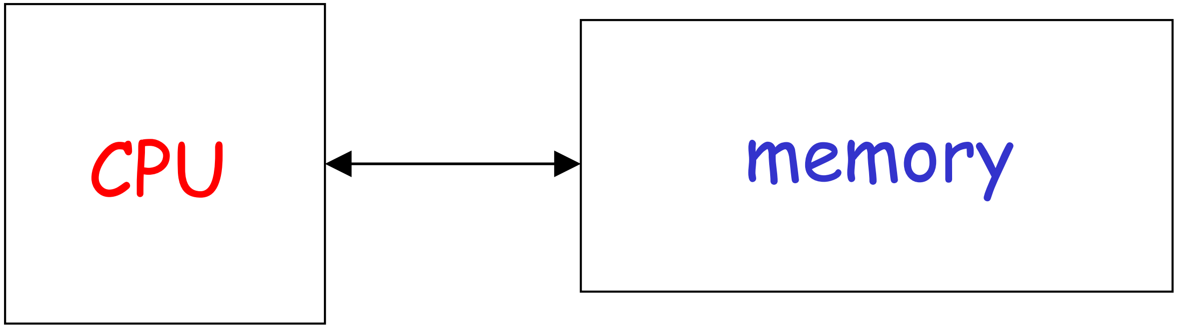
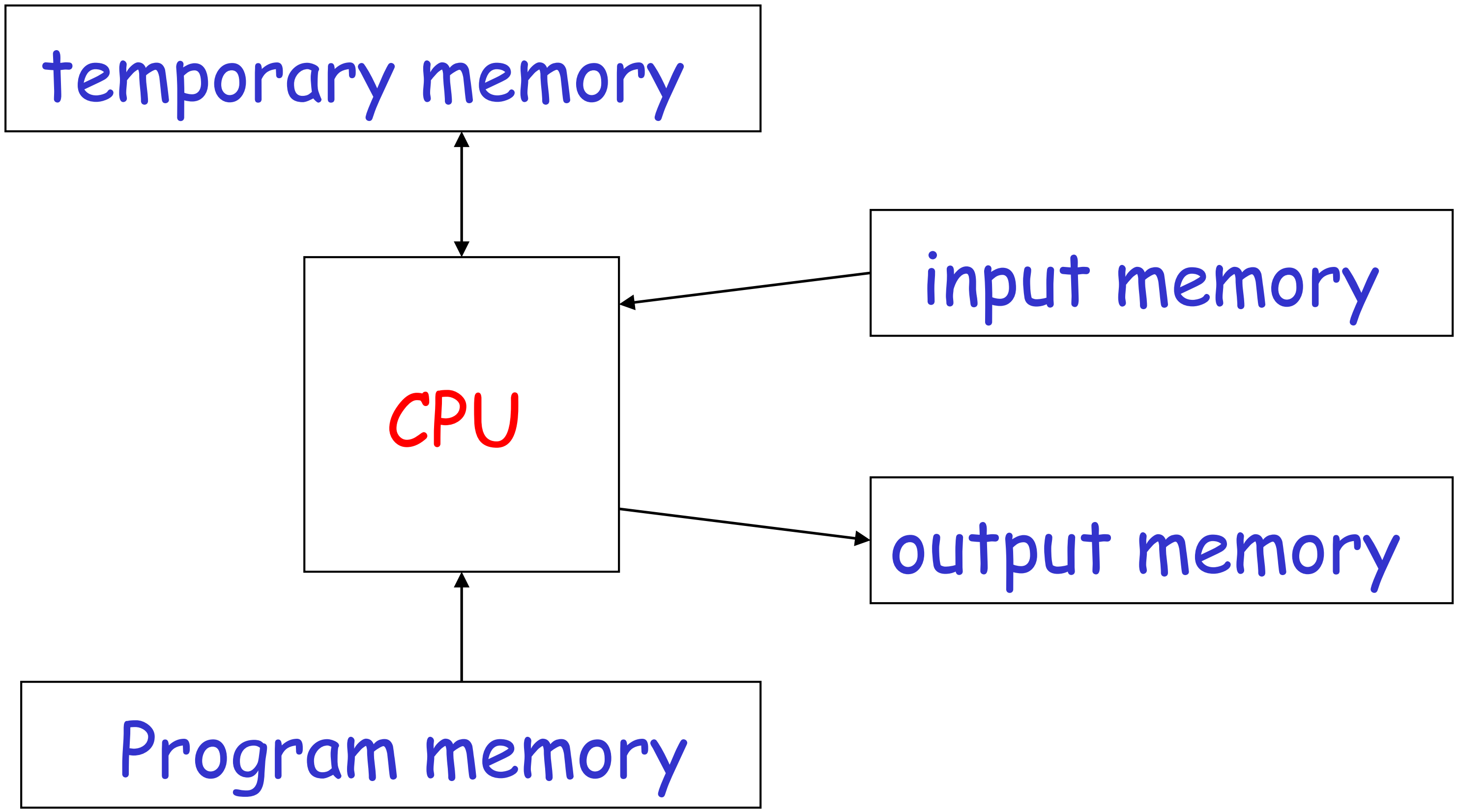
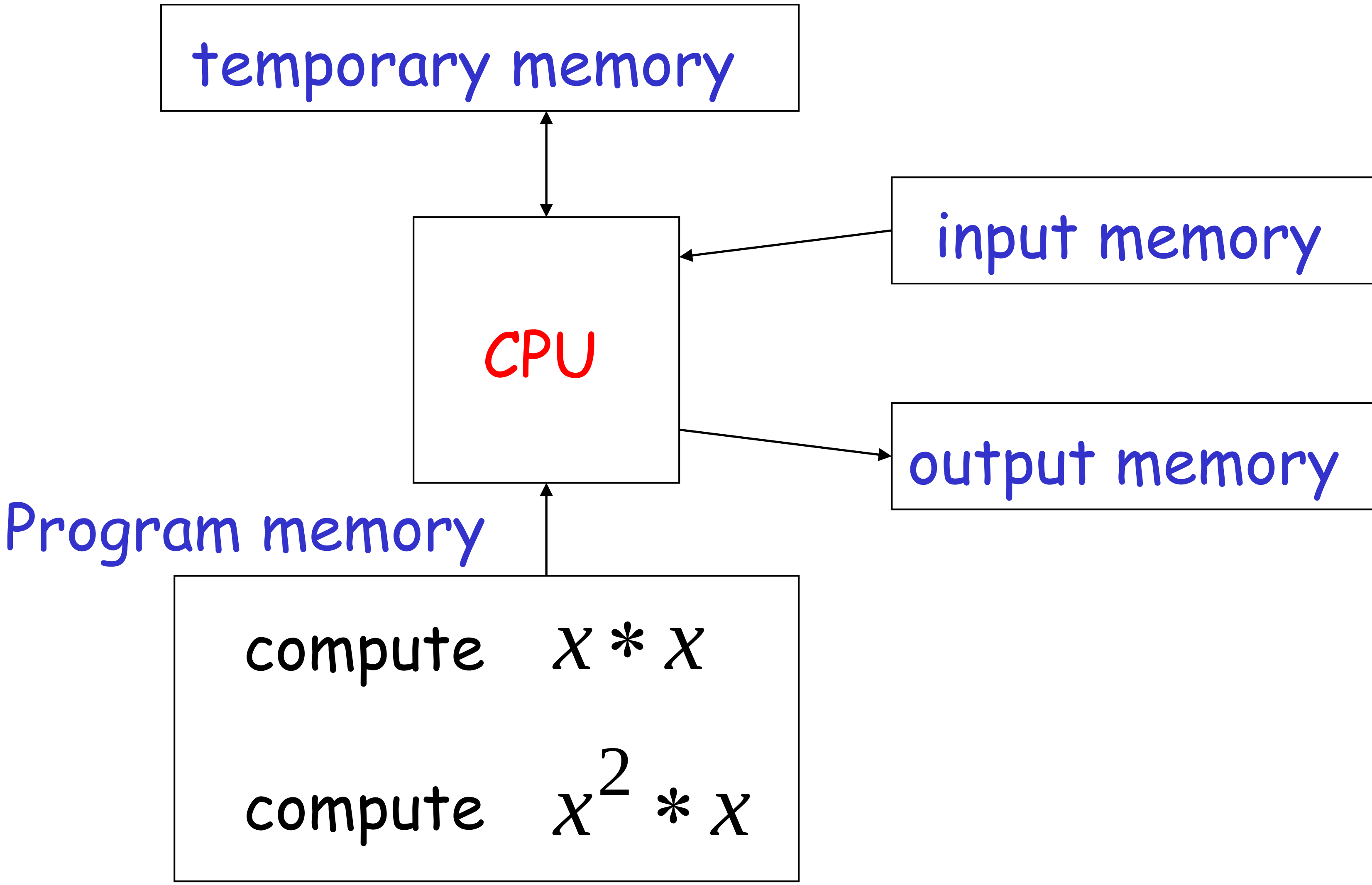
Models of Computation

Computation

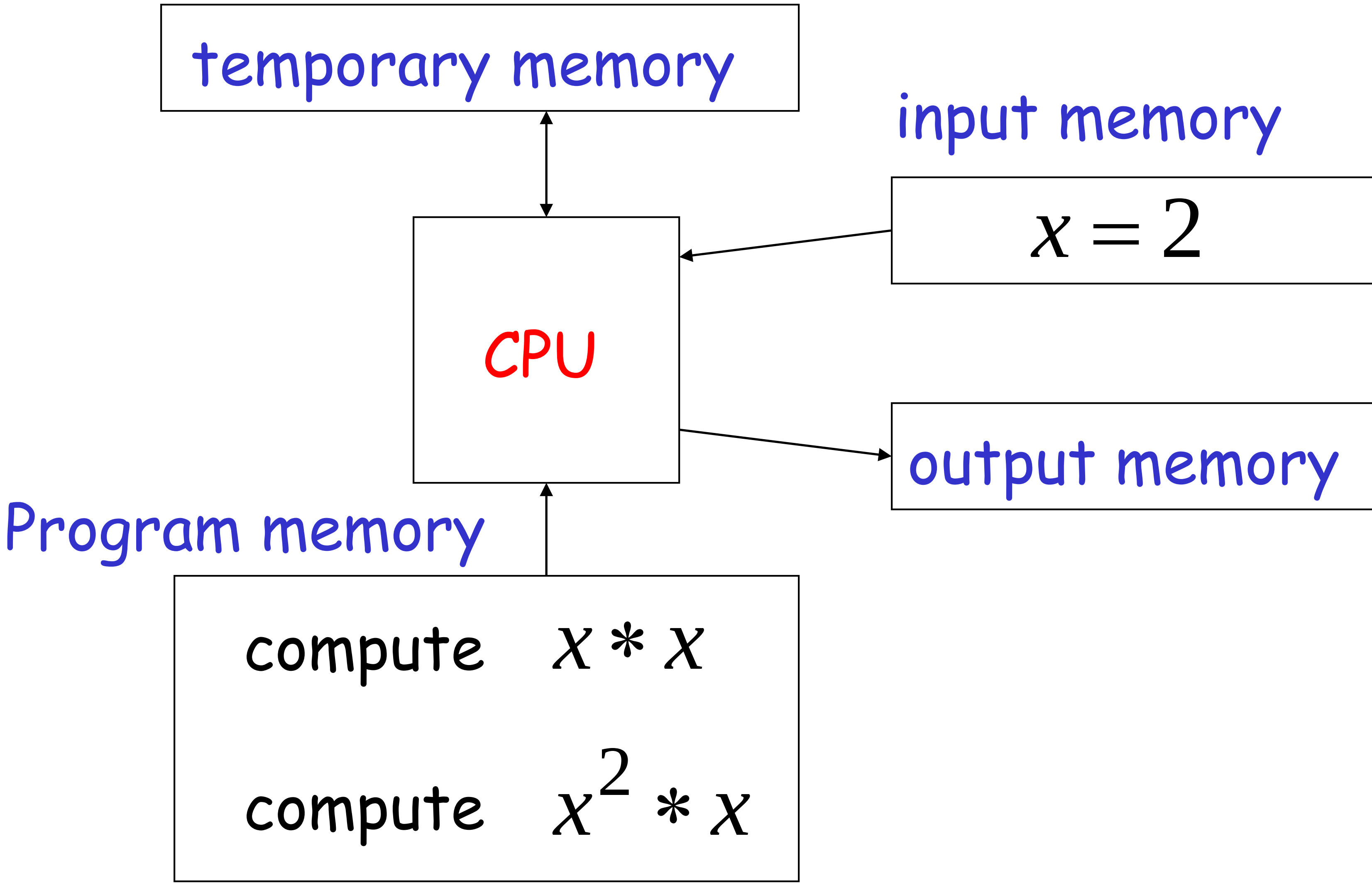



temporary memory
input memory
CPU
output memory
Program memory

Example: $f(x) = x^3$

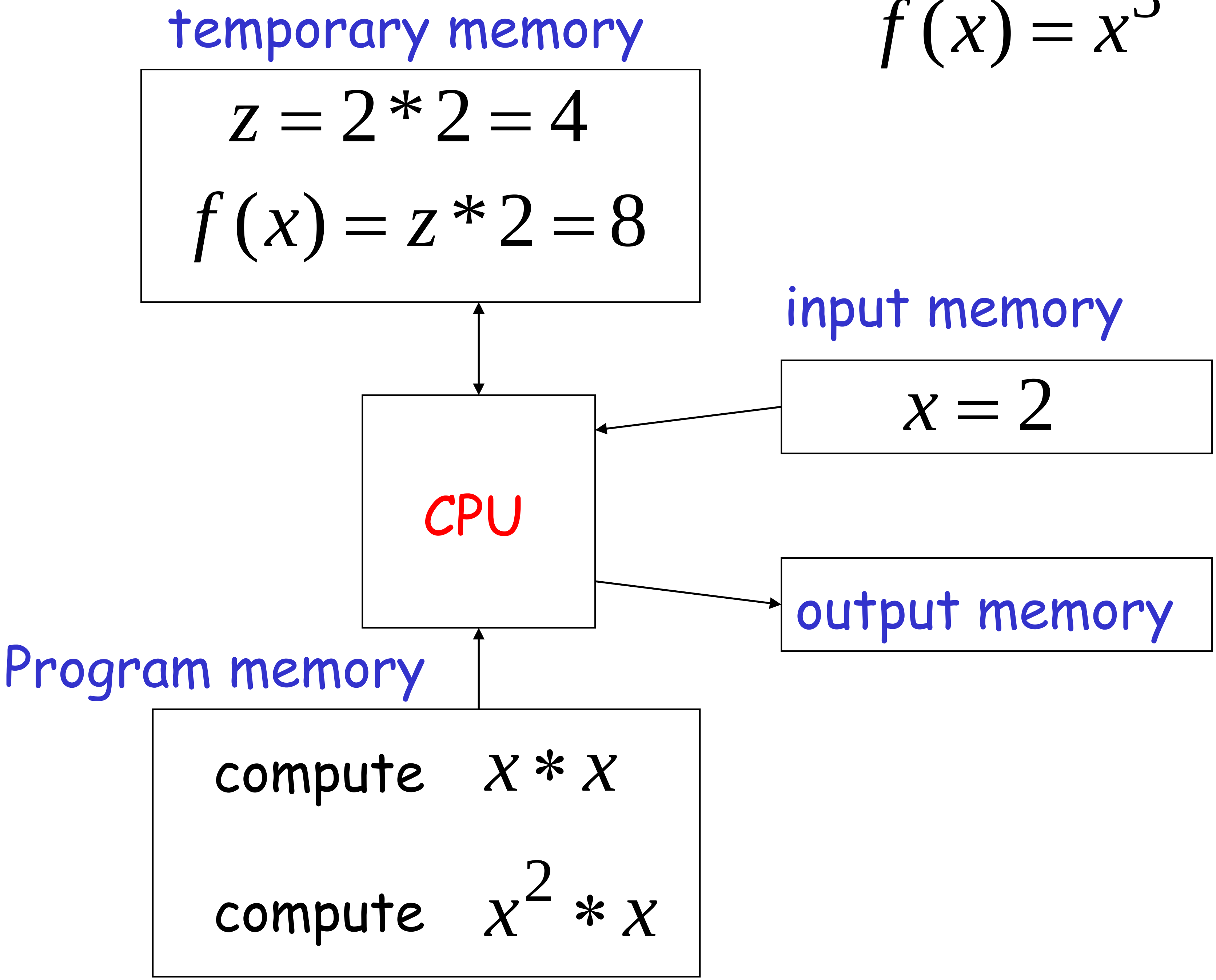


$$f(x) = x^3$$

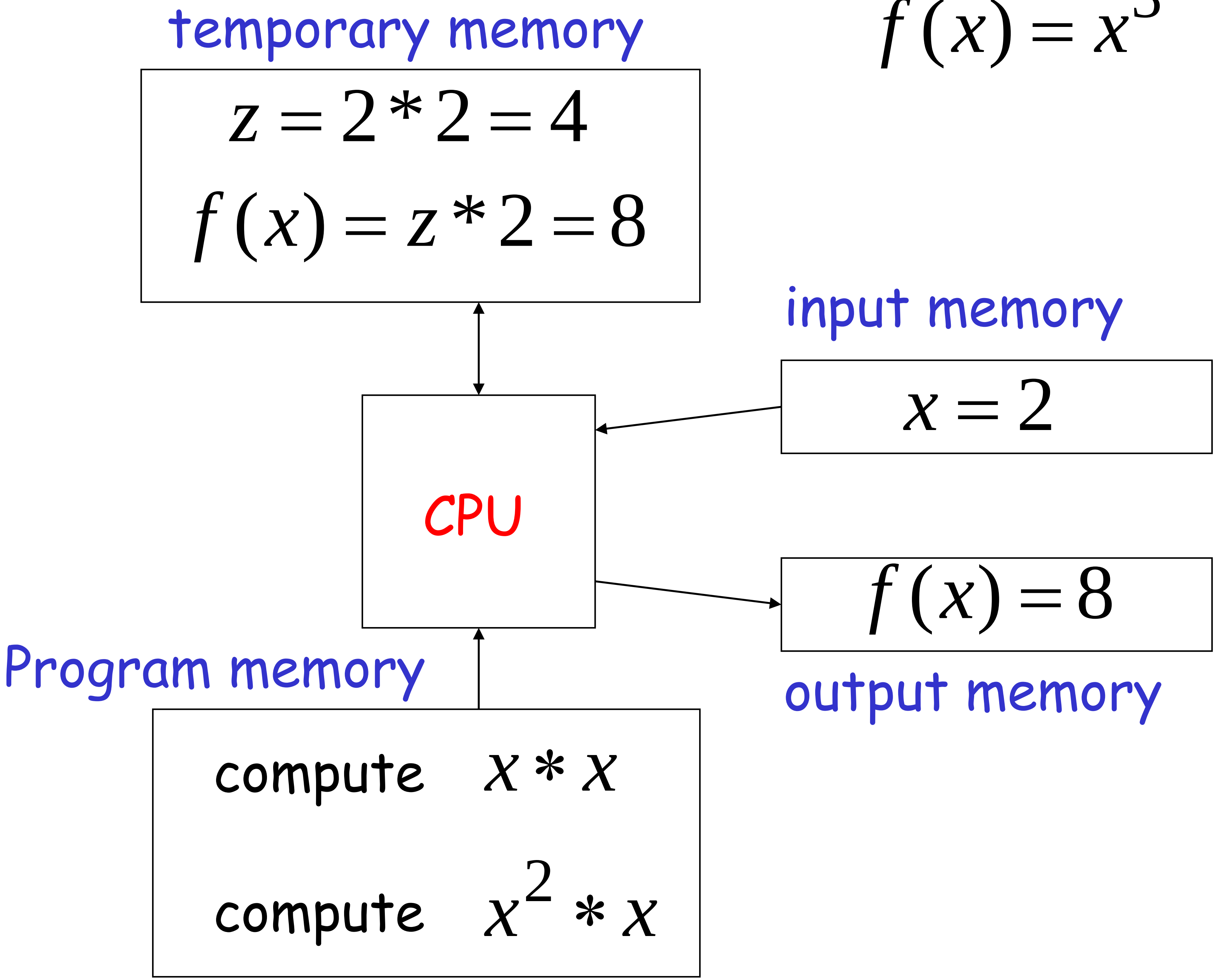


temporary memory
input memory
$x = 2$
CPU
output memory
Program memory
compute $x * x$
compute $x^2 * x$

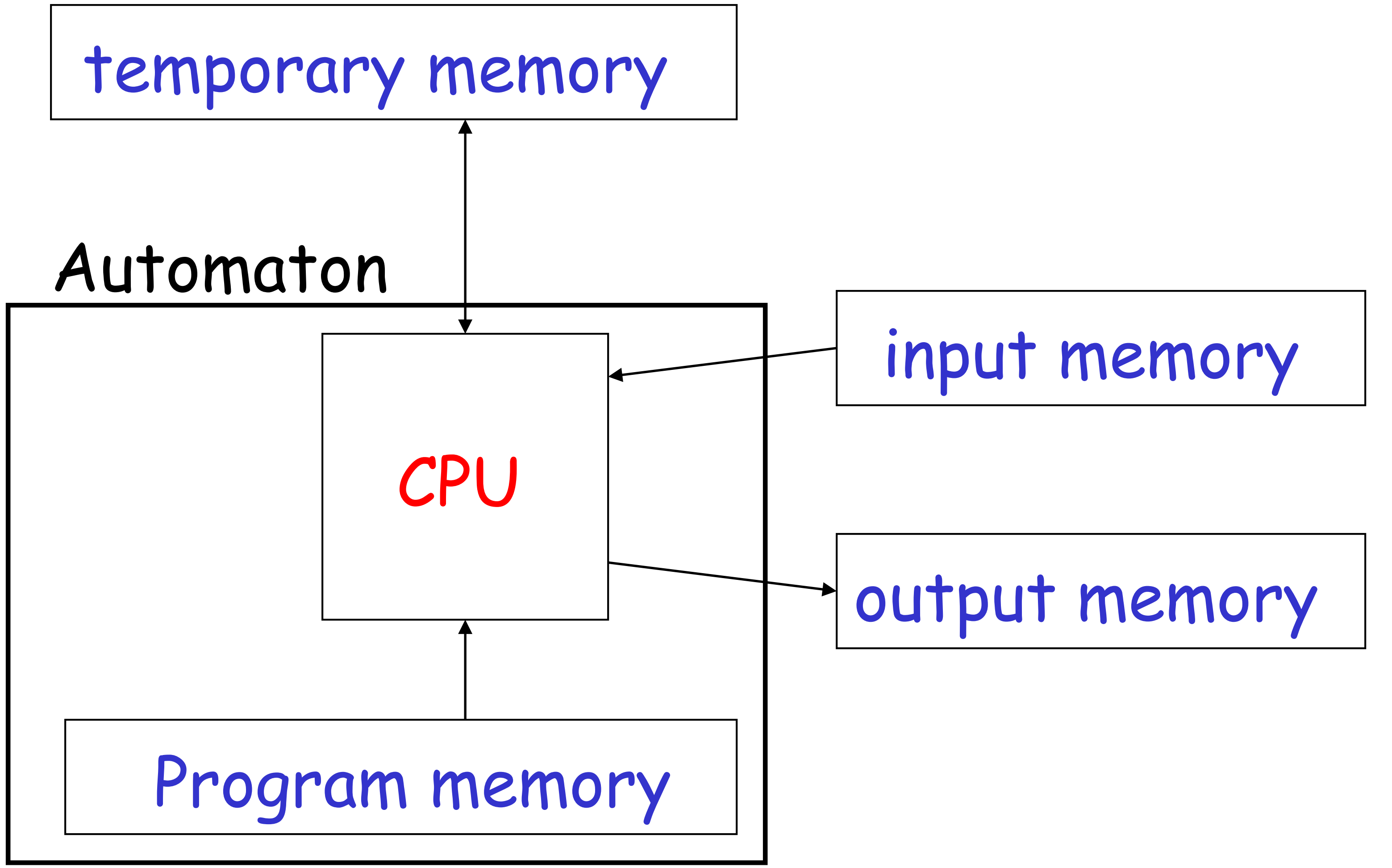
$$f(x) = x^3$$


temporary memory
$z = 2 * 2 = 4$
$f(x) = z * 2 = 8$
input memory
$x = 2$
CPU
output memory
Program memory
compute $x * x$
compute $x^2 * x$

$$f(x) = x^3$$


temporary memory
$z = 2*2 = 4$
$f(x) = z*2 = 8$
input memory
$x = 2$
CPU
$f(x) = 8$
output memory
Program memory
compute $x*x$
compute x^2*x

Automaton

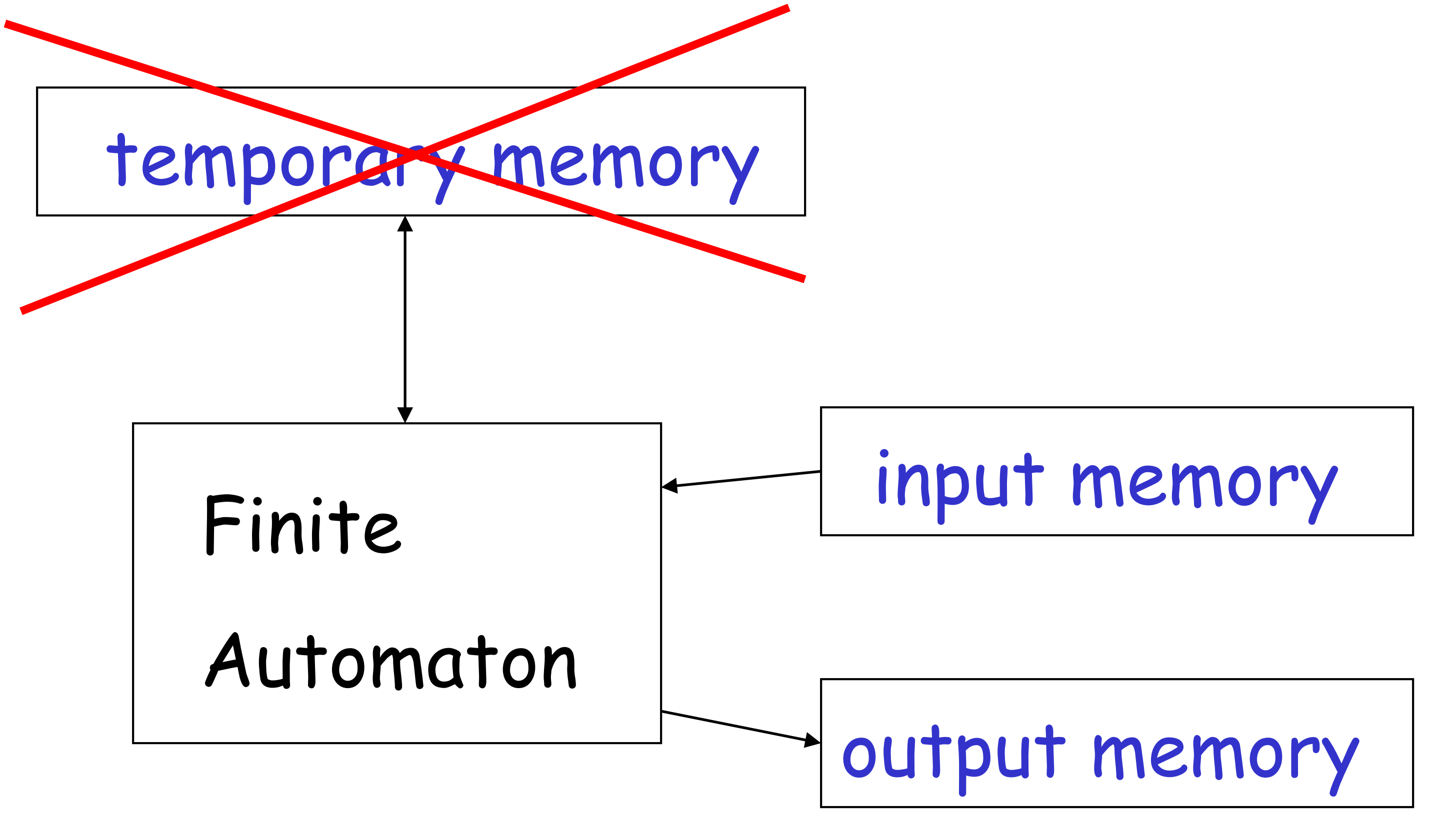

temporary memory
Automaton
CPU
input memory
output memory
Program memory

Different Kinds of Automata

Automata are distinguished by the temporary memory

- **Finite Automata:** no temporary memory
- **Pushdown Automata:** stack
- **Turing Machines:** random access memory

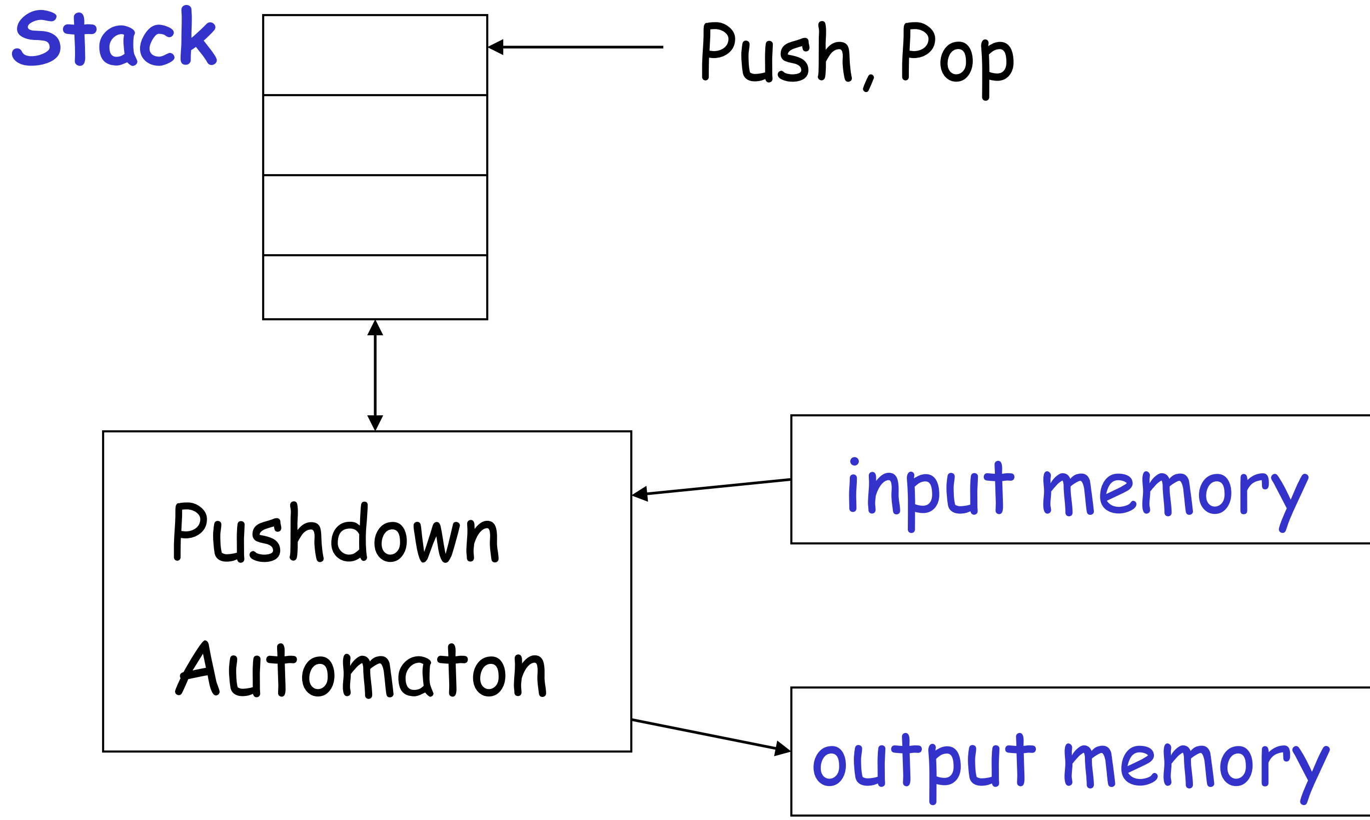
Finite Automaton



Example: Vending Machines

(small computing power)

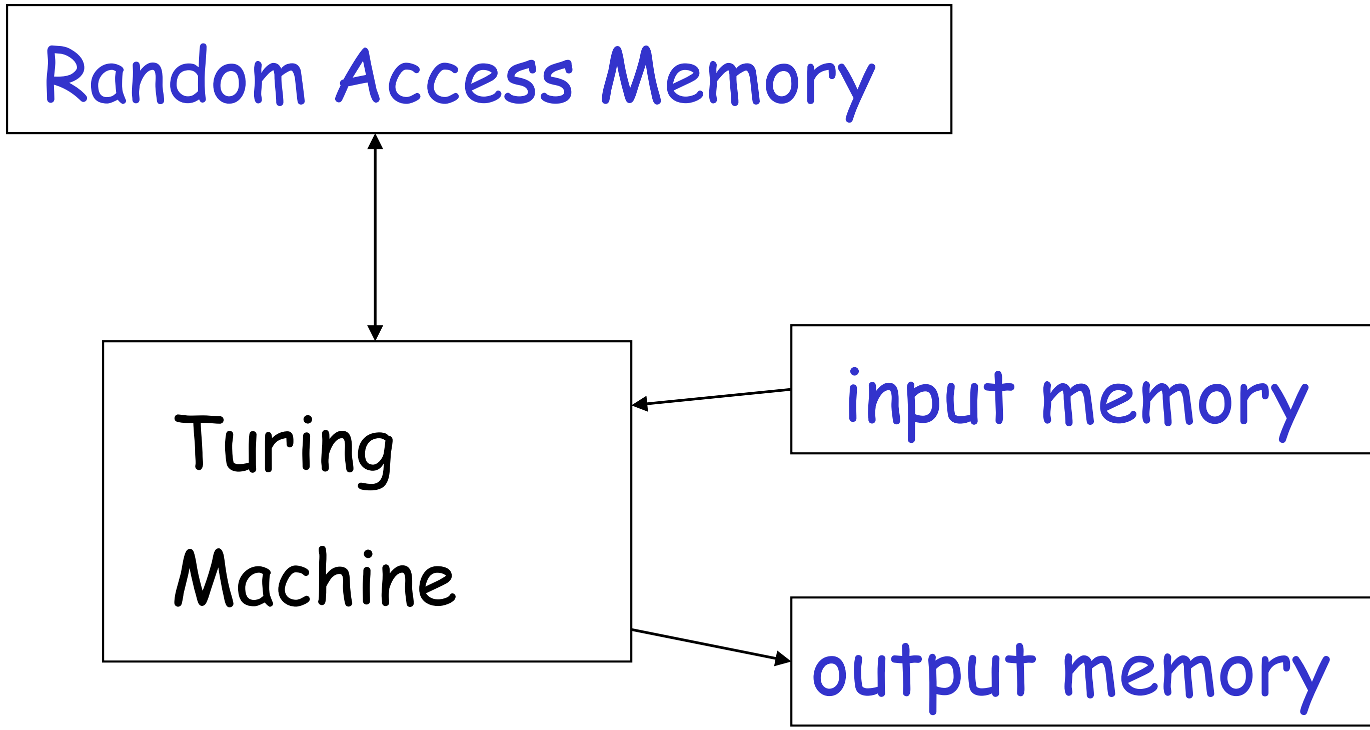
Pushdown Automaton



Example: Compilers for Programming Languages

(medium computing power)

Turing Machine



Examples: Any Algorithm

(highest computing power)

Power of Automata

Finite Automata < Pushdown Automata < Turing Machine

Less power → More power

Solve more computational problems

Languages

A language is a set of strings

String: A sequence of letters

Examples: "cat", "dog", "house", ...

Defined over an alphabet:

$$\Sigma = \{a, b, c, \ldots, z\}$$

Alphabets and Strings

We will use small alphabets: $\Sigma = \{a, b\}$

Strings

a

ab $\qquad u = ab$

$abba$ $\qquad v = bbbaaa$

$baba$ $\qquad w = abba$

$aaabbbaabab$

String Operations

$$w = a_1 a_2 \cdots a_n \qquad\qquad abba$$

$$v = b_1 b_2 \cdots b_m \qquad\qquad bbbaaa$$

Concatenation

$$wv = a_1 a_2 \cdots a_n b_1 b_2 \cdots b_m \qquad\qquad abbabbbaaa$$

$$w = a_1 a_2 \cdots a_n \qquad ababaaabbb$$

Reverse

$$w^R = a_n \cdots a_2 a_1 \qquad bbbaaababa$$

String Length

$$w = a_1 a_2 \cdots a_n$$

Length: $|w| = n$

Examples:

$$|abba| = 4$$

$$|aa| = 2$$

$$|a| = 1$$

Length of Concatenation

$$|uv| = |u| + |v|$$

Example:

$$u = aab, \quad |u| = 3$$

$$v = abaab, \quad |v| = 5$$

$$|uv| = |aababaab| = 8$$

$$|uv| = |u| + |v| = 3 + 5 = 8$$

Empty String

A string with no letters: λ

Observations: $|\lambda| = 0$

$$\lambda w = w\lambda = w$$

$$\lambda abba = abba\lambda = abba$$

Another Operation

$$w^n = \underbrace{ww\cdots w}_{n}$$

Example: $(abba)^2 = abbaabba$

Definition: $w^0 = \lambda$

$$(abba)^0 = \lambda$$

The * Operation

Σ^*: the set of all possible strings from alphabet Σ

$$\Sigma = \{a, b\}$$

$$\Sigma^* = \{\lambda, a, b, aa, ab, ba, bb, aaa, aab, \ldots\}$$

Languages

A language is any subset of Σ^*

Example: $\Sigma = \{a, b\}$

$$\Sigma^* = \{\lambda, a, b, aa, ab, ba, bb, aaa, \ldots\}$$

Languages: $\{\lambda\}$

$$\{a, aa, aab\}$$

$$\{\lambda, abba, baba, aa, ab, aaaaaa\}$$

Note that:

Sets $\emptyset = \{\ \} \neq \{\lambda\}$

Set size $|\{\ \}| = |\emptyset| = 0$

Set size $|\{\lambda\}| = 1$

String length $|\lambda| = 0$

Another Example

An infinite language $L = \{a^n b^n : n \geq 0\}$

$$\left.\begin{array}{l} \lambda \\ ab \\ aabb \\ aaaaabbbbb \end{array}\right\} \in L \qquad abb \notin L$$

Operations on Languages

The usual set operations

$$\{a, ab, aaaa\} \bigcup \{bb, ab\} = \{a, ab, bb, aaaa\}$$

$$\{a, ab, aaaa\} \bigcap \{bb, ab\} = \{ab\}$$

$$\{a, ab, aaaa\} - \{bb, ab\} = \{a, aaaa\}$$

Complement: $\overline{L} = \Sigma^* - L$

$$\overline{\{a, ba\}} = \{\lambda, b, aa, ab, bb, aaa, \ldots\}$$

Reverse

Definition: $L^R = \{w^R : w \in L\}$

Examples: $\{ab, aab, baba\}^R = \{ba, baa, abab\}$

$$L = \{a^n b^n : n \geq 0\}$$

$$L^R = \{b^n a^n : n \geq 0\}$$

Concatenation

Definition: $L_1L_2 = \{xy : x \in L_1, y \in L_2\}$

Example: $\{a, ab, ba\}\{b, aa\}$

$$= \{ab, aaa, abb, abaa, bab, baaa\}$$

Another Operation

Definition: $$L^n = \underbrace{LL \cdots L}_{n}$$

$$\{a,b\}^3 = \{a,b\}\{a,b\}\{a,b\} =$$
$$\{aaa, aab, aba, abb, baa, bab, bba, bbb\}$$

Special case: $L^0 = \{\lambda\}$

$$\{a, bba, aaa\}^0 = \{\lambda\}$$

More Examples

$$L = \{a^n b^n : n \geq 0\}$$

$$L^2 = \{a^n b^n a^m b^m : n, m \geq 0\}$$

$$aabbaaabbb \in L^2$$

Star-Closure (Kleene *)

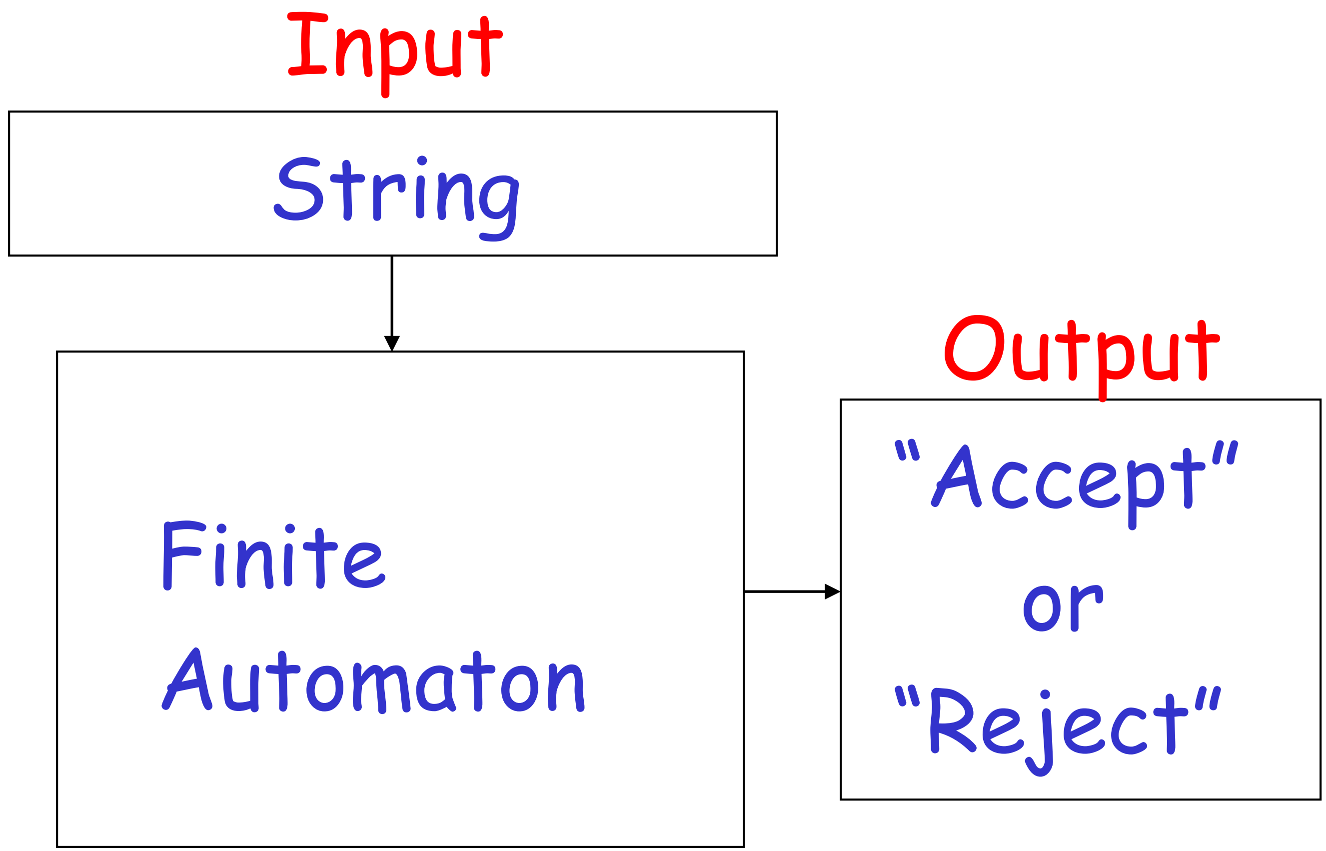
Definition: $L^* = L^0 \bigcup L^1 \bigcup L^2 \cdots$

Example:

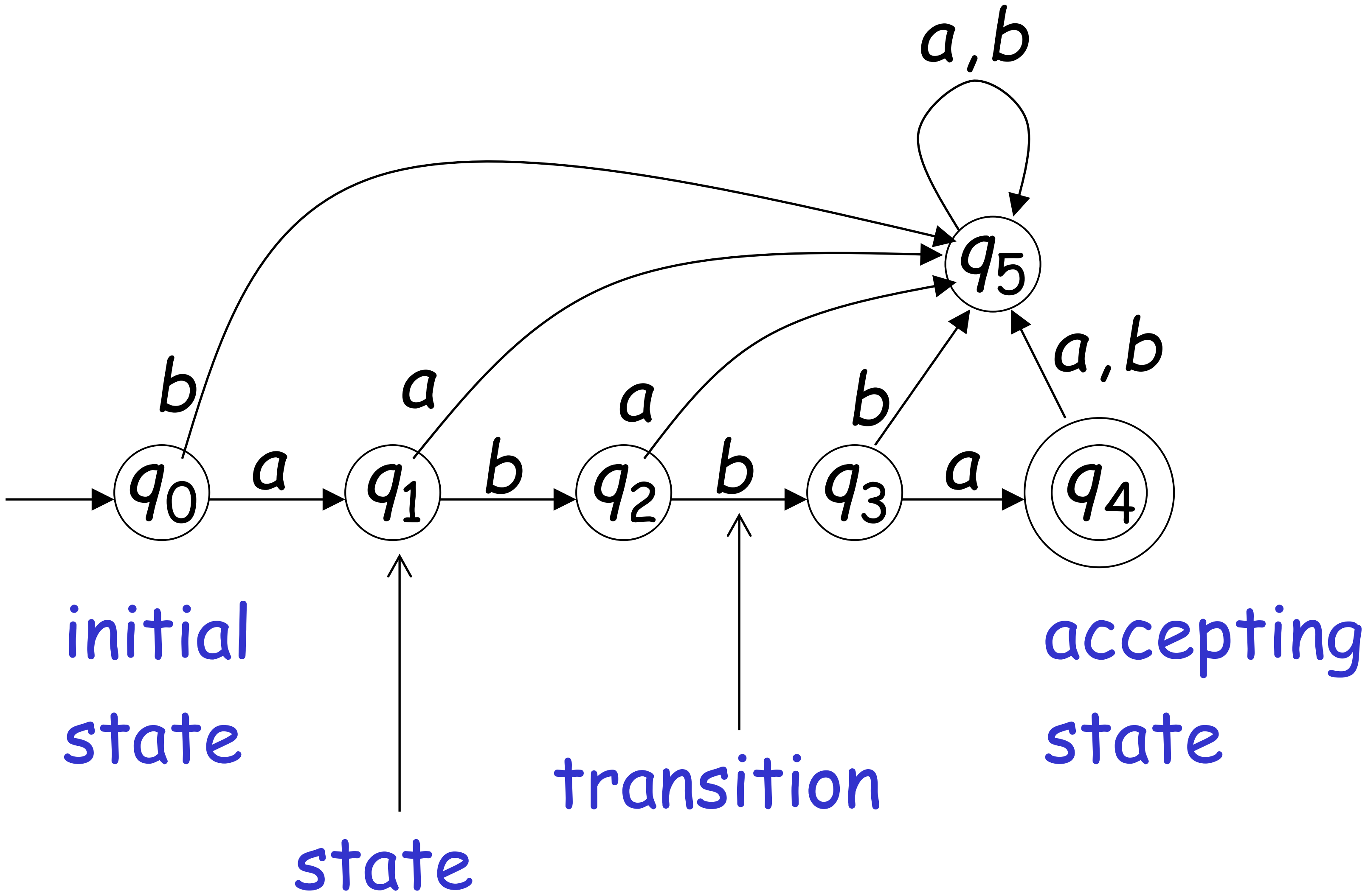
$$\{a, bb\}^* = \left\{ \begin{array}{l} \lambda, \\ a, bb, \\ aa, abb, bba, bbbb, \\ aaa, aabb, abba, abbbb, \ldots \end{array} \right\}$$

Finite Automata

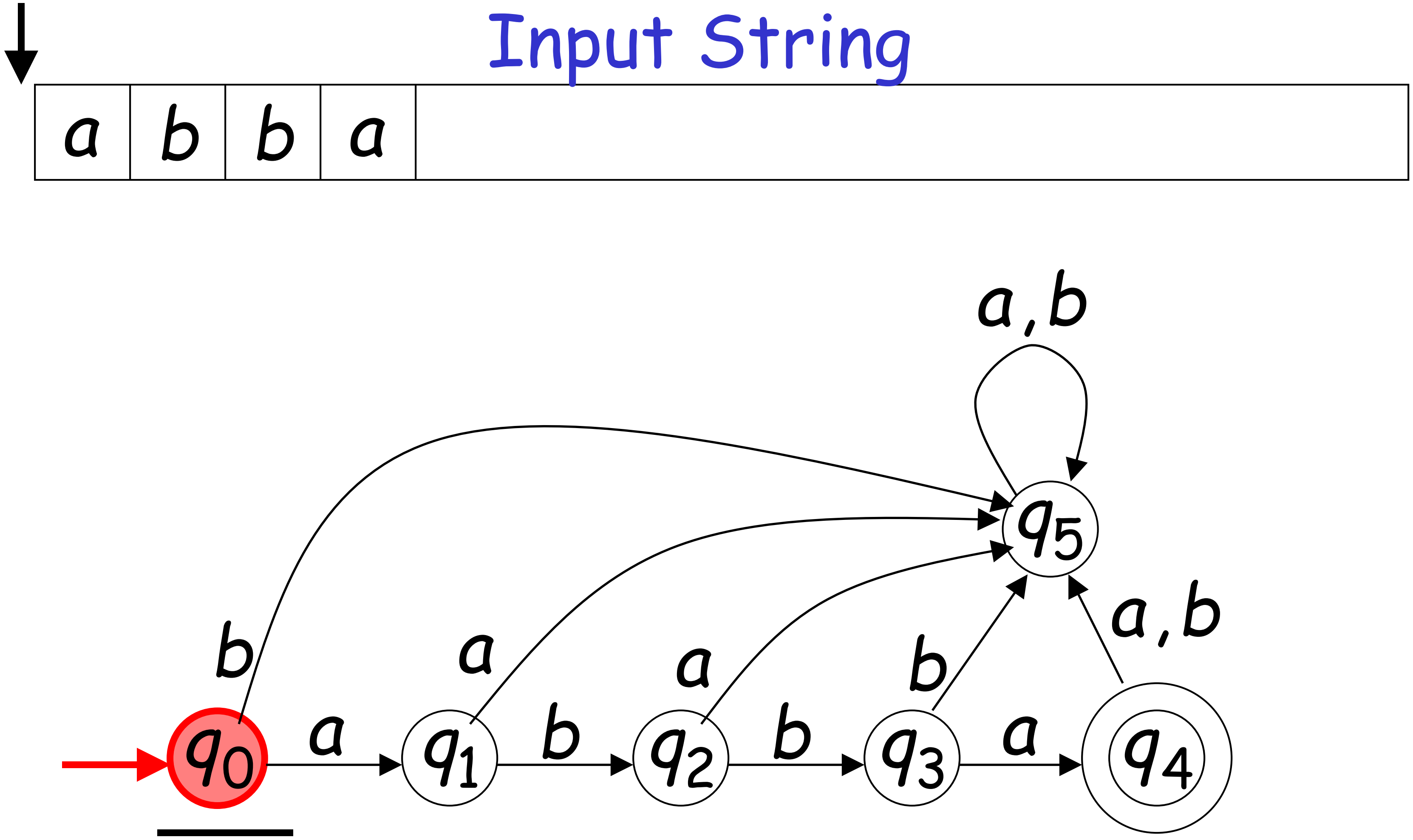
Finite Automaton



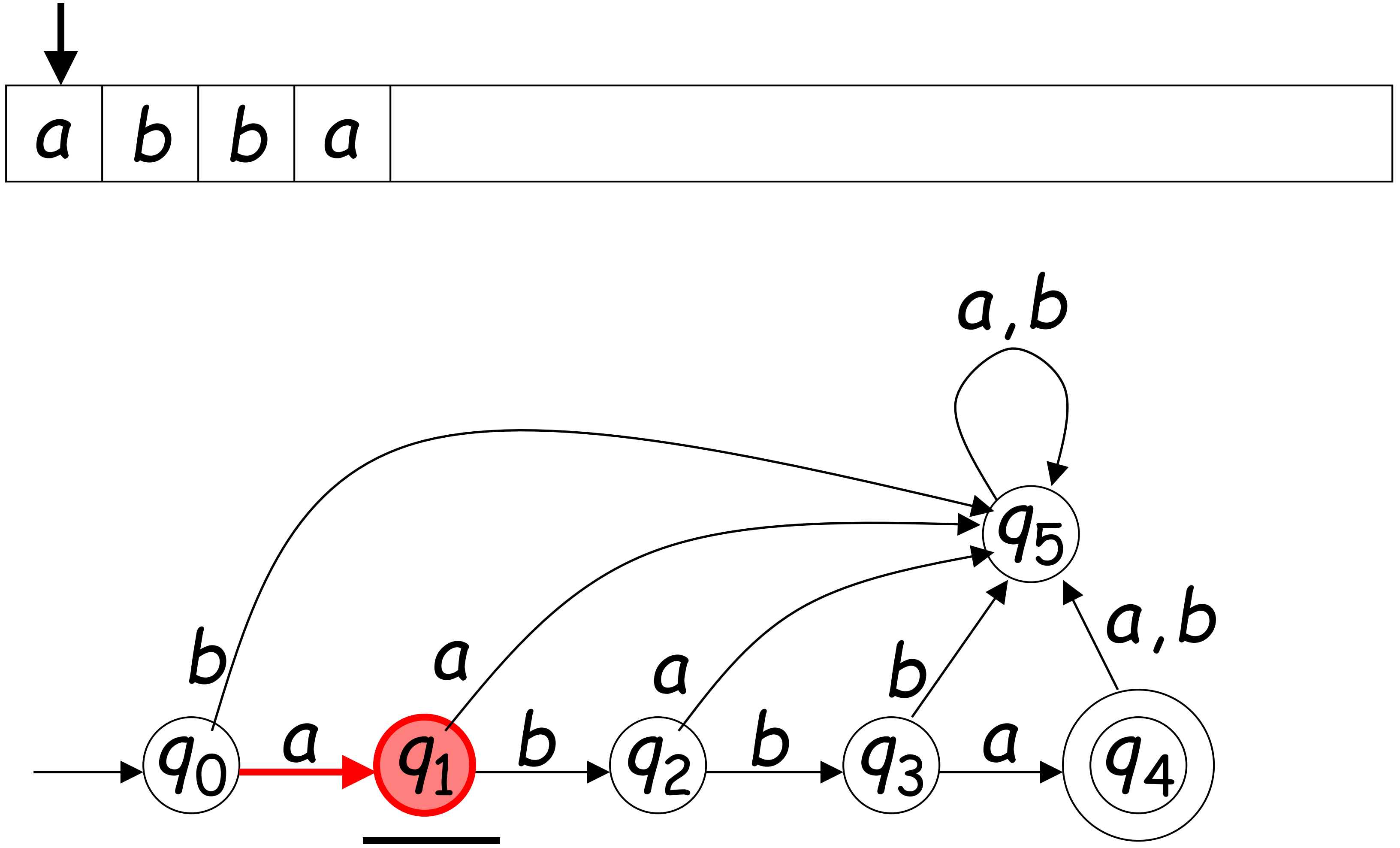
Transition Graph

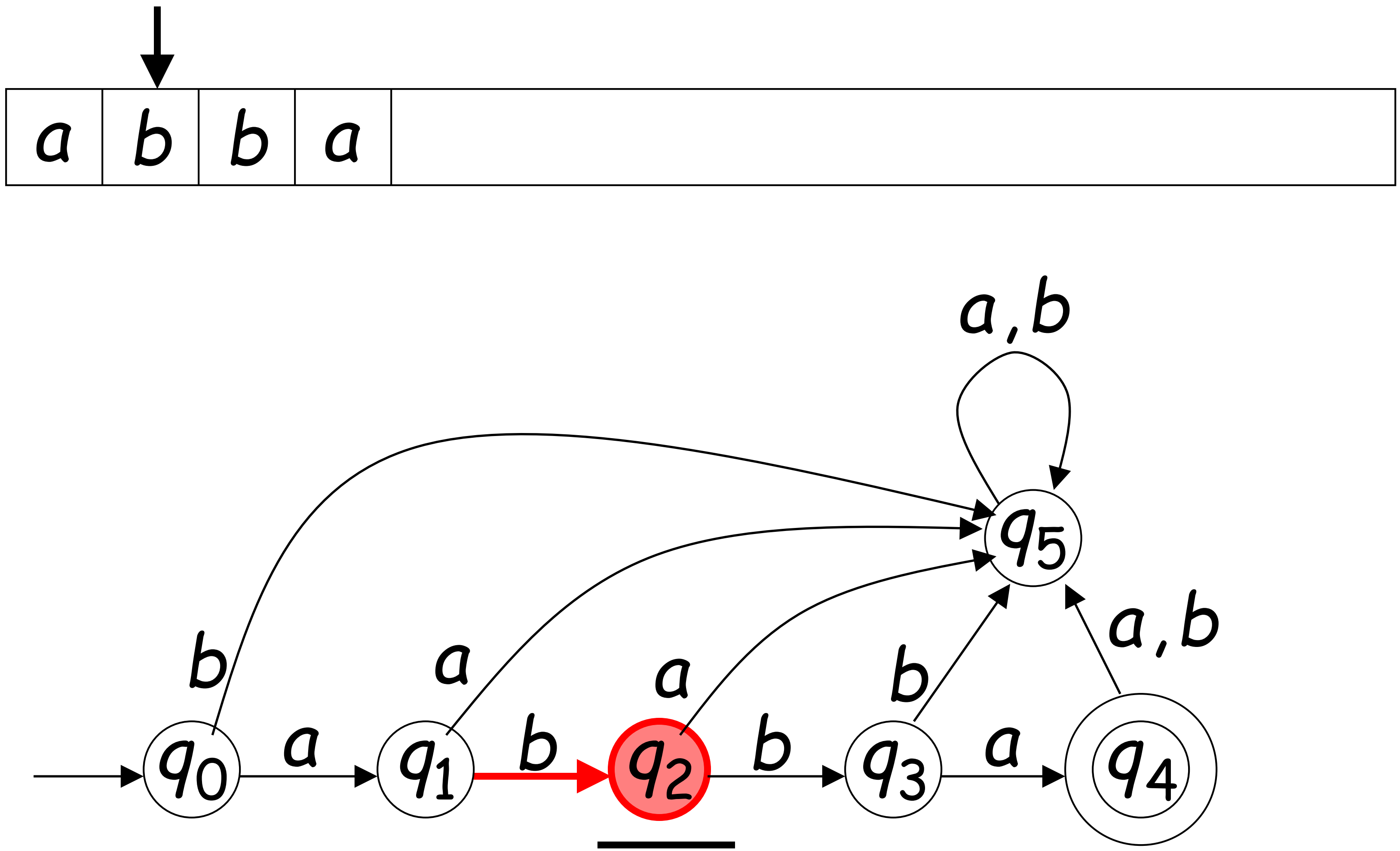


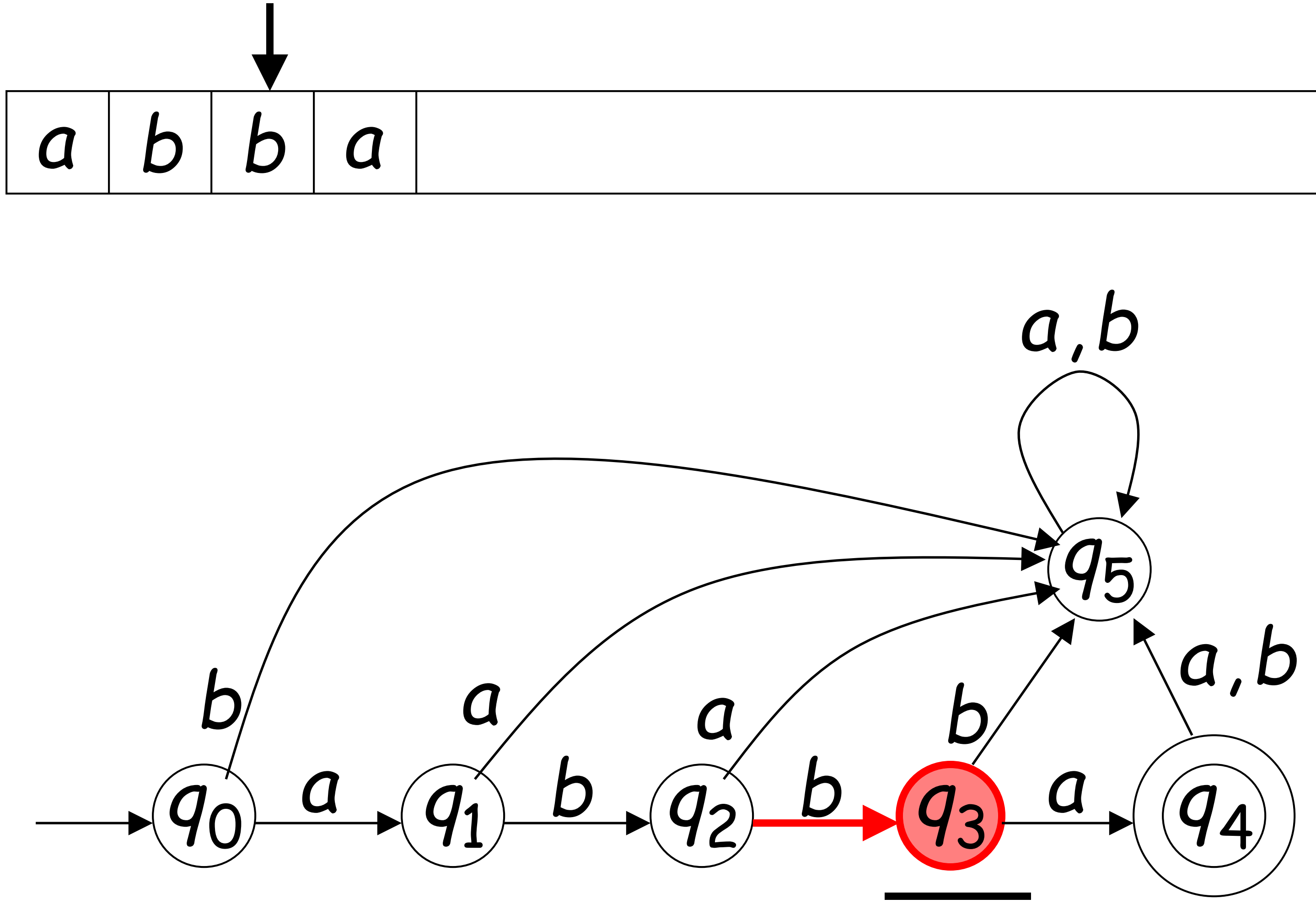
Initial Configuration



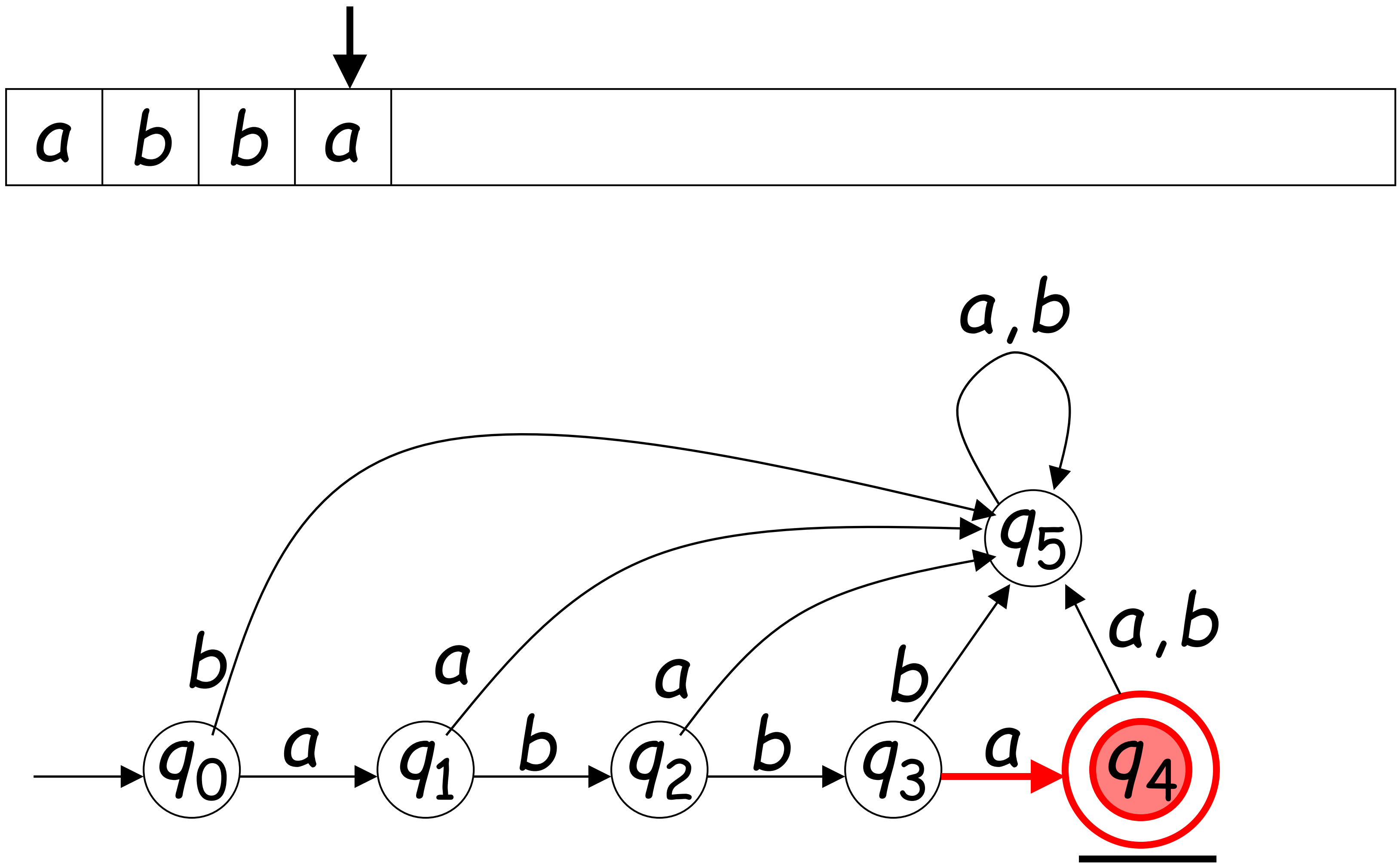
Reading the Input



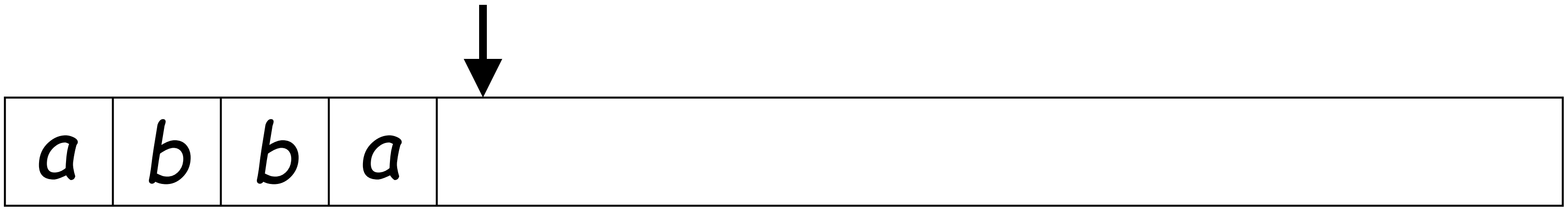

a b b a
a,b
b
a
a
a,b
b
a
b
b
a
q_0
q_1
q_2
q_3
q_4
q_5

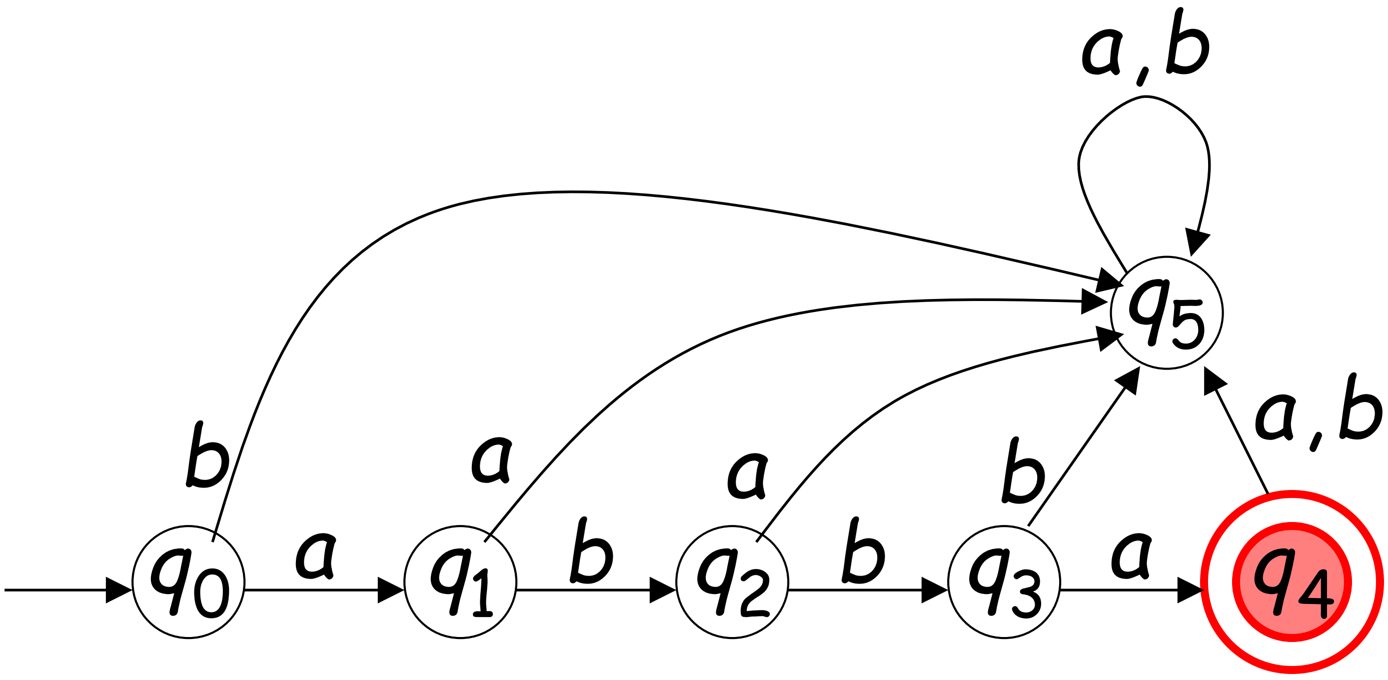


a b b a
a,b
q5
b
a
a
b
a,b
q0
a
q1
b
q2
b
q3
a
q4


a
b
b
a
a, b
b
a
a
b
a, b
q0
a
q1
b
q2
b
q3
a
q4
q5

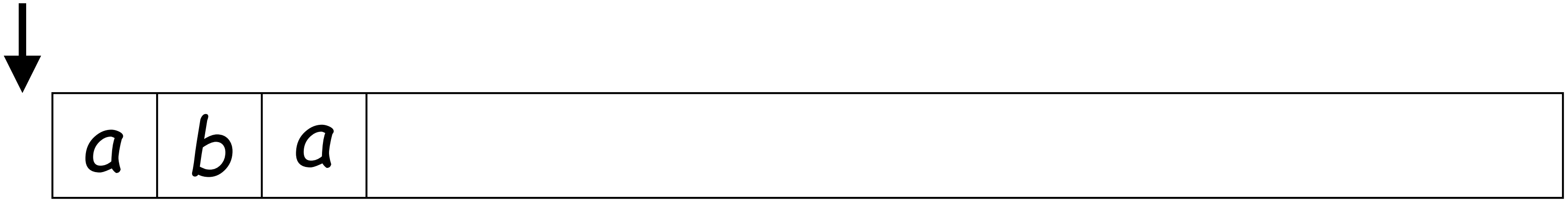
Input finished

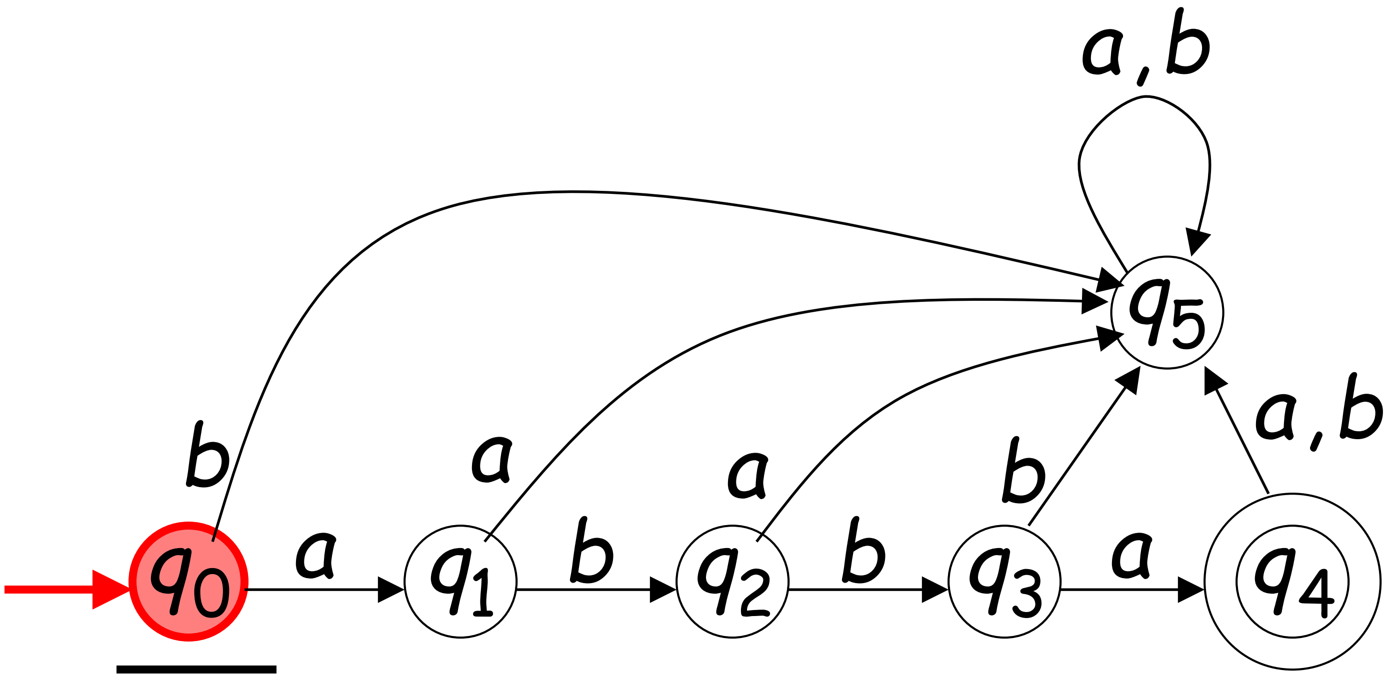


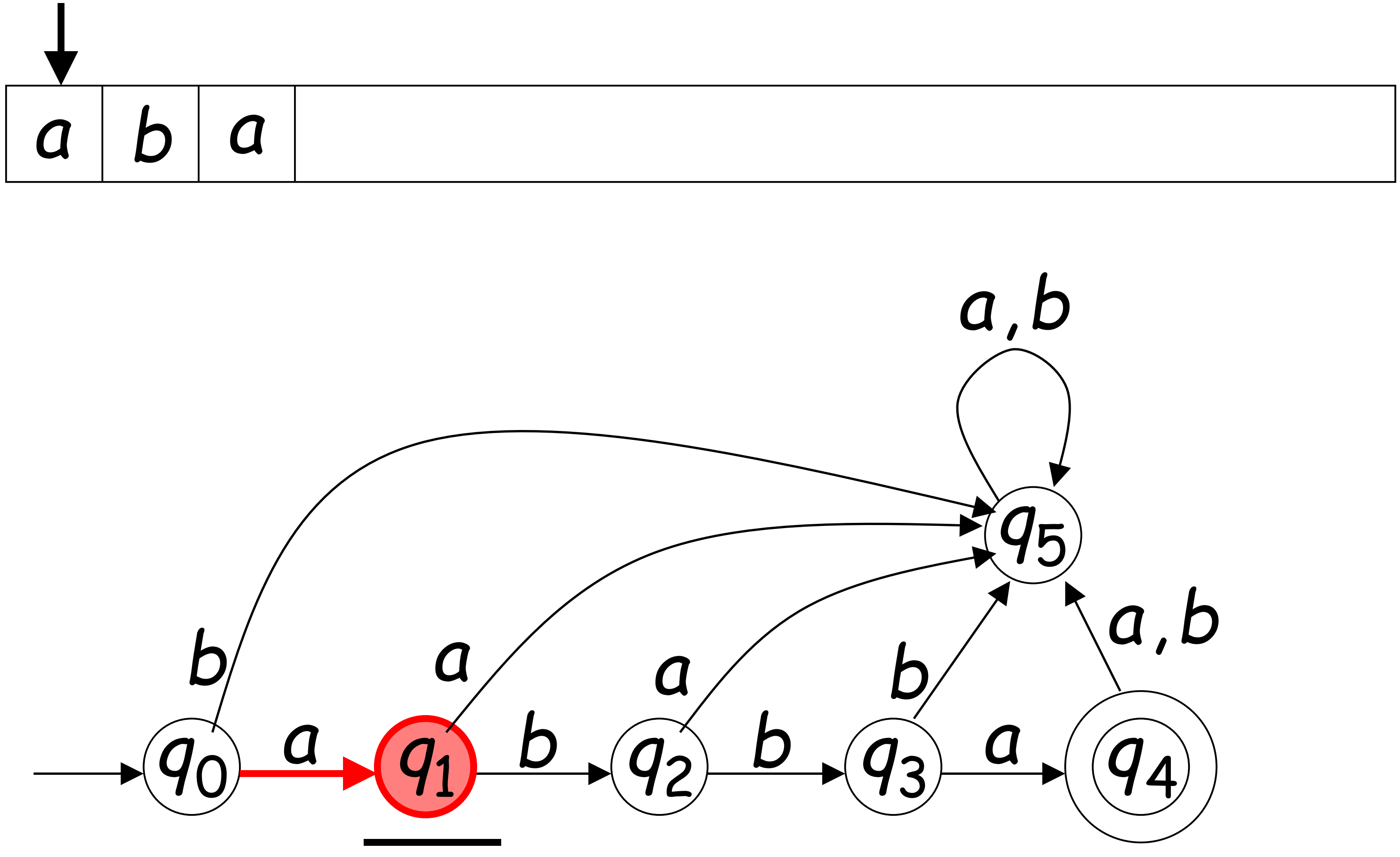


accept

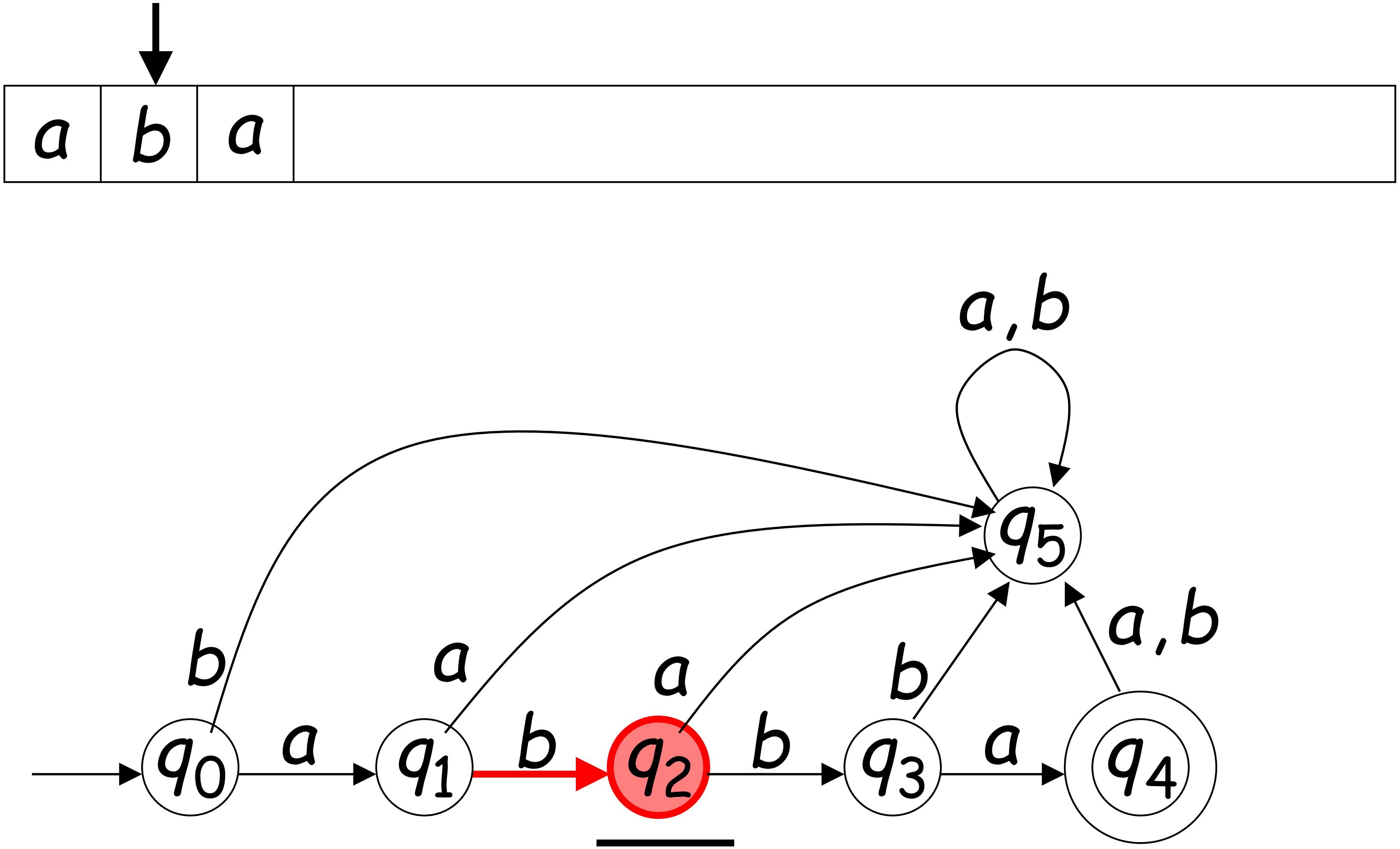
Rejection

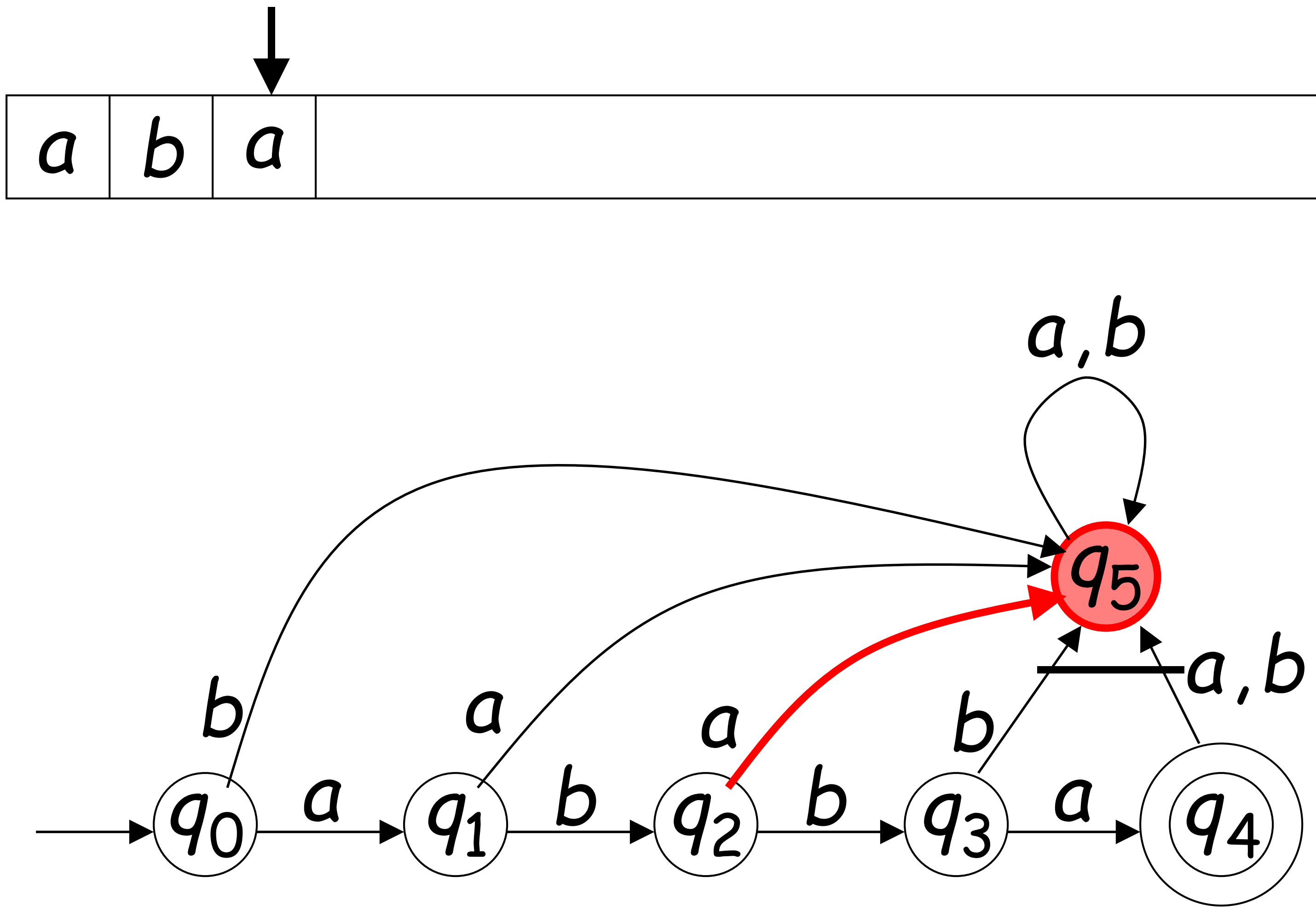




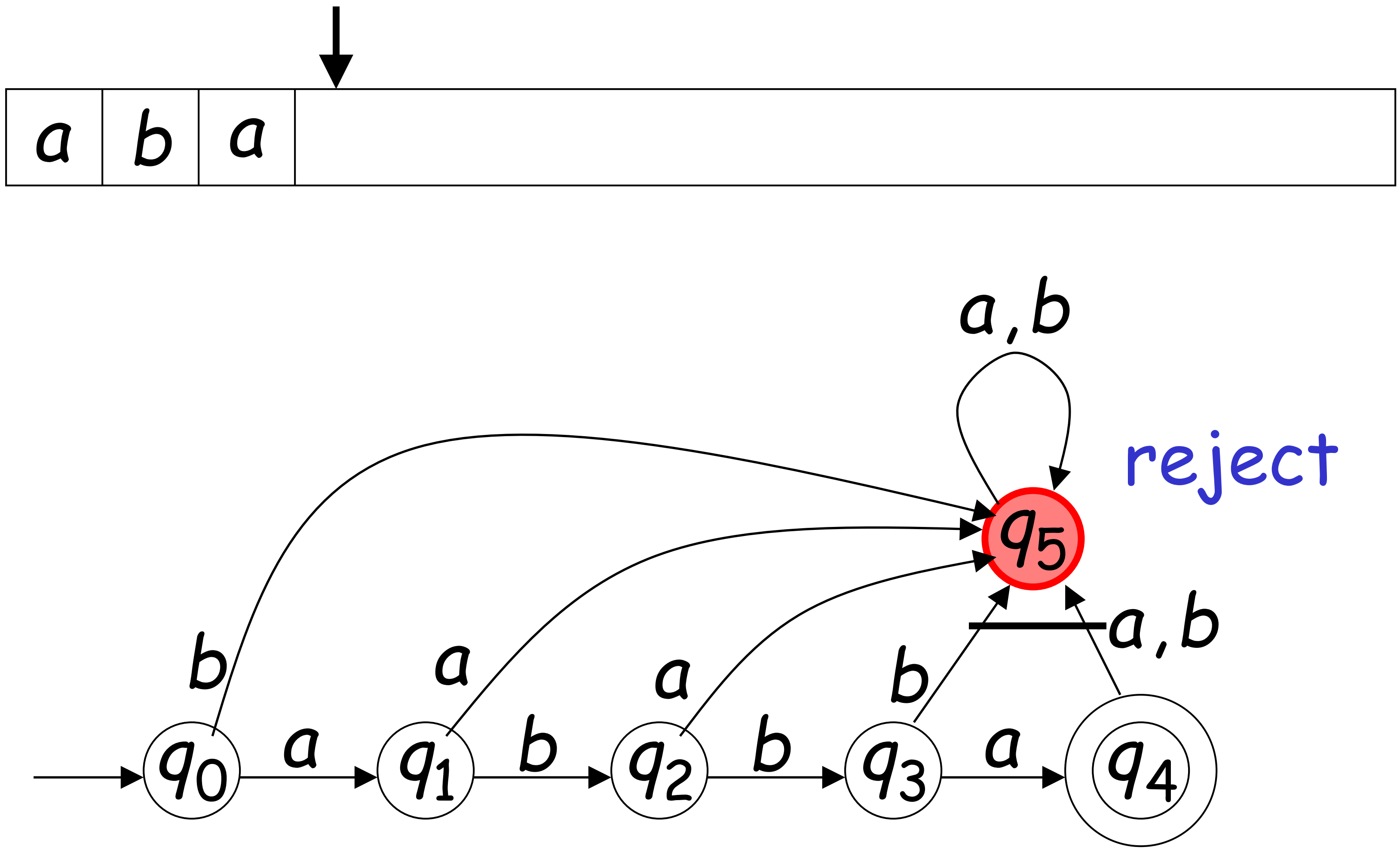


a
b
a
a,b
q5
a,b
b
a
a
b
q0
a
q1
b
q2
b
q3
a
q4


a
b
a
a, b
a, b
b
a
a
b
a
b
b
a
q_0
q_1
q_2
q_3
q_4
q_5

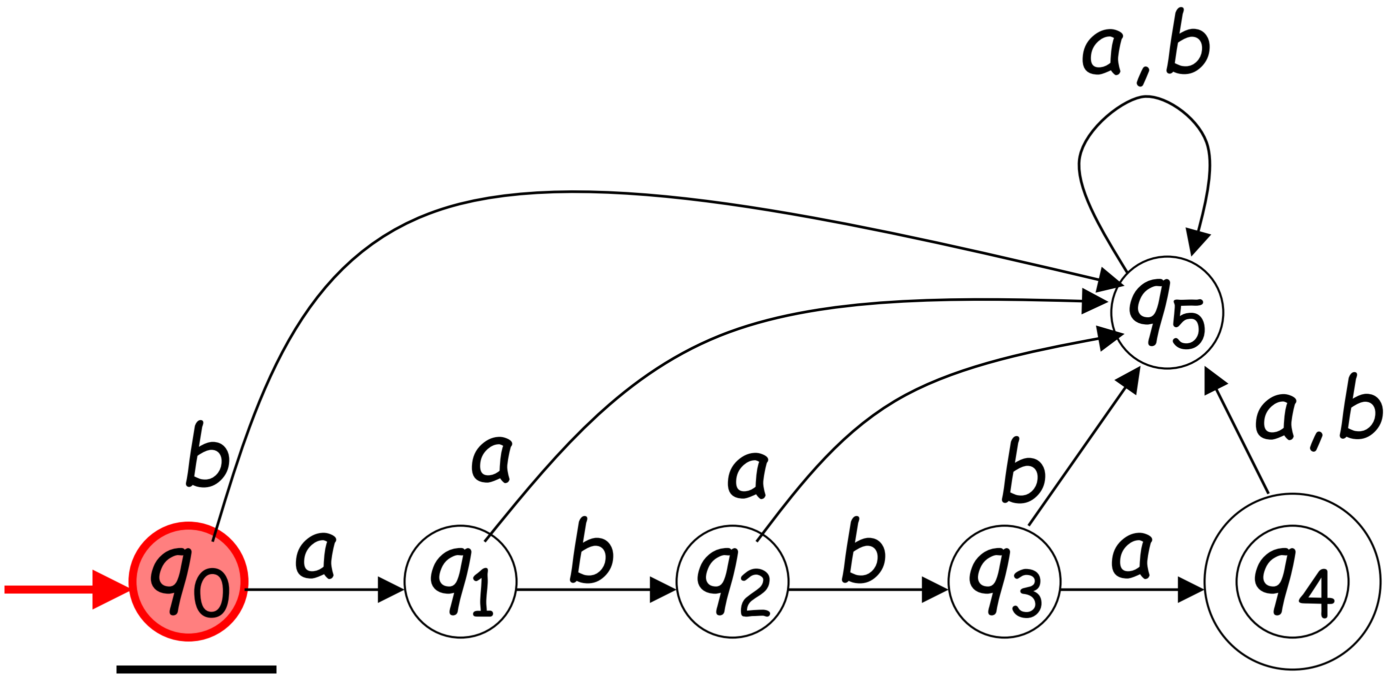

a
b
a
a,b
b
a
a
b
a,b
q0
a
q1
b
q2
b
q3
a
q4
q5

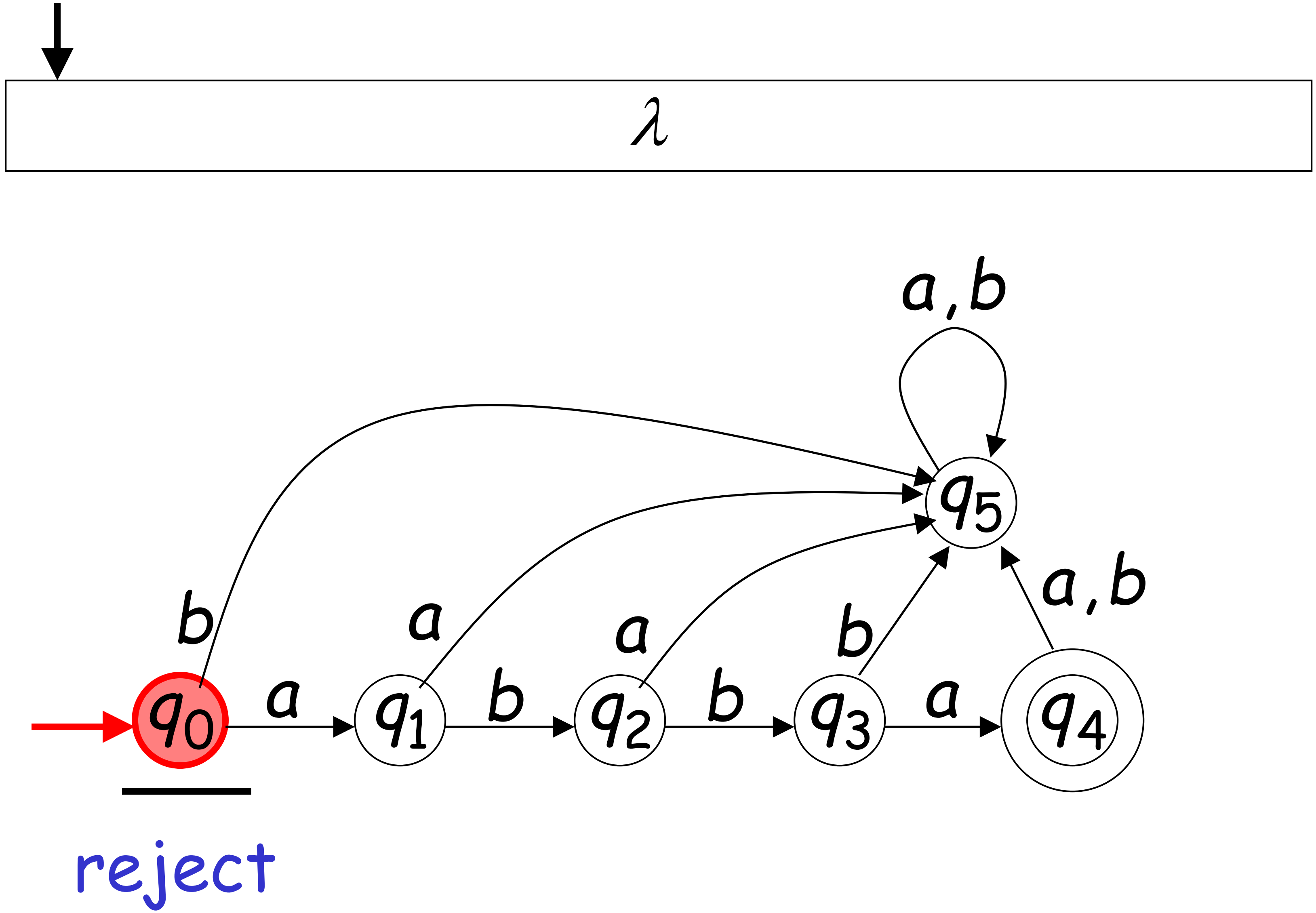
Input finished



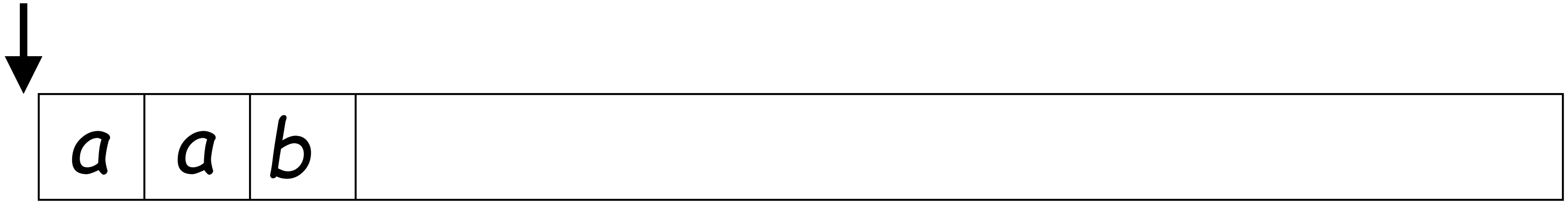
Another Rejection

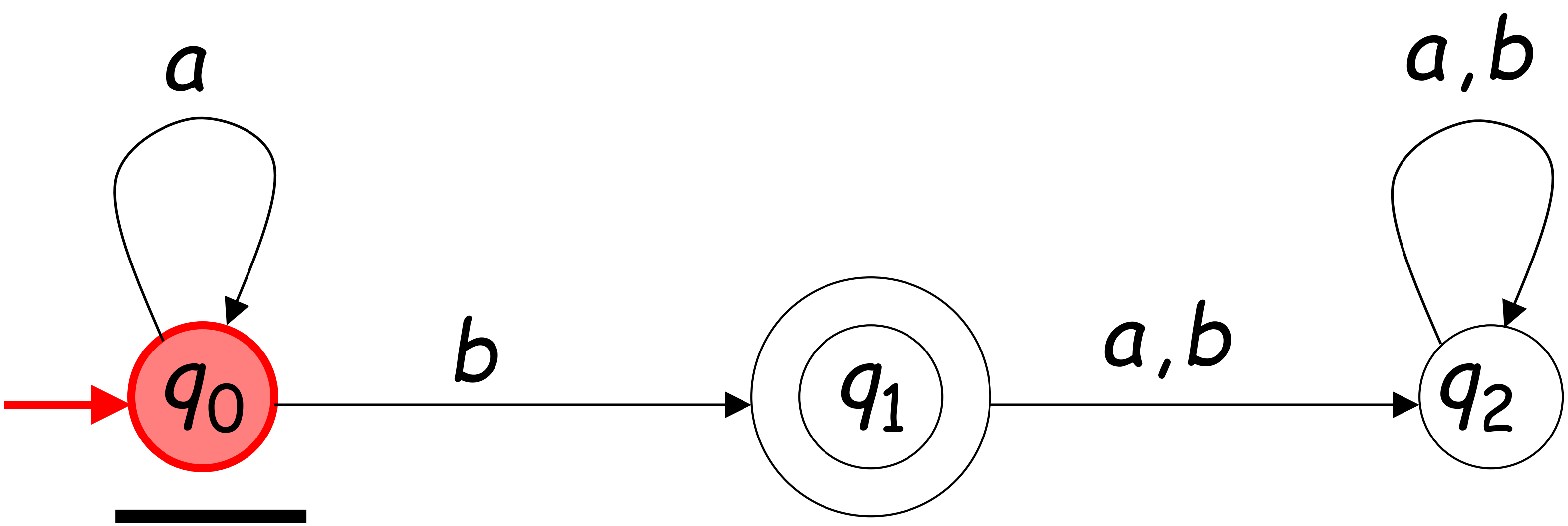
λ

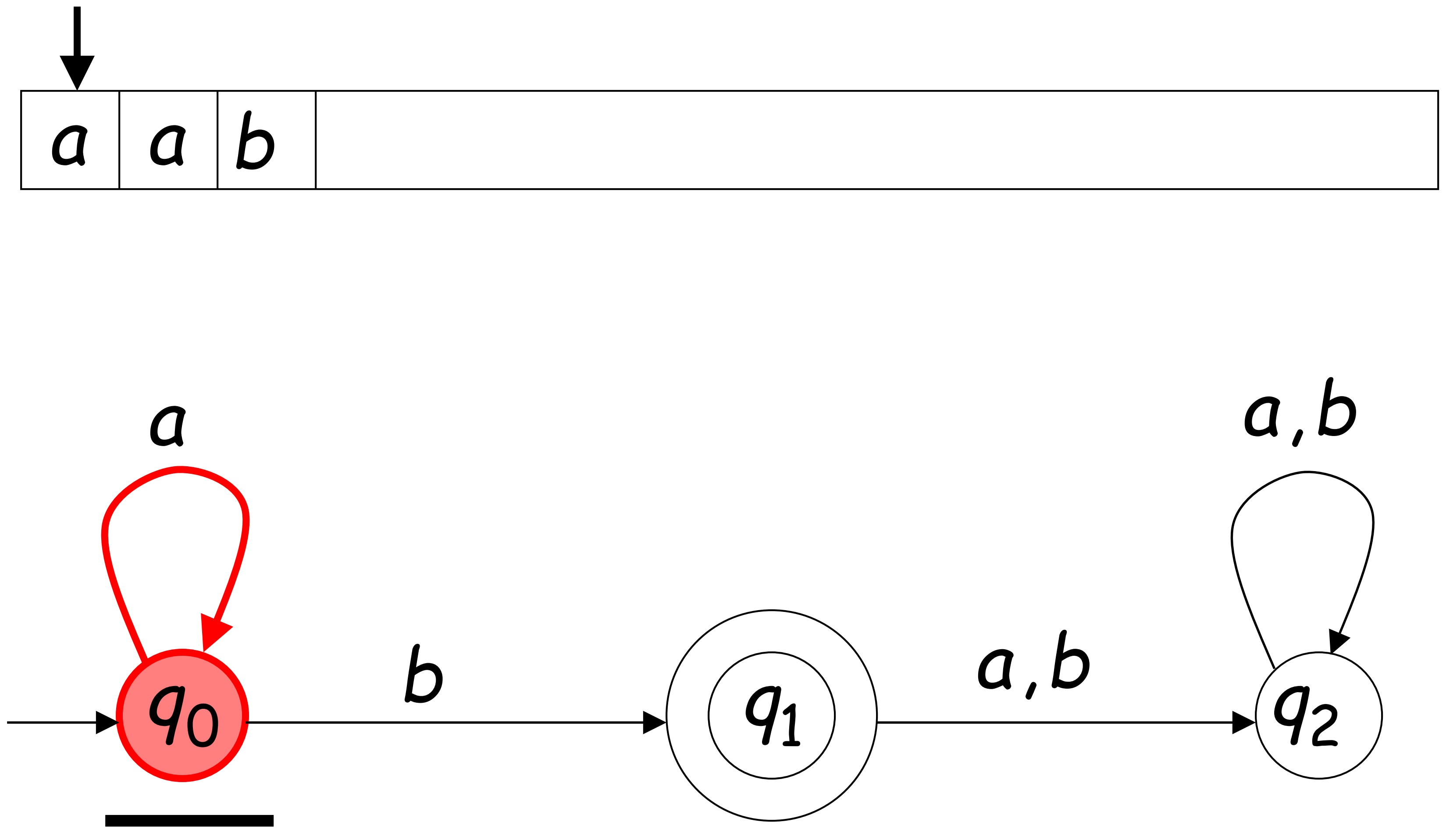



λ
a,b
b
a
a
b
a,b
q_0
a
q_1
b
q_2
b
q_3
a
q_4
q_5
reject

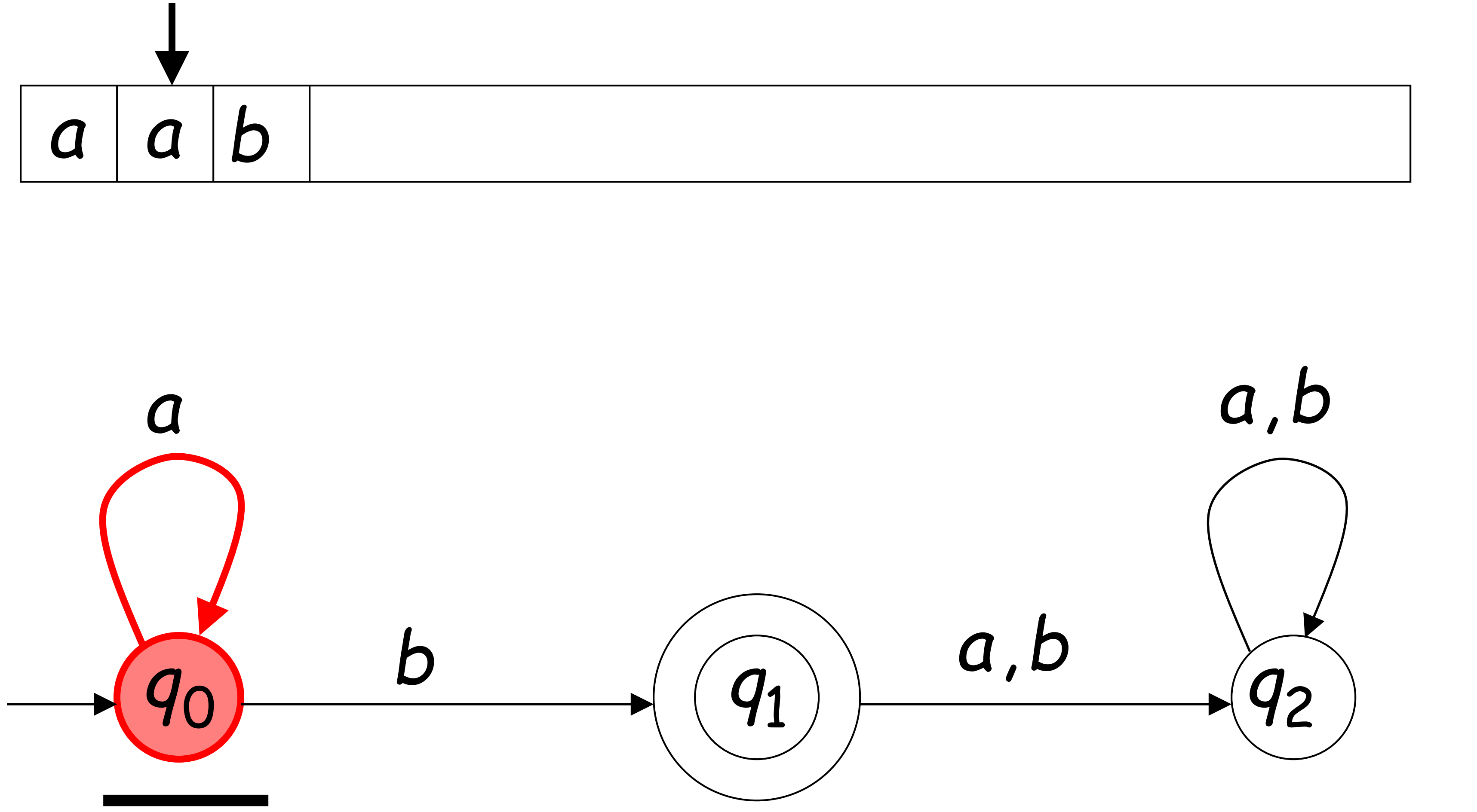
Another Example

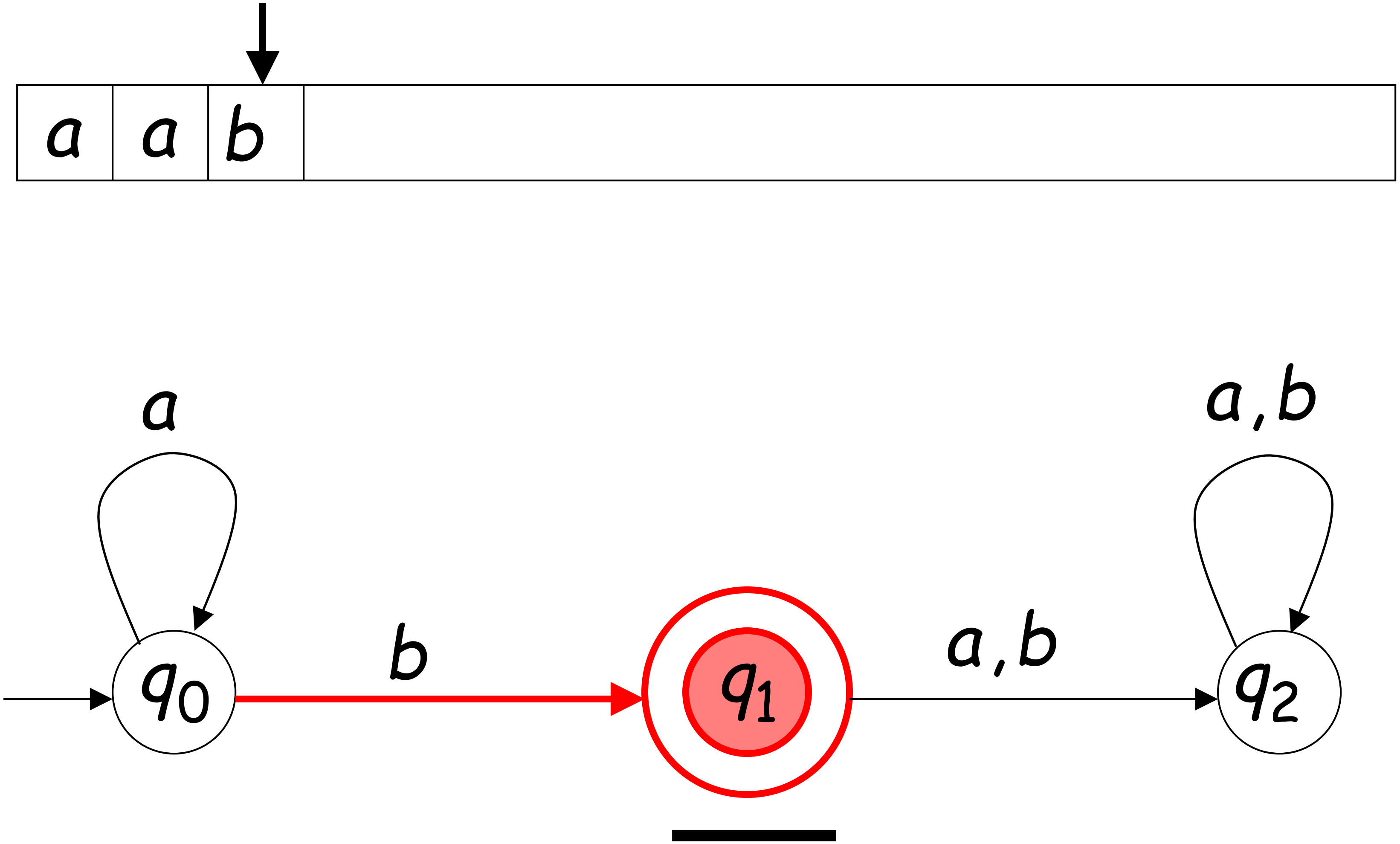




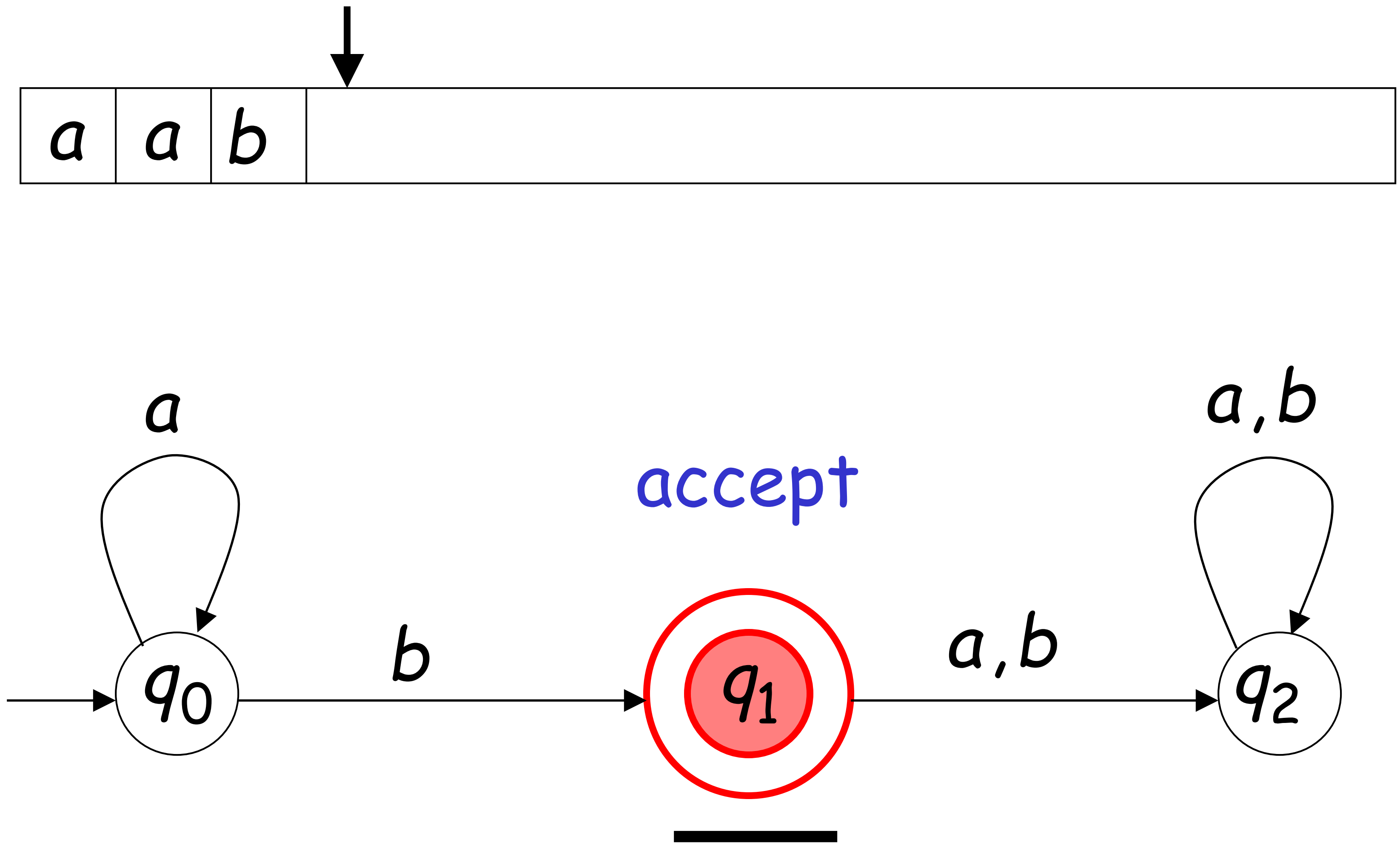


a
a
b
a
a,b
b
a,b
q0
q1
q2

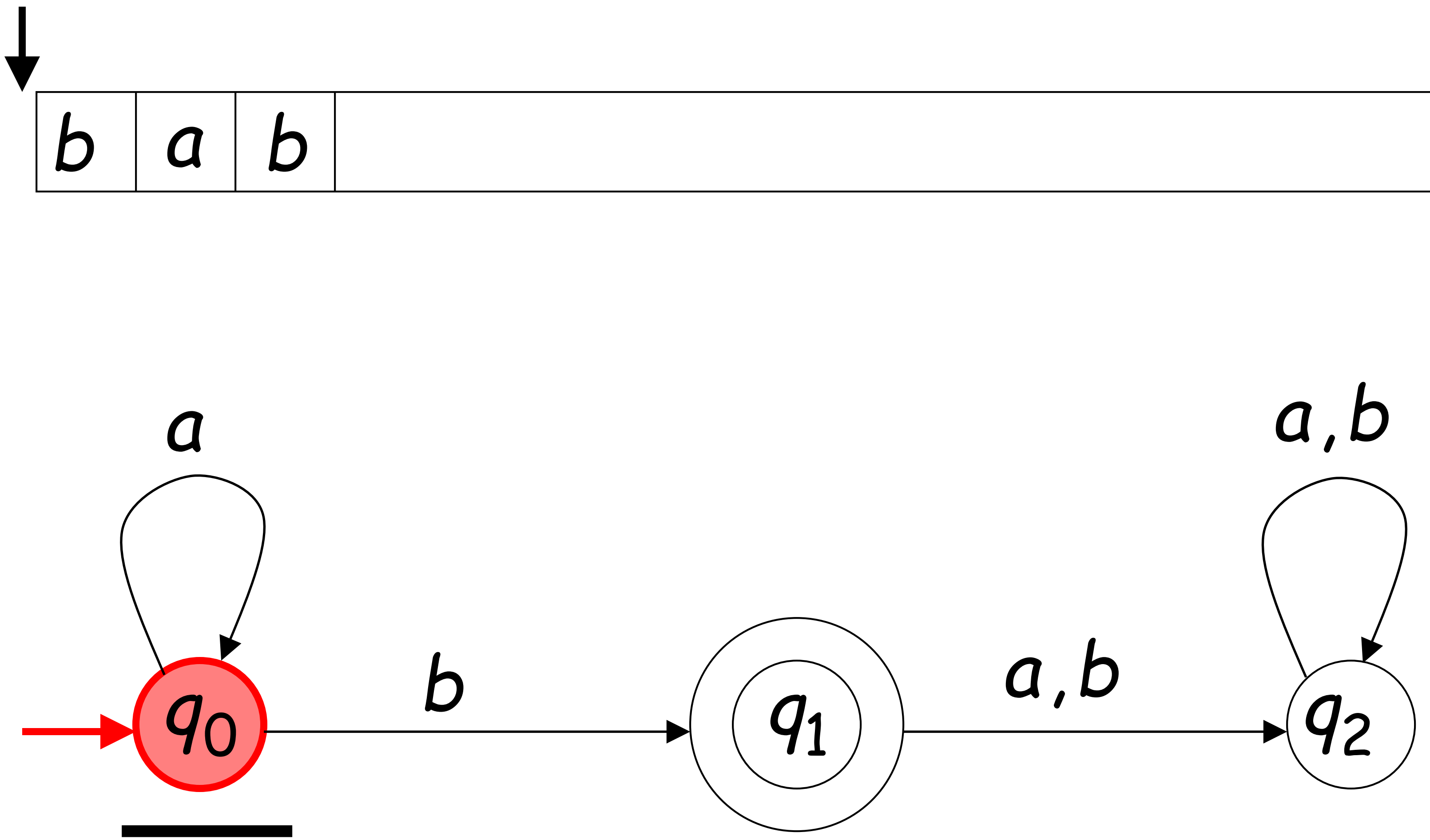

a a b
a
a,b
b
a,b
q_0
q_1
q_2

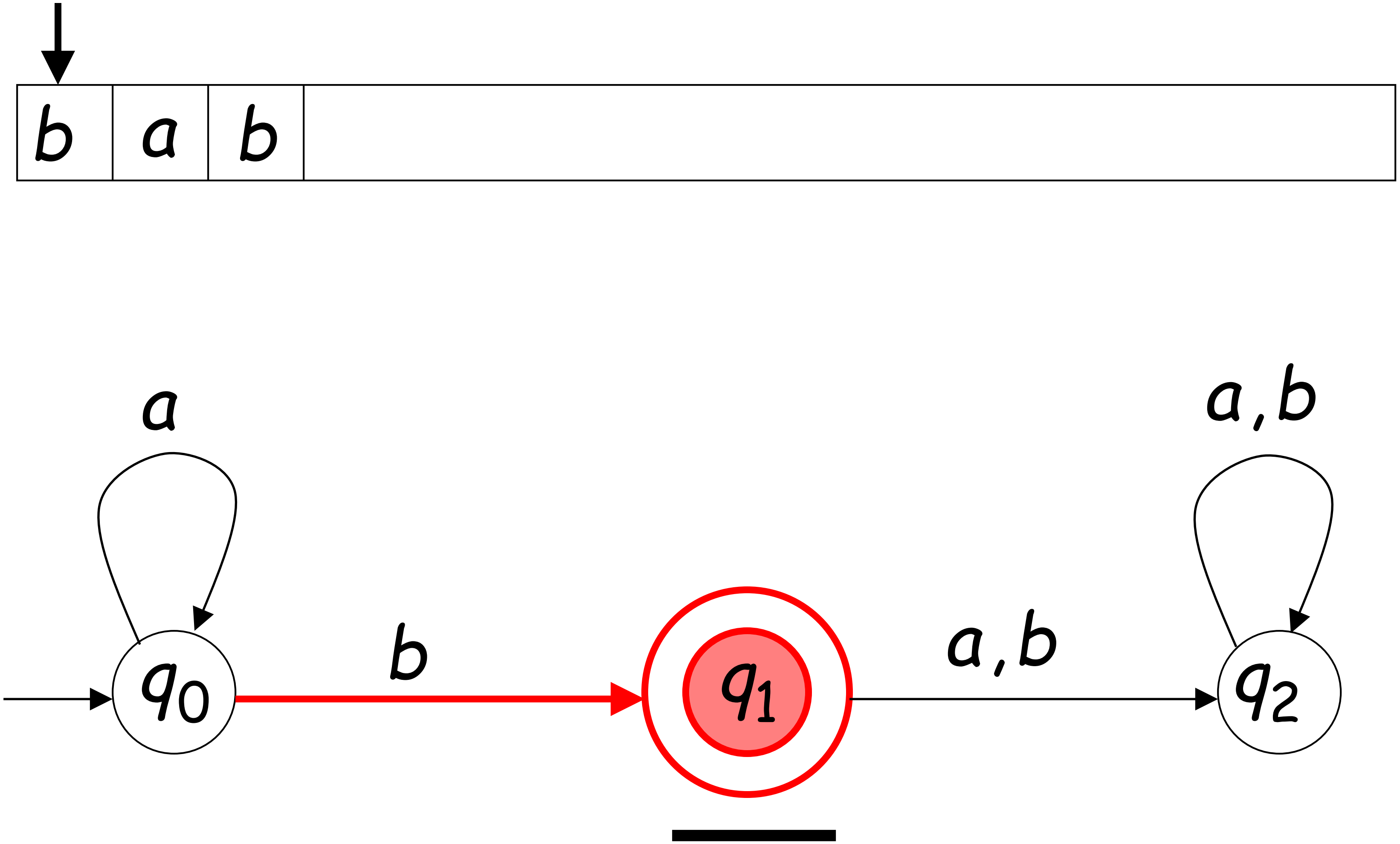

a a b
a
a,b
b
a,b
q0
q1
q2

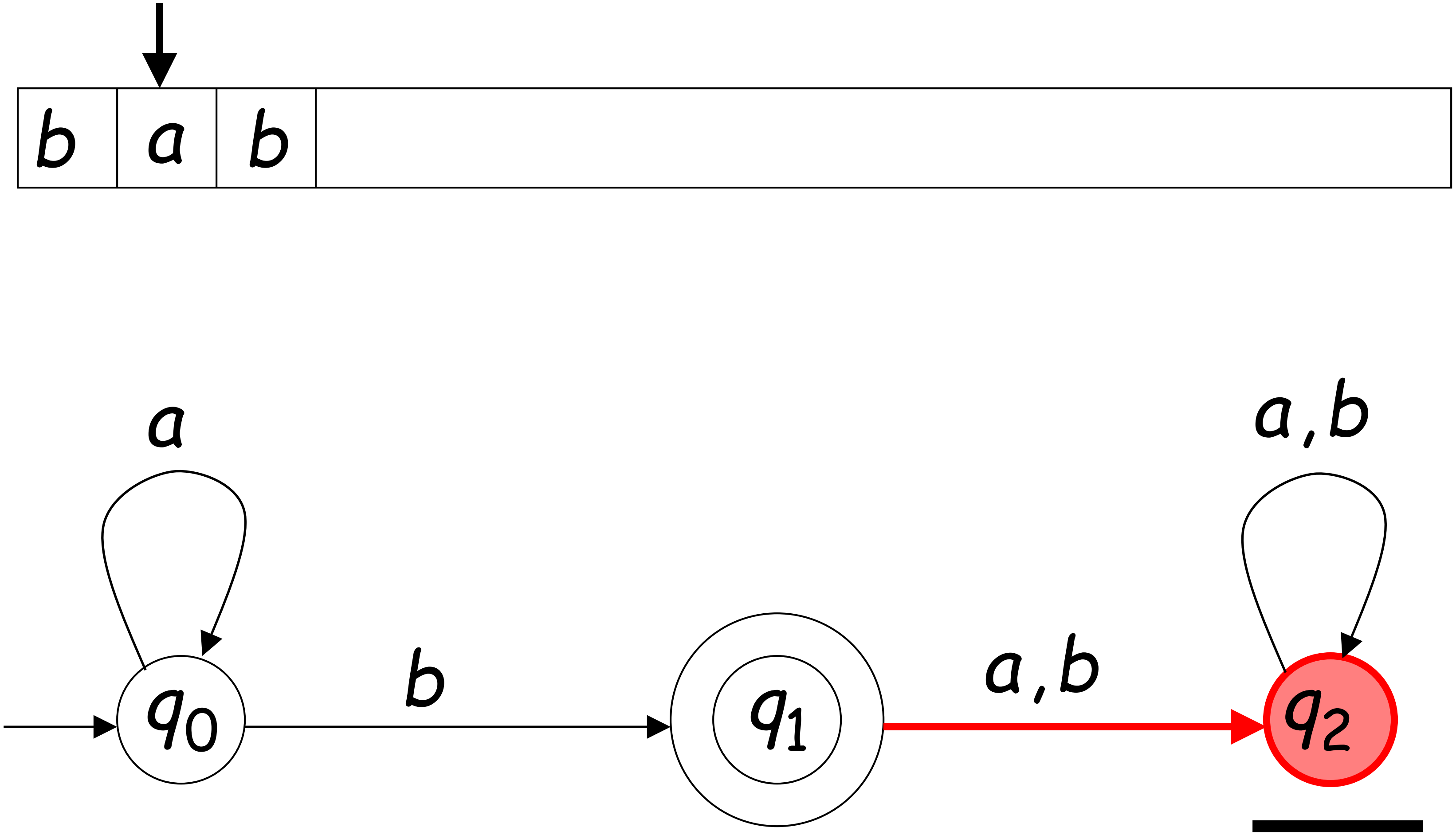
Input finished



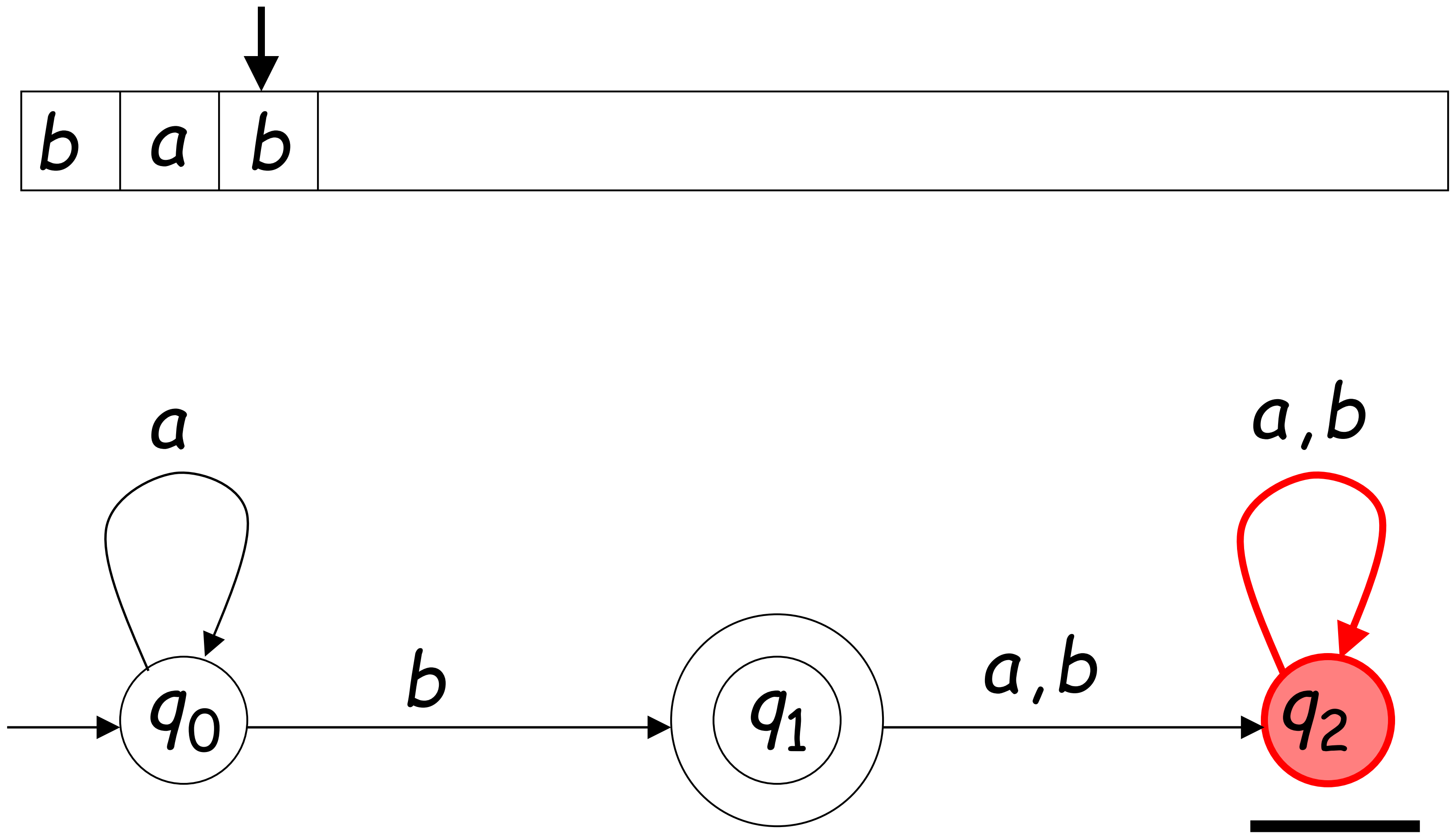
Rejection Example




b
a
b
a
a,b
b
q0
q1
a,b
q2

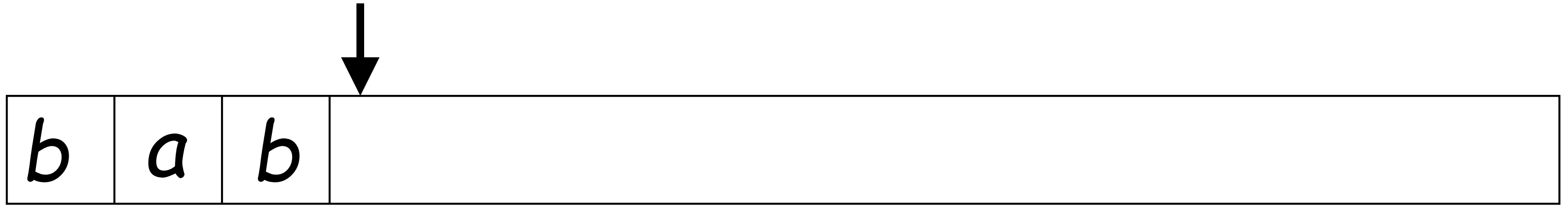


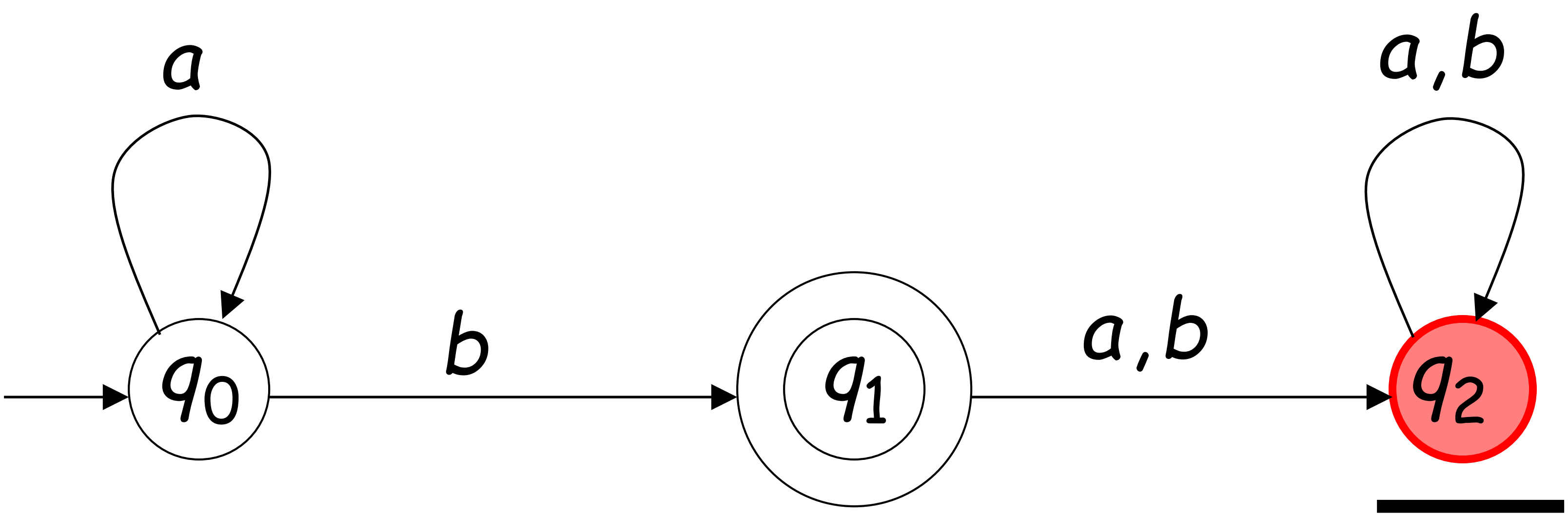
b a b
a
a,b
b
a,b
q0
q1
q2



b
a
b
a
a,b
b
q_0
q_1
a,b
q_2

Input finished





reject

Languages Accepted by FAs

FA M

Definition:

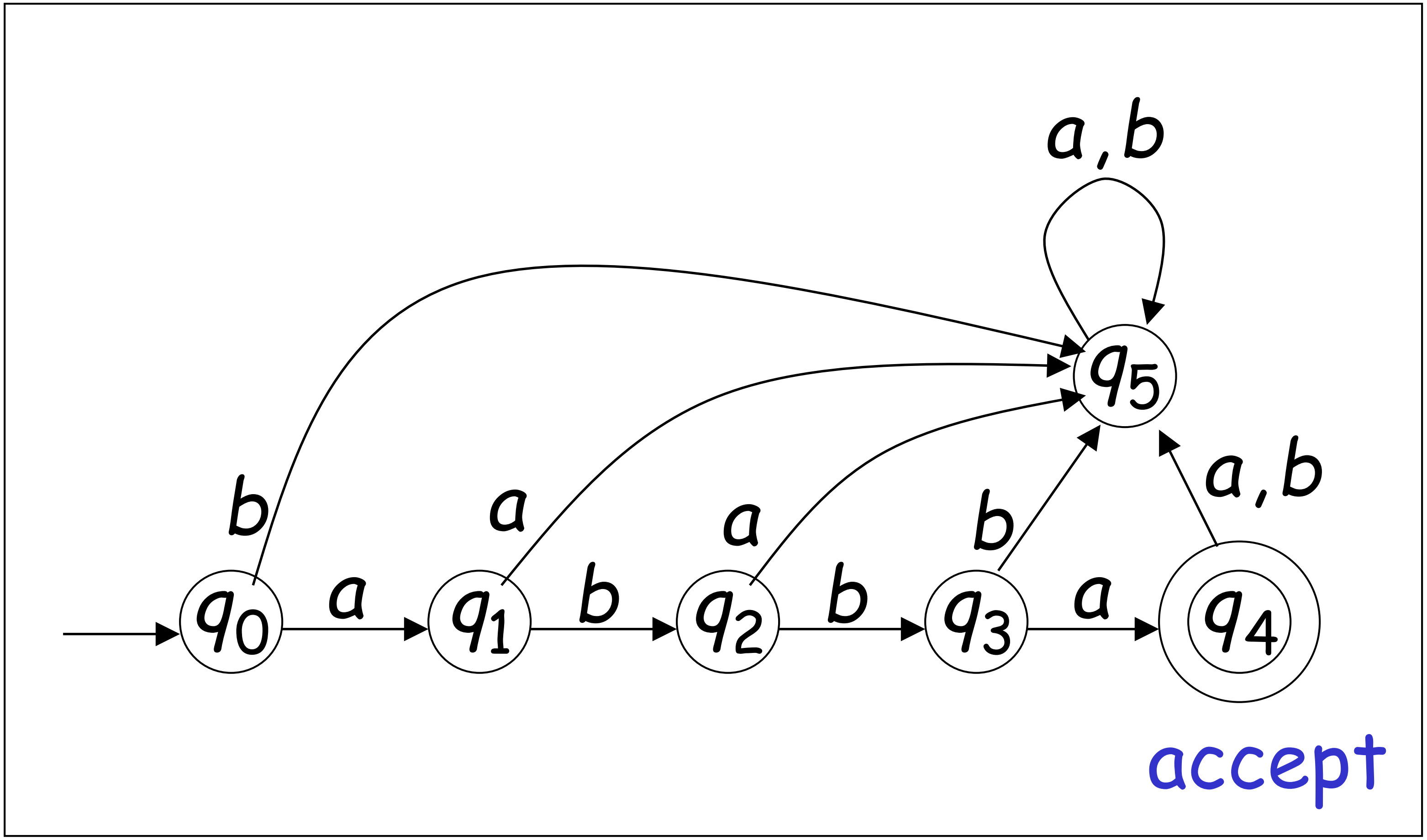
The language $L(M)$ contains all input strings accepted by M

$L(M)$ = { strings that bring M to an accepting state}

Example

$$L(M) = \{abba\}$$

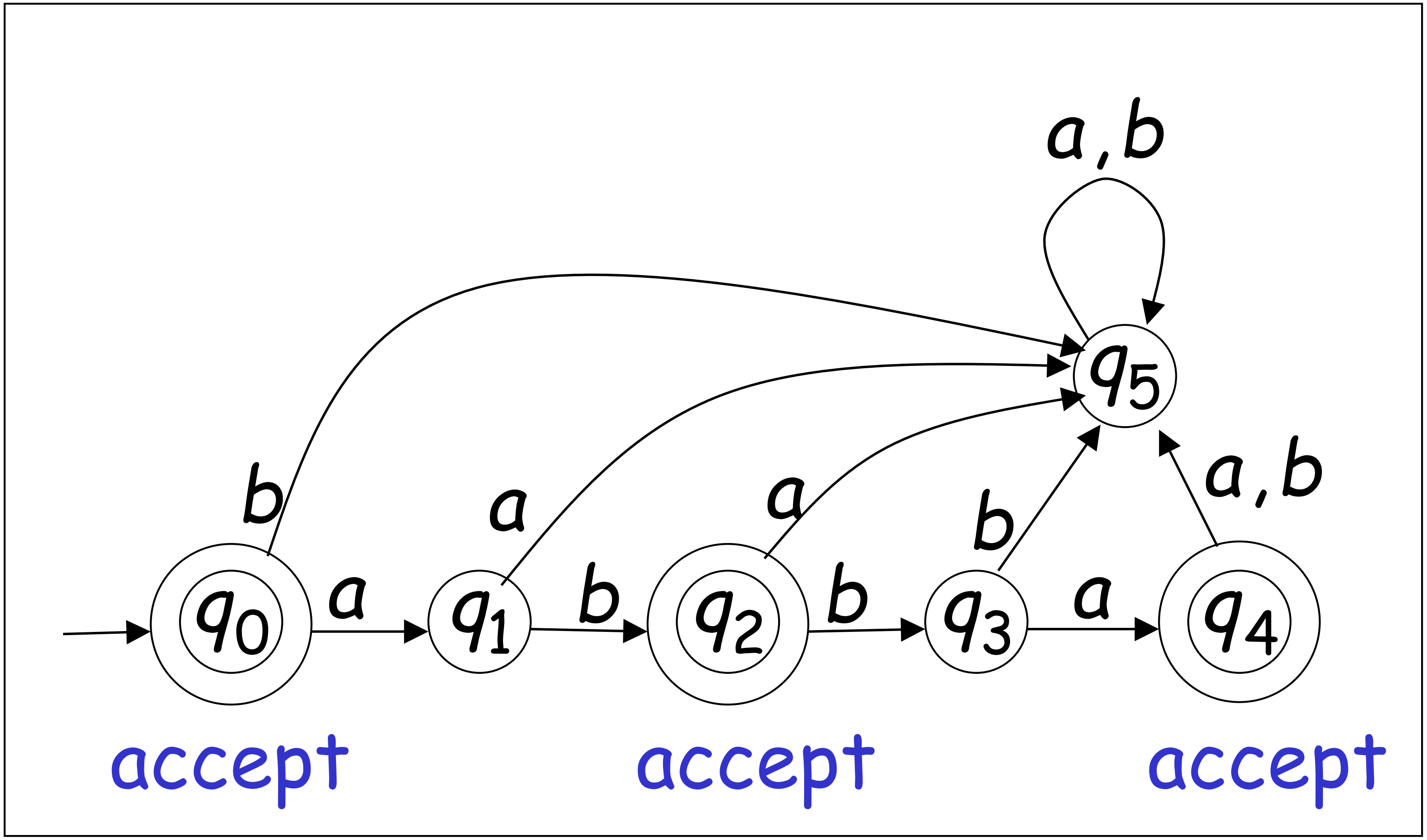
M



Example

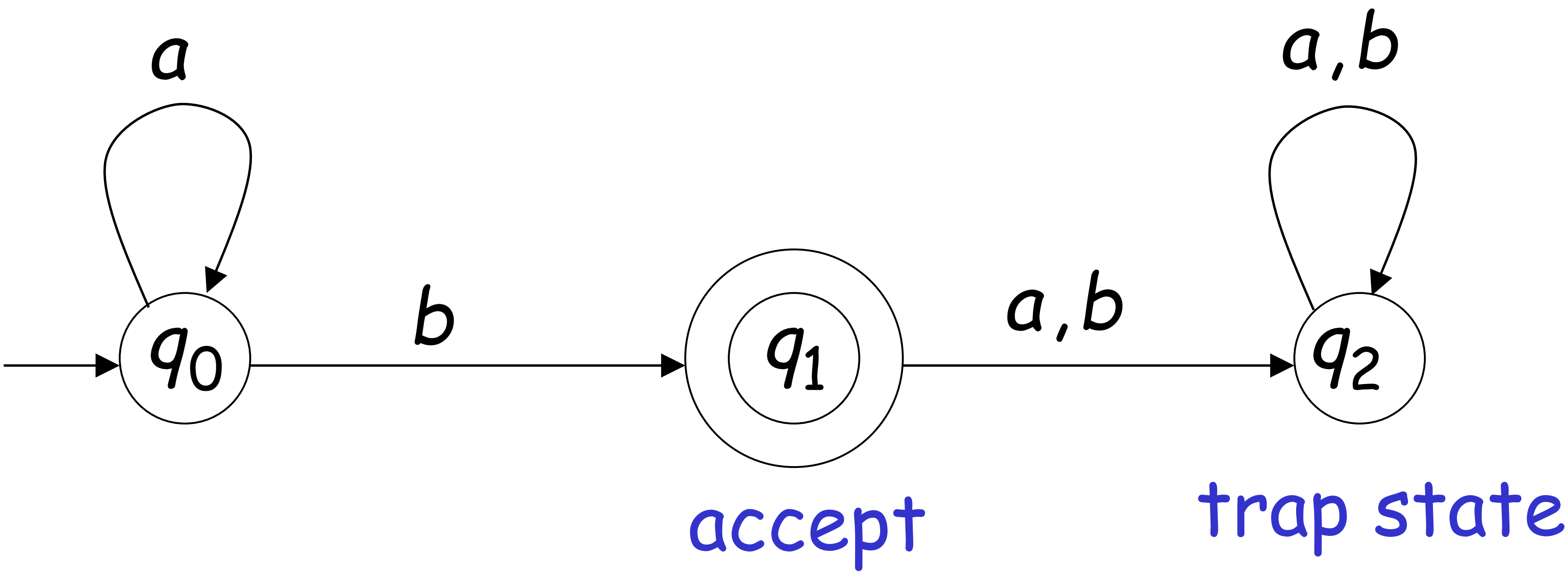
$$L(M) = \{\lambda, ab, abba\}$$

M



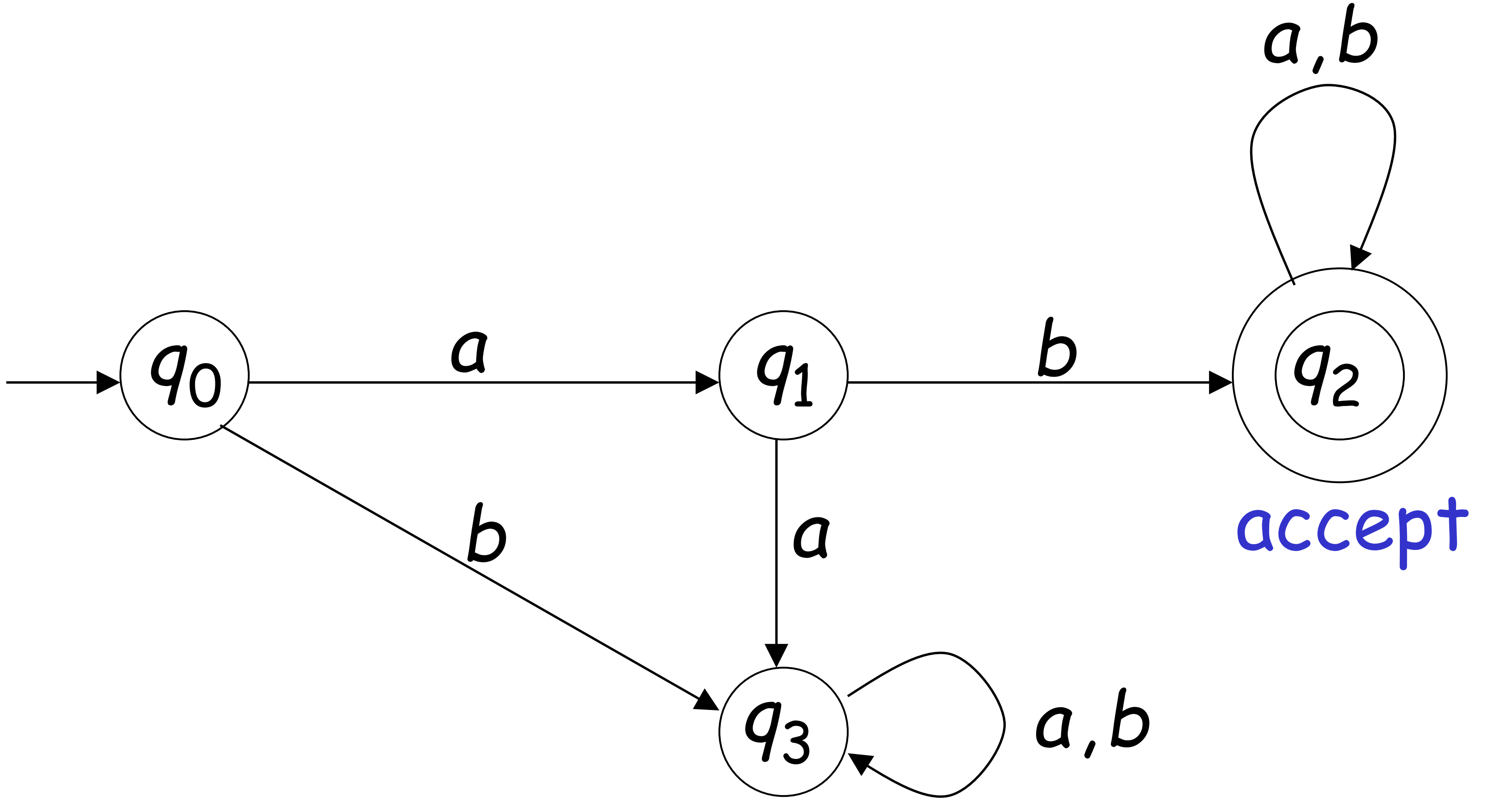
Example

$$L(M) = \{a^n b : n \geq 0\}$$



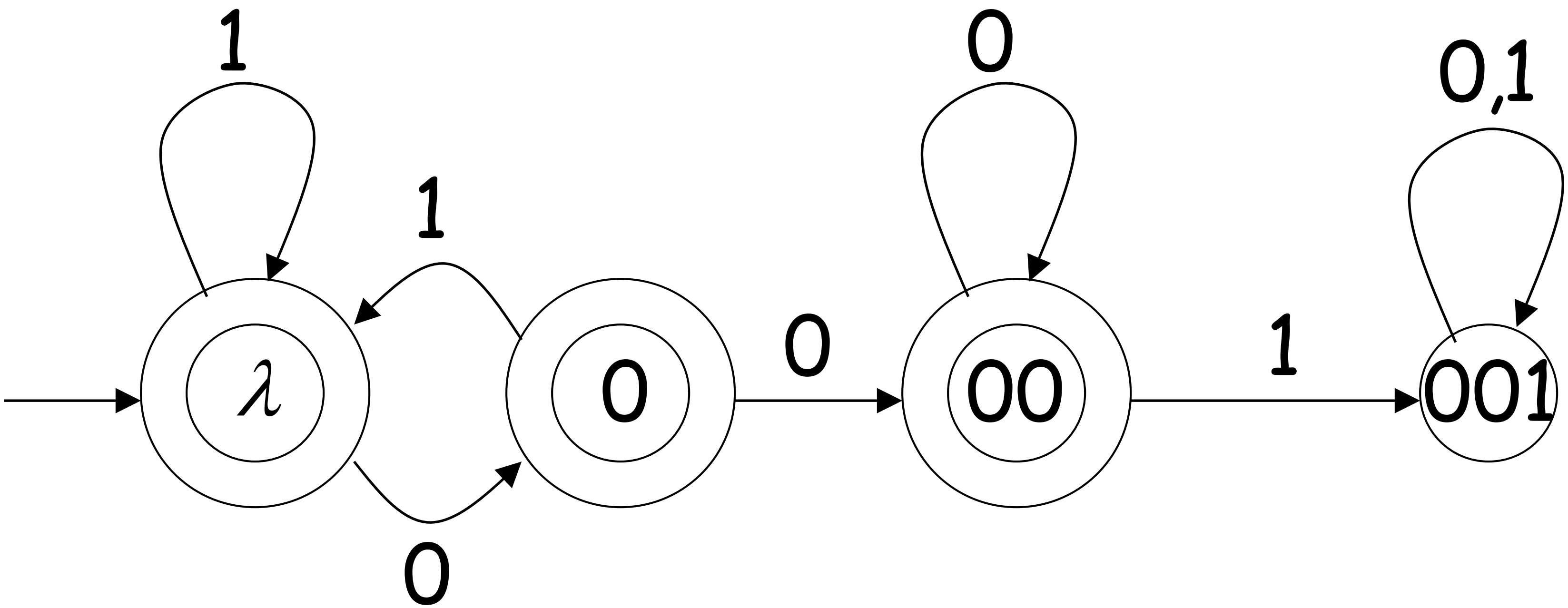
Example

$L(M)$ = { all strings with prefix ab }



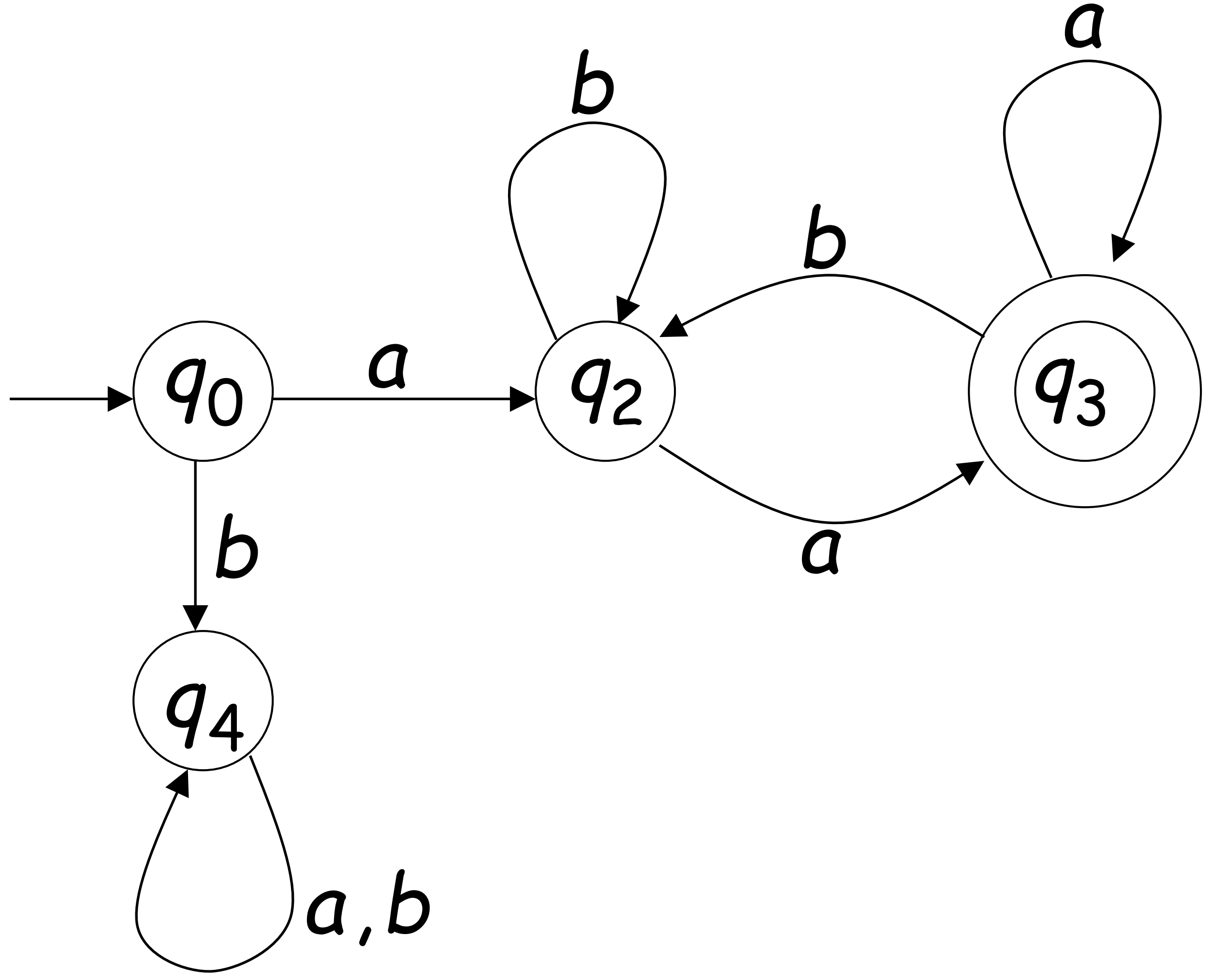
Example

$L(M) = \{$ all strings without substring 001 $\}$



Example

$$L(M) = \{awa : w \in \{a,b\}^*\}$$



Regular Languages

Definition:

A language L is regular if there is FA M such that $L = L(M)$

Observation:

All languages accepted by FAs form the family of regular languages

Examples of regular languages:

$\{abba\}$ $\{\lambda, ab, abba\}$

$\{awa : w \in \{a,b\}^*\}$ $\{a^n b : n \geq 0\}$

{ all strings with prefix ab }

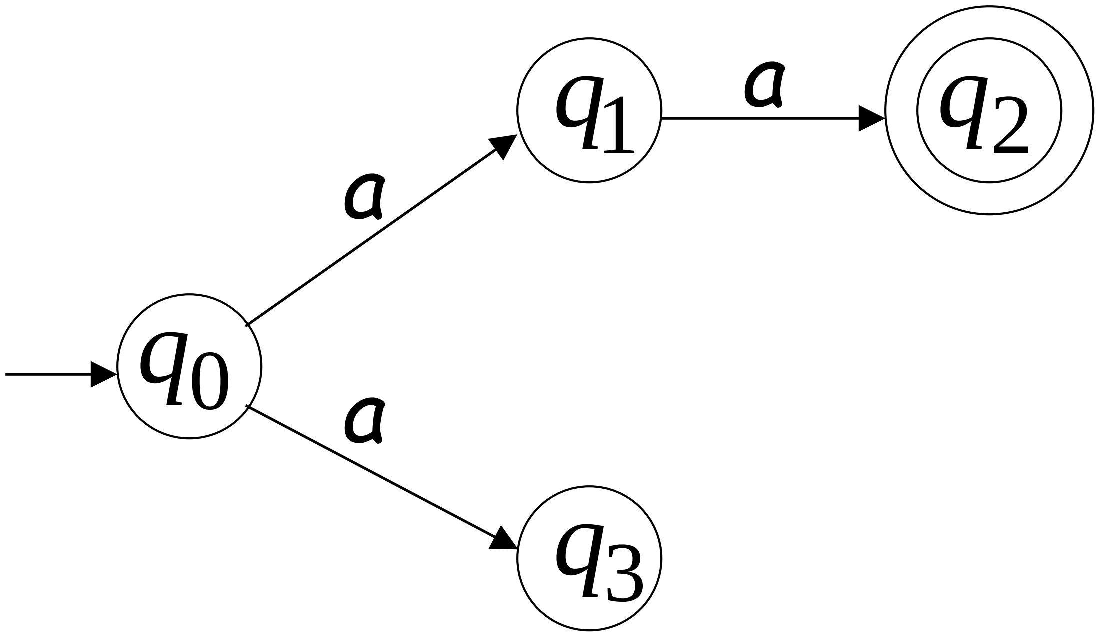
{ all strings without substring 001 }

There exist automata that accept these Languages.

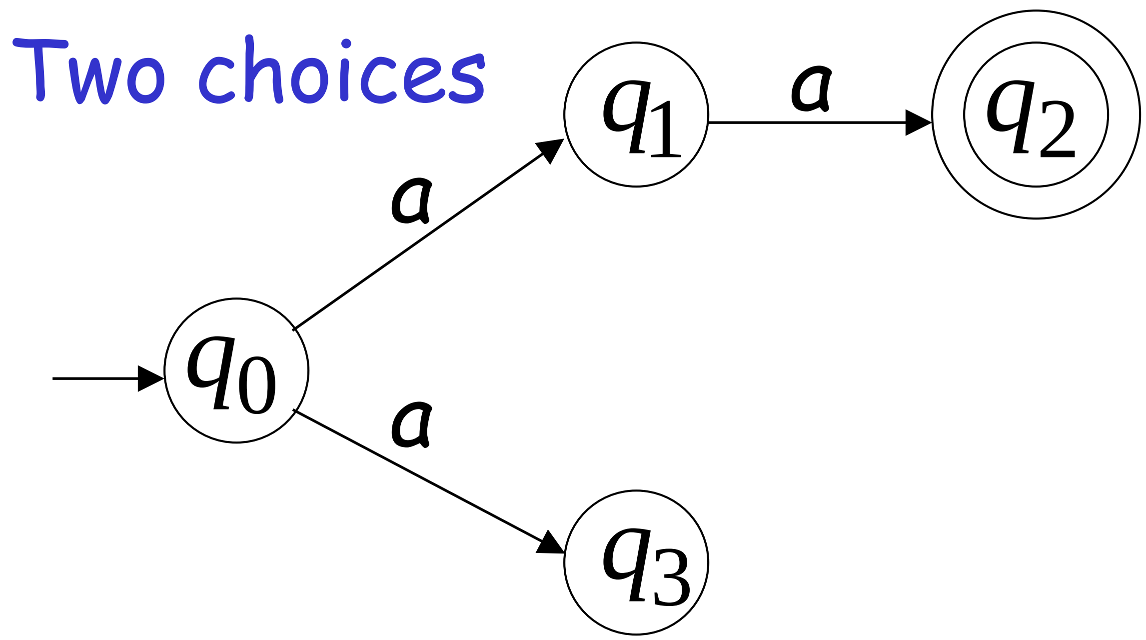
Non-Deterministic Finite Automata

Nondeterministic Finite Automaton (NFA)

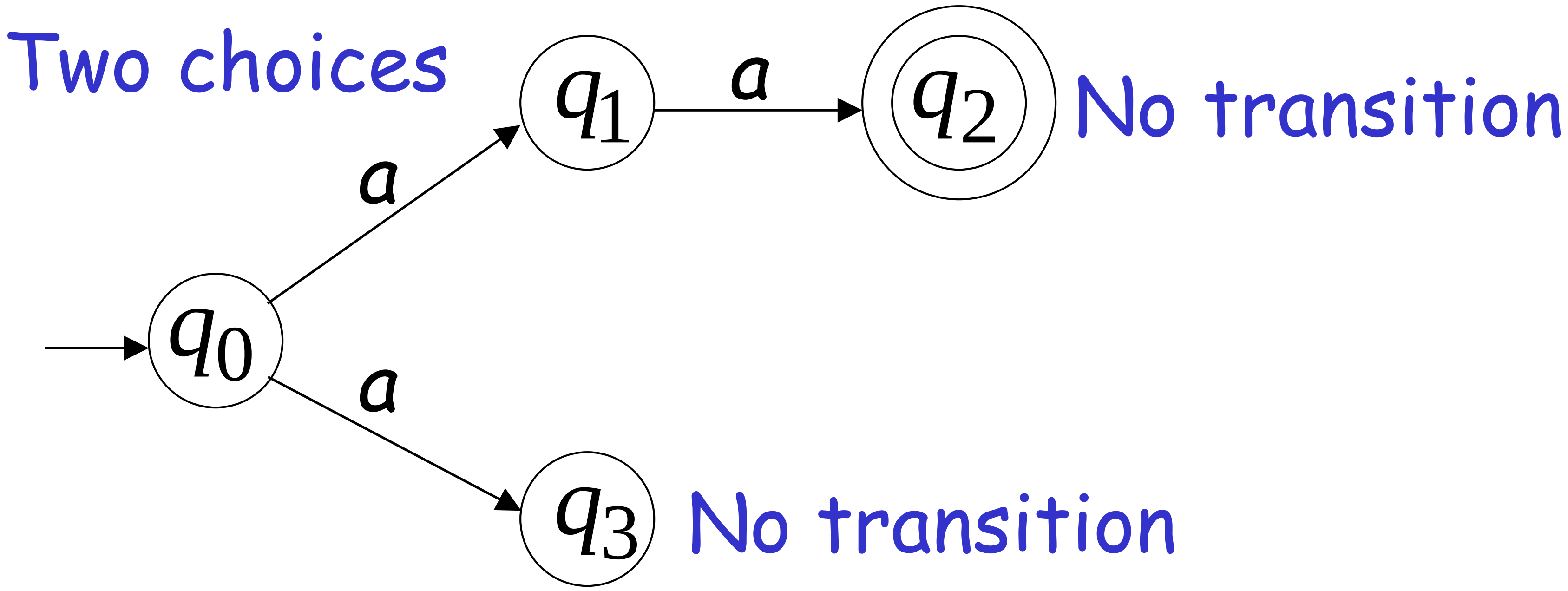
Alphabet = $\{a\}$



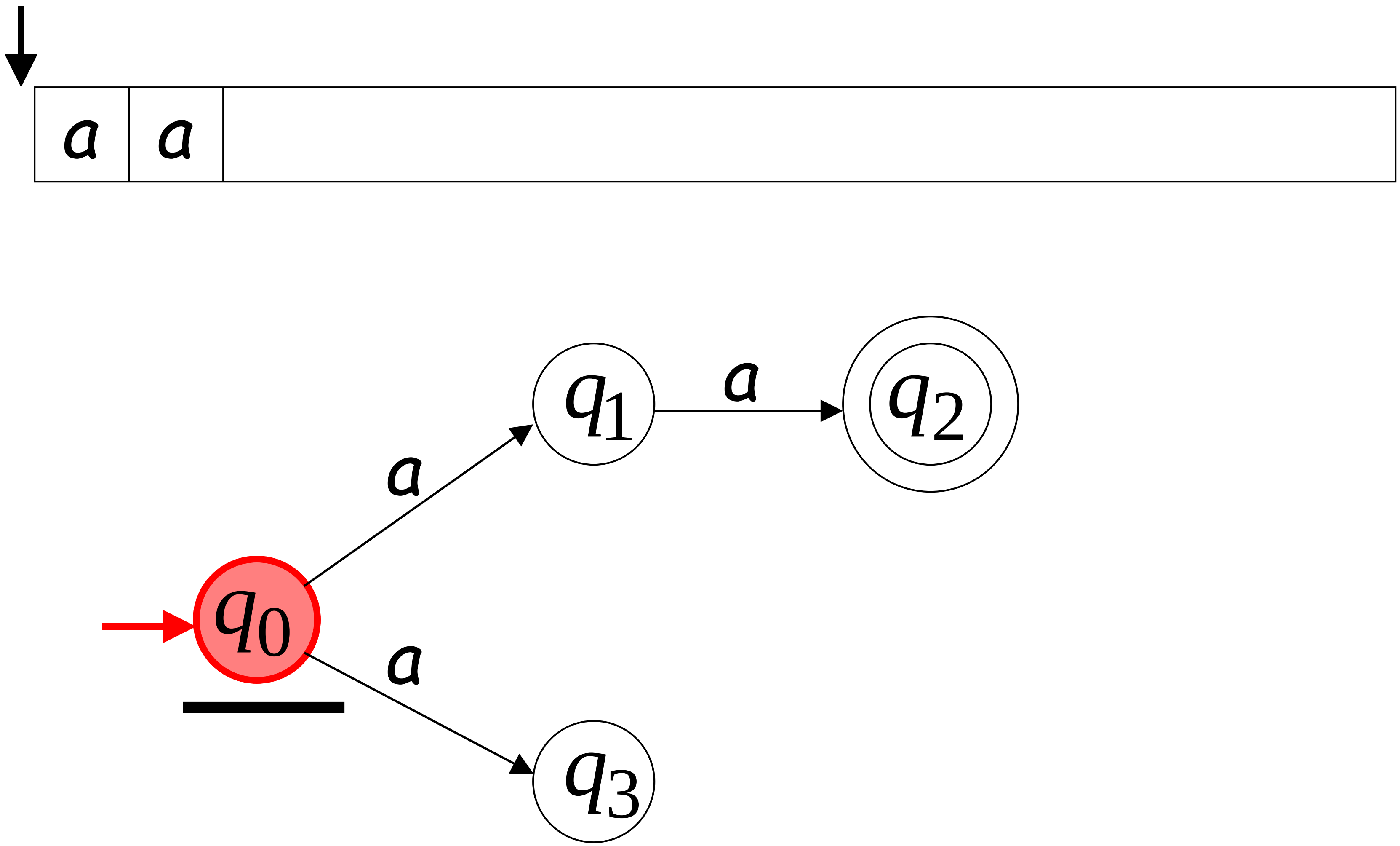
Alphabet = $\{a\}$



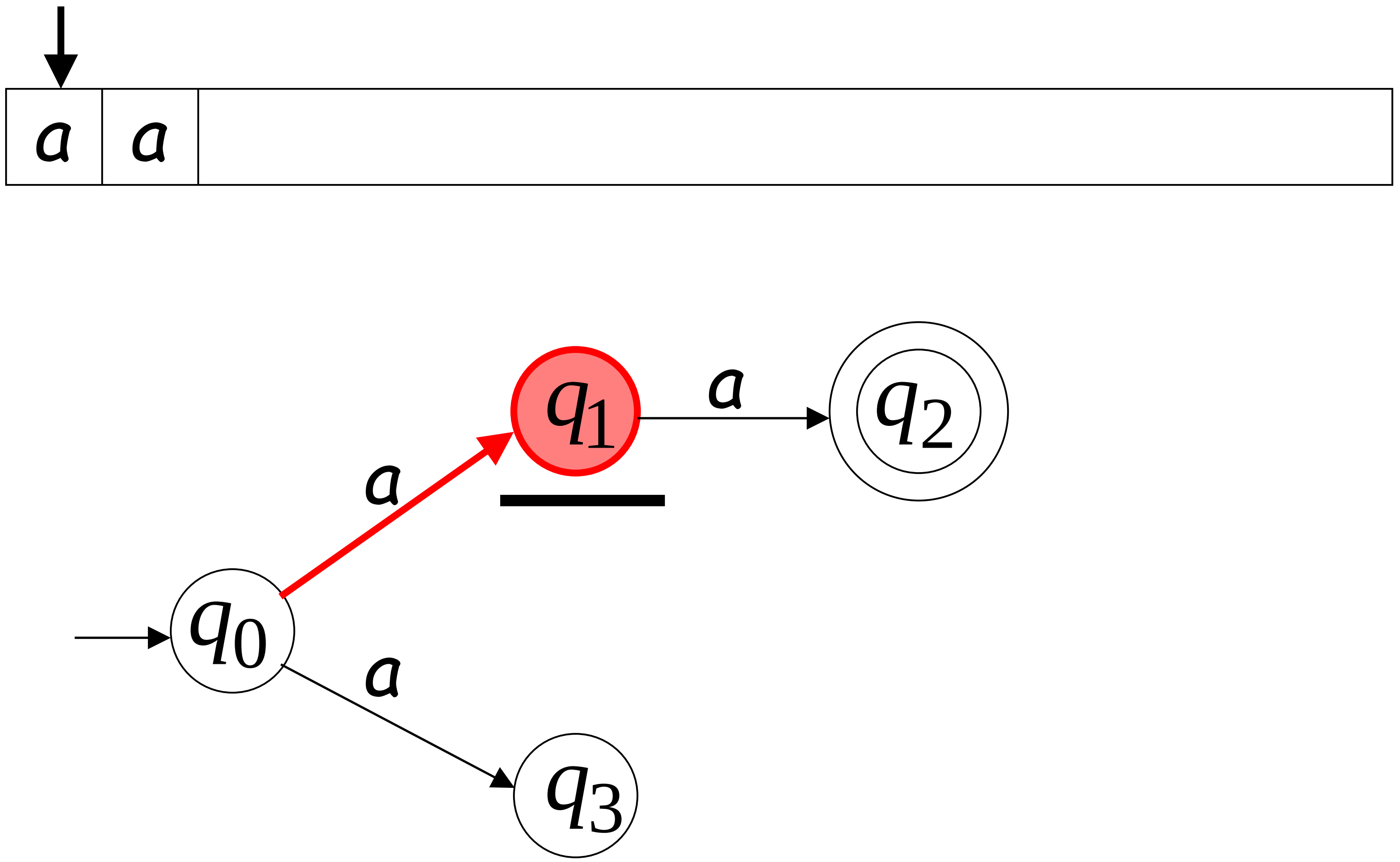
Alphabet = $\{a\}$



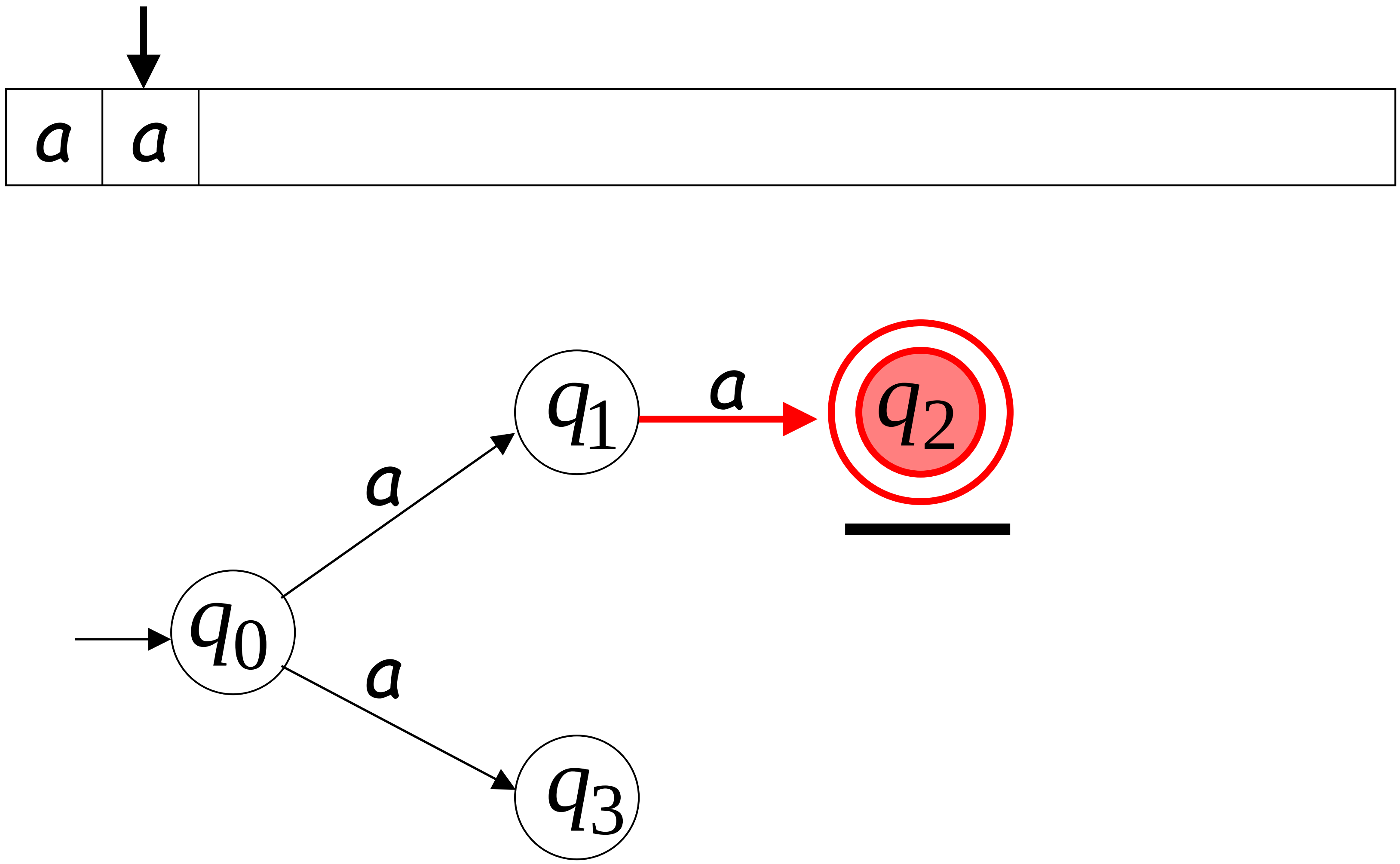
First Choice



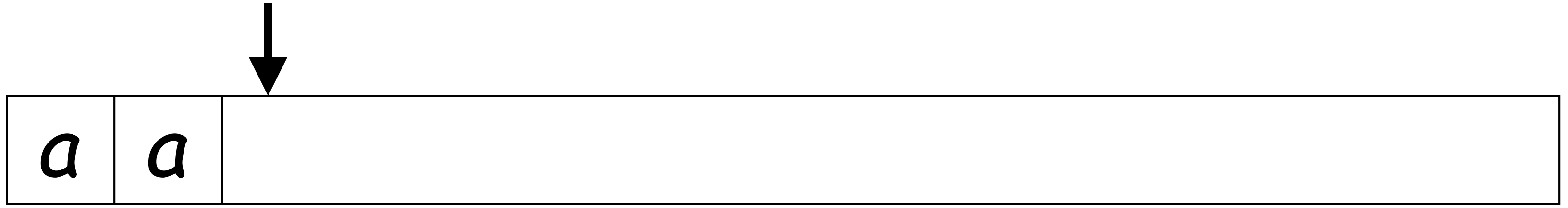
First Choice



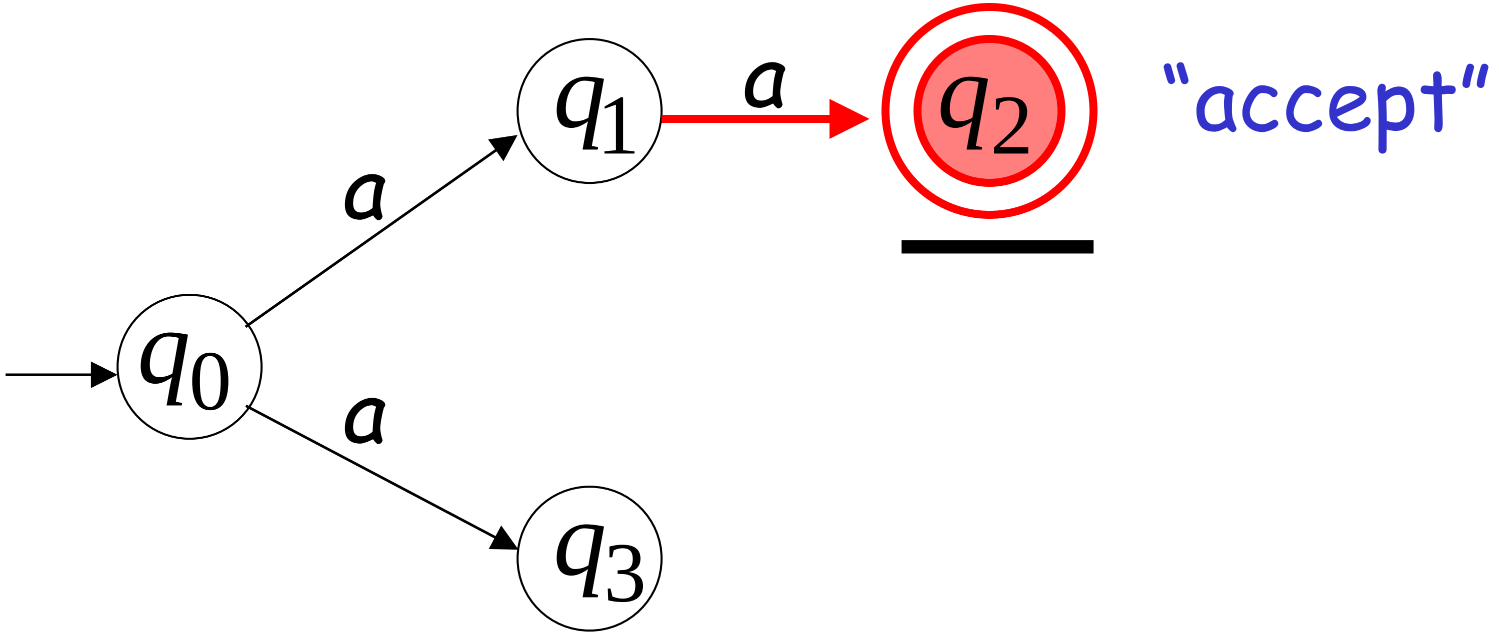
First Choice



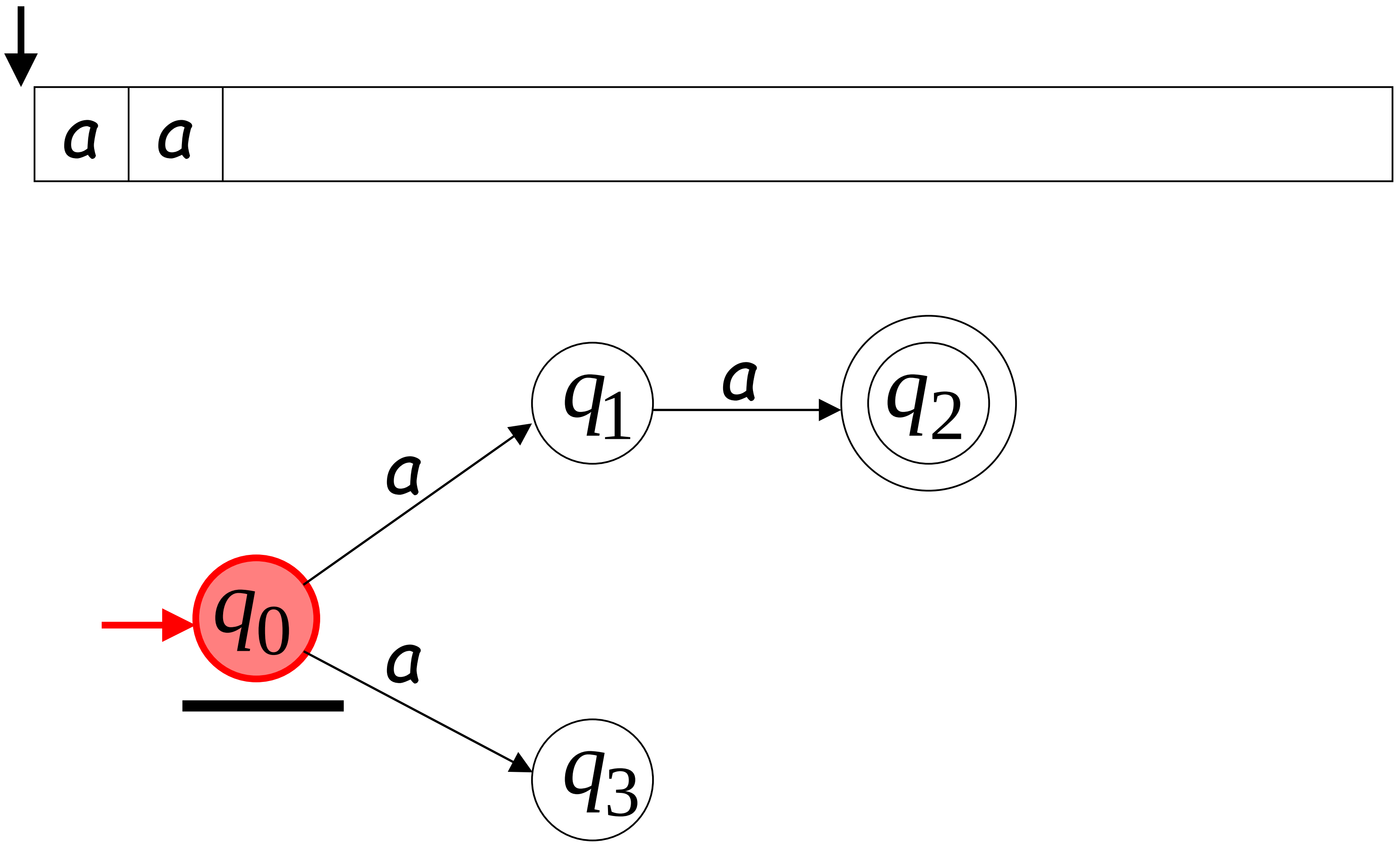
First Choice



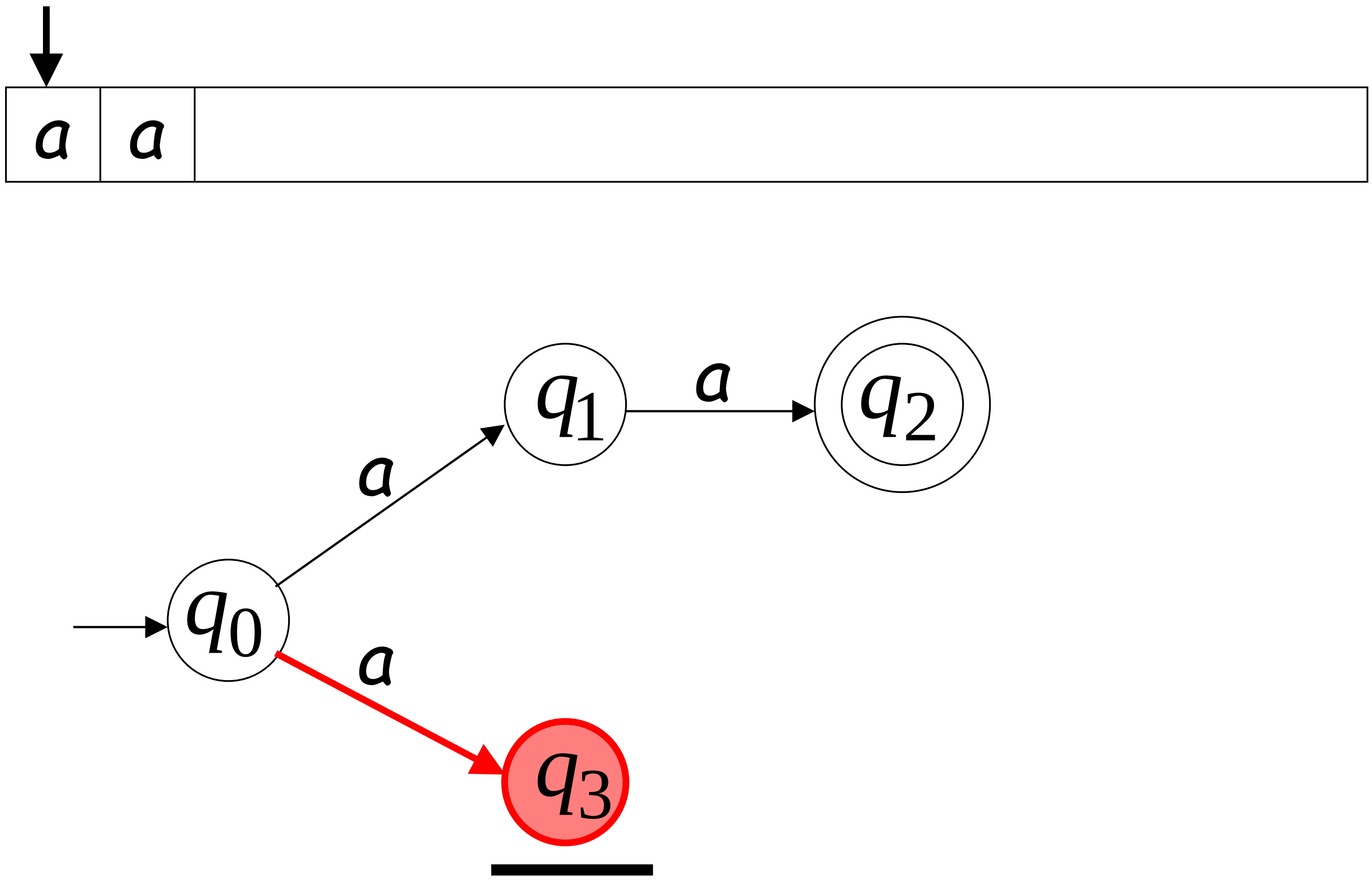
All input is consumed



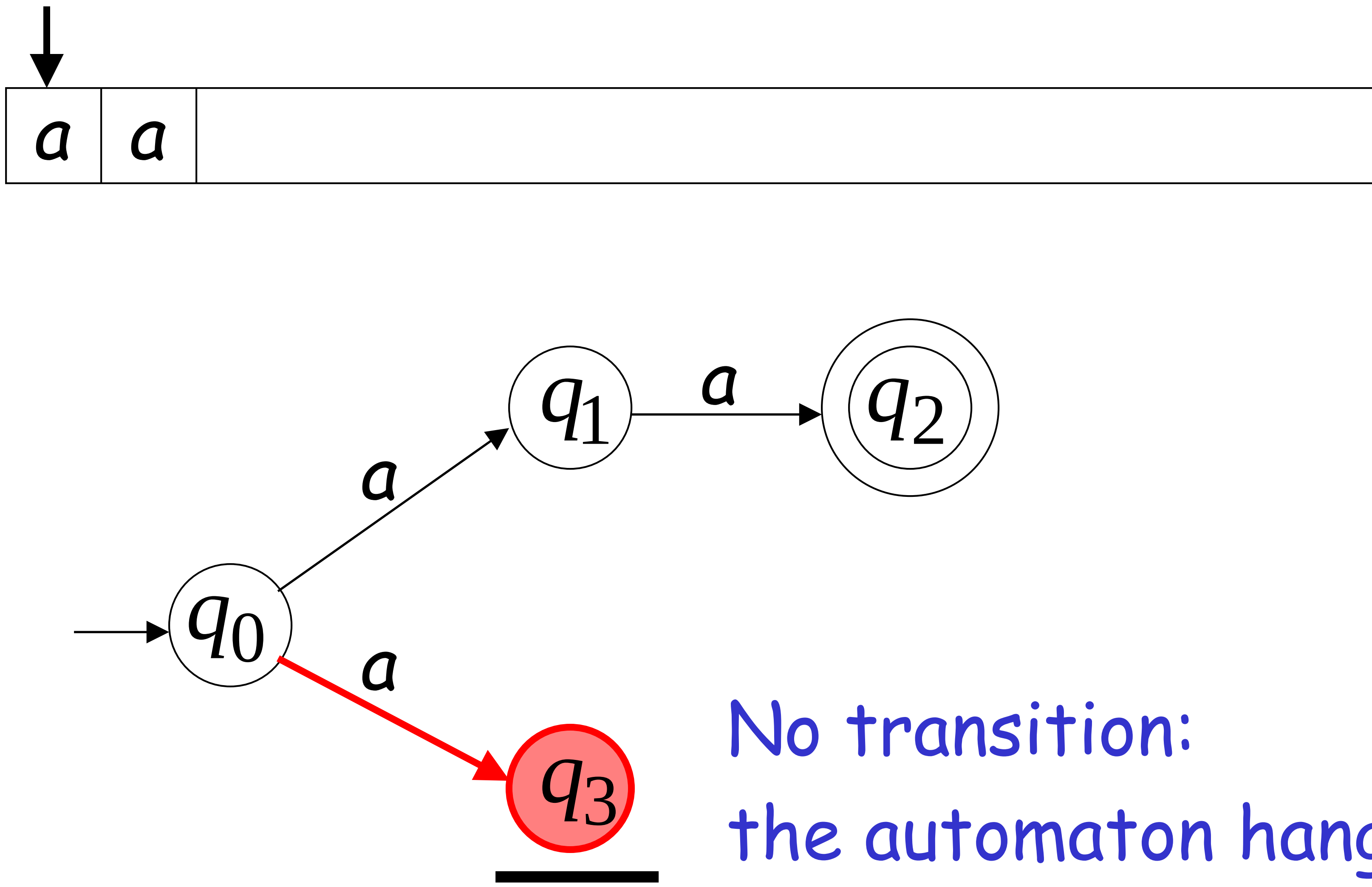
Second Choice



Second Choice

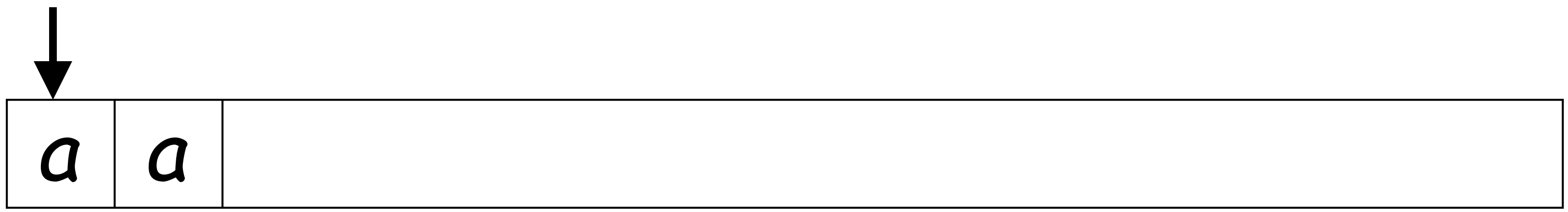


Second Choice

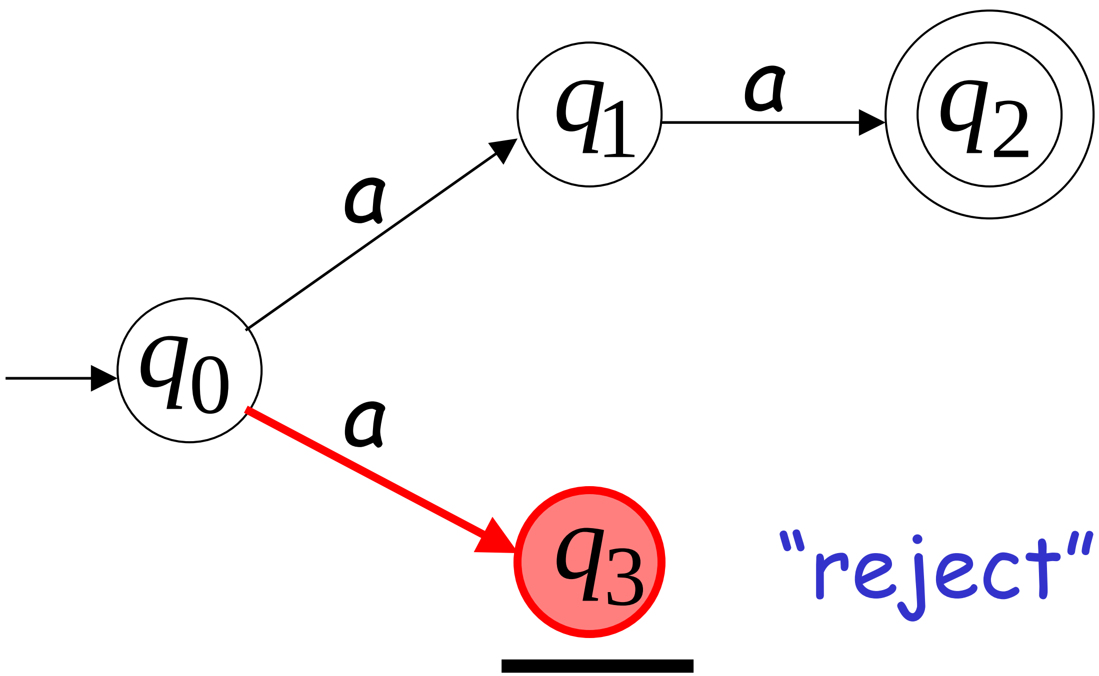


No transition:
the automaton hangs

Second Choice



Input cannot be consumed



"reject"

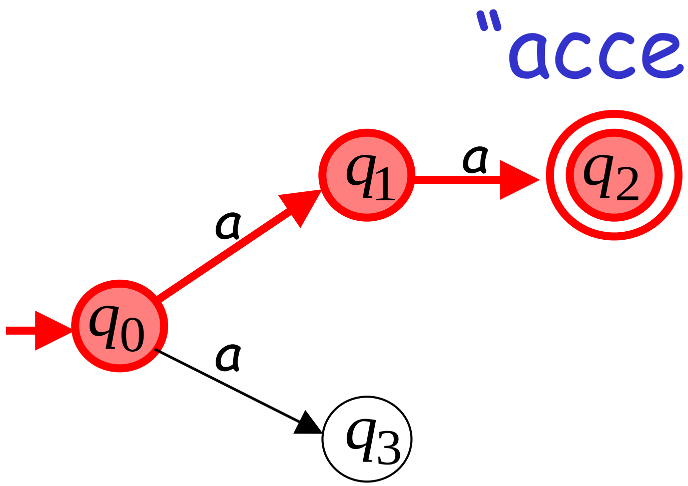
An NFA accepts a string:

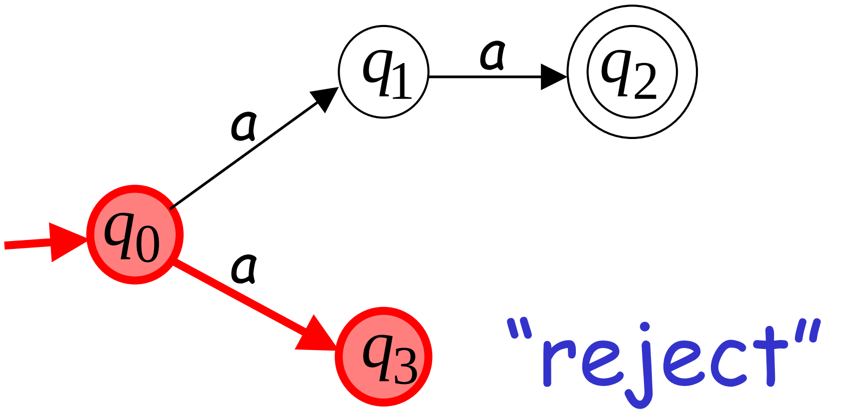
when there is a computation of the NFA
that accepts the string

There is a computation:
all the input is consumed and the automaton
is in an accepting state

Example

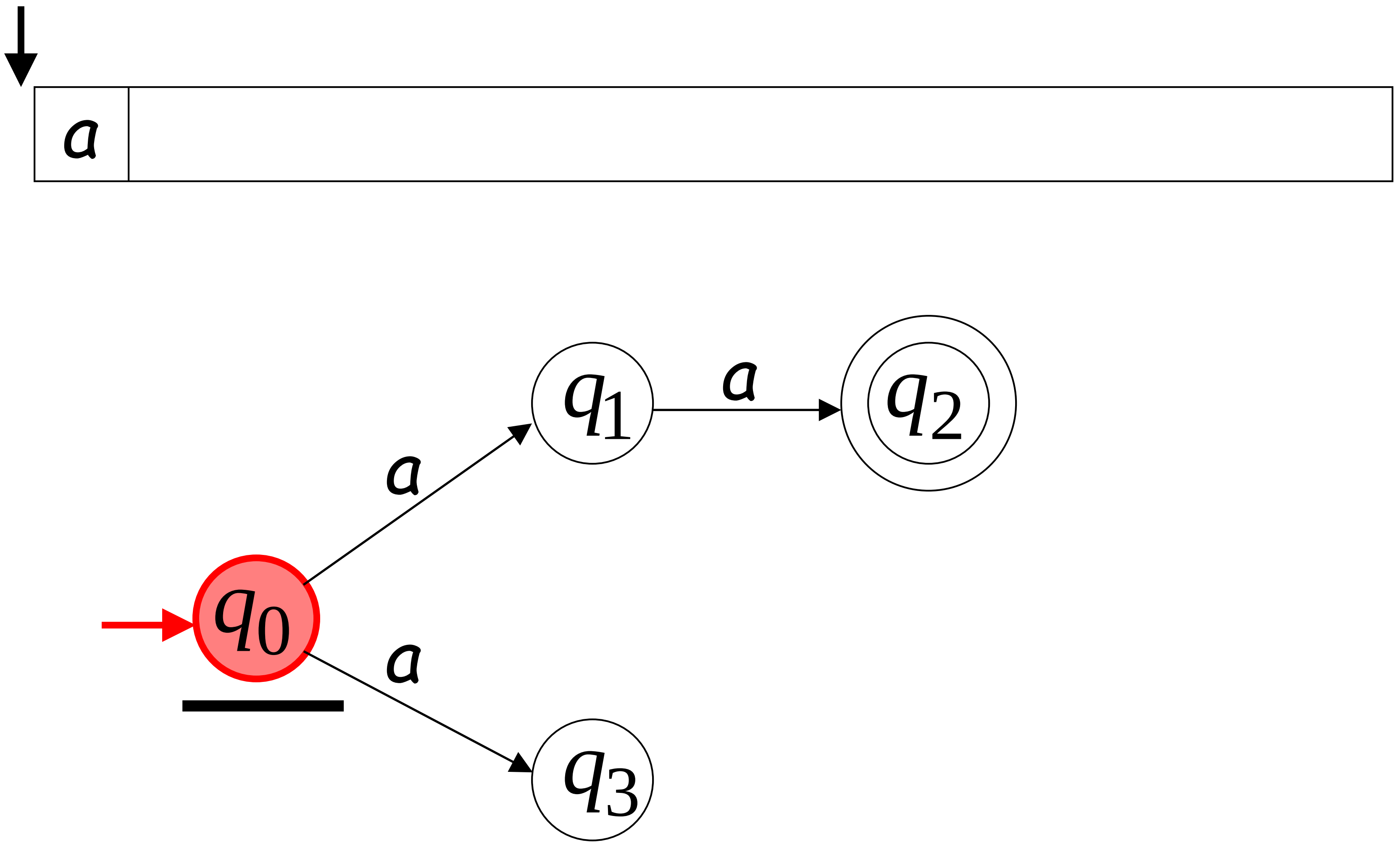
aa is accepted by the NFA:



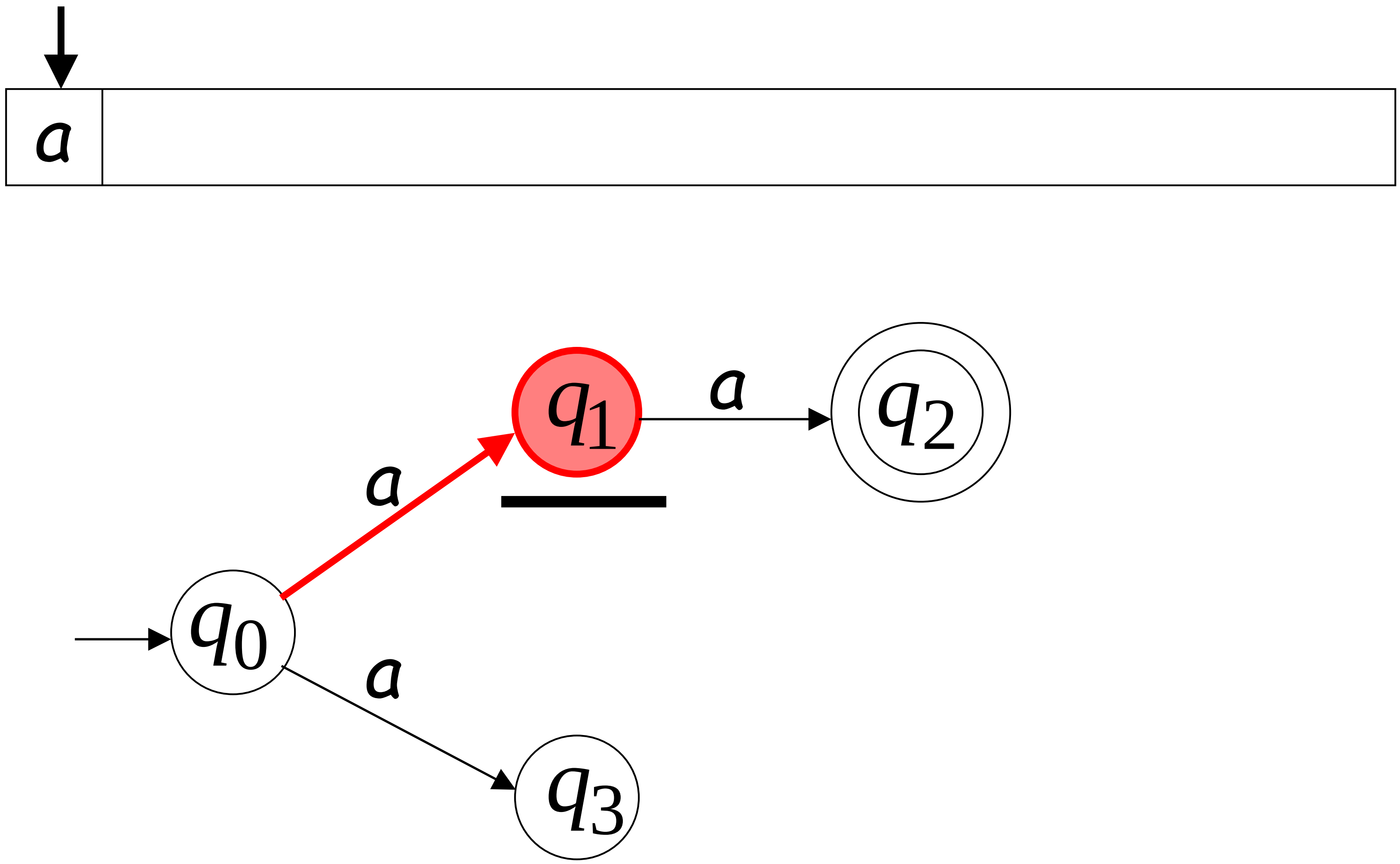


because this computation accepts aa

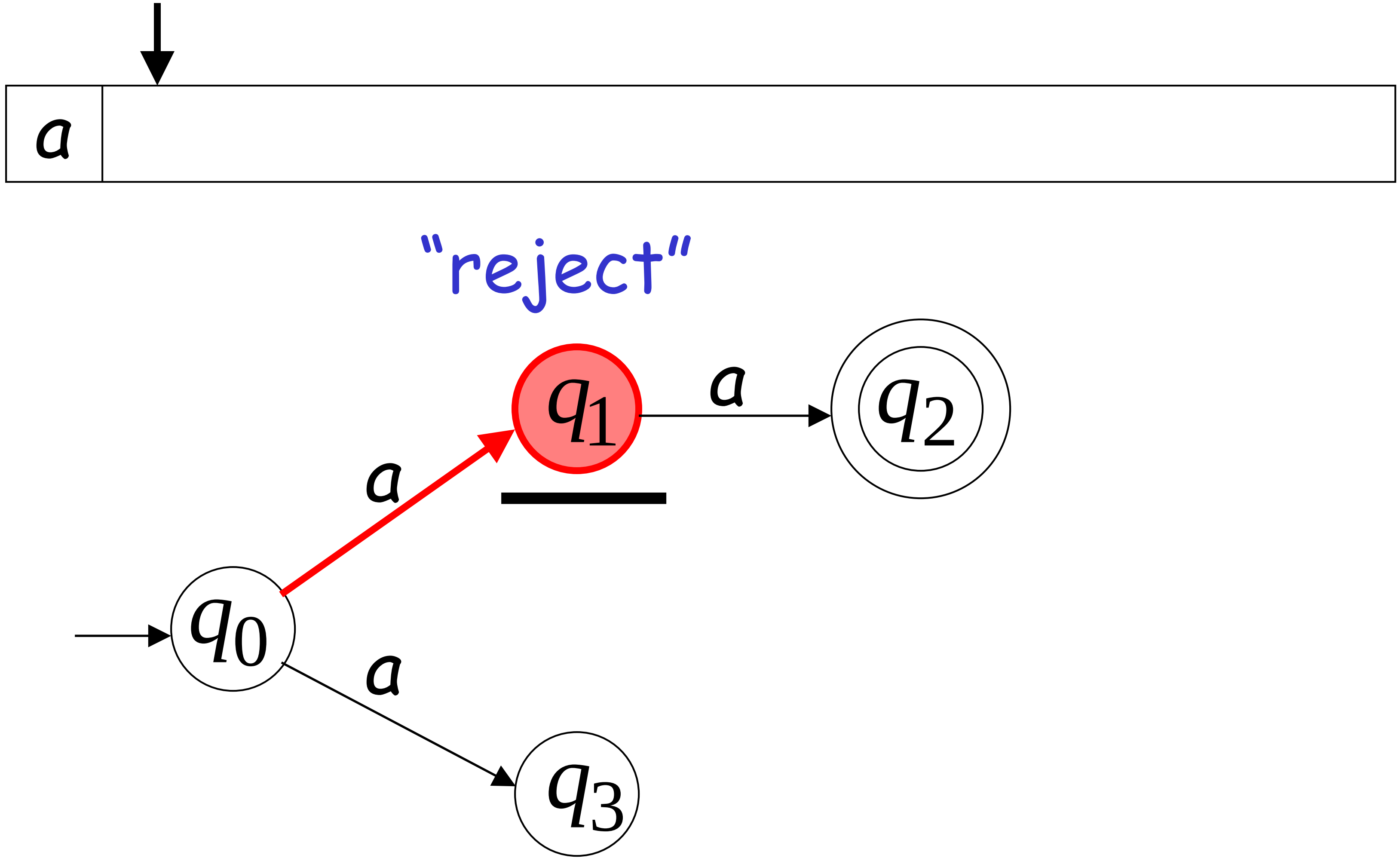
Rejection example



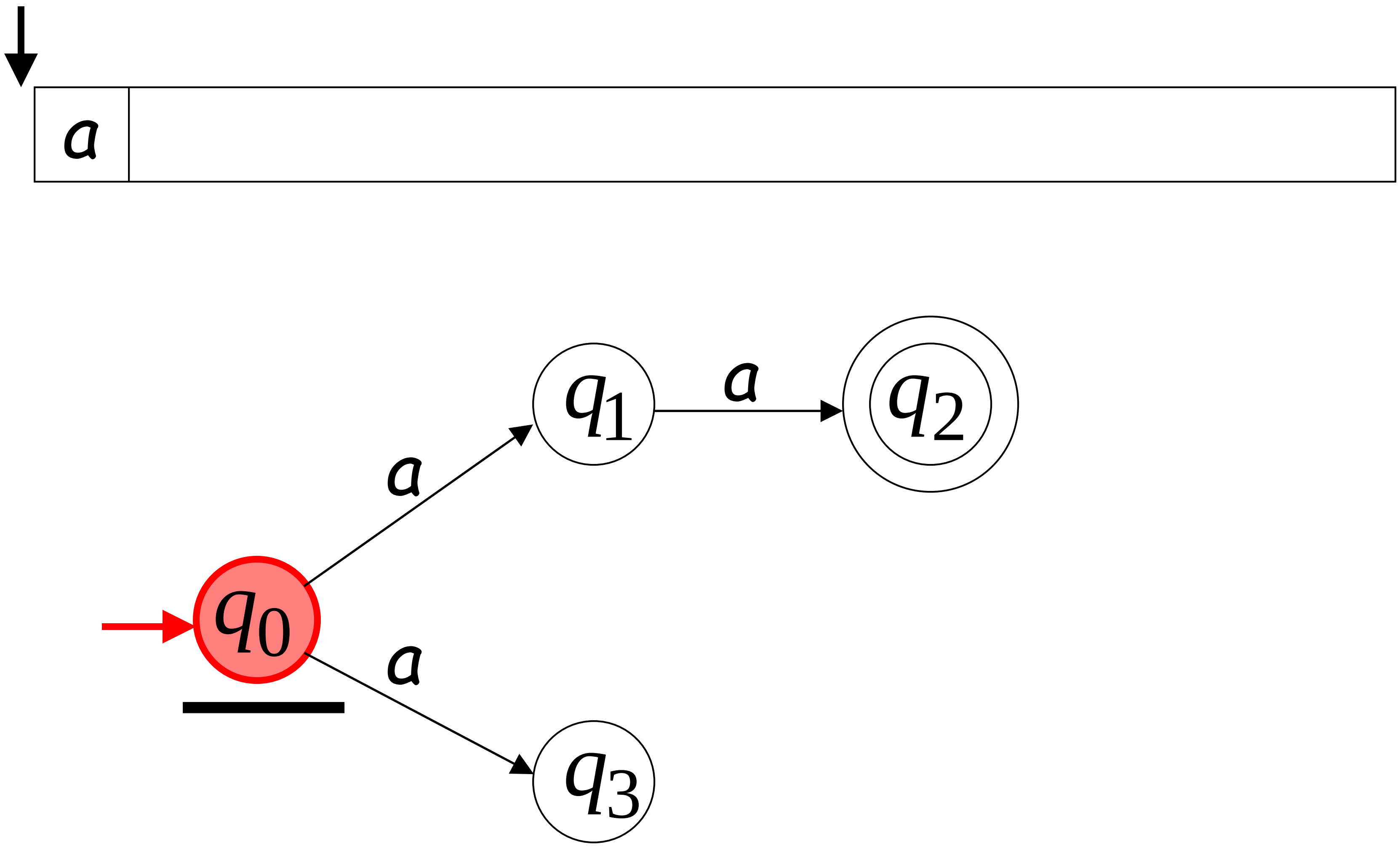
First Choice



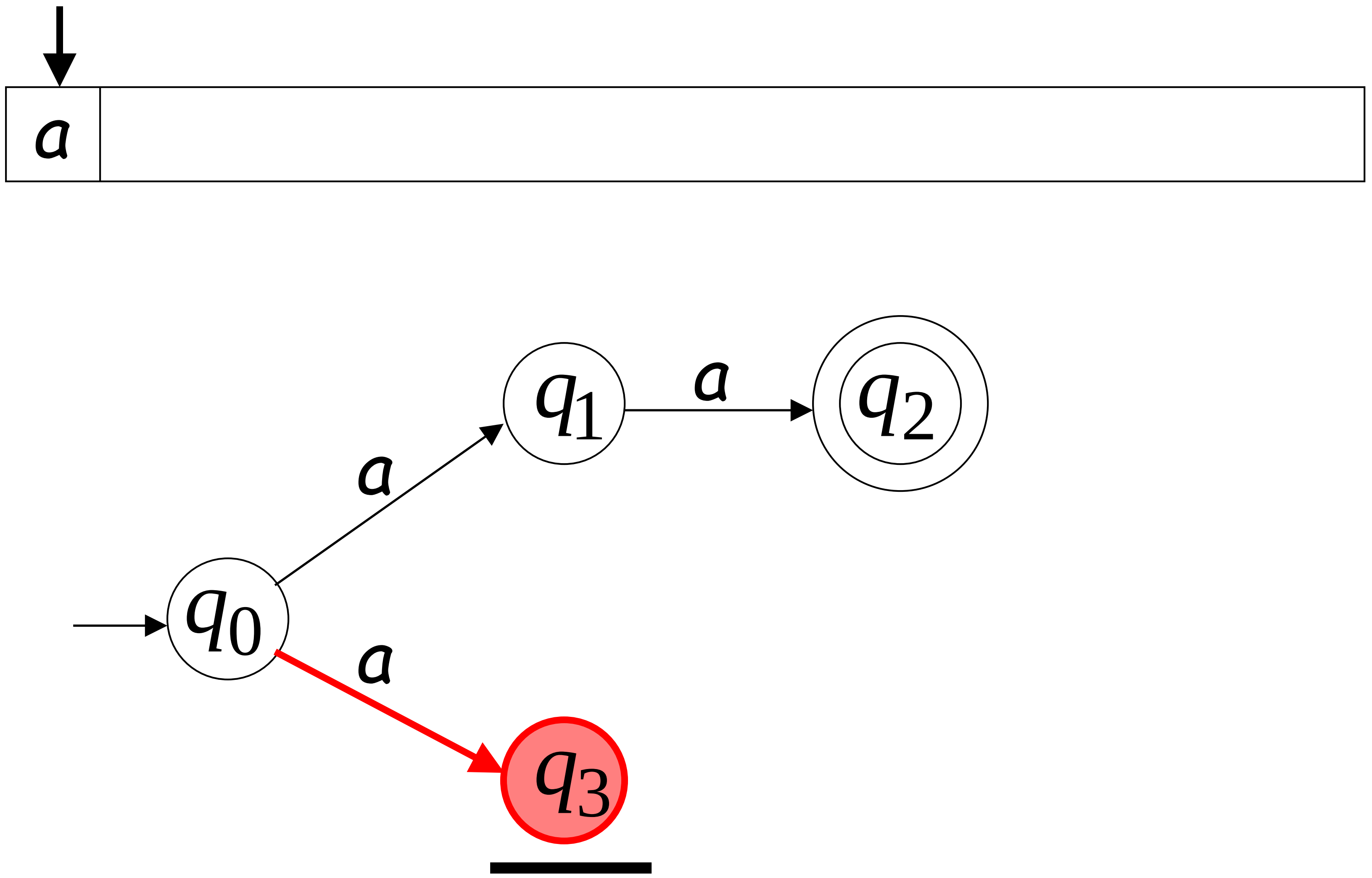
First Choice



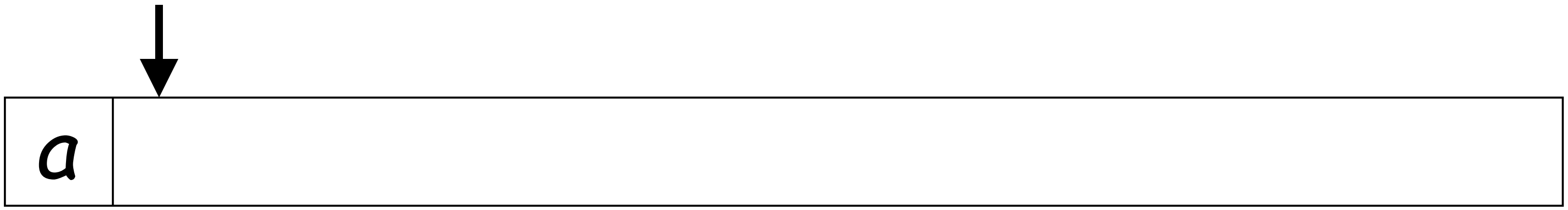
Second Choice

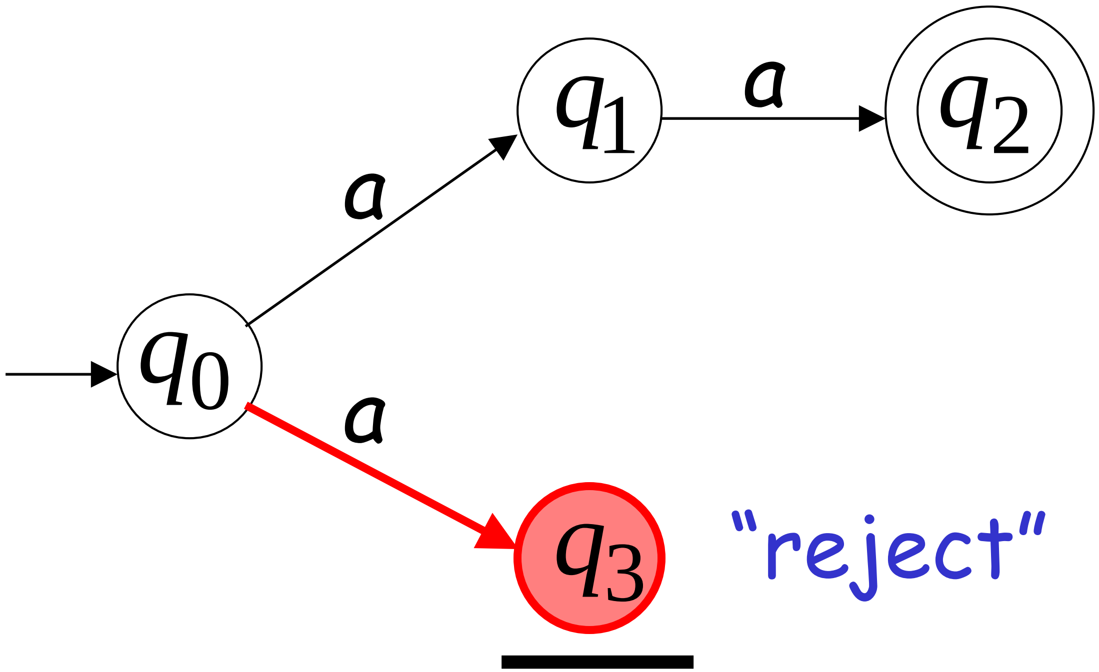


Second Choice



Second Choice





An NFA rejects a string:

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

For each computation:

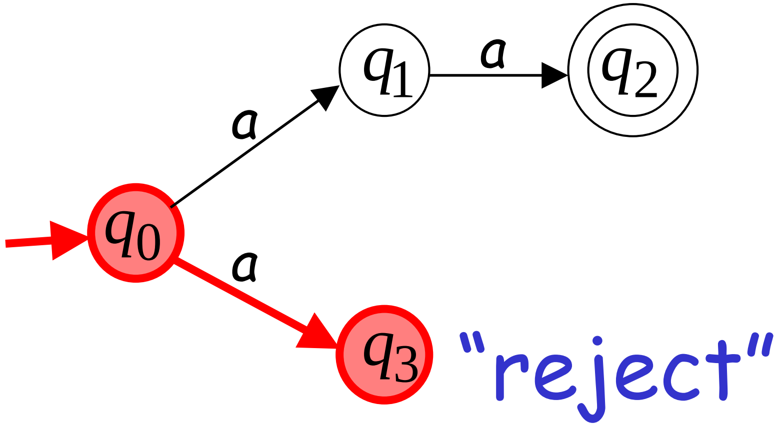
- All the input is consumed and the automaton is in a non final state

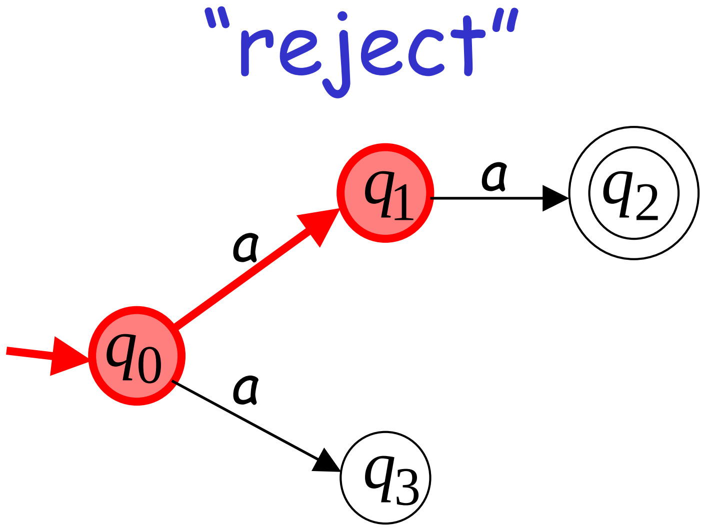
OR

- The input cannot be consumed

Example

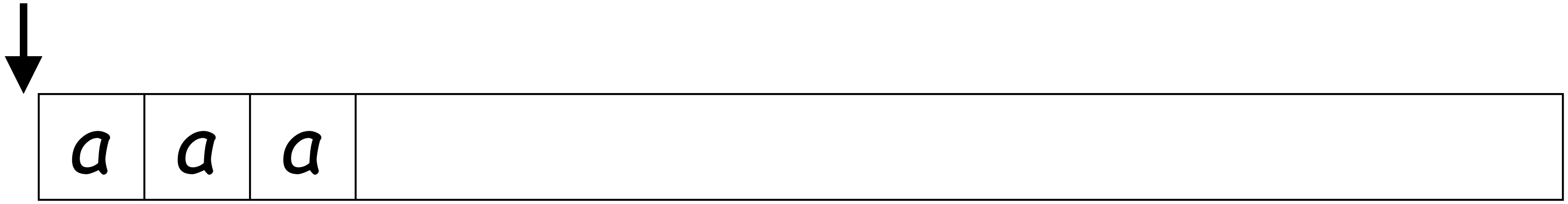
a is rejected by the NFA:

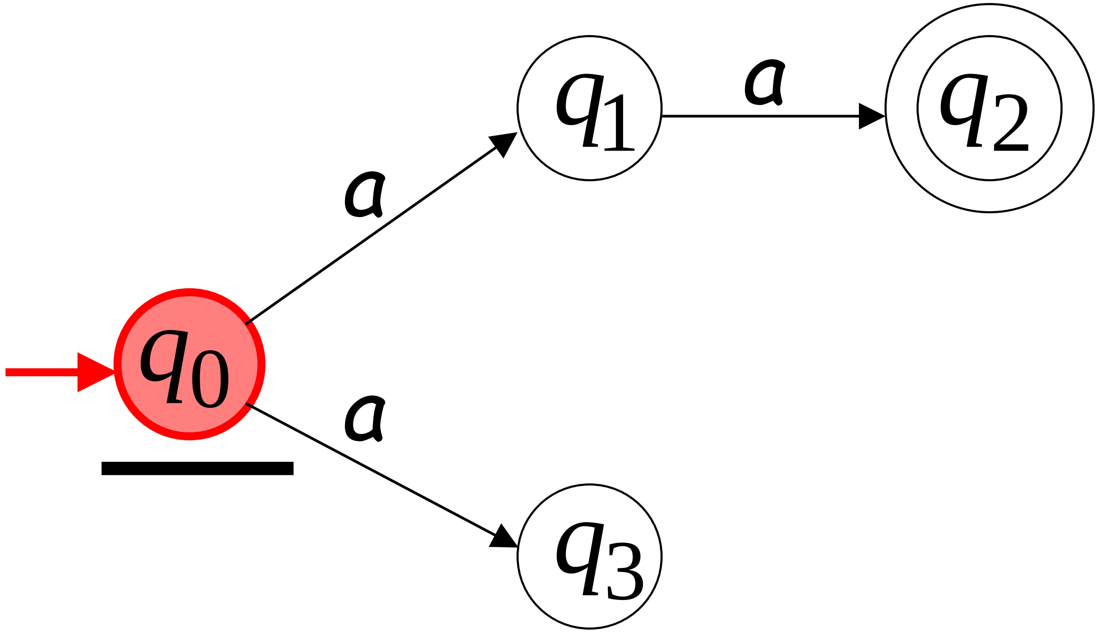




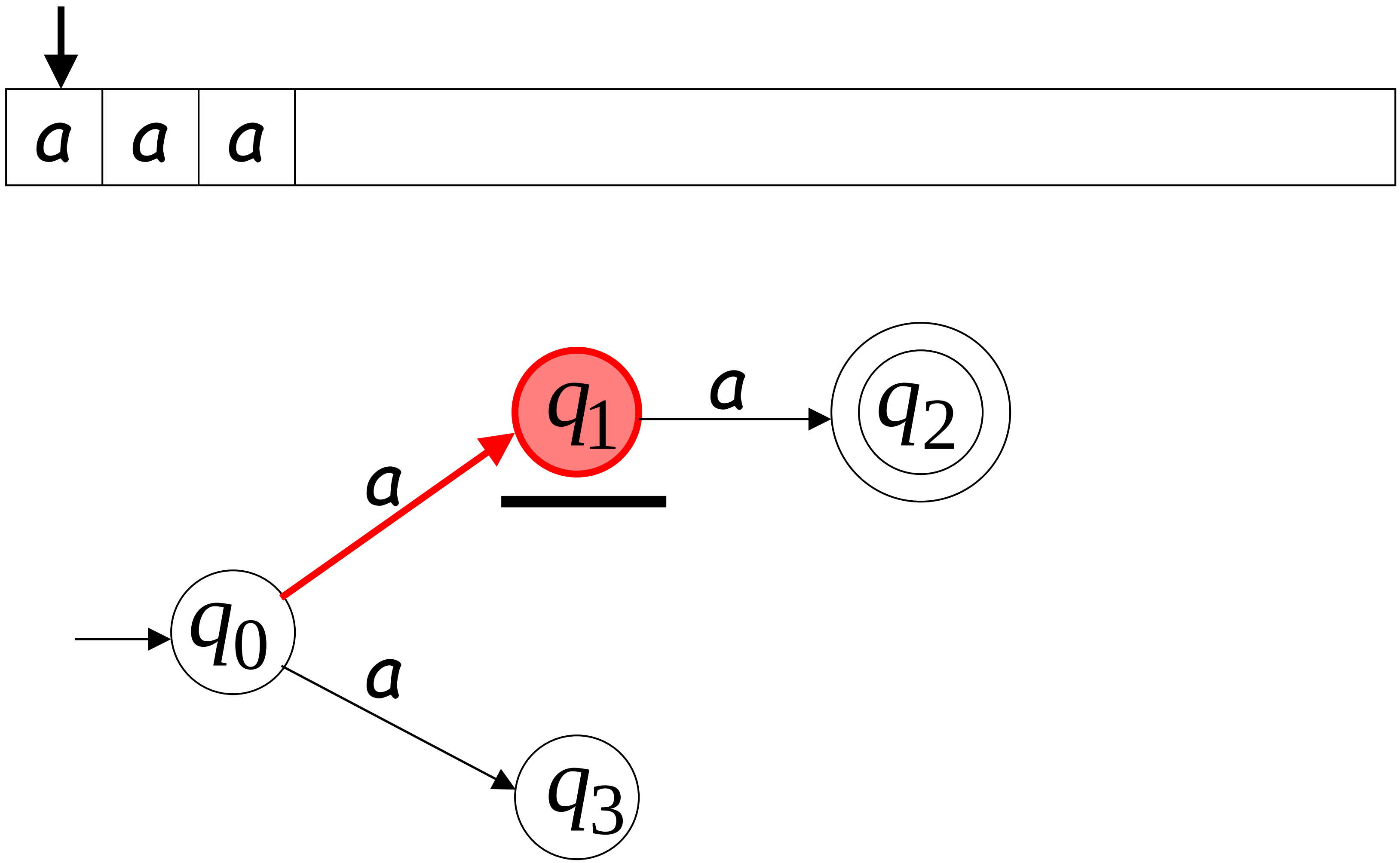
All possible computations lead to rejection

Rejection example

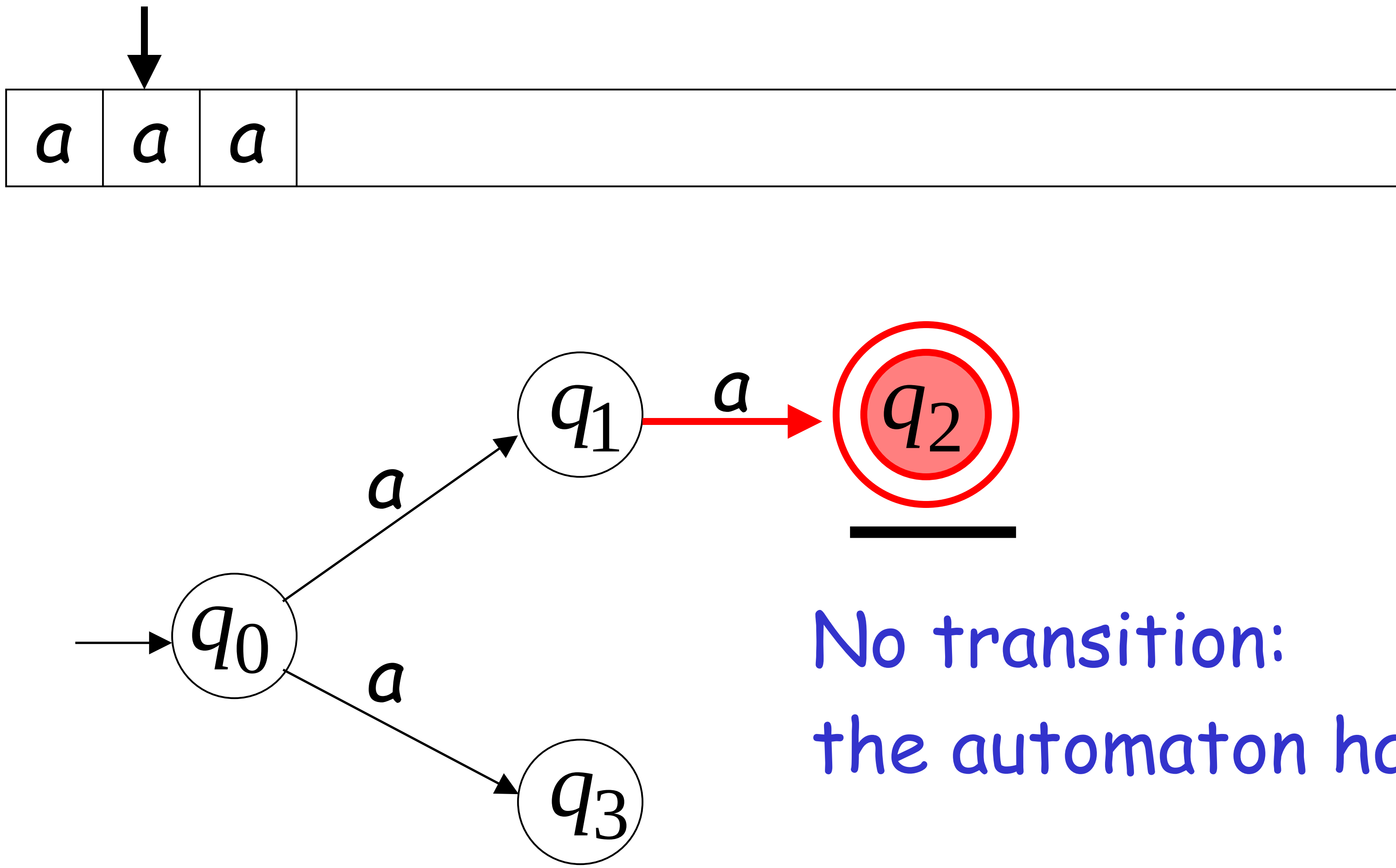




First Choice

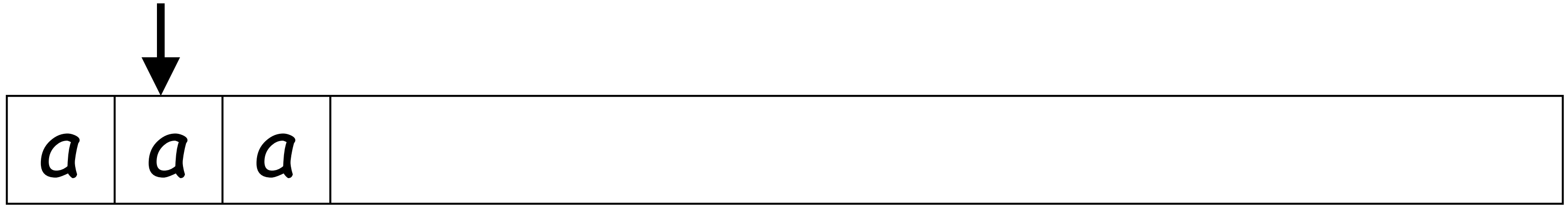


First Choice

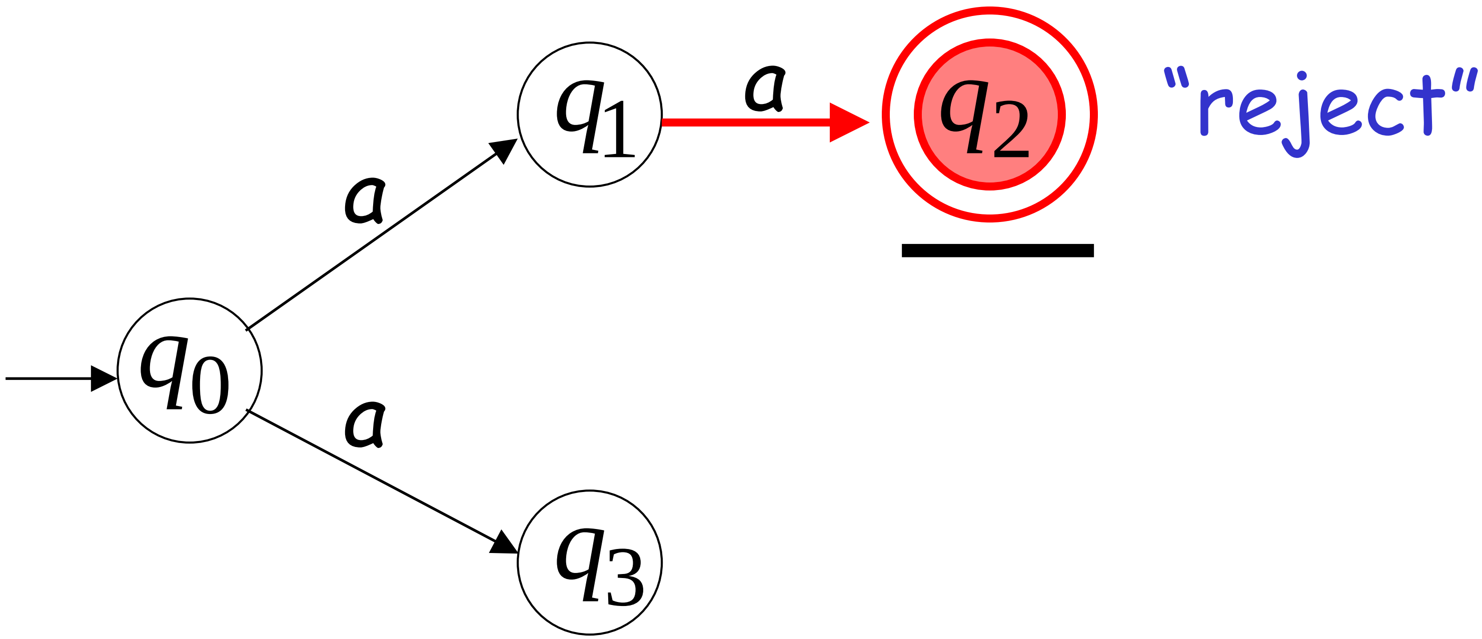


No transition:
the automaton hangs

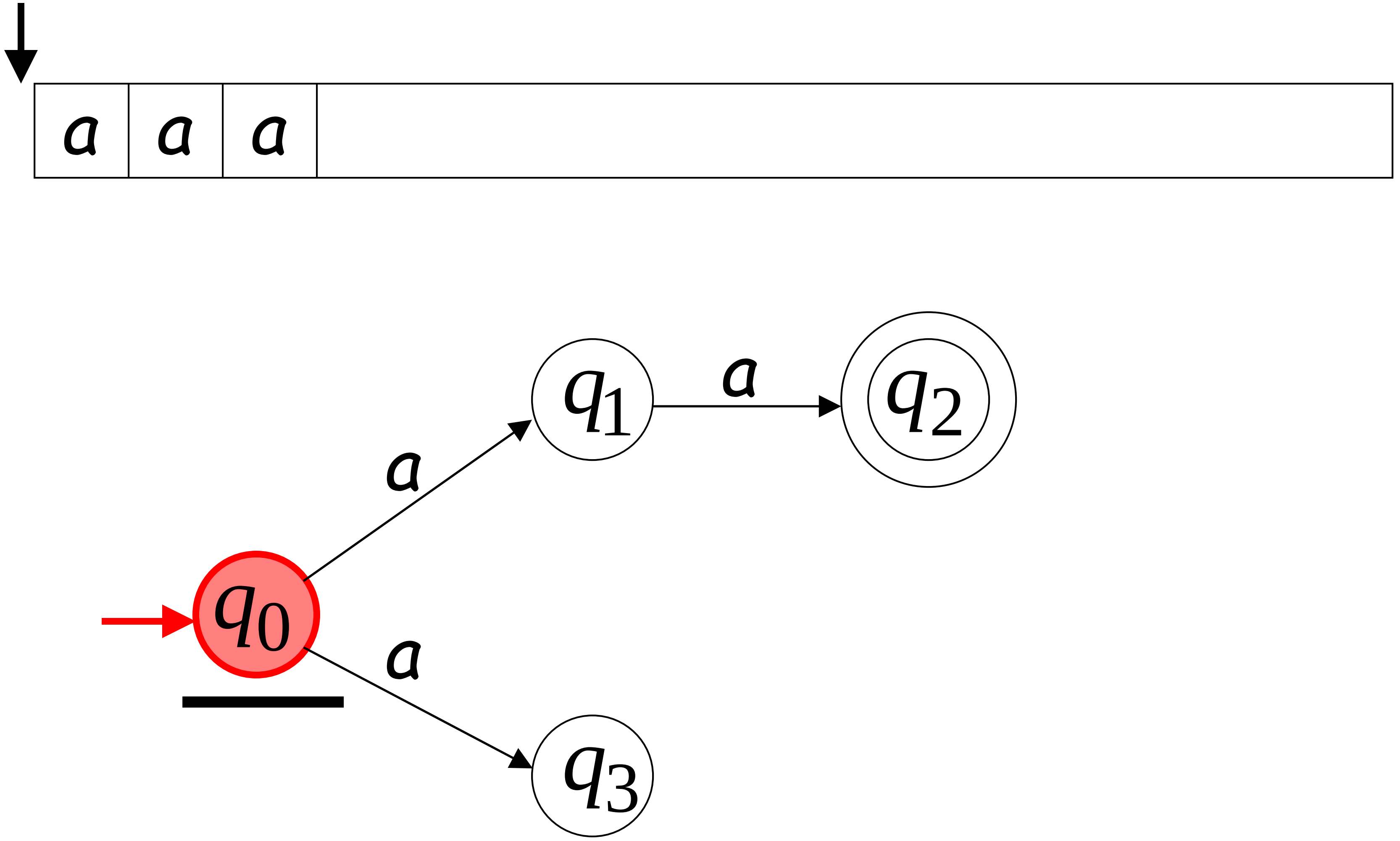
First Choice



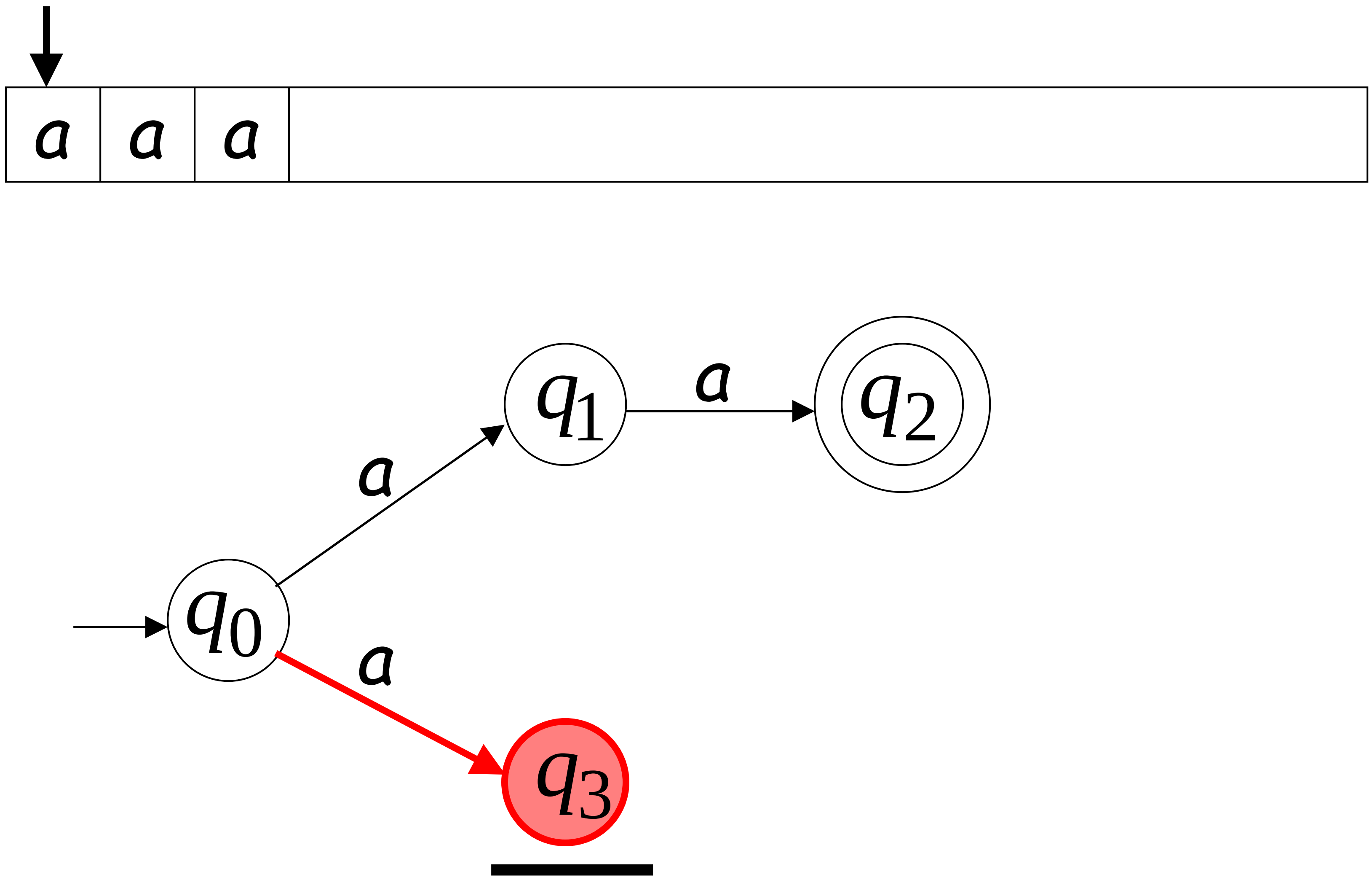
Input cannot be consumed



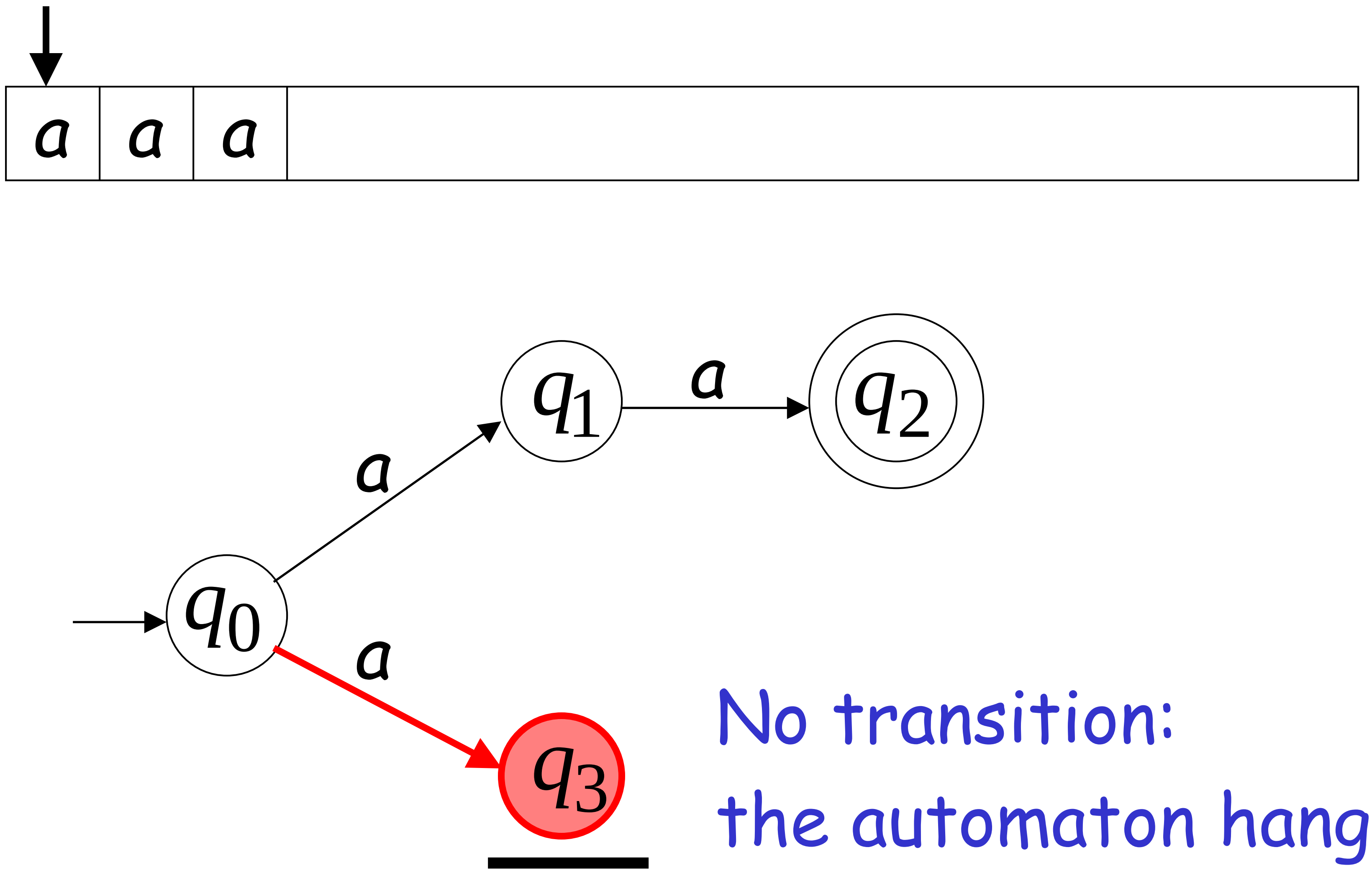
Second Choice



Second Choice

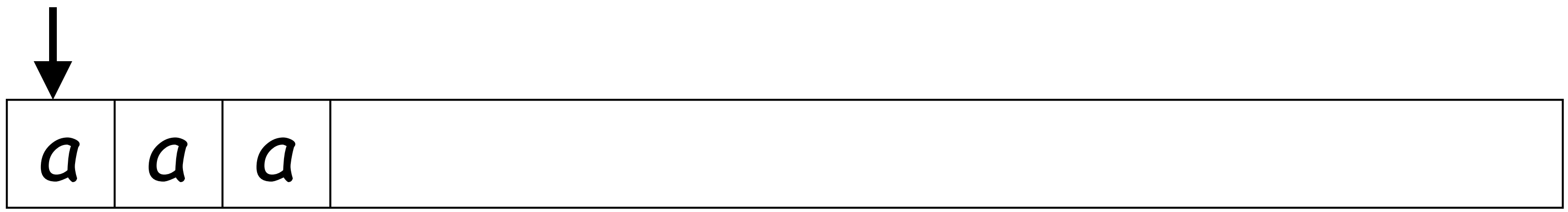


Second Choice

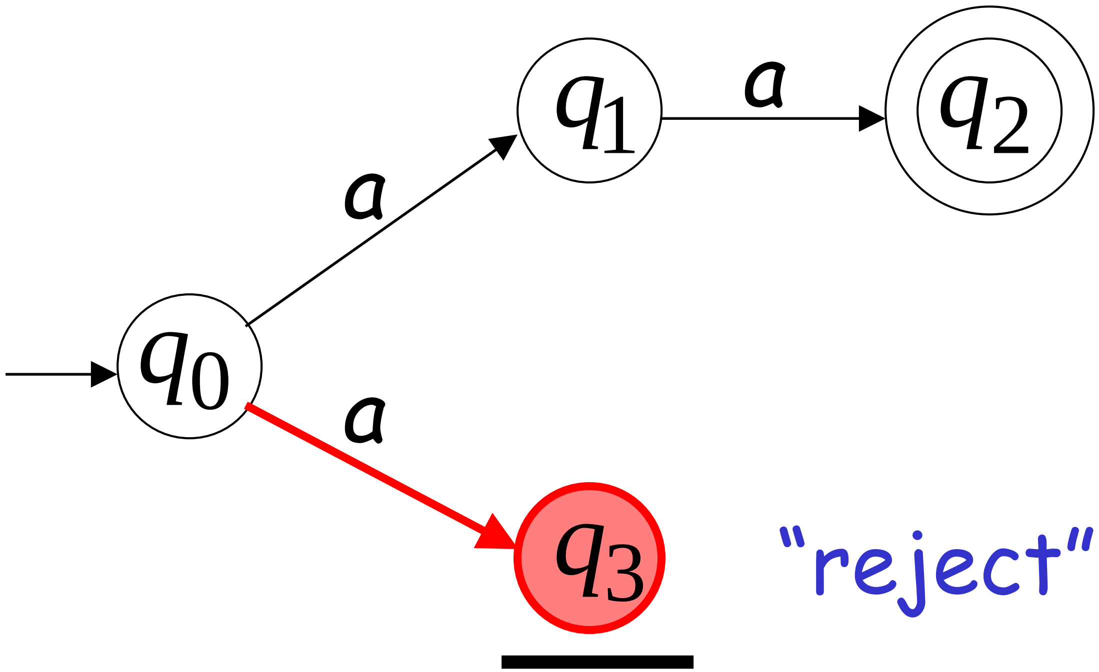


No transition:
the automaton hangs

Second Choice



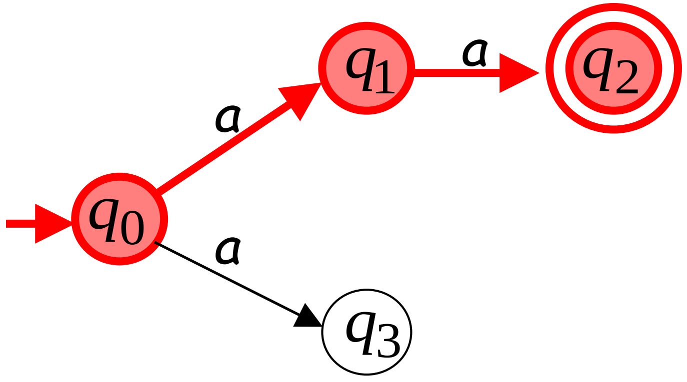
Input cannot be consumed

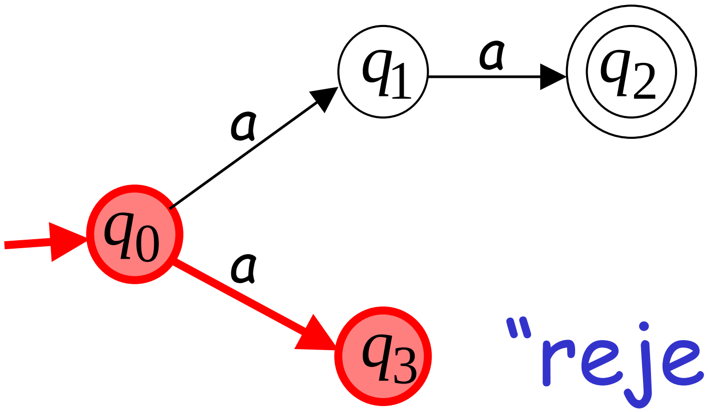


"reject"

aaa is rejected by the NFA:

"reject"

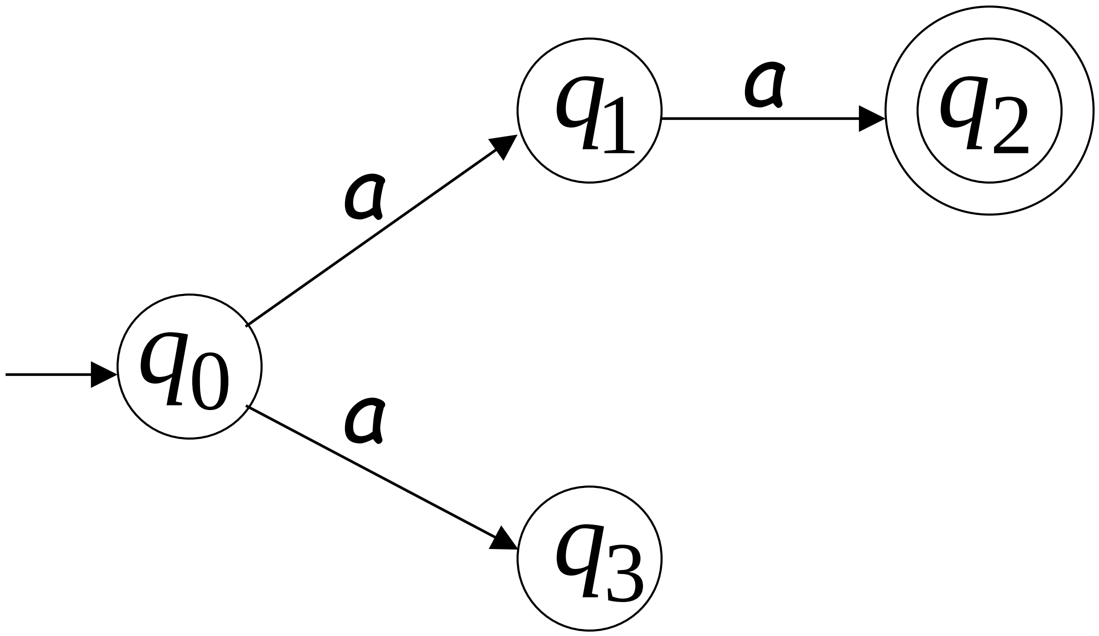




"reject"

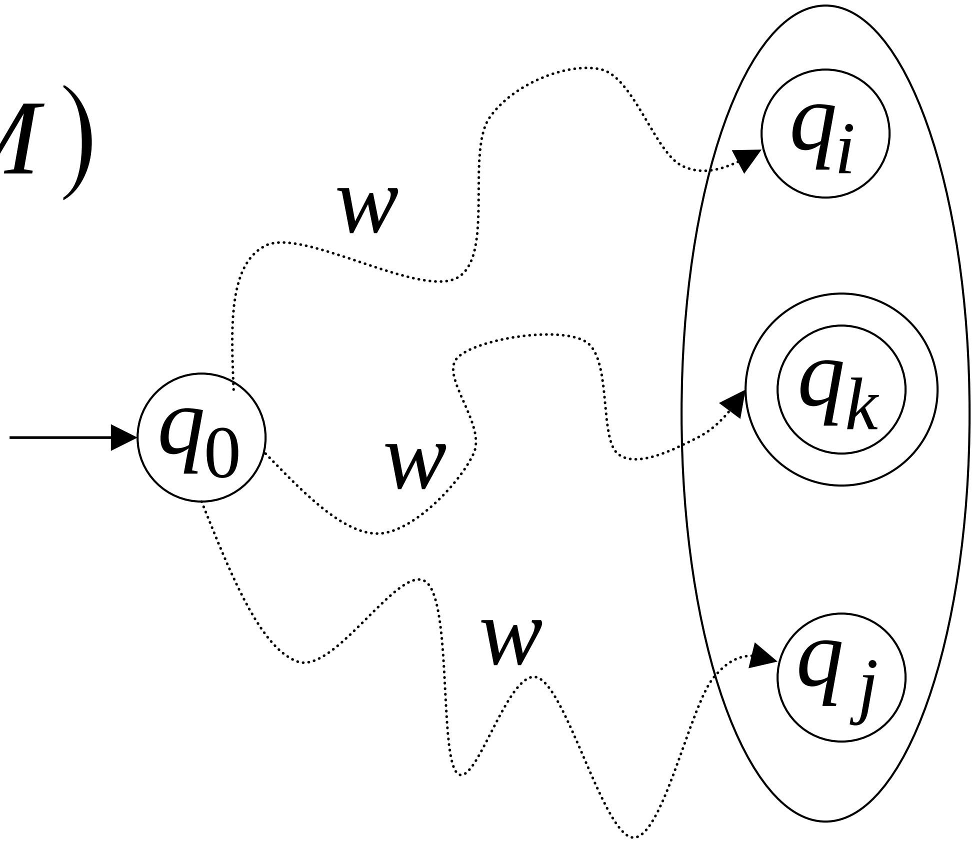
All possible computations lead to rejection

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

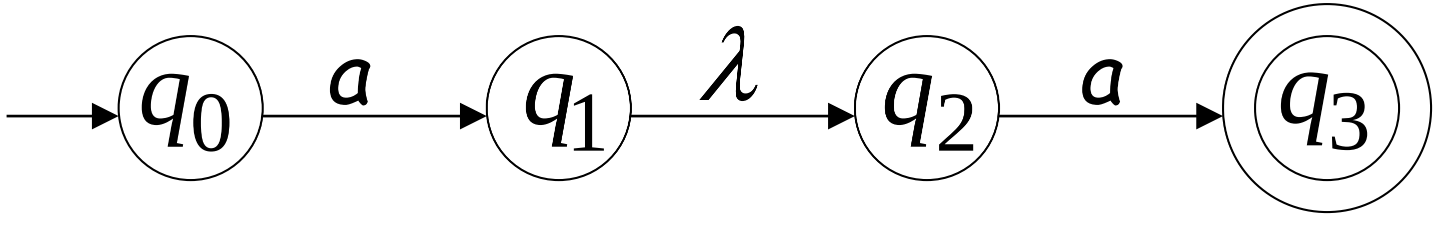


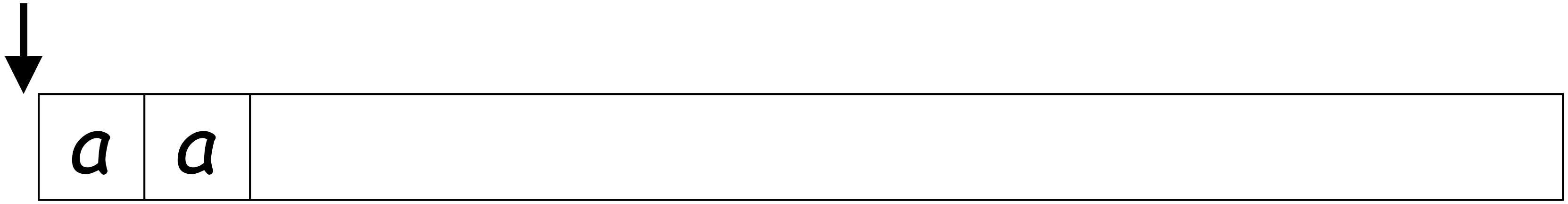
One path from q_0 to an accepting state suffices

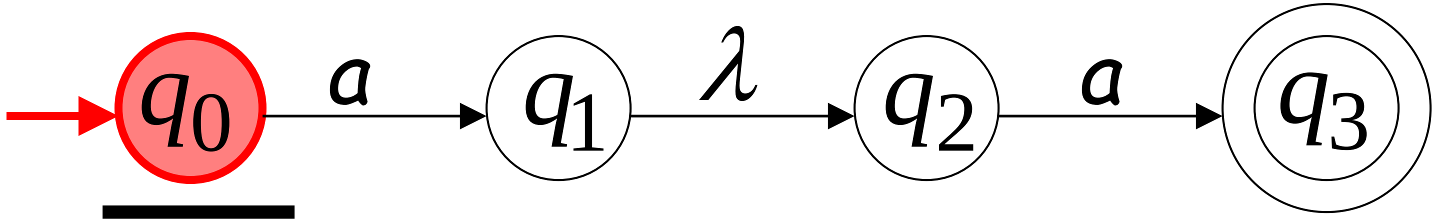
$w \in L(M)$

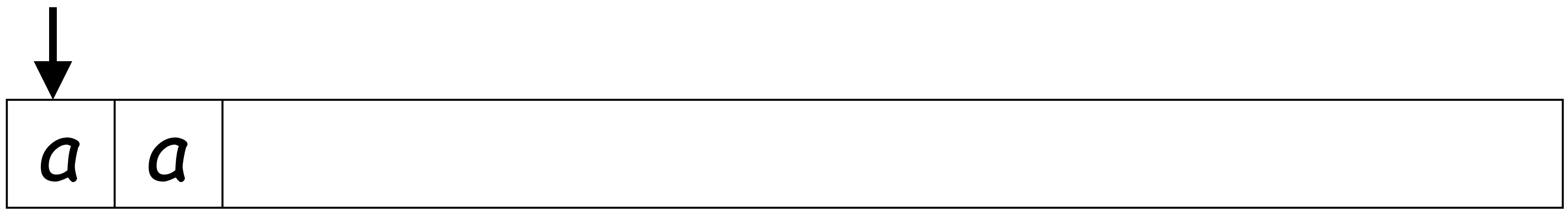


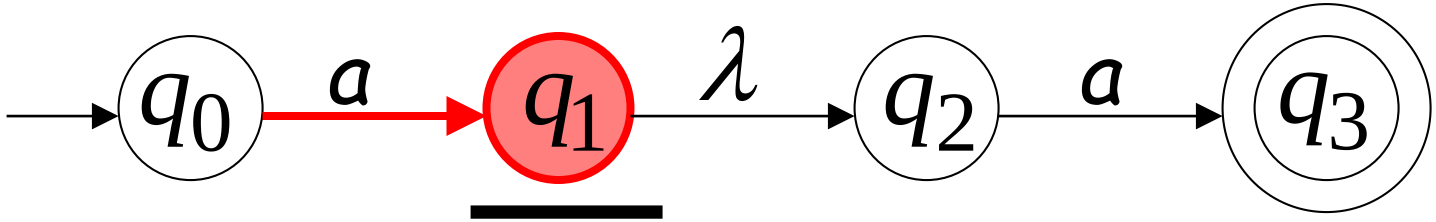
Lambda Transitions



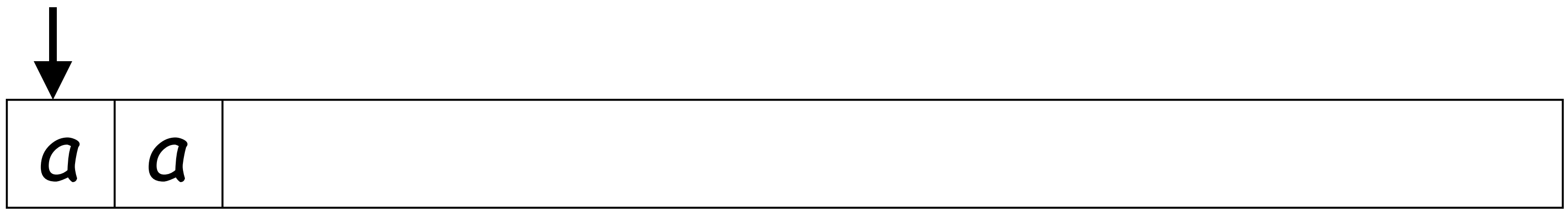

a
a

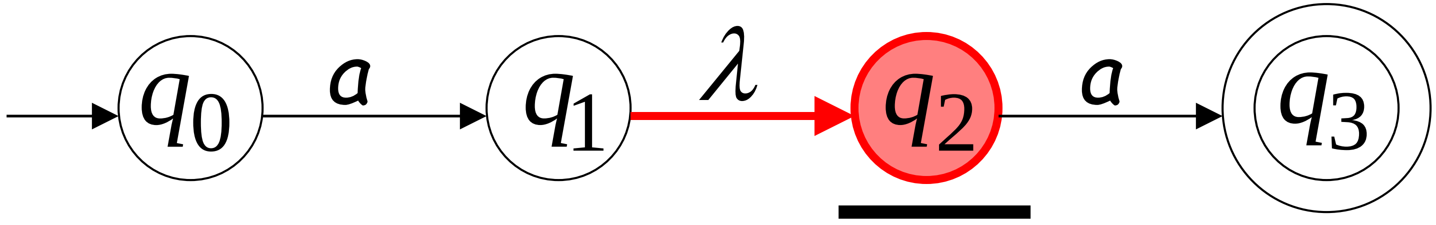

q_0
a
q_1
λ
q_2
a
q_3

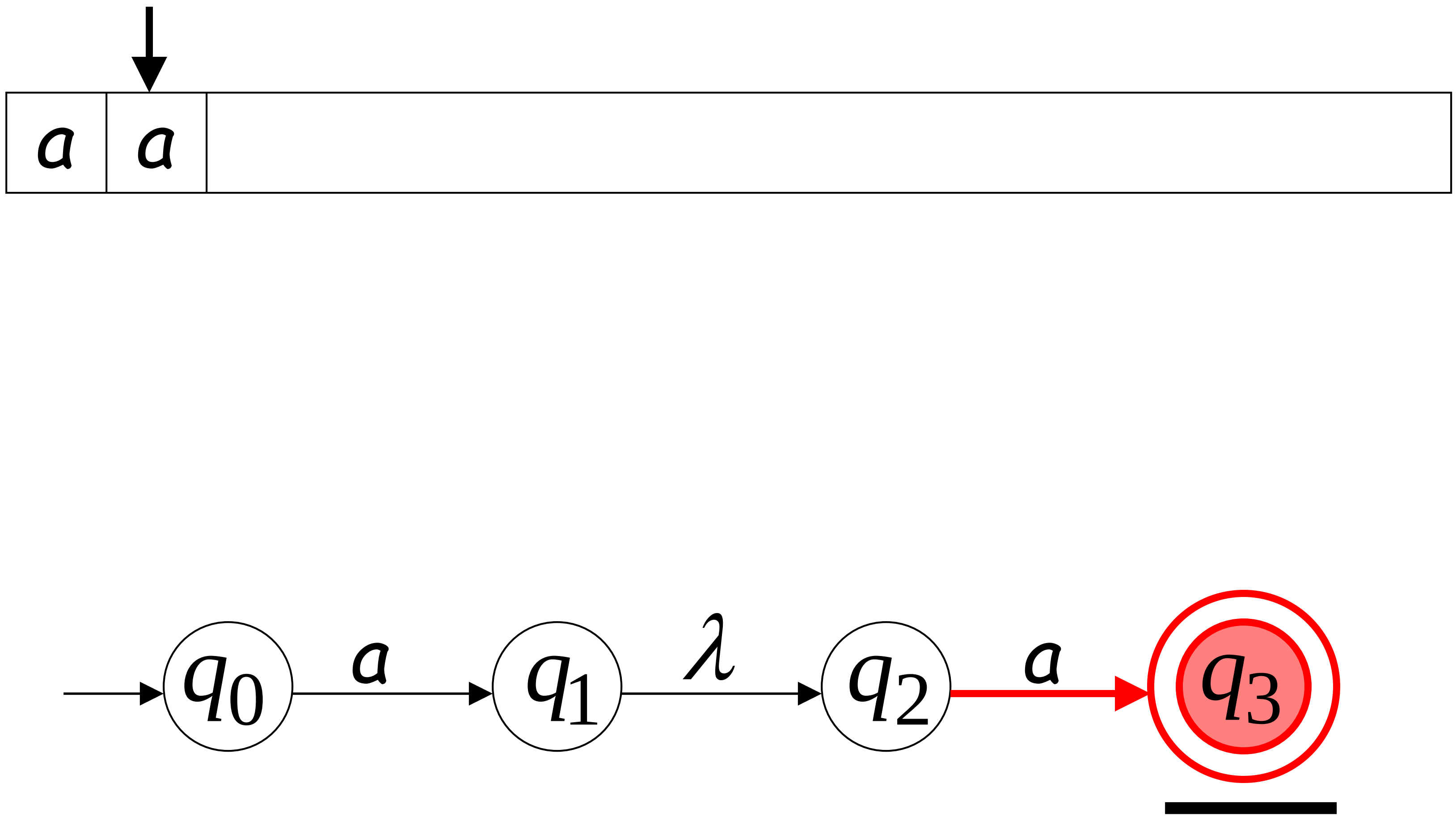

a
a


q0
a
q1
λ
q2
a
q3

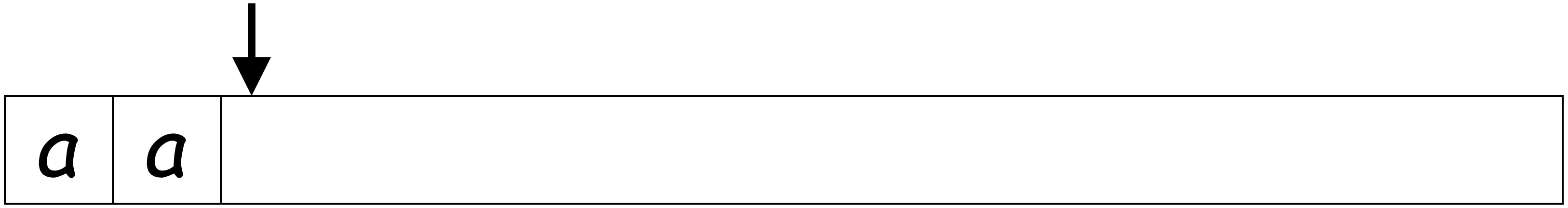
(read head does not move)



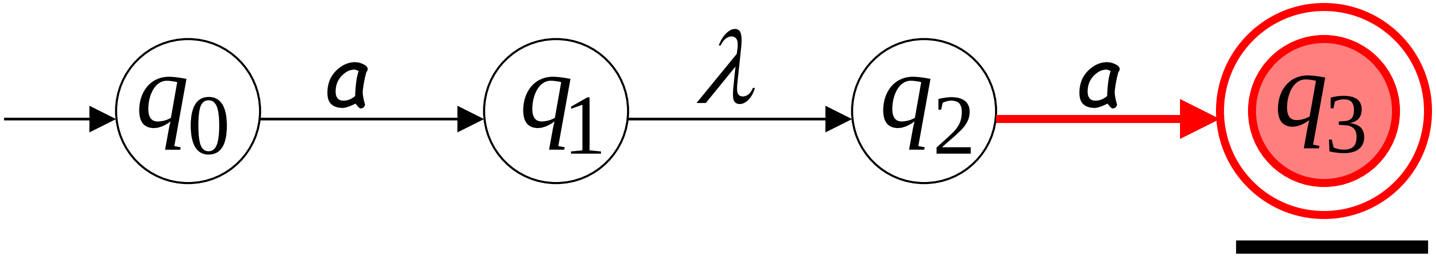



a
a
a
λ
a
q0
q1
q2
q3

all input is consumed

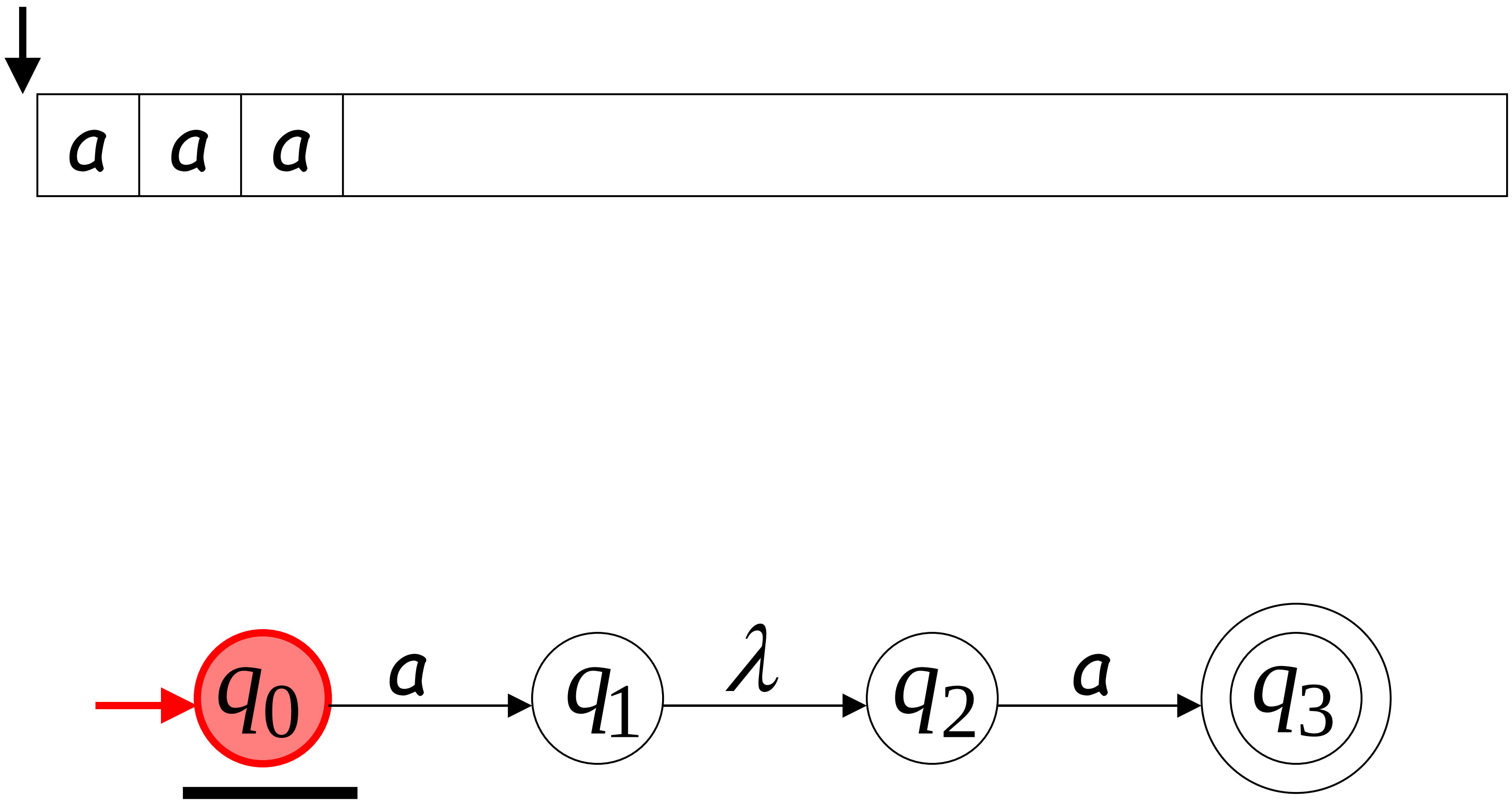


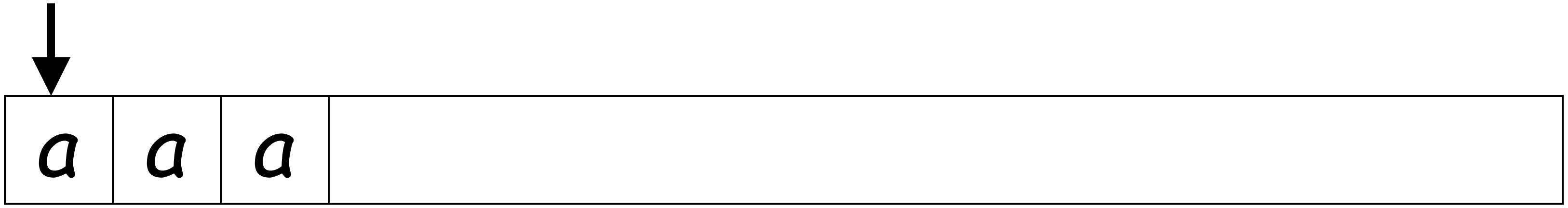
“accept”

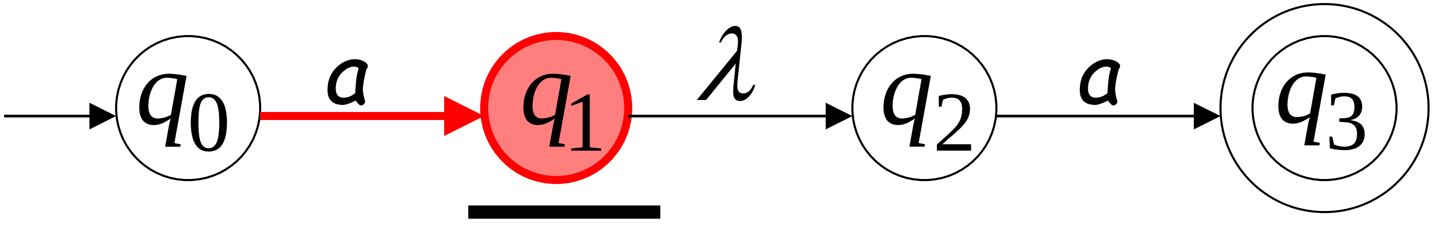


String aa is accepted

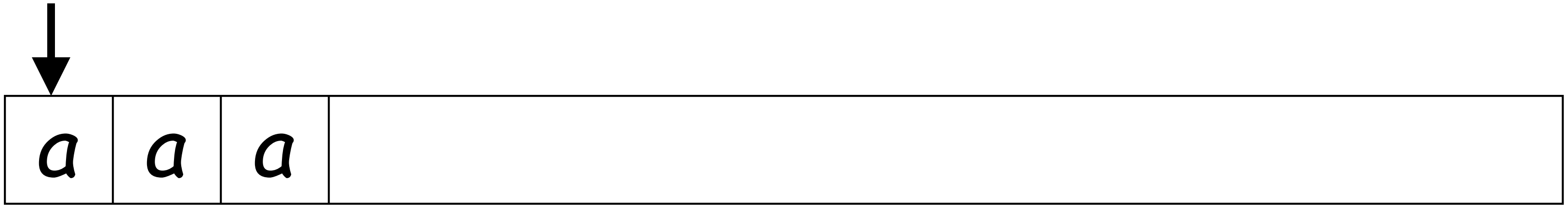
Rejection Example

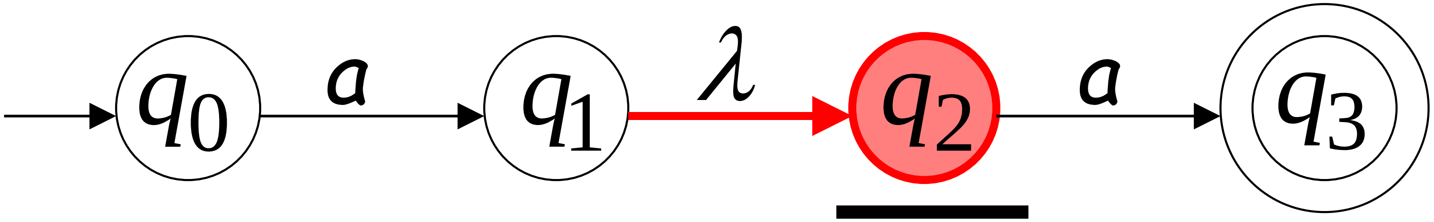


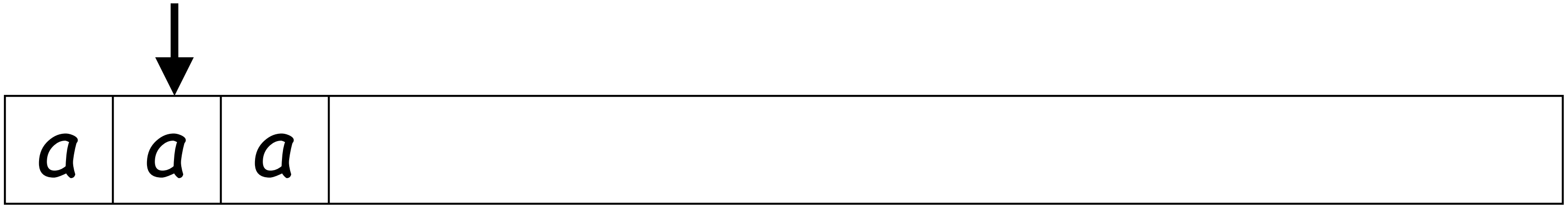

a a a

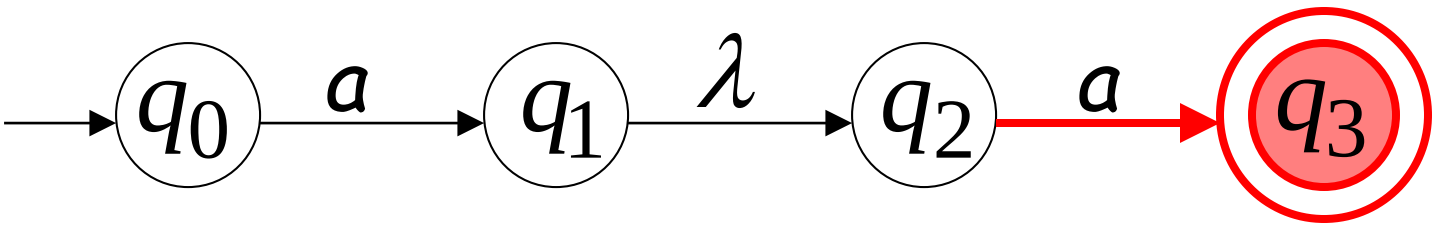

q_0
a
q_1
λ
q_2
a
q_3

(read head doesn't move)



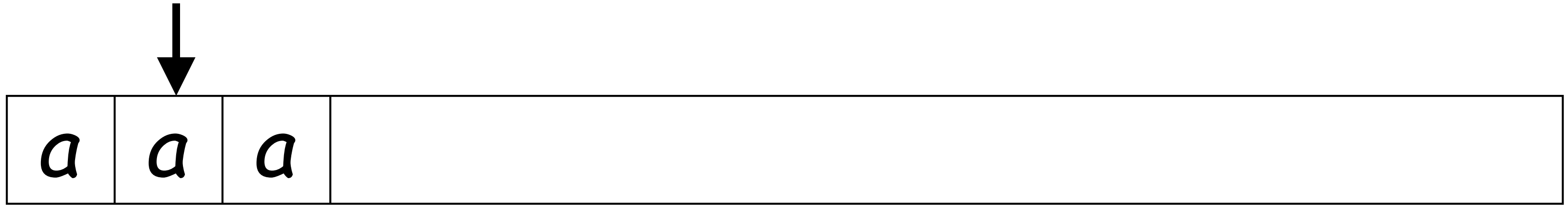




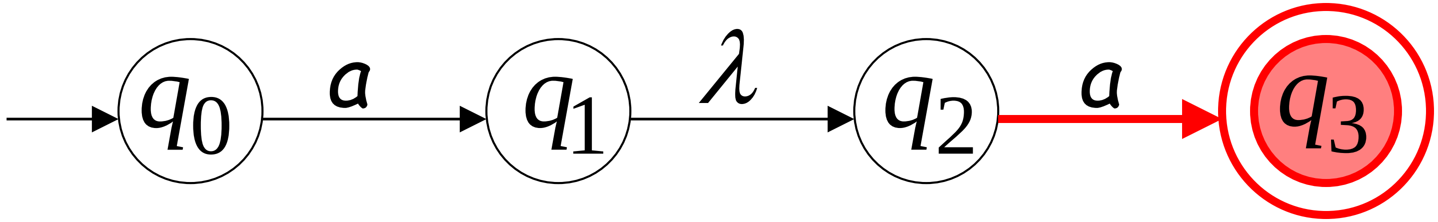


No transition:
the automaton hangs

Input cannot be consumed

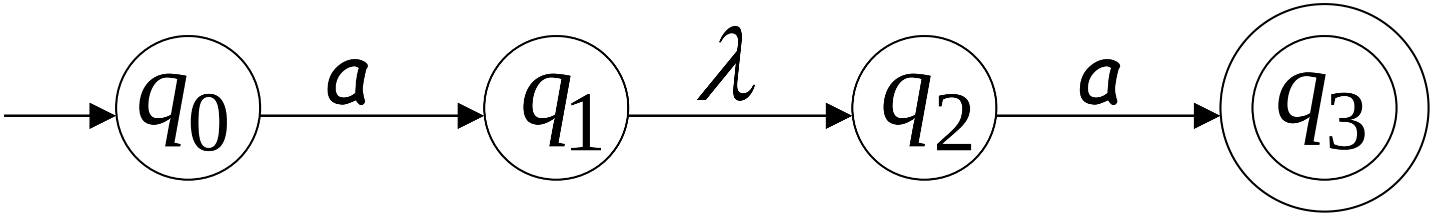


“reject”



String aaa is rejected

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



Remarks:

- The λ symbol never appears on the input tape

Theorem:

$$\left\{ \begin{array}{l} \text{Languages} \\ \text{accepted} \\ \text{by NFAs} \end{array} \right\} = \left\{ \begin{array}{l} \text{Regular} \\ \text{Languages} \end{array} \right\}$$

Languages
accepted
by FAs

NFAs and FAs have the same computation power

We can show:

$$\left\{ \begin{array}{l} \text{Languages} \\ \text{accepted} \\ \text{by NFAs} \end{array} \right\} \supseteq \left\{ \begin{array}{l} \text{Regular} \\ \text{Languages} \end{array} \right\}$$

$$\left\{ \begin{array}{l} \text{Languages} \\ \text{accepted} \\ \text{by NFAs} \end{array} \right\} \subseteq \left\{ \begin{array}{l} \text{Regular} \\ \text{Languages} \end{array} \right\}$$

Proof-Step 1

$$\left\{\begin{array}{l}\text{Languages}\\ \text{accepted}\\ \text{by NFAs}\end{array}\right\} \supseteq \left\{\begin{array}{l}\text{Regular}\\ \text{Languages}\end{array}\right\}$$

Proof: Every FA is trivially an NFA

$\Downarrow$

Any language L accepted by a FA is also accepted by an NFA

Proof-Step 2

$$\left\{ \begin{array}{l} \text{Languages} \\ \text{accepted} \\ \text{by NFAs} \end{array} \right\} \subseteq \left\{ \begin{array}{l} \text{Regular} \\ \text{Languages} \end{array} \right\}$$

Proof: Any NFA can be converted to an equivalent FA

$\Downarrow$

Any language L accepted by an NFA is also accepted by a FA

Properties of Regular Languages

For regular languages L_1 and L_2:

Union: $L_1 \cup L_2$

Concatenation: $L_1 L_2$

Star: $L_1 *$

Reversal: L_1^R

Complement: $\overline{L_1}$

Intersection: $L_1 \cap L_2$

$\left.\right\}$ Are regular Languages

We say: Regular languages are **closed under**

Union: $L_1 \cup L_2$

Concatenation: $L_1 L_2$

Star: $L_1 *$

Reversal: L_1^R

Complement: $\overline{L_1}$

Intersection: $L_1 \cap L_2$

Non-regular languages

Non-regular languages

$$\{a^n b^n : \quad n \geq 0\}$$

$$\{vv^R : \quad v \in \{a, b\}^*\}$$

Regular languages

$$a^* b$$

$$b^* c + a$$

$$b + c(a + b)^*$$

$$etc...$$

How can we prove that a language L is not regular?

Prove that there is no DFA that accepts L

Problem: this is not easy to prove

Solution: the Pumping Lemma !!!

Regular Expressions

Regular Expressions

Regular expressions
describe regular languages

Example: $(a + b \cdot c)^*$

describes the language

$$\{a, bc\}^* = \{\lambda, a, bc, aa, abc, bca, \ldots\}$$

Recursive Definition

Primitive regular expressions: $\emptyset, \quad \lambda, \quad \alpha$

Given regular expressions r_1 and r_2

$$\left.\begin{array}{l} r_1 + r_2 \\ r_1 \cdot r_2 \\ r_1\,* \\ (r_1) \end{array}\right\} \text{Are regular expressions}$$

Examples

A regular expression: $(a+b\cdot c)^*\cdot(c+\varnothing)$

Not a regular expression: $(a+b+)$

Languages of Regular Expressions

$L(r)$: language of regular expression r

Example

$$L((a+b \cdot c)^*) = \{\lambda, a, bc, aa, abc, bca, ...\}$$

Definition

For primitive regular expressions:

$$L(\varnothing) = \varnothing$$

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

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

Definition (continued)

For regular expressions r_1 and r_2

$$L(r_1 + r_2) = L(r_1) \cup L(r_2)$$

$$L(r_1 \cdot r_2) = L(r_1)\, L(r_2)$$

$$L(r_1 *) = (L(r_1))*$$

$$L((r_1)) = L(r_1)$$

Example

Regular expression: $(a+b)\cdot a^*$

$$
\begin{aligned}
L((a+b)\cdot a^*) &= L((a+b))\,L(a^*) \\
&= L(a+b)\,L(a^*) \\
&= (L(a)\cup L(b))\,(L(a))^* \\
&= (\{a\}\cup\{b\})\,(\{a\})^* \\
&= \{a,b\}\,\{\lambda,a,aa,aaa,\ldots\} \\
&= \{a,aa,aaa,\ldots,b,ba,baa,\ldots\}
\end{aligned}
$$

Example

Regular expression $r = (a+b)^*(a+bb)$

$$L(r) = \{a, bb, aa, abb, ba, bbb, \ldots\}$$

Example

Regular expression $r = (aa)^*(bb)^*b$

$$L(r) = \{a^{2n}b^{2m}b: \quad n, m \geq 0\}$$

Example

Regular expression $r = (0+1)^*\, 00\, (0+1)^*$

$L(r)$ = { all strings with at least two consecutive 0 }

Theorem

$$\left\{ \begin{array}{l} \text{Languages} \\ \text{Generated by} \\ \text{Regular Expressions} \end{array} \right\} = \left\{ \begin{array}{l} \text{Regular} \\ \text{Languages} \end{array} \right\}$$

Standard Representations of Regular Languages