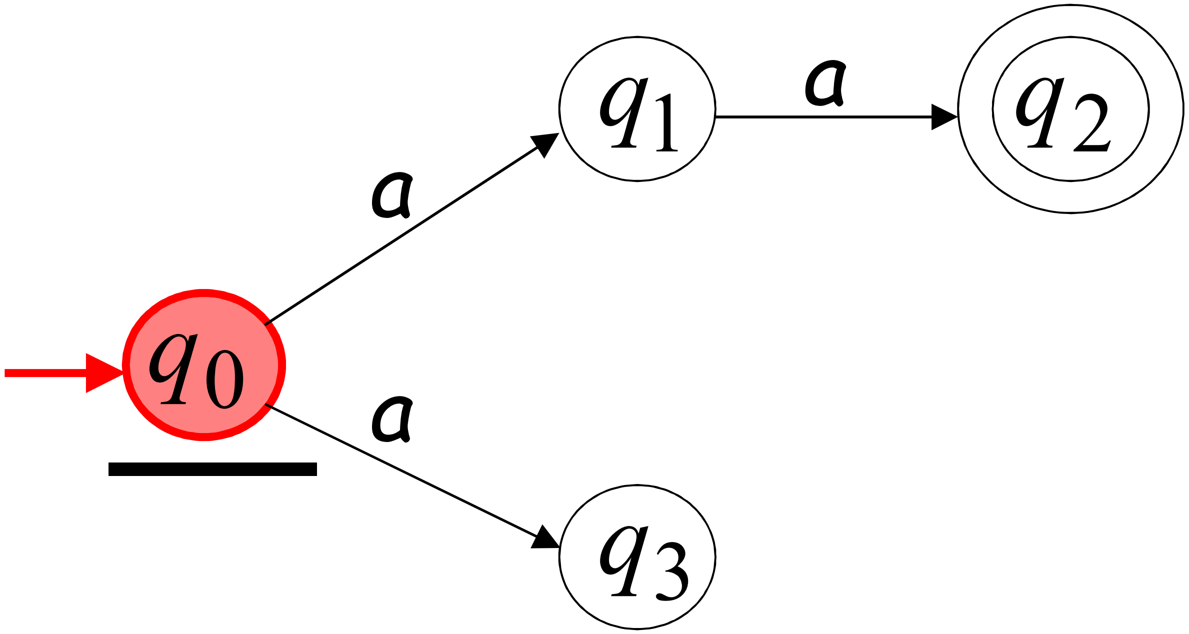


because this computation accepts **aa**

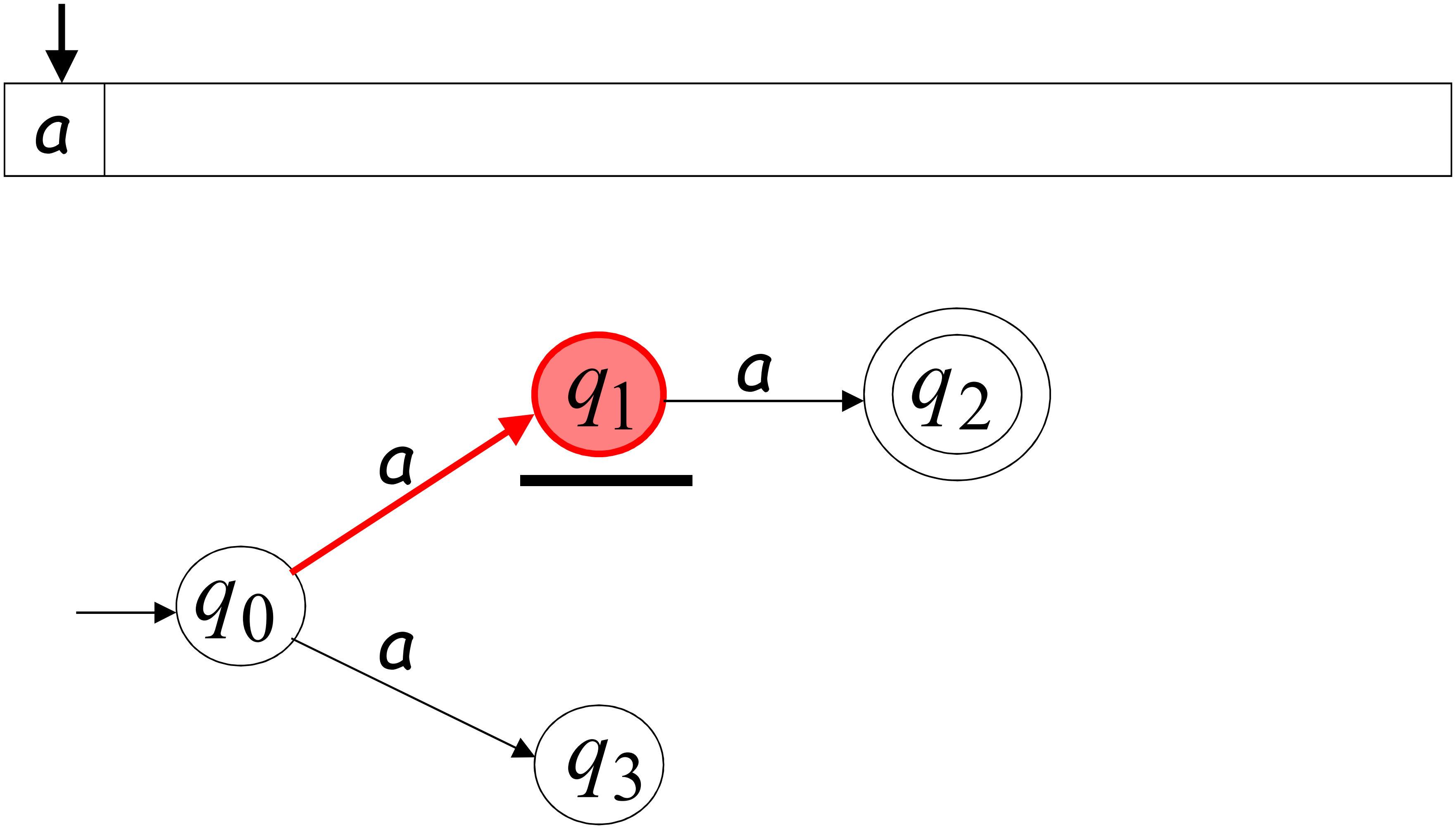
- A NFA accepts a string if all the input is consumed and the automata is in an accepting state

Rejection

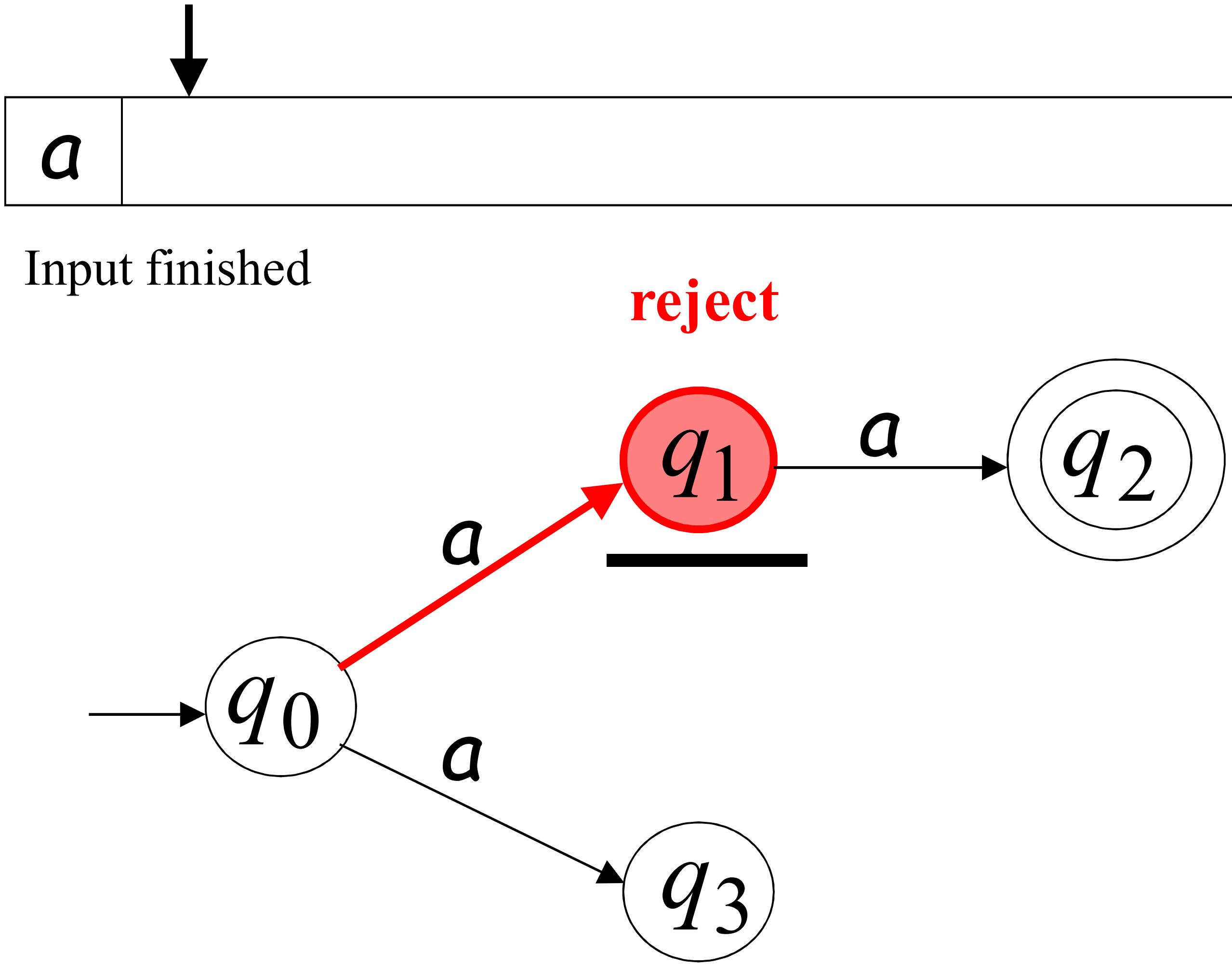




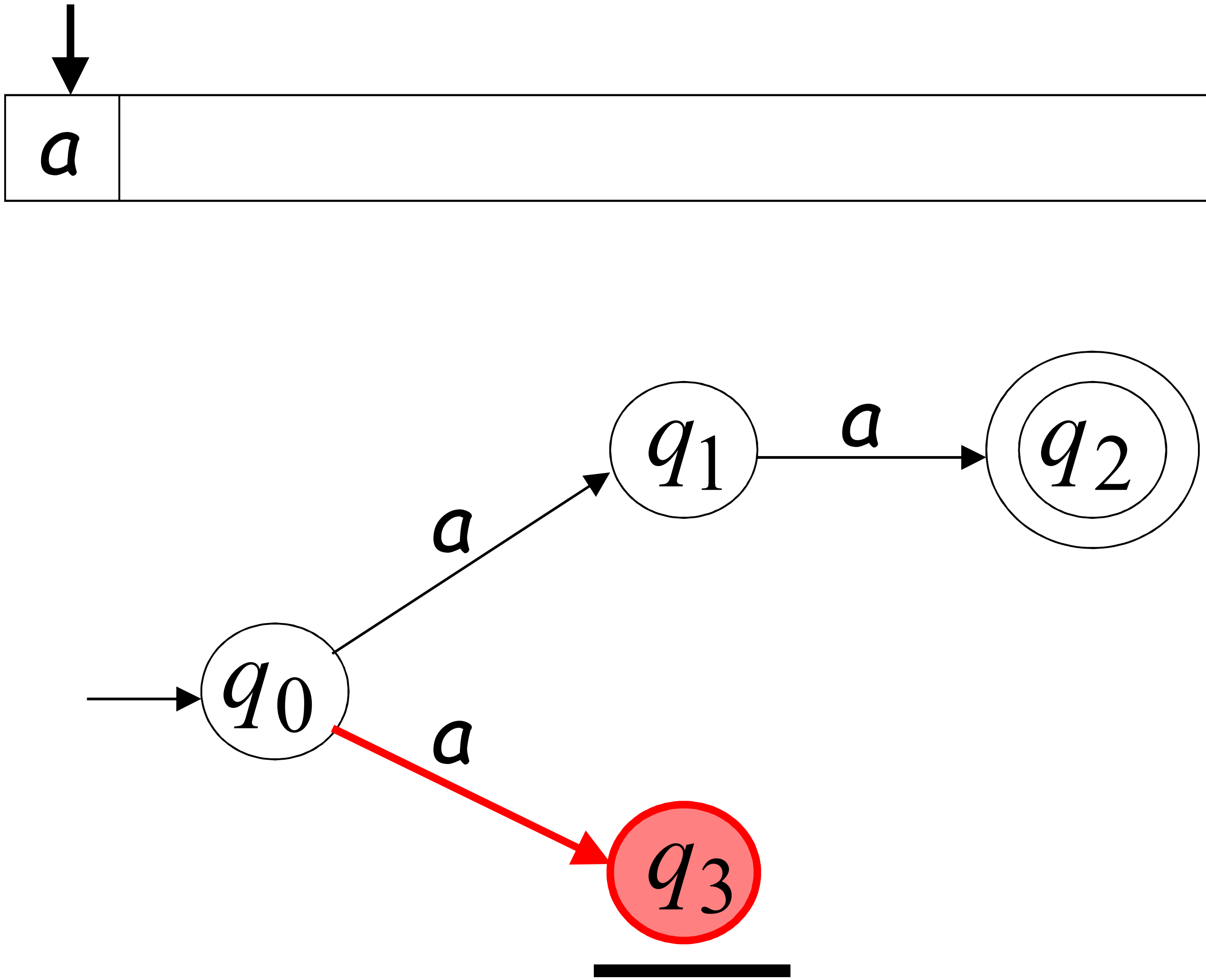
First Choice



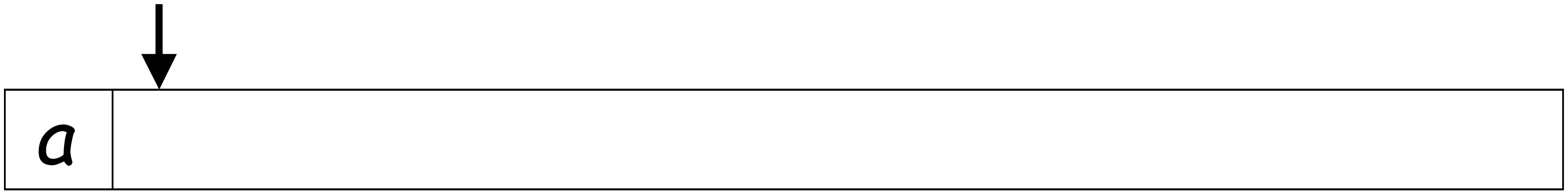
First Choice



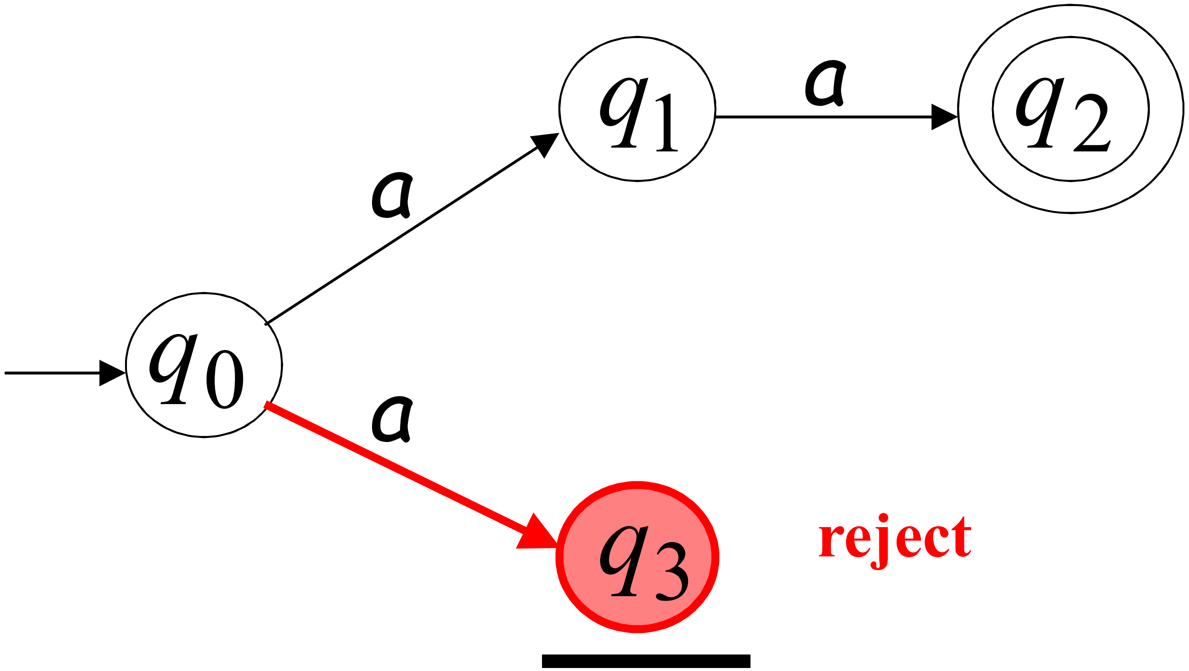
Second Choice



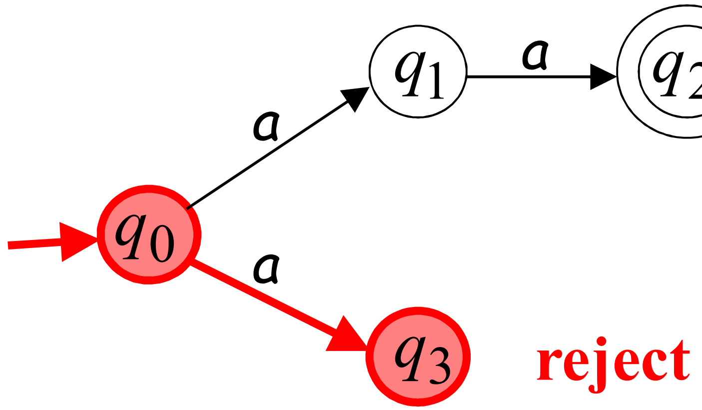
Second Choice

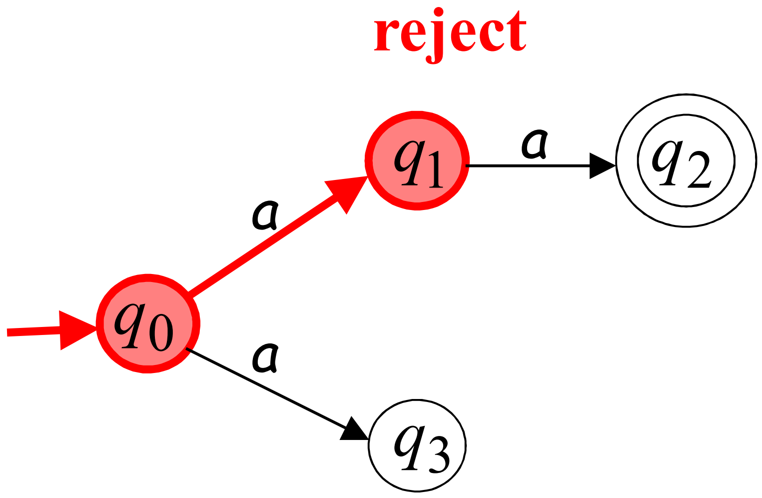


Input finished



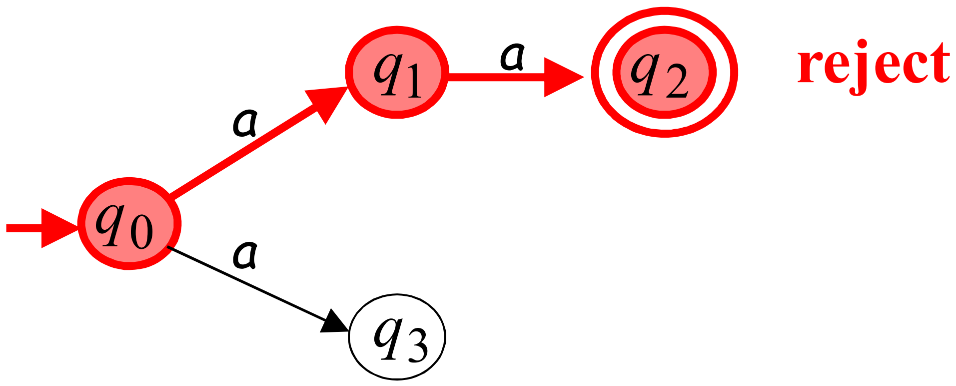
a is rejected by the NFA:

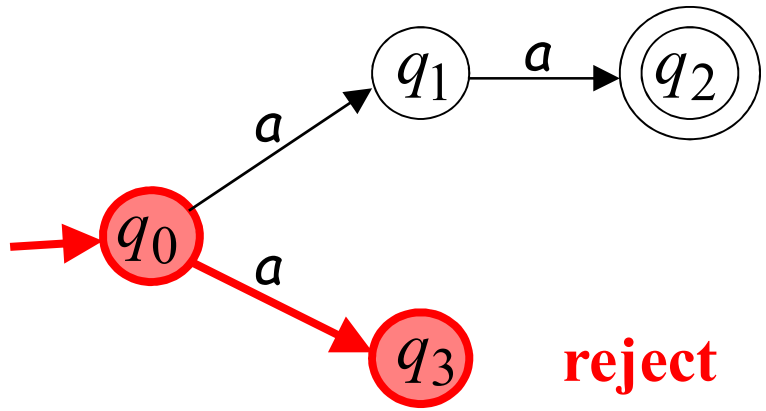




because all possible computations lead to rejection

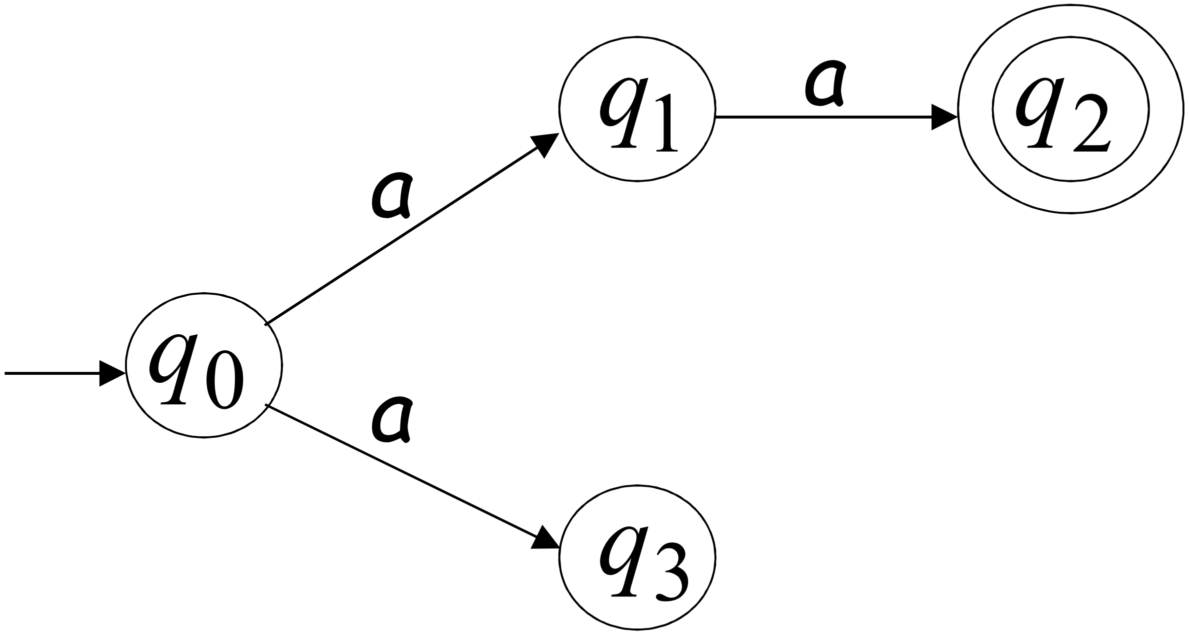
aaa is rejected by the NFA:



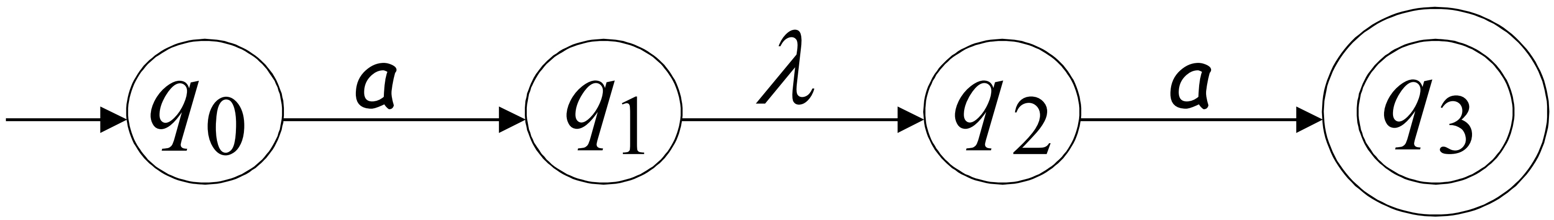


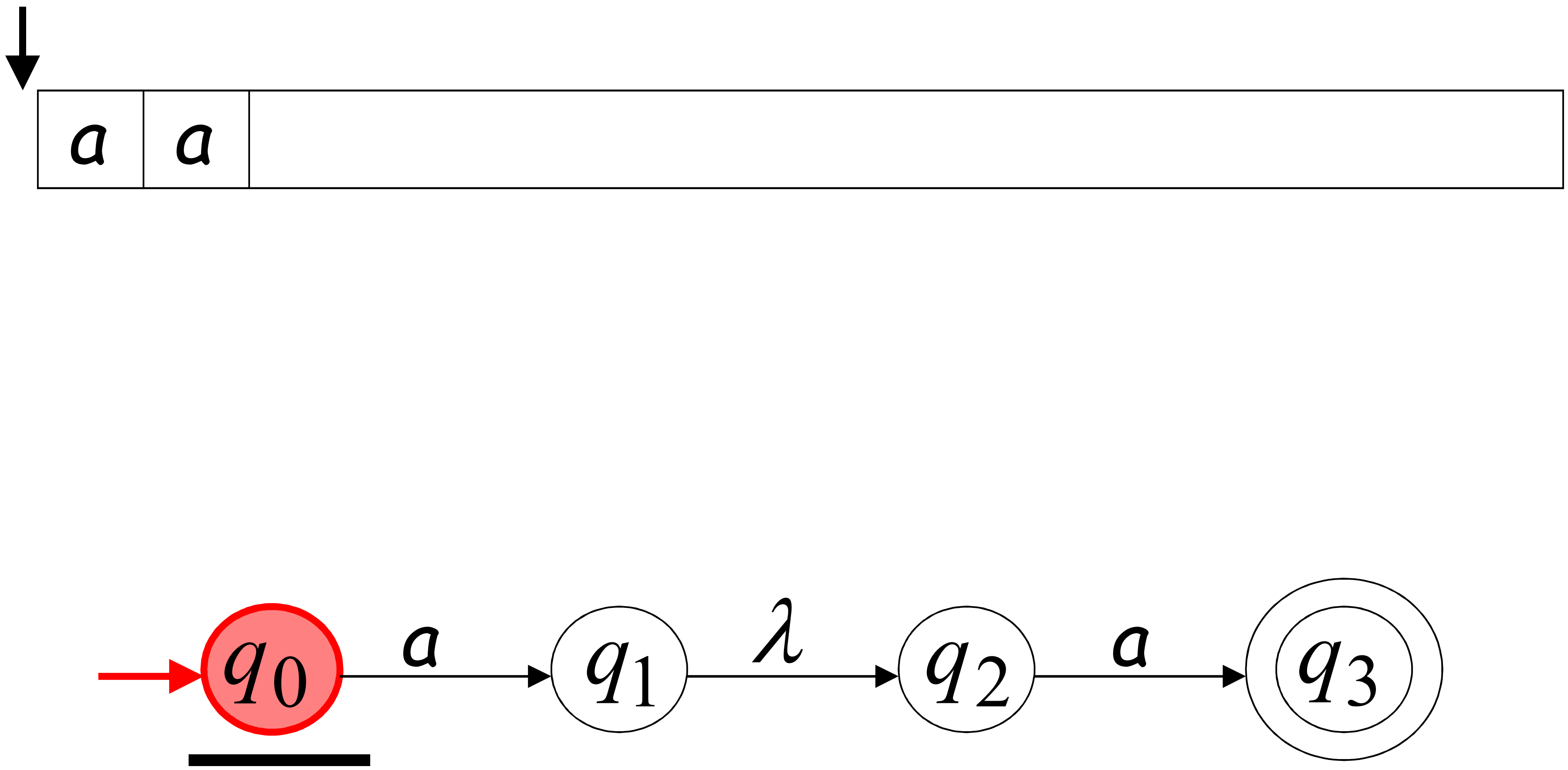
because all possible computations lead to rejection

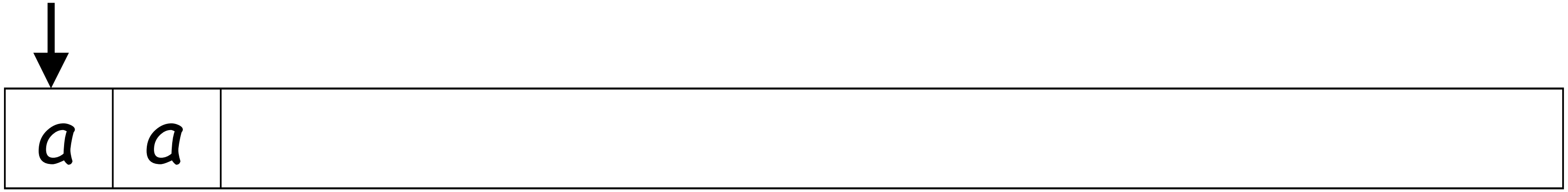
Language accepted: $L = \{aa\}$



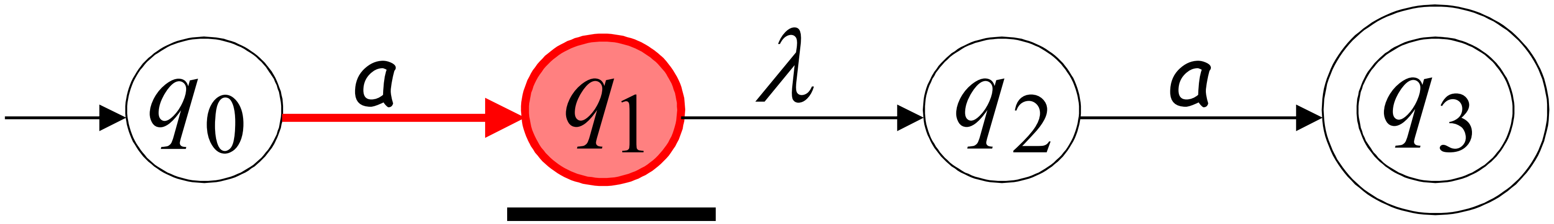
NFA – Example with ε (or λ) transitions




a
a
q0
a
q1
λ
q2
a
q3

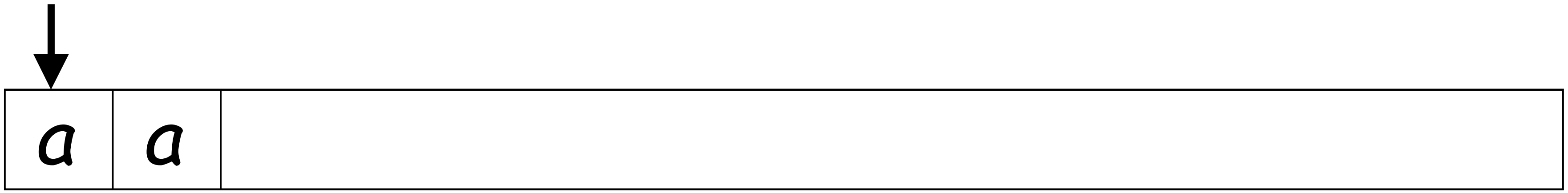


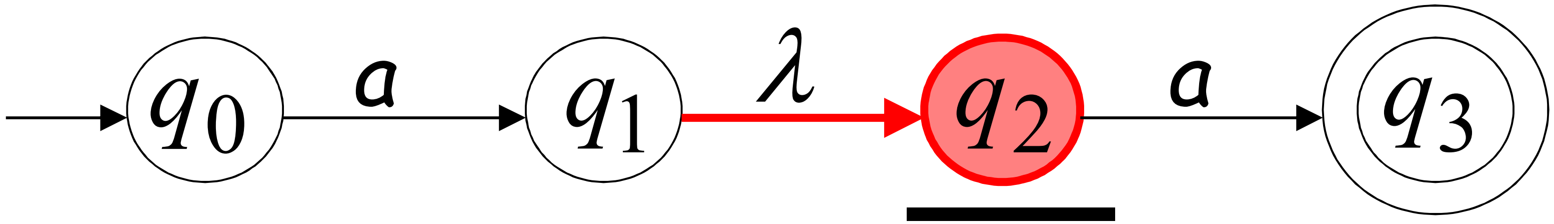
a
a

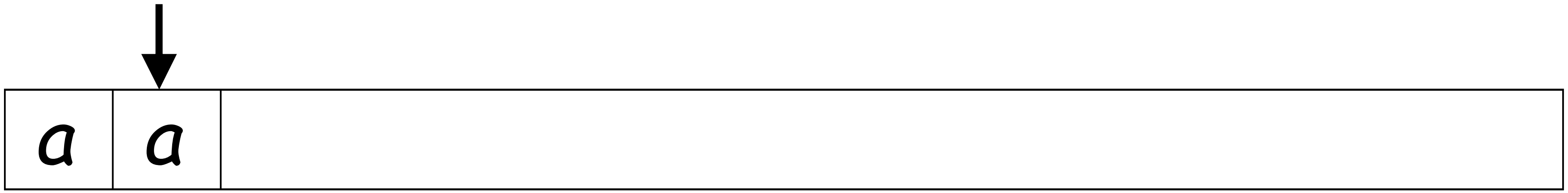


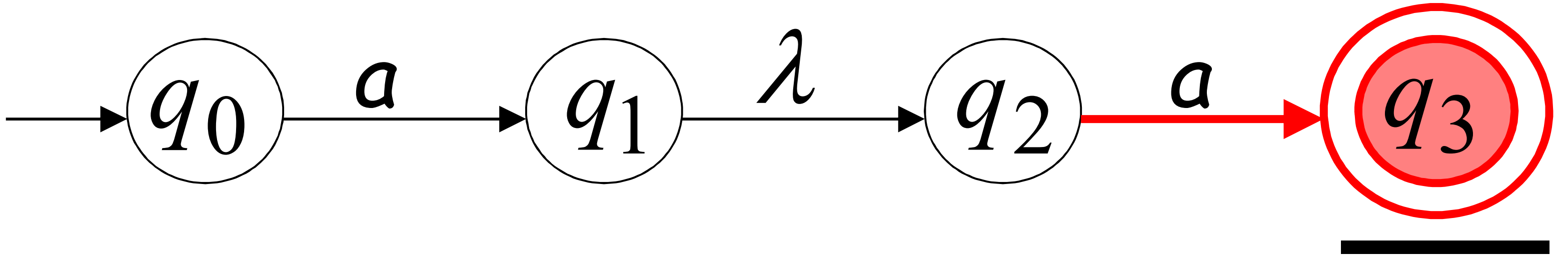
q_0
a
q_1
λ
q_2
a
q_3

(read head does not move)

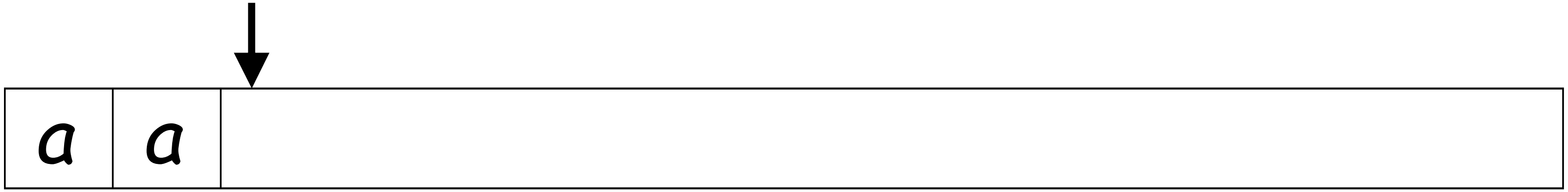


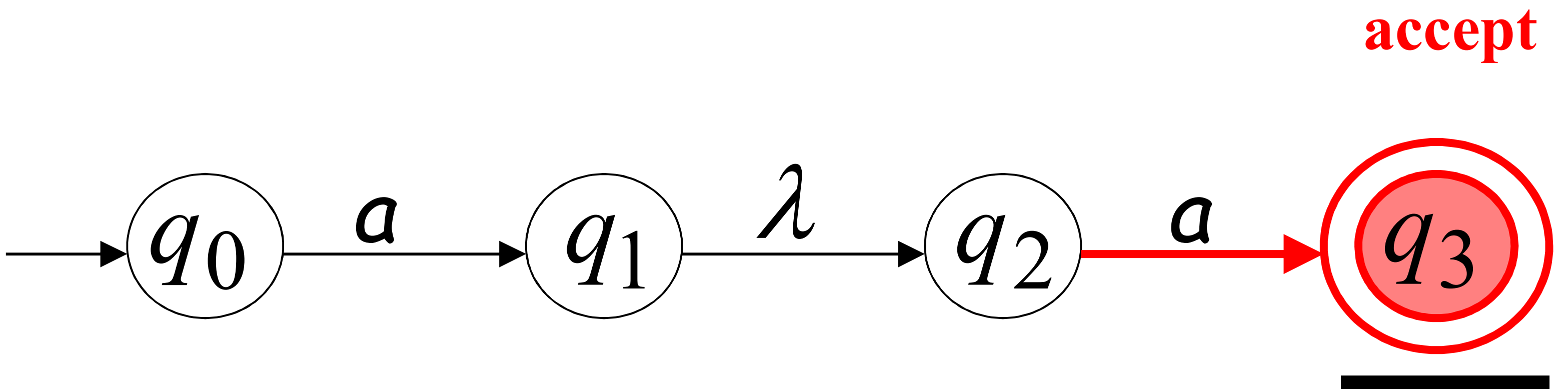



a
a


q_0
a
q_1
λ
q_2
a
q_3

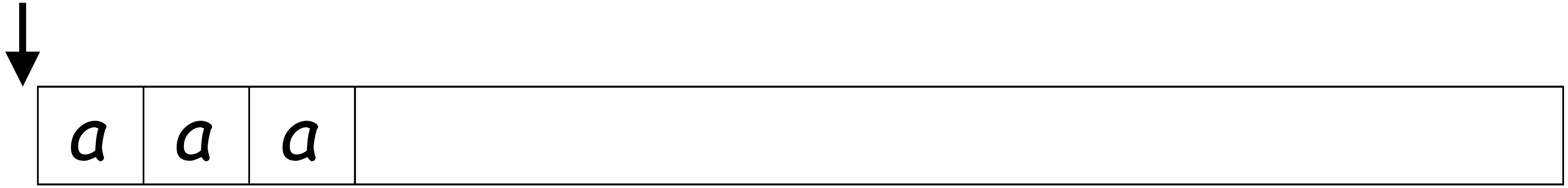
all input is consumed

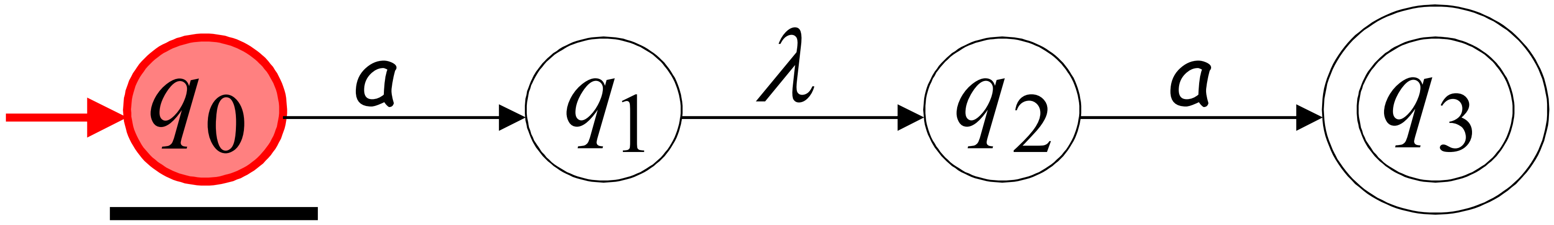




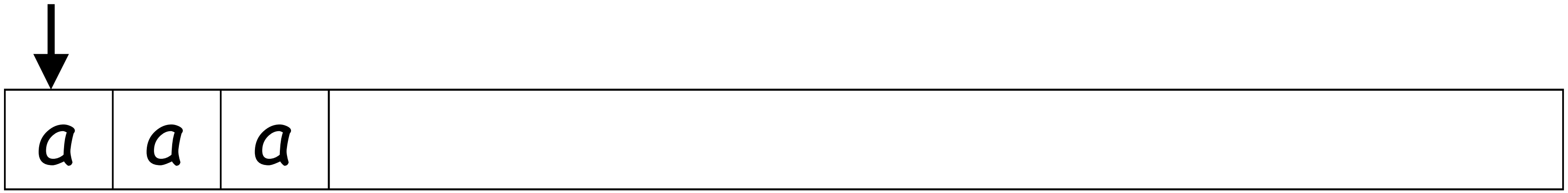
String **aa** is accepted

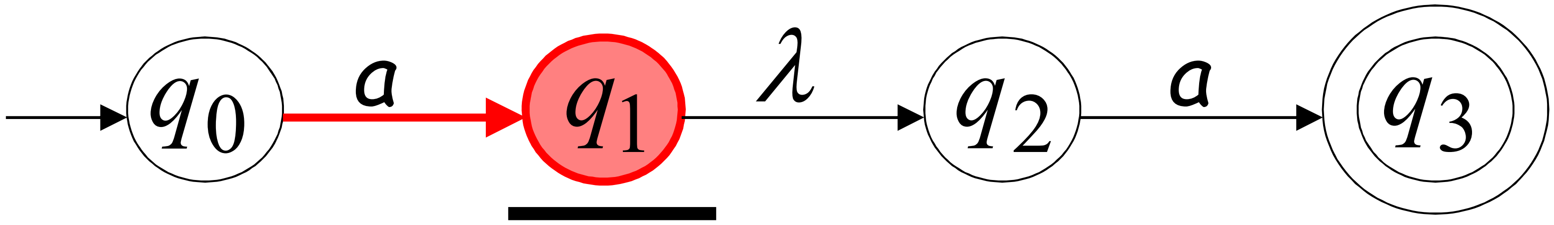
Rejection



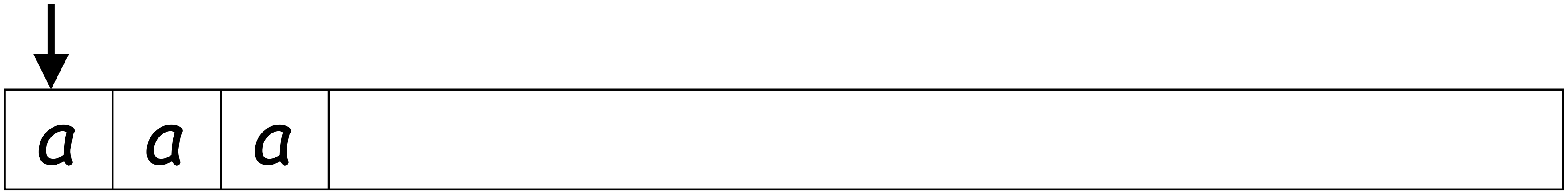


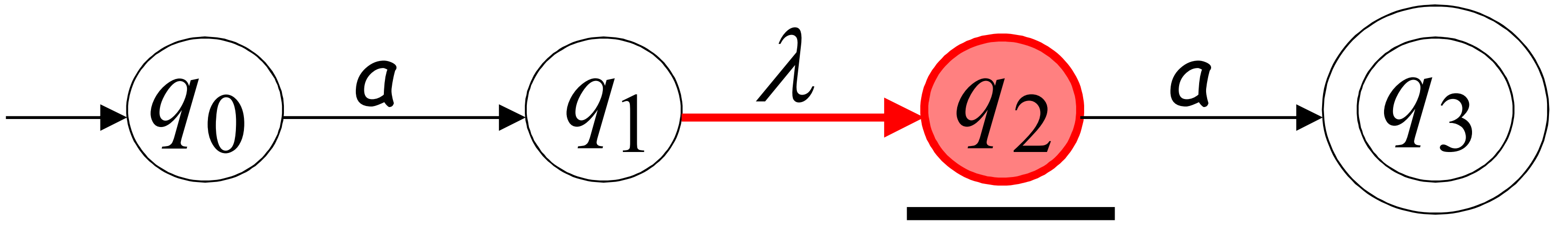
Rejection

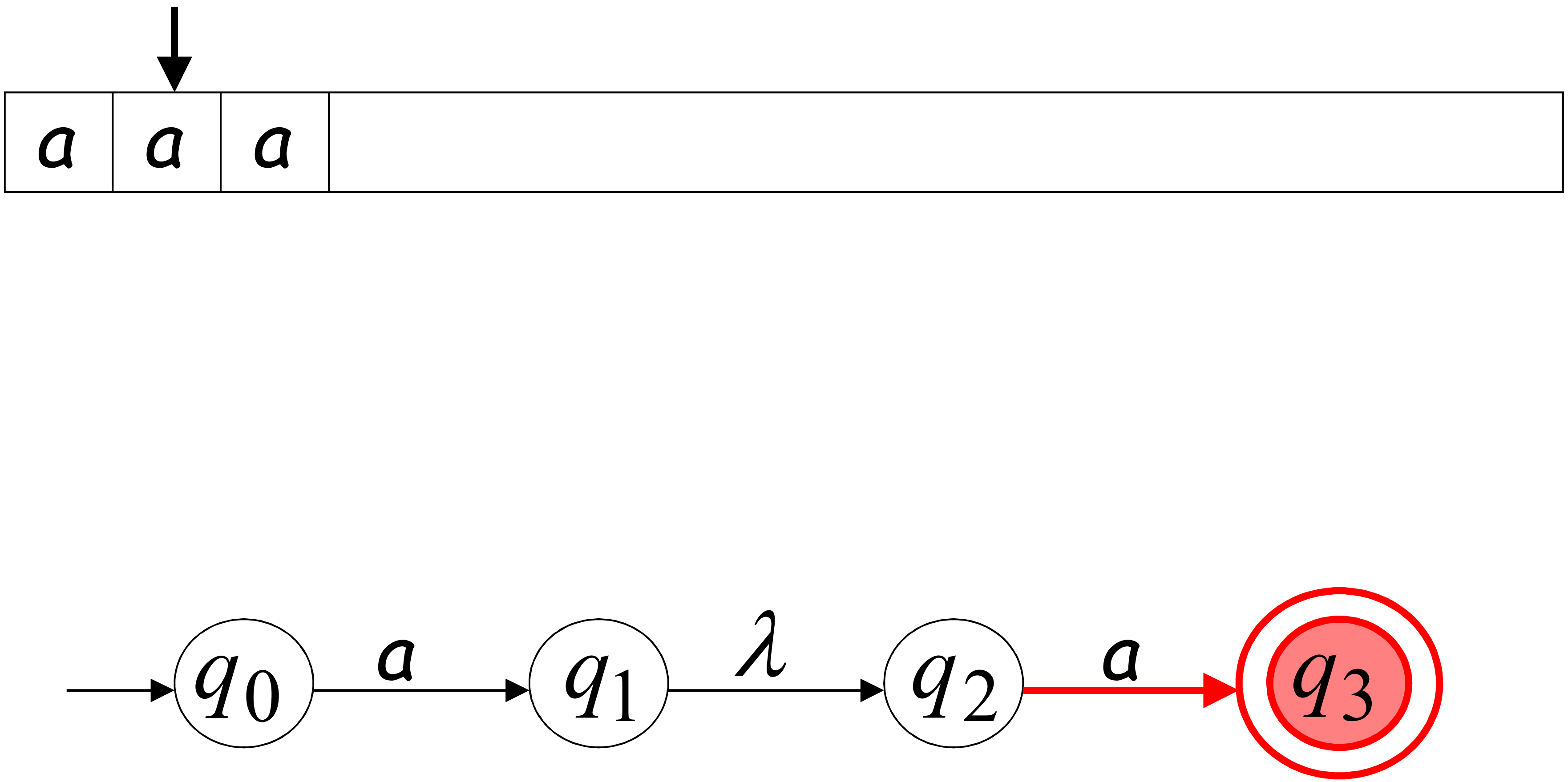




(read head does not move)

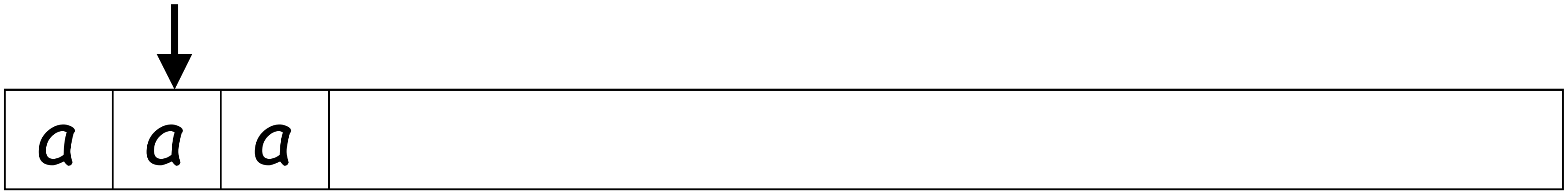


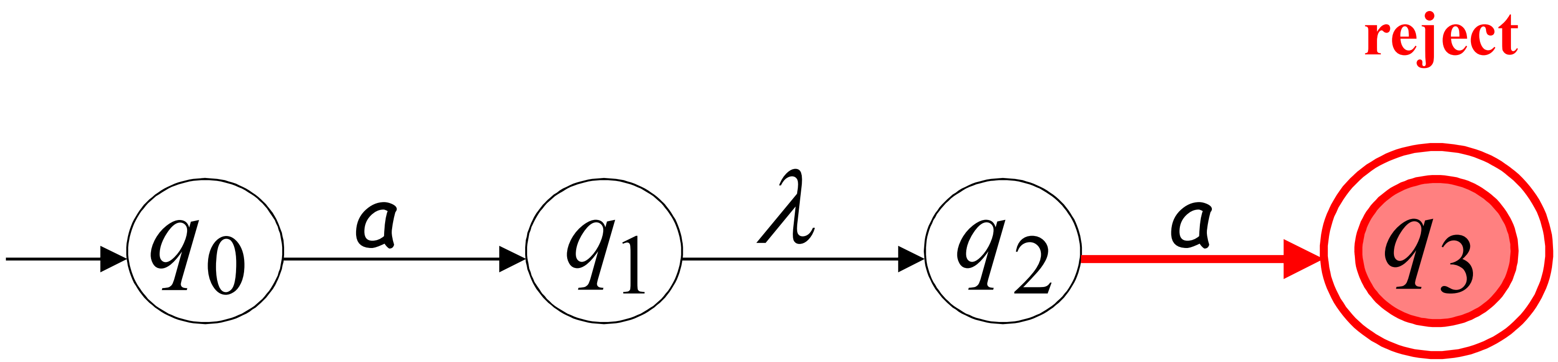




No transition: the automata stucks

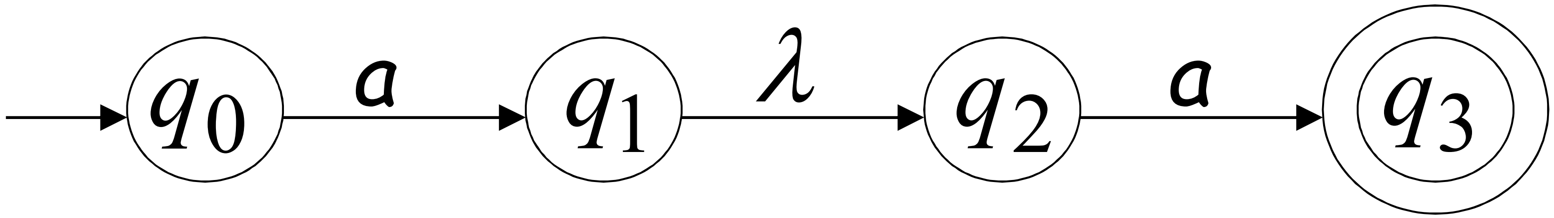
Input cannot be consumed



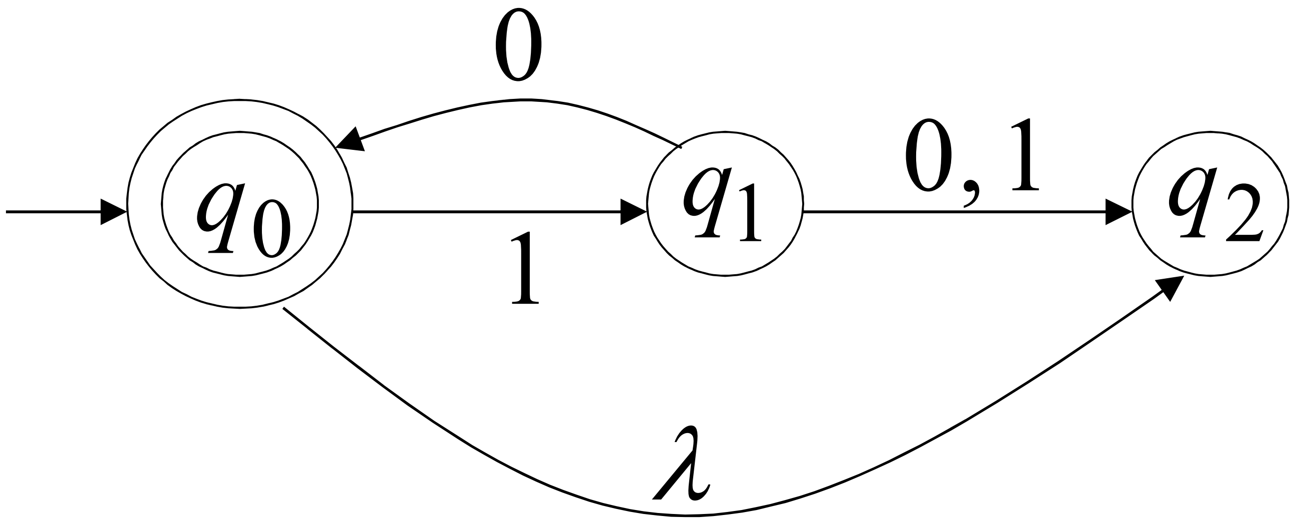


String **aaa** is rejected

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



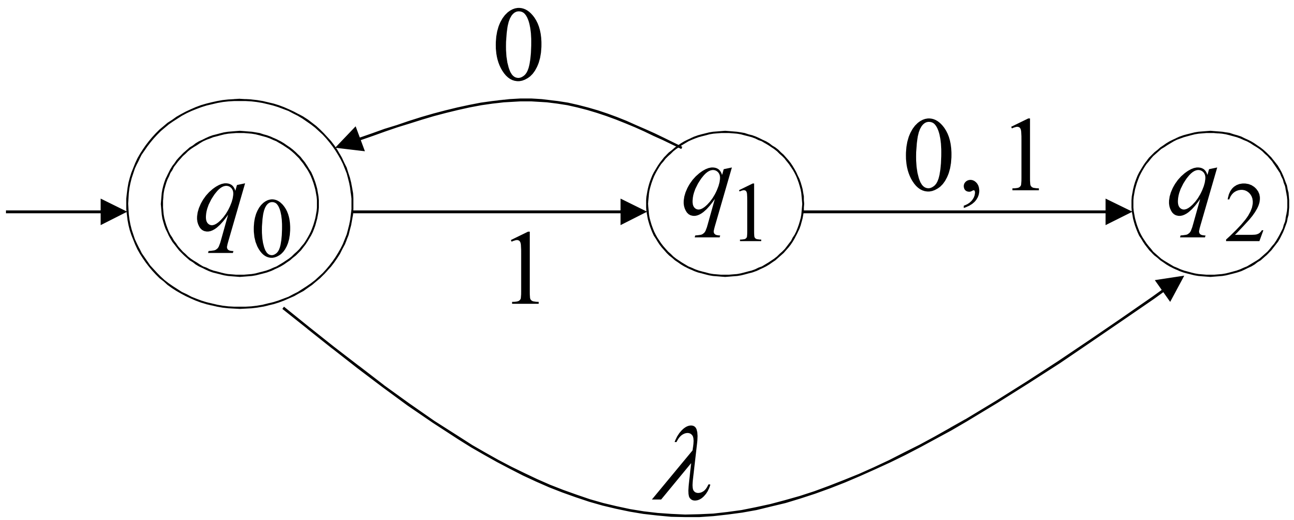
Another NFA Example



Another NFA Example

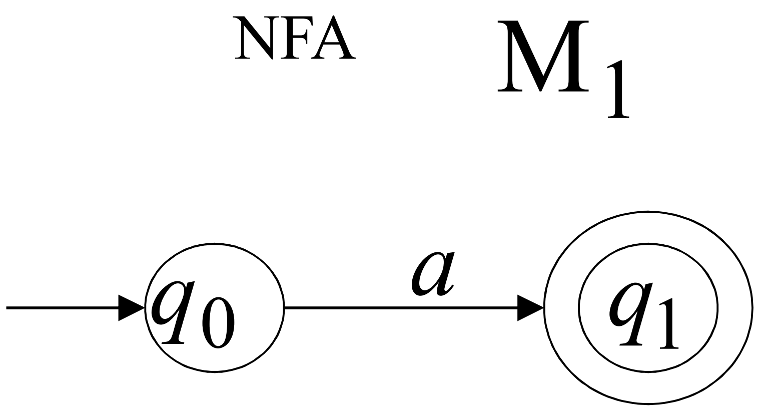
Language accepted

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

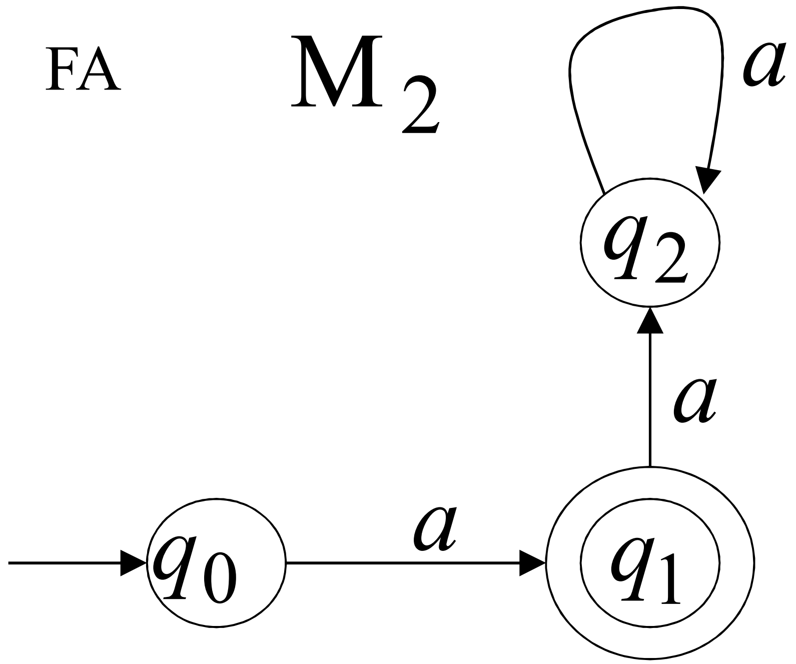


(redundant state)

NFAs are interesting because we can express languages easier than FAs



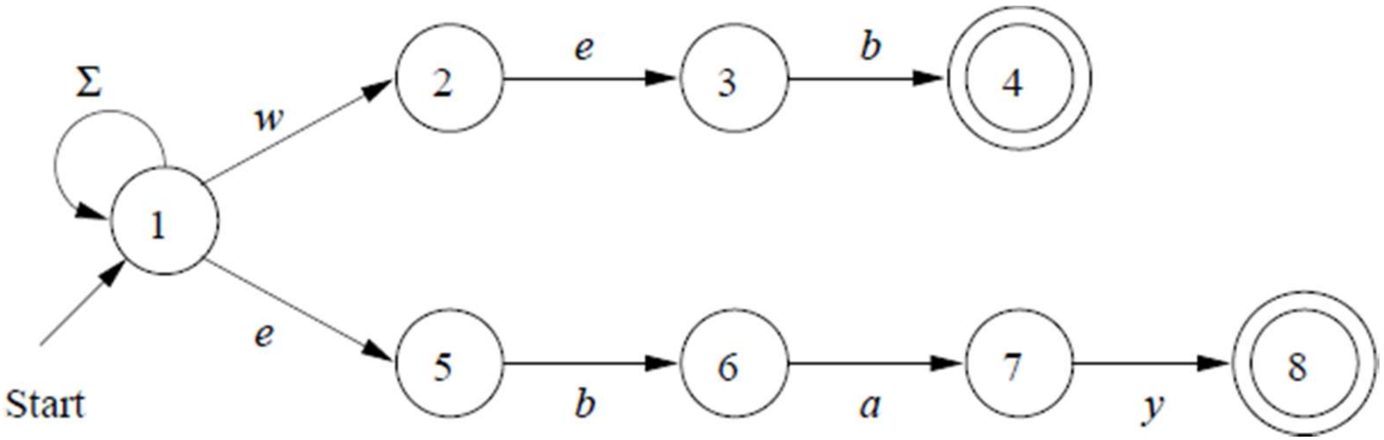
$$L(M_1) = \{a\}$$



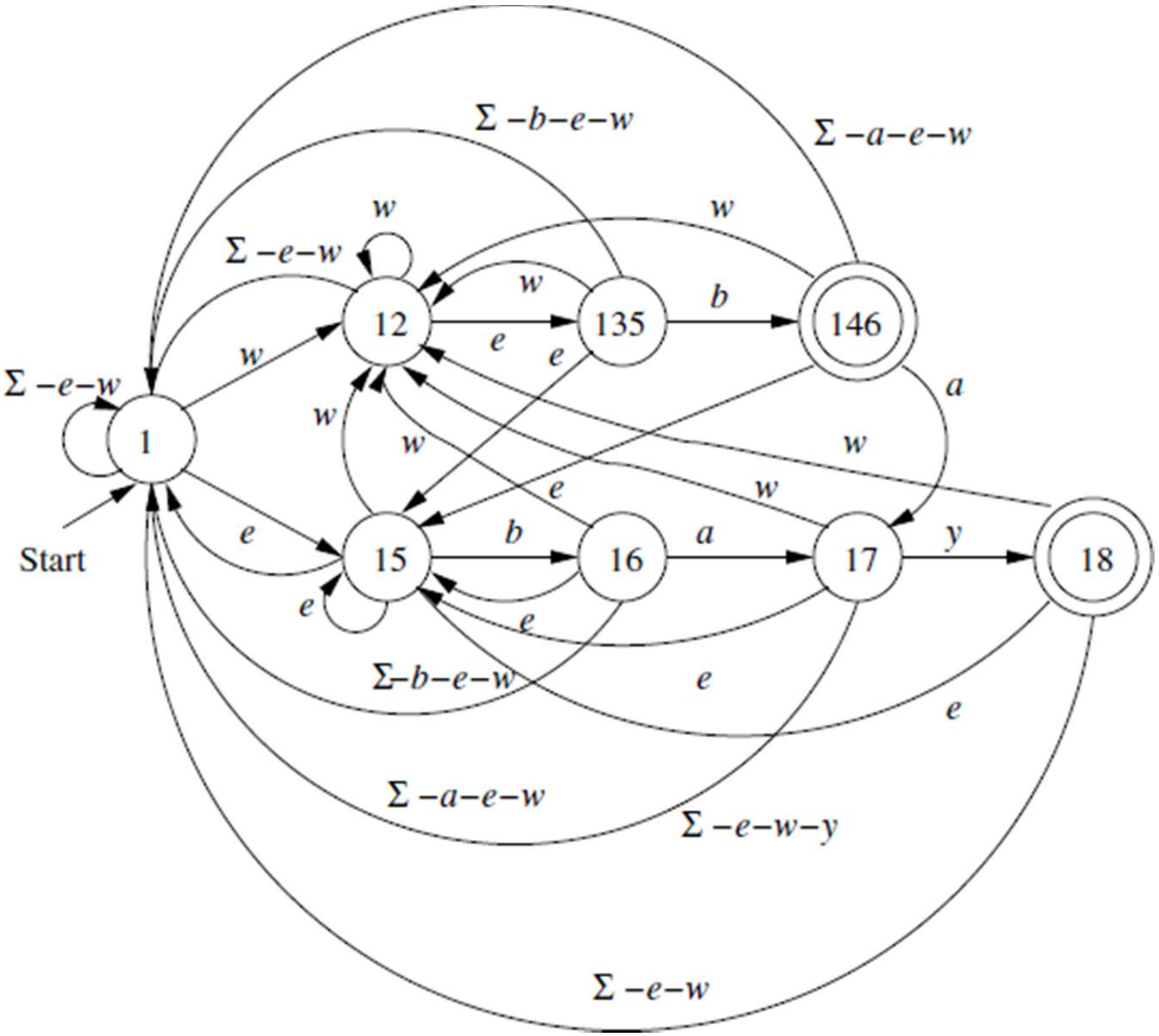
$$L(M_2) = \{a\}$$

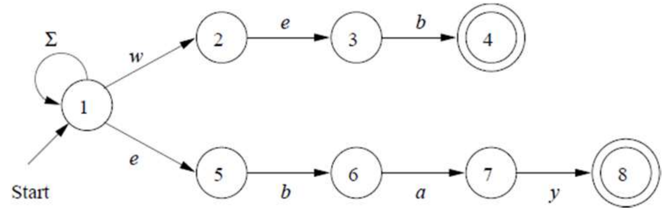
NFA for Text Search

- An NFA accepting the set of words ending with **ebay** or **web**



Corresponding DFA for Text Search





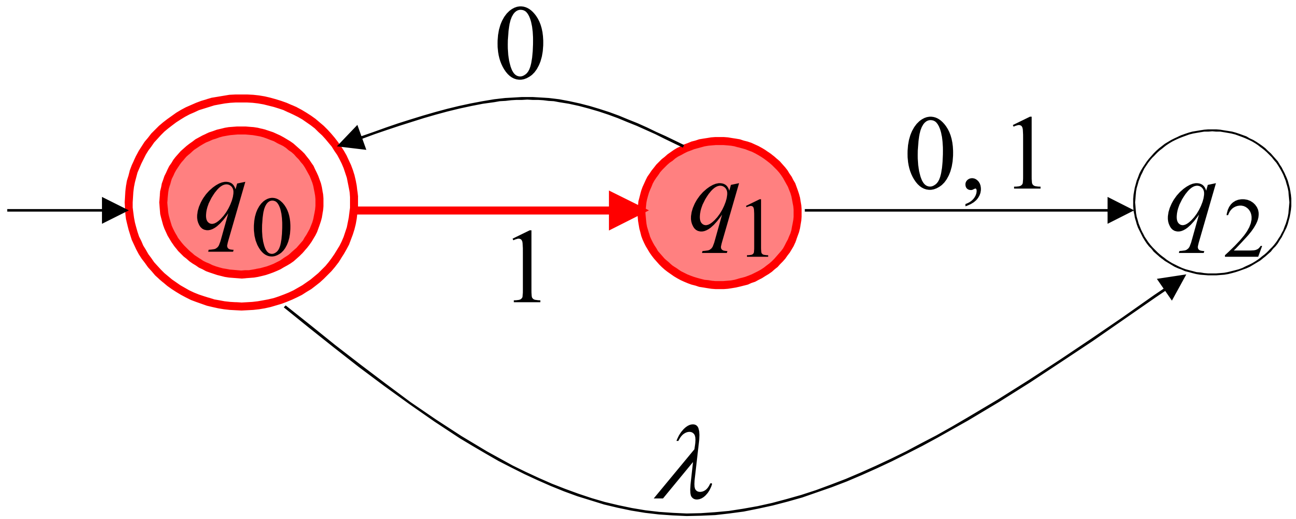
Formal Definition of NFA

- A **Nondeterministic Finite Automaton (NFA)** is a 5-tuple **(Q, Σ, δ, q_0, F)**
  1. **Q is a finite set of states**
  2. **Σ is a finite set of symbols (alphabet)**
  3. **Delta (δ) is a transition function from** $\delta: Q \times \underbrace{\Sigma_{\varepsilon}}_{\Sigma \cup \{\varepsilon\}} \rightarrow \underbrace{\boldsymbol{\mathcal{P}}(Q)}_{\text{power set}} = \{R | R \subseteq Q\}$
  4. **q_0 is the start state ($q_0 \in Q$)**
  5. **F is a set of final (accepting) states (F $\subseteq$ Q)**

- Transition function takes two arguments: a state and an input symbol or ε.
- δ(q,a) = the set of the states that the NFA goes to when it is in state **q** and **a** is received.
  - where **a** is an input symbol or ε.

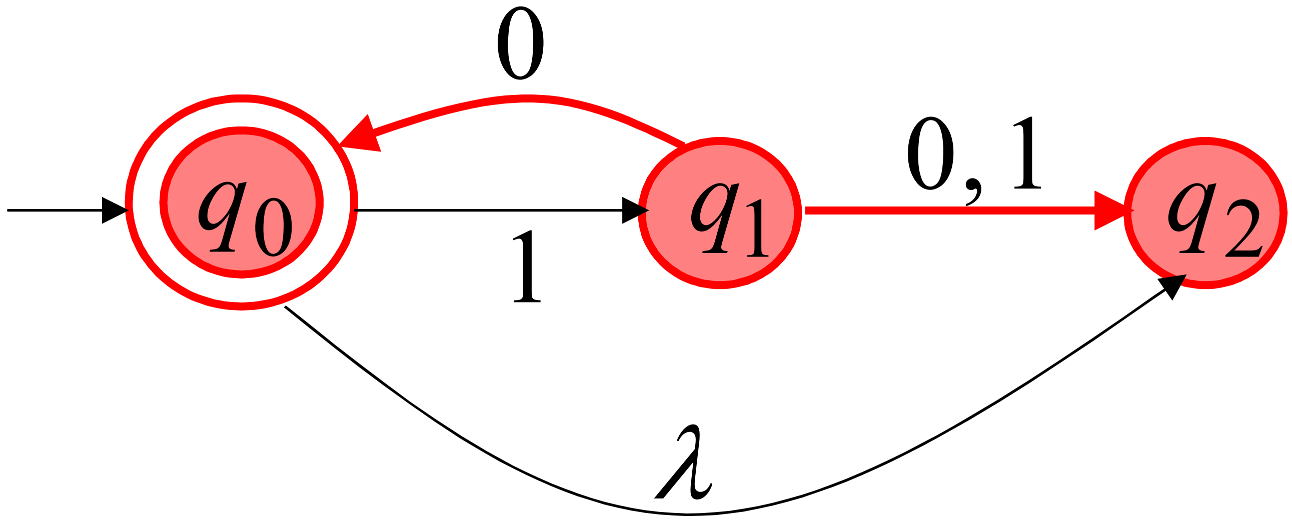
Transition Function - δ

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



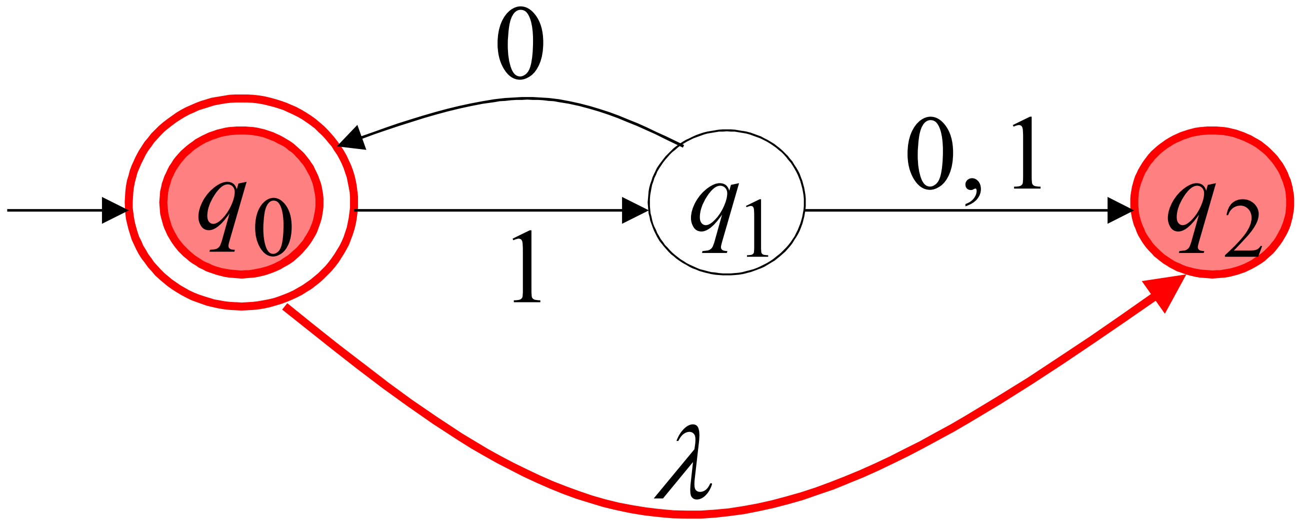
Transition Function - δ

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



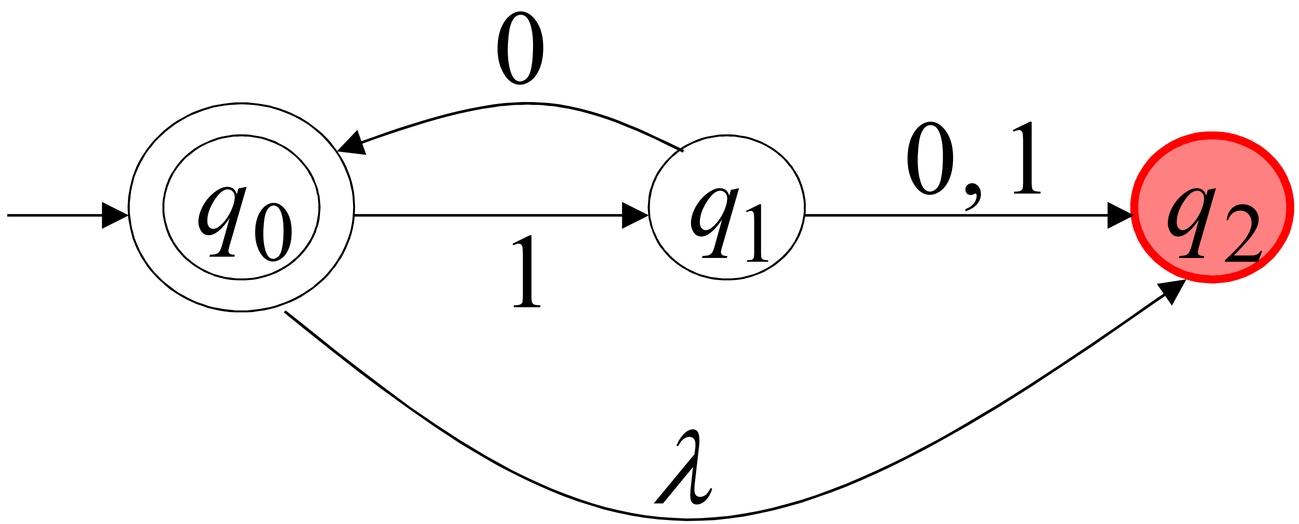
Transition Function - δ

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



Transition Function - δ

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



Language of a NFA

- Formally, the language accepted by a NFA *A* is:

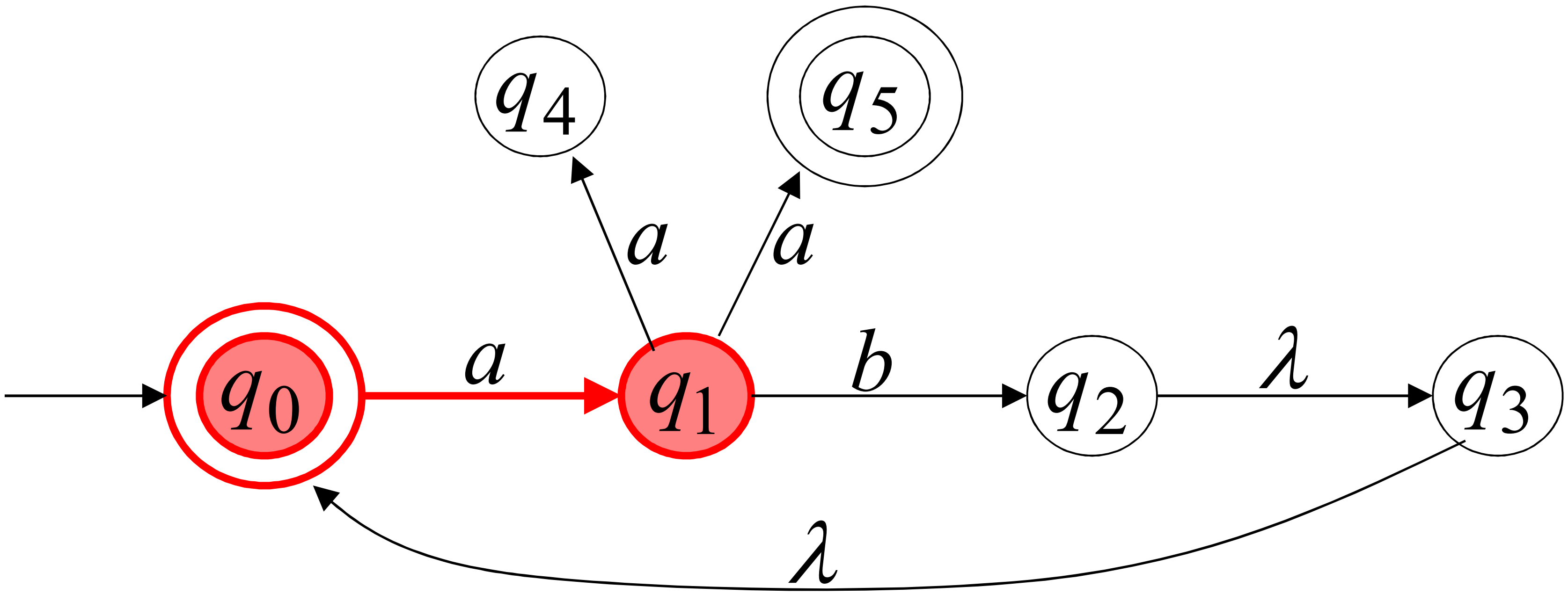
$$L(A) = \{ w \mid \hat{\delta}^{*}(q_0,w) \cap F \neq \phi \}$$

- The transition function delta is extended delta star that operates on states and strings (as opposed to states and symbols).

- A string *w* is accepted by a NFA *A* iff the states that are reachable from the starting state by consuming *w* contain at least one final state.

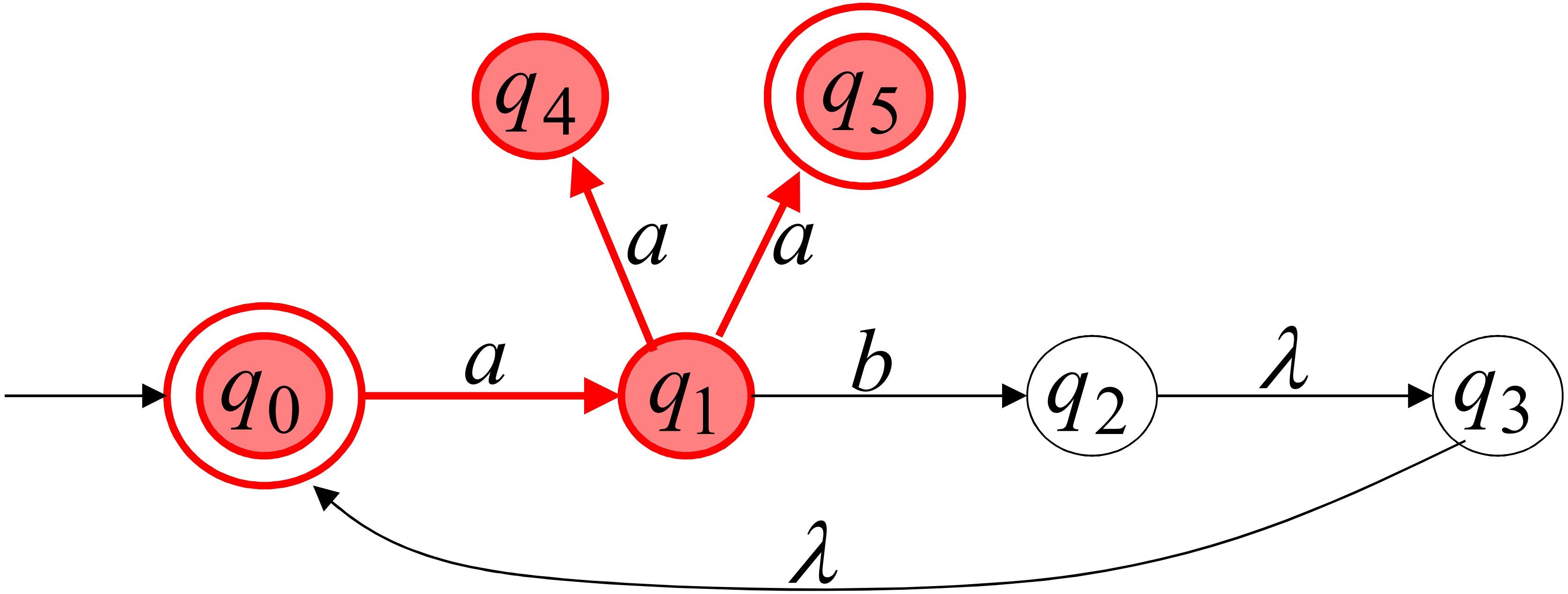
Extended Transition Function - δ^*

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



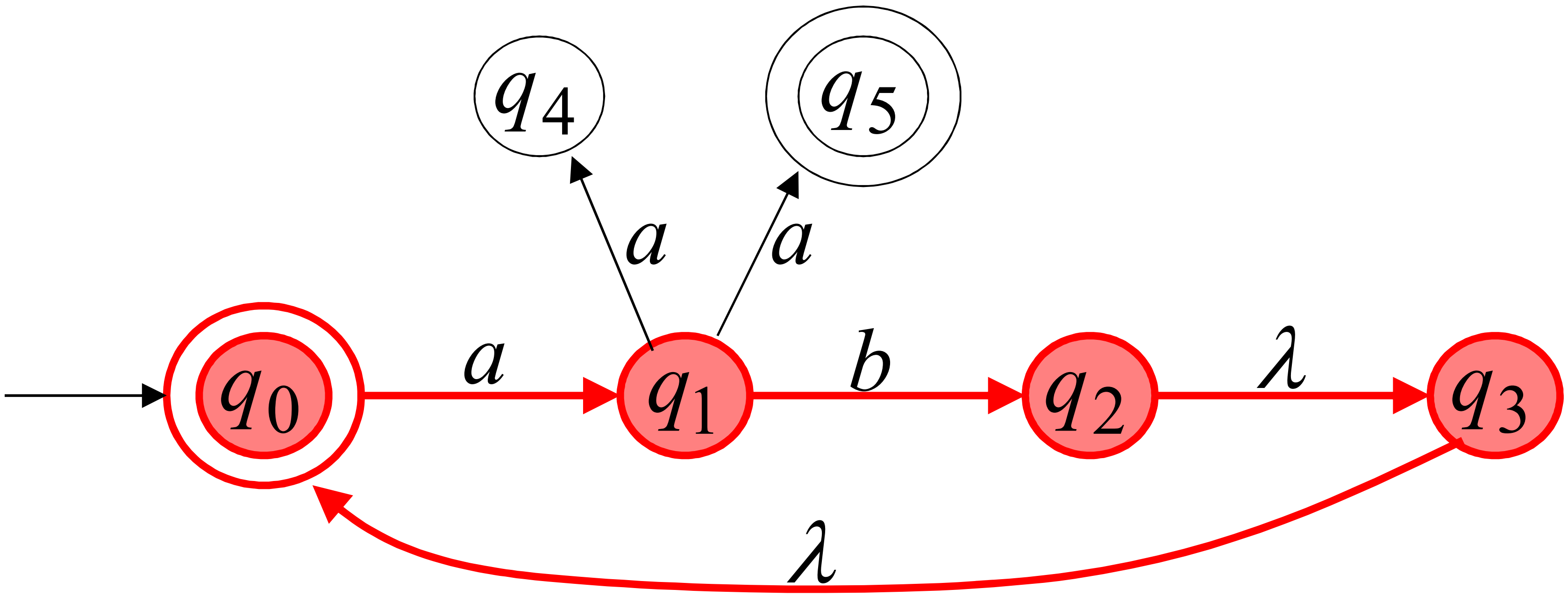
Extended Transition Function - δ^*

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



Extended Transition Function - δ^*

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



Epsilon Closure

- We **close a state** by **adding all states reachable by a sequence** εε**…**ε **.**
- **ECLOSE(q)** is the epsilon closure of the state q.

Epsilon Closure

ECLOSE(1) = {1,2,3,4,6}

ECLOSE(2) = {2,3,6}

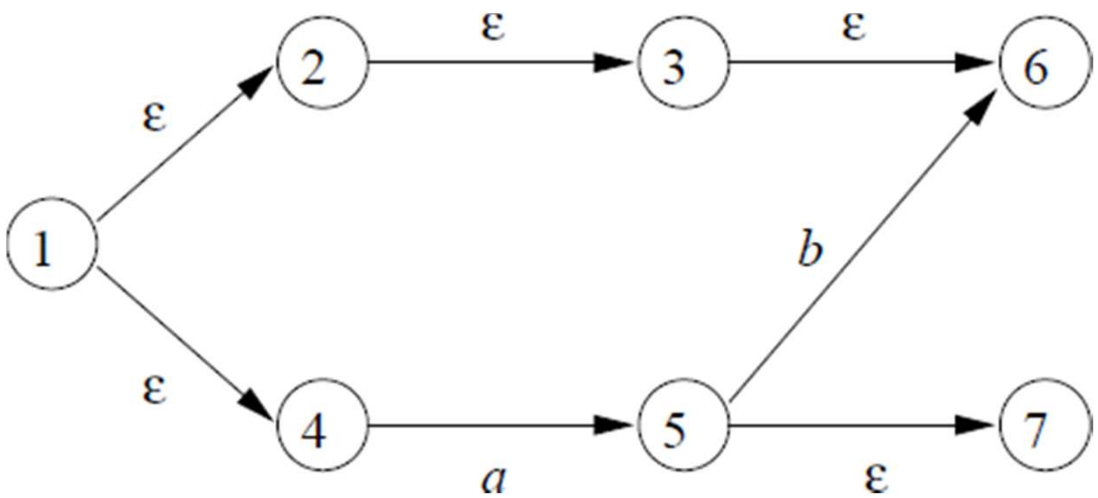
ECLOSE(3) = {3,6}

ECLOSE(4) = {4}

ECLOSE(5) = {5,7}

ECLOSE(6) = {6}

ECLOSE(7) = {7}



NFA Examples:

- Give NFA's accepting the following languages over the alphabet {0,1}.

1. The set of all strings ending in 00.
2. The set of all strings ending in 1010.
3. The strings whose second characters from the right end are 1.
4. The strings whose third characters from the right end are 1.

Equivalence of DFA and NFA

- NFA's are usually easier to construct.
- For any NFA N there is a DFA M, such that $L(M) = L(N)$, and vice versa.
- Given a NFA N

$$\boldsymbol{N} = (\boldsymbol{Q}, \boldsymbol{\Sigma}, \boldsymbol{\delta}, \boldsymbol{q_0}, \boldsymbol{F})$$

We can construct an equivalent DFA M

$$\boldsymbol{M} = (\boldsymbol{Q'}, \boldsymbol{\Sigma}, \boldsymbol{\delta'}, \boldsymbol{q_0}', \boldsymbol{F'})$$

such that $L(M) = L(N)$

Equivalence of DFA and NFA

- Given a NFA N

$$\boldsymbol{N} = (\boldsymbol{Q}, \boldsymbol{\Sigma}, \boldsymbol{\delta}, \boldsymbol{q_0}, \boldsymbol{F})$$

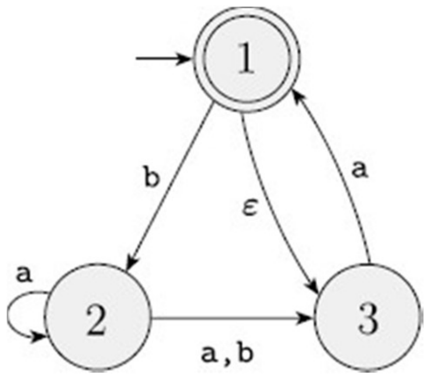
we can construct an equivalent DFA M

$$\boldsymbol{M} = (\boldsymbol{Q'}, \boldsymbol{\Sigma}, \boldsymbol{\delta'}, \boldsymbol{q_0}', \boldsymbol{F'})$$

- $Q' = \mathcal{P}(Q)$
- For $R \in Q'$, $\delta'(R, a) = \{q \in Q \mid q \in \delta(r, a)$ for some $r \in R\}$
- $q_0' = ECLOSE\{\mathrm{q}_0\}$
- $F' = \{R \in Q' \mid R$ contains an accept state of $N\}$

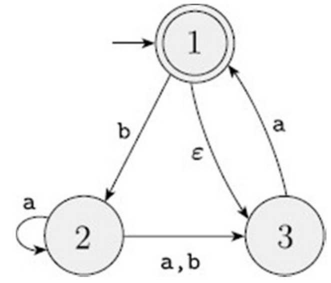
Equivalence of DFA and NFA

- Construct a DFA that is equivalent to the following NFA.



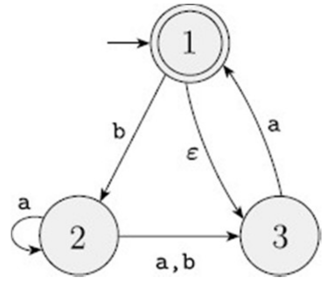
$$N = (\{1, 2, 3\}, \{a, b\}, \delta, \{1\}, \{1\})$$

Equivalence of DFA and NFA



- $Q' = \mathcal{P}(Q)$ $\{\emptyset, \{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{2,3\}, \{1,2,3\}\}$
- $q_0' = \text{ECLOSE}\{1\} = \{1,3\}$
- $F' = \{\{1\}, \{1,2\}, \{1,3\}, \{1,2,3\}\}$
- $\delta'(\{1\}, a) = ECLOSE\{\delta(1, a)\} = \emptyset$
- $\delta'(\{1\}, b) = ECLOSE\{\delta(1, b)\} = \{2\}$
- $\delta'(\{2\}, a) = ECLOSE\{\delta(2, a)\} = \{2,3\}$
- $\delta'(\{2\}, b) = ECLOSE\{\delta(2, b)\} = \{3\}$
- $\delta'(\{3\}, a) = ECLOSE\{\delta(3, a)\} = \{1,3\}$
- $\delta'(\{3\}, b) = ECLOSE\{\delta(3, b)\} = \emptyset$
- $\delta'(\{1,2\}, a) = ECLOSE\{\delta(1, a)\} \cup ECLOSE\big(\delta(2, a)\big) = \emptyset \cup \{2,3\} = \{2,3\}$
- $\delta'(\{1, 2\}, b) = ECLOSE\{\delta(1, b)\} \cup ECLOSE\big(\delta(2, b)\big) = \{2\} \cup \{3\} = \{2,3\}$

Equivalence of DFA and NFA



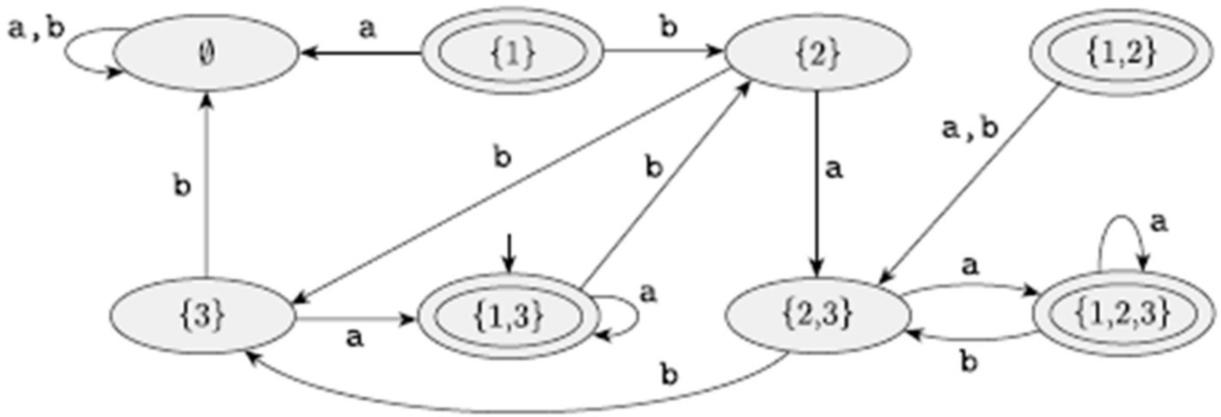
- $Q' = \mathcal{P}(Q)$ $\{\emptyset, \{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{2,3\}, \{1,2,3\}\}$

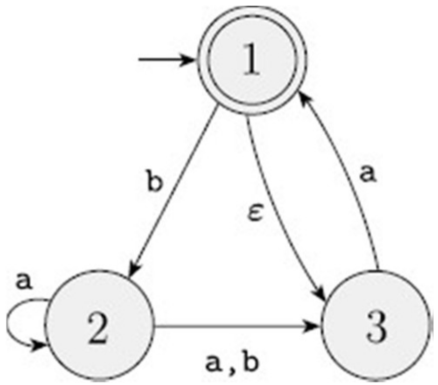
- $q_0' = \text{ECLOSE}\{1\} = \{1,3\}$

- $F' = \{\{1\}, \{1,2\}, \{1,3\}, \{1,2,3\}\}$

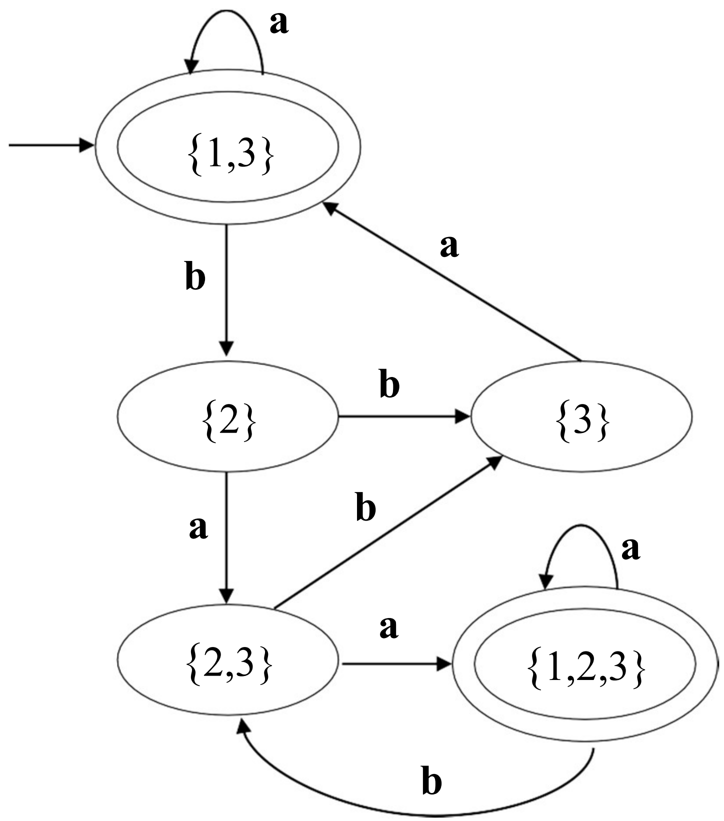
- $\delta'(\{1,3\}, a) = ECLOSE\{\delta(1,a)\} \cup ECLOSE\big(\delta(3,a)\big) = \emptyset \cup \{1,3\} = \{1,3\}$
- $\delta'(\{1,3\}, b) = ECLOSE\{\delta(1,b)\} \cup ECLOSE\big(\delta(3,b)\big) = \{2\} \cup \emptyset = \{2\}$
- $\delta'(\{2,3\}, a) = ECLOSE\{\delta(2,a)\} \cup ECLOSE\big(\delta(3,a)\big) = \{2,3\} \cup \{1,3\}$
  $= \{1,2,3\}$
- $\delta'(\{2,3\}, b) = ECLOSE\{\delta(2,b)\} \cup ECLOSE\big(\delta(3,b)\big) = 3 \cup \emptyset = \{3\}$
- $\delta'(\{1,2,3\}, a) = ECLOSE\{\delta(1,a)\} \cup ECLOSE\big(\delta(2,a)\big) \cup ECLOSE\big(\delta(3,a)\big)$
  $= \emptyset \cup \{2,3\} \cup \{1,3\} = \{1,2,3\}$
- $\delta'(\{1,2,3\}, b) = ECLOSE\{\delta(1,b)\} \cup ECLOSE\big(\delta(2,b)\big) \cup ECLOSE\big(\delta(3,b)\big)$
  $= \{2\} \cup \{3\} \cup \emptyset = \{2,3\}$

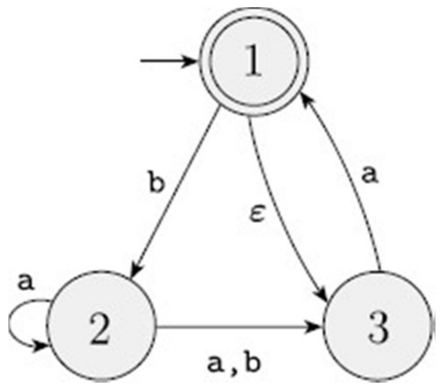
Equivalence of DFA and NFA





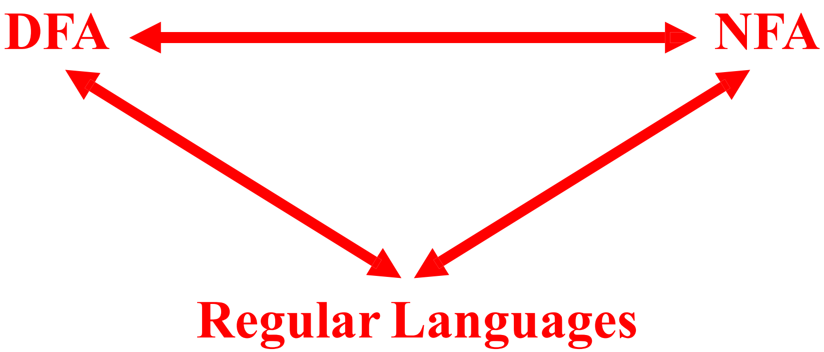
Equivalence of DFA and NFA





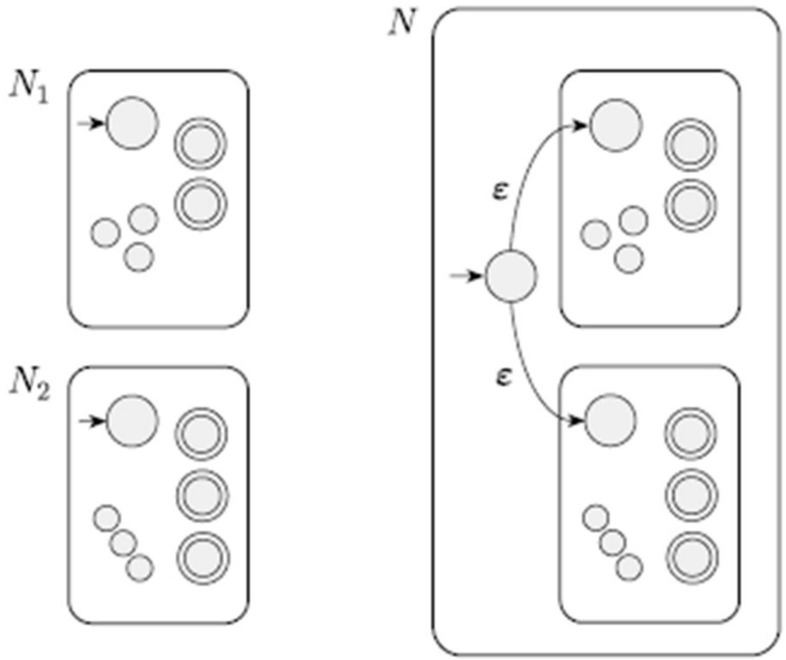
Equivalence of DFA and NFA- Summary

- **Every DFA recognizes a regular language, and there is a DFA for every regular language.**
- **There is an equivalent DFA (their languages are equal) for every NFA, and there is an equivalent NFA for every DFA.**
- **Thus, every NFA recognizes a regular language, and there is a NFA for every regular language.**



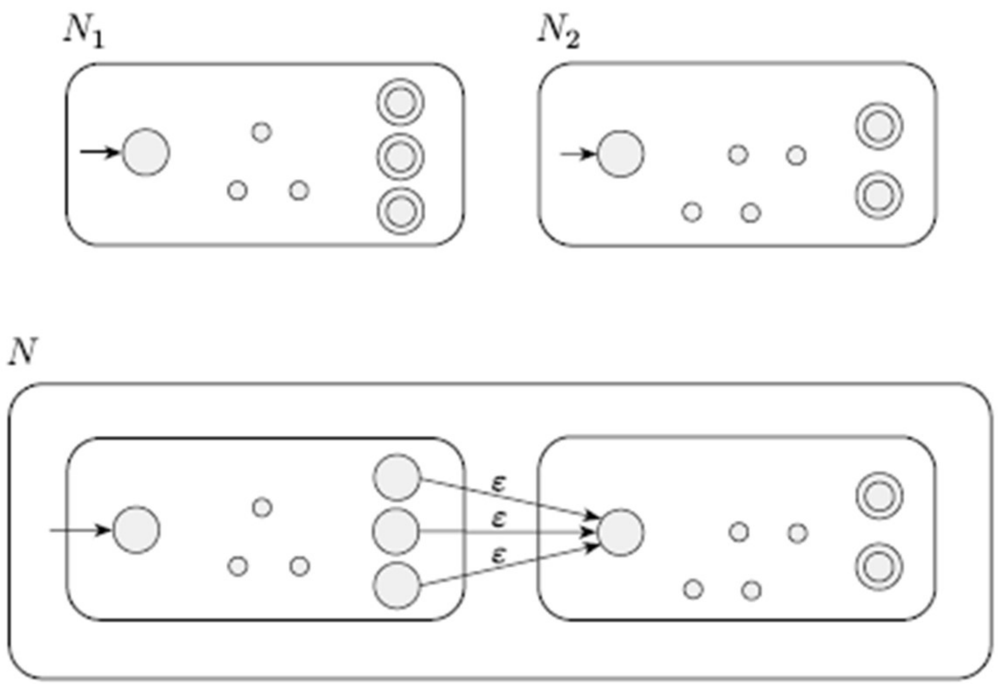
Closure Under The Regular Operations

- The class of regular languages is closed under the union operation.
- We have regular languages $A1$ and $A2$ and want to prove that $A1 \cup A2$ is regular.
- **Idea:**
  - Take two NFAs, $N1$ and $N2$ for $A1$ and $A2$, and combine them into one new NFA, N.
  - Machine N must accept its input if either $N1$ or $N2$ accepts this input, so it has a new start state that branches to the start states of the old machines with ε arrows.



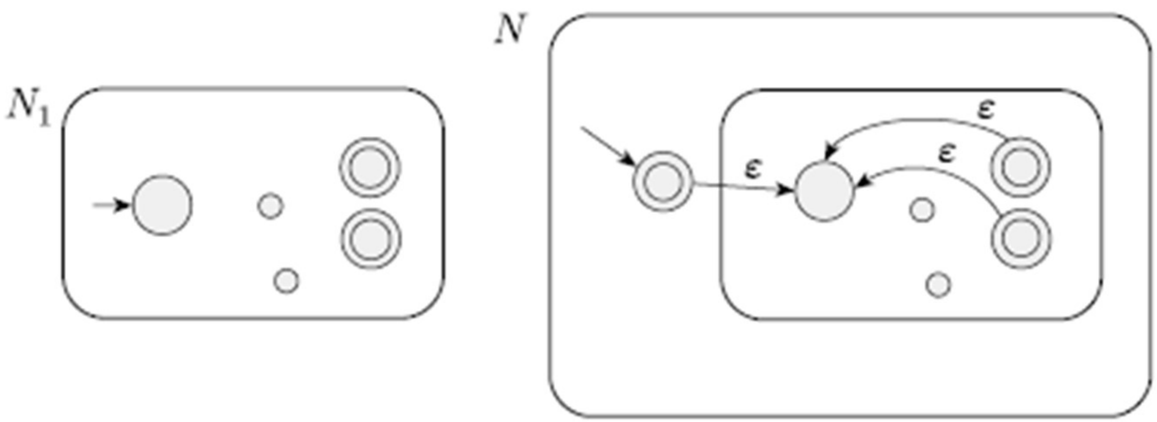
Closure Under The Regular Operations

- The class of regular languages is closed under the concatenation operation ($A1 \circ A2$).
- **Idea:**
  - Assign N's start state to be the start state of $N1$.
  - The accept states of $N1$ have additional ε arrows that nondeterministically allow branching to $N2$ whenever $N1$ is in an accept state
  - The accept states of N are the accept states of $N2$ only.



Closure Under The Regular Operations

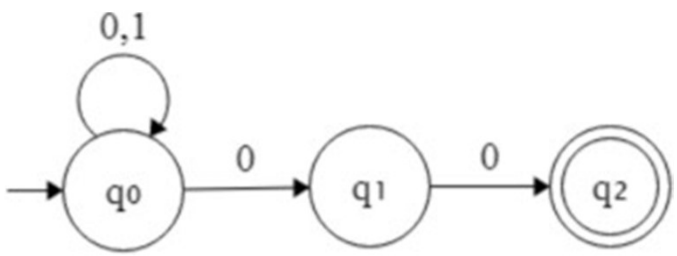
- The class of regular languages is closed under the star operation (A^*).
- **Idea:**
  - Construct N like $N1$ with additional ε arrows returning to the start state from the accept states.
  - Modify N so that it accepts ε by adding a new start state to $N1$, which also is an accept state, and which has an ε arrow to the old start state.



NFA Examples:

- Give NFA's accepting the following languages over the alphabet {0,1}.

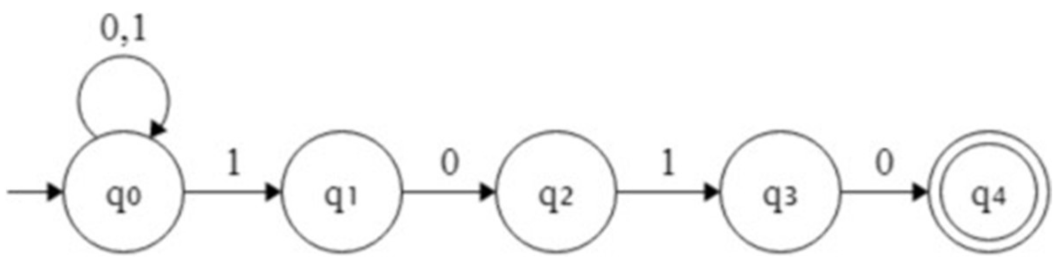
  The set of all strings ending in 00.



NFA Examples:

- Give NFA's accepting the following languages over the alphabet {0,1}.

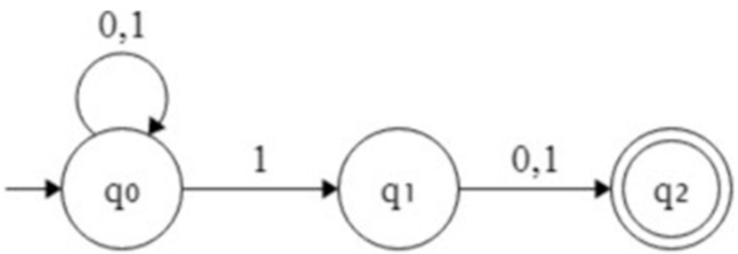
  The set of all strings ending in 1010.



NFA Examples:

- Give NFA's accepting the following languages over the alphabet {0,1}.

  The strings whose second characters from the right end are 1.



NFA Examples:

- Give NFA's accepting the following languages over the alphabet {0,1}.

  The strings whose third characters from the right end are 1.