CS-305

Formal Language & Automata Theory

Instructor: Dr. Ashish Phophalia
Asst. Prof., IIIT-V

ashish_p@iiitvadodara.ac.in

Reference Books:

1. *Peter Linz*
2. *Micheal Sipser*
3. K. L. Mishra & Chandrashekran
4. Kamala Kirtivasan,
5. John C. Martin
6. Aho, Ullman, Sethi
7. Dexter Kozen
8. Lewis & Papadimitriou
9. John Sevage
10. Vivek Kulkarni
11. ...

Course Resources

- Offered to all major universities/colleges around the globe in CS stream
- NPTEL video lectures
- You are free to refer course website of other reputed universities/faculties

Video Lectures

1. Prof. Somnath Biswas, IIT Kanpur
2. Prof. Kamala K., IIT Madras
3. Prof. J. Ullman, Coursera/Stanford
4. Prof. Shai Simonson, ArsDigita University

Purpose of Course

- Historical Perceptive – Current Computation modeling
- Foundation course to computer science & research in relevant areas
- Major part in many competitive exams like GATE

Course Content

Mathematical Preliminaries: Set, Functions, Relation, Graph Theory, Mathematical Induction, Proof Techniques

Finite Automata: DFA, NDFA, Conversion b/w DFA & NDFA, Melay & Moore Machine, Minimization of automata

Languages & Grammars: Types and Properties of Chomsky classification

Regular Languages & Grammar, Pumpimg Leema

Context Free Language, Grammar & Pushdown Automata, Deterministic Context Free Language and Automatam, Pumping Leema

Context Sensitive Language, Grammar & Linear Bounded automata

Turning Machines & its variants, Undecideability & Reduceability

Computational Complexity: P, NP, NP Complete and Hard Problems, Post Correspondence Problem (PCP)

Course Goals

Provide computation Models

Analyze power of Models

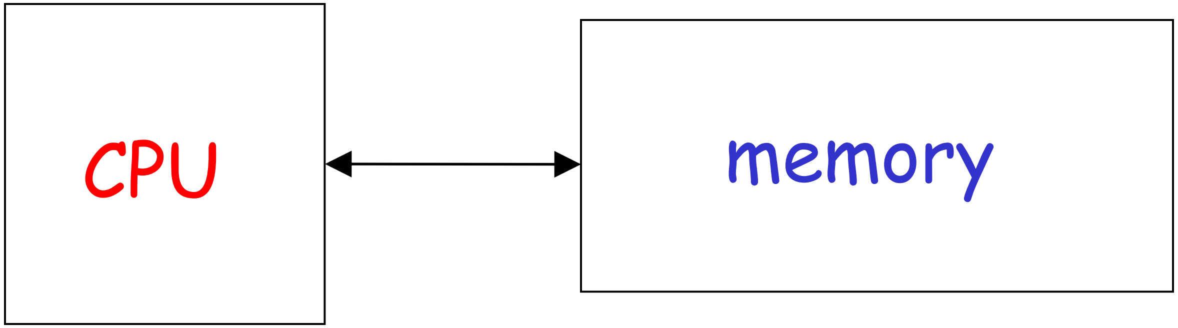
Answer Intractability questions:

What computational problems can each model solve?

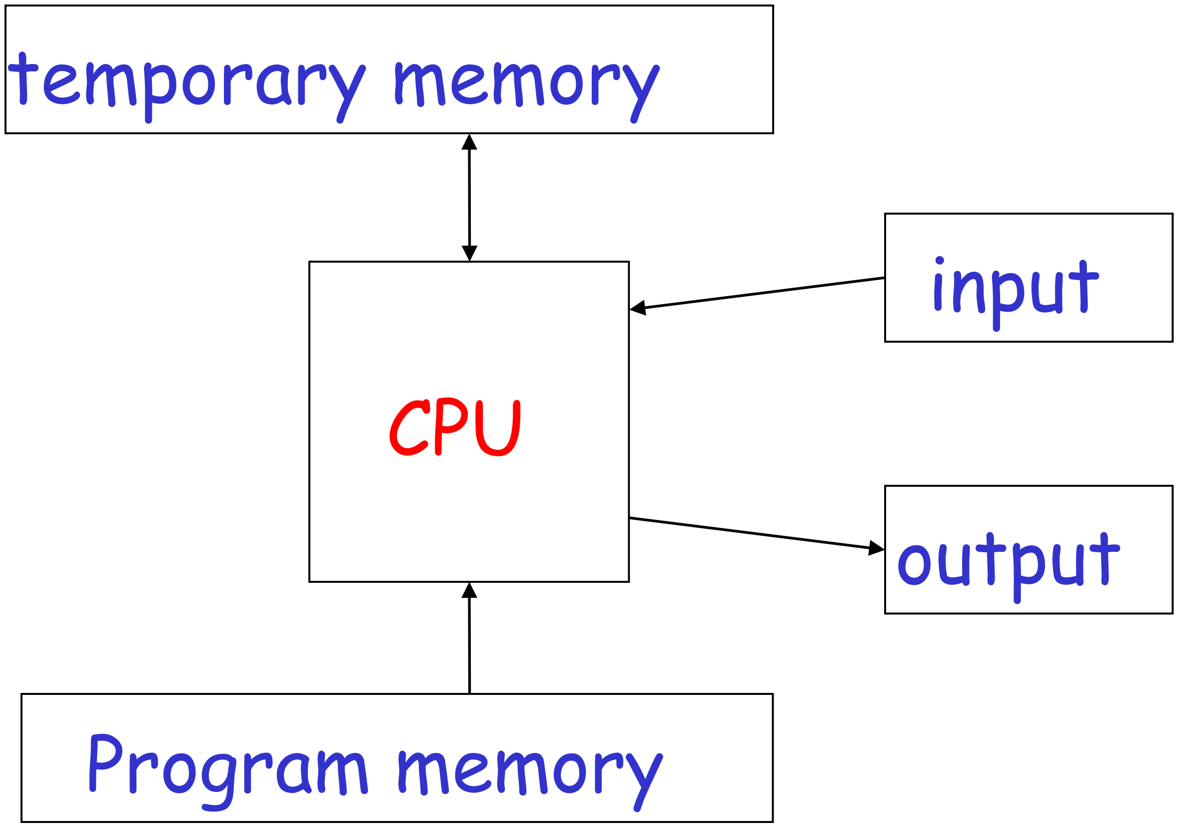
Answer Time Complexity questions:

How much time we need to solve the problems?

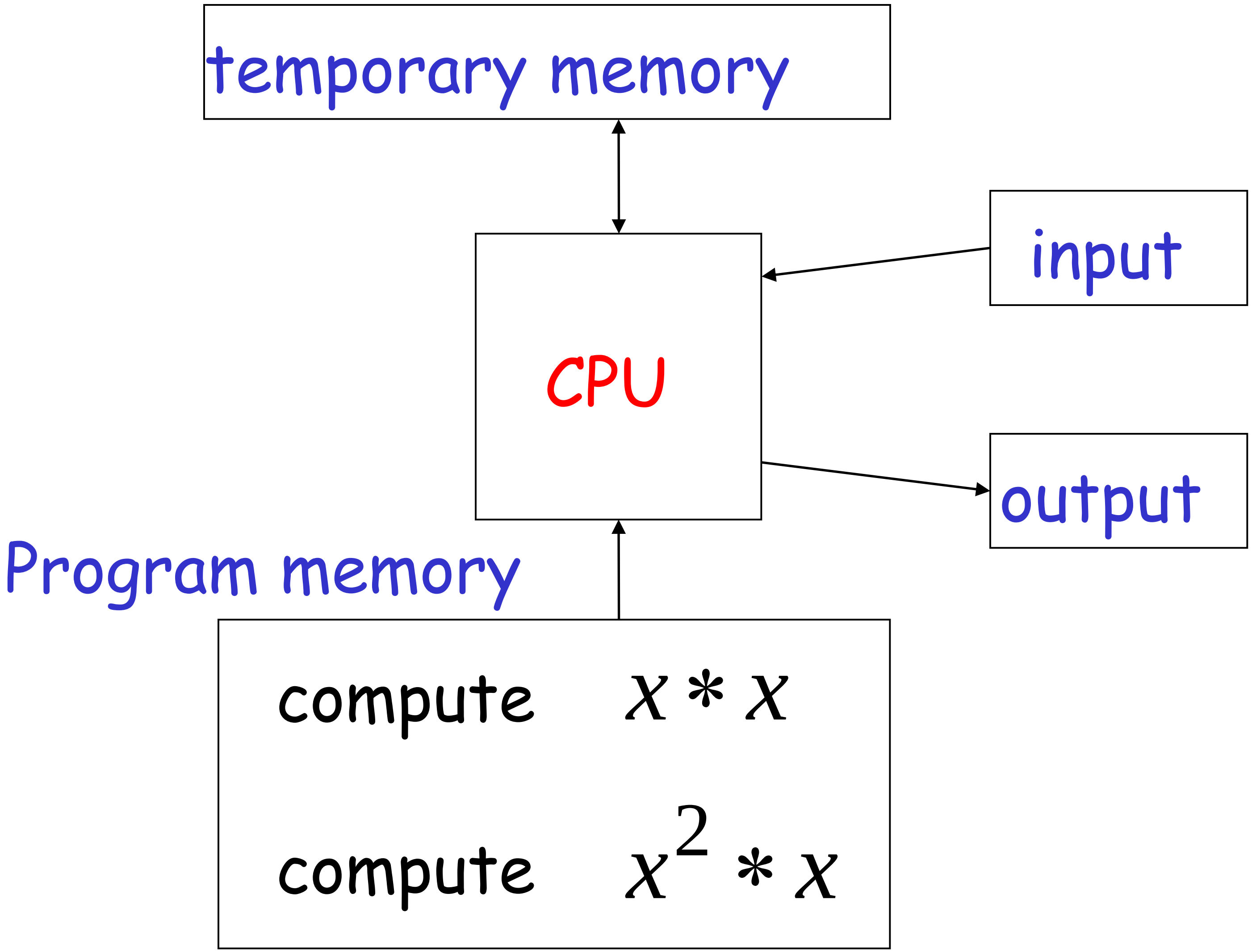
A widely accepted model of computation



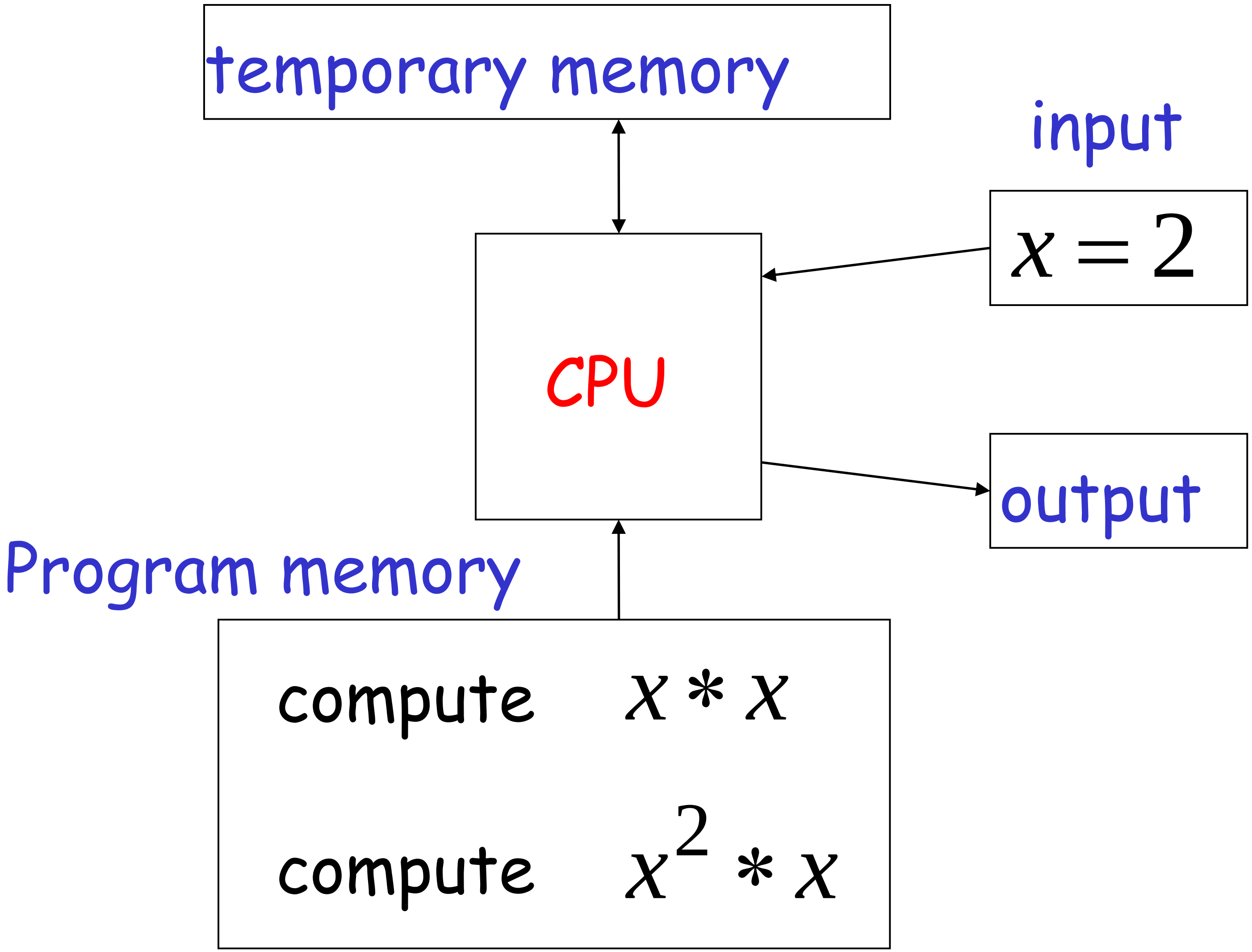
The different components of memory



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

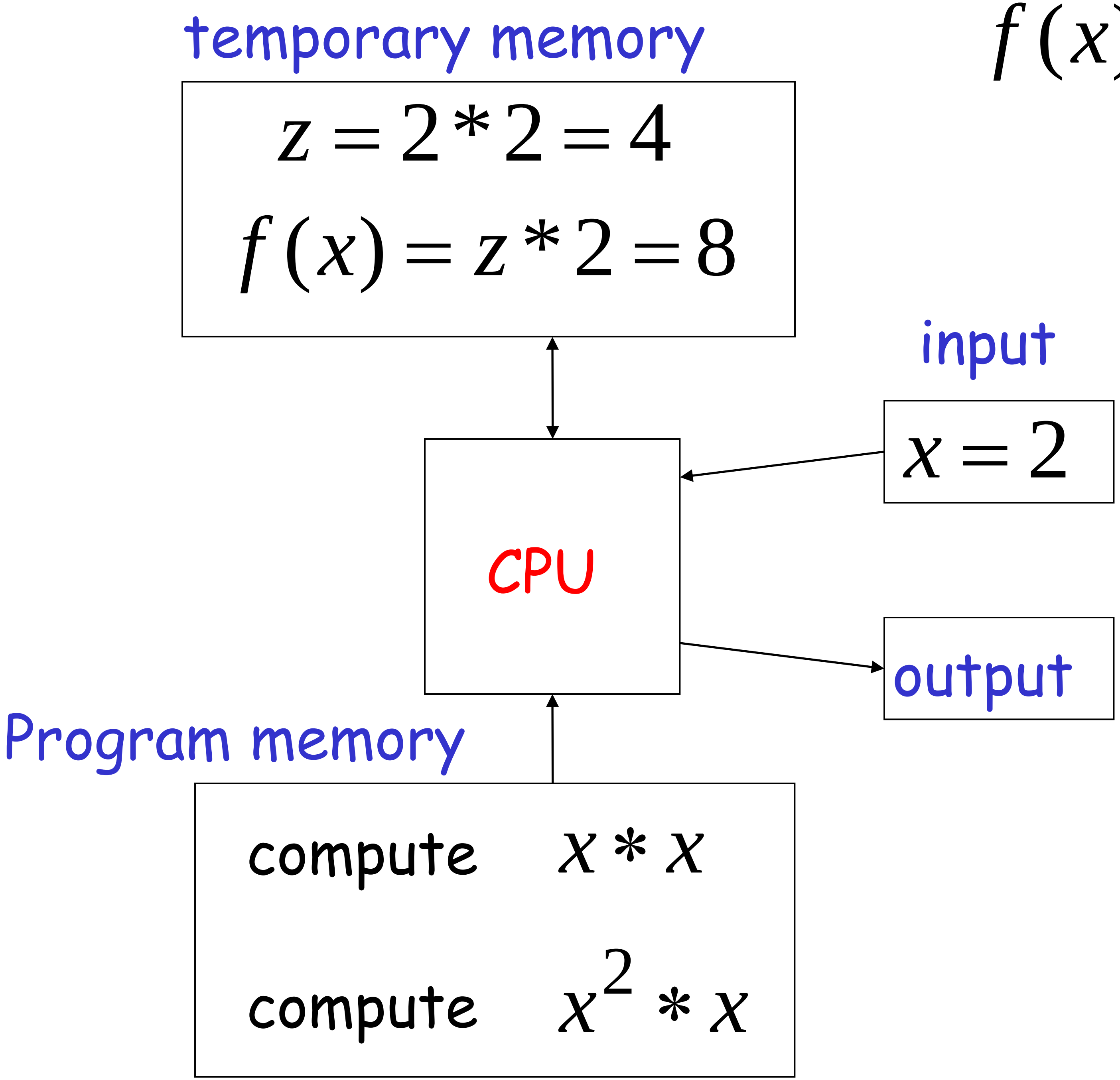


$$f(x) = x^3$$

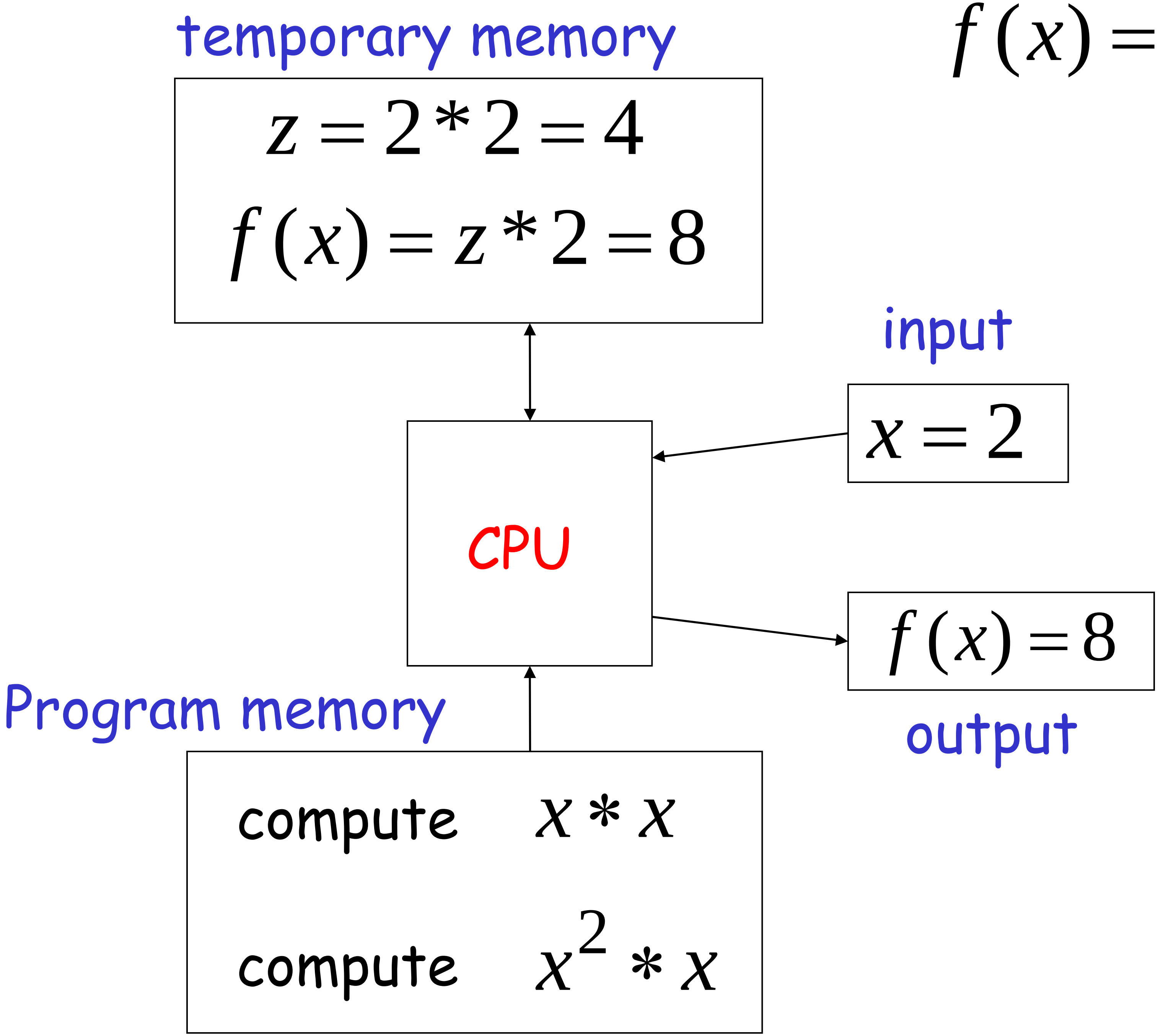


temporary memory
input
$x = 2$
CPU
output
Program memory
compute $x * x$
compute $x^2 * x$

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

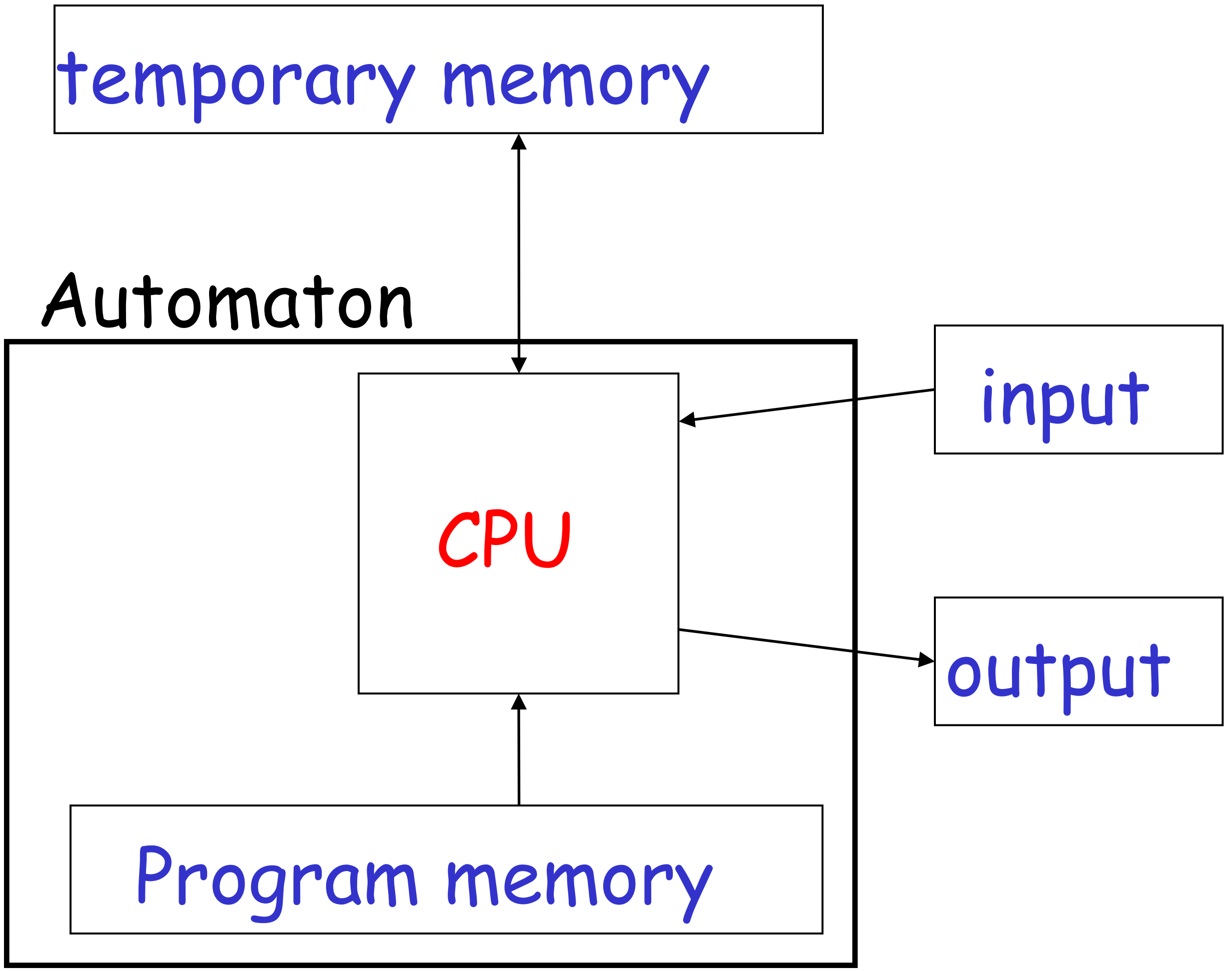

temporary memory
$z = 2 * 2 = 4$
$f(x) = z * 2 = 8$
input
$x = 2$
CPU
output
Program memory
compute $x * x$
compute $x^2 * x$

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

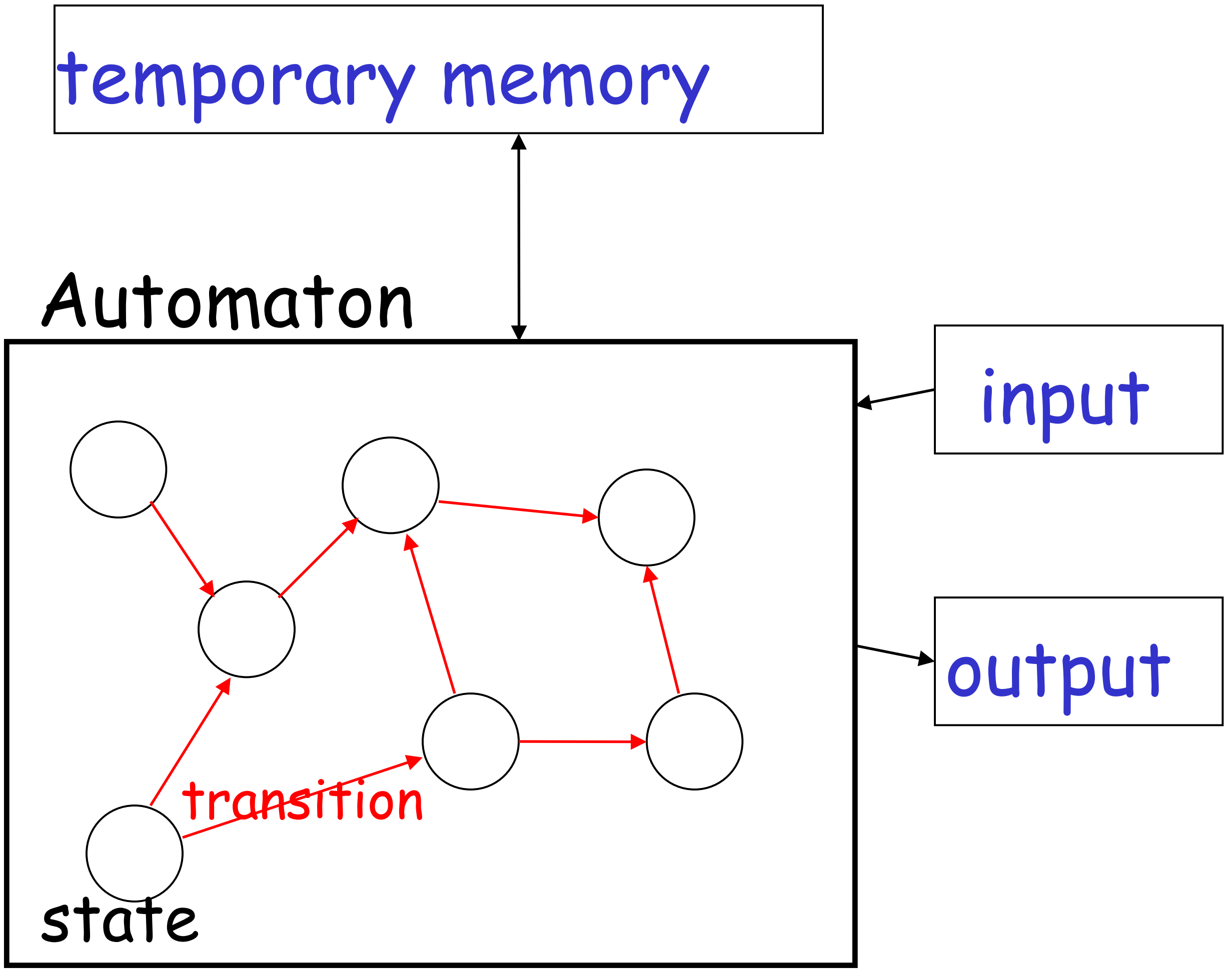


temporary memory
$z = 2*2 = 4$
$f(x) = z*2 = 8$
input
$x = 2$
CPU
$f(x) = 8$
output
Program memory
compute $x*x$
compute $x^2 * x$

Automaton


temporary memory
Automaton
CPU
input
output
Program memory

Automaton



CPU+ProgramMem = States + Transitions

Different Kinds of Automata

utomata are distinguished by the temporary memory

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

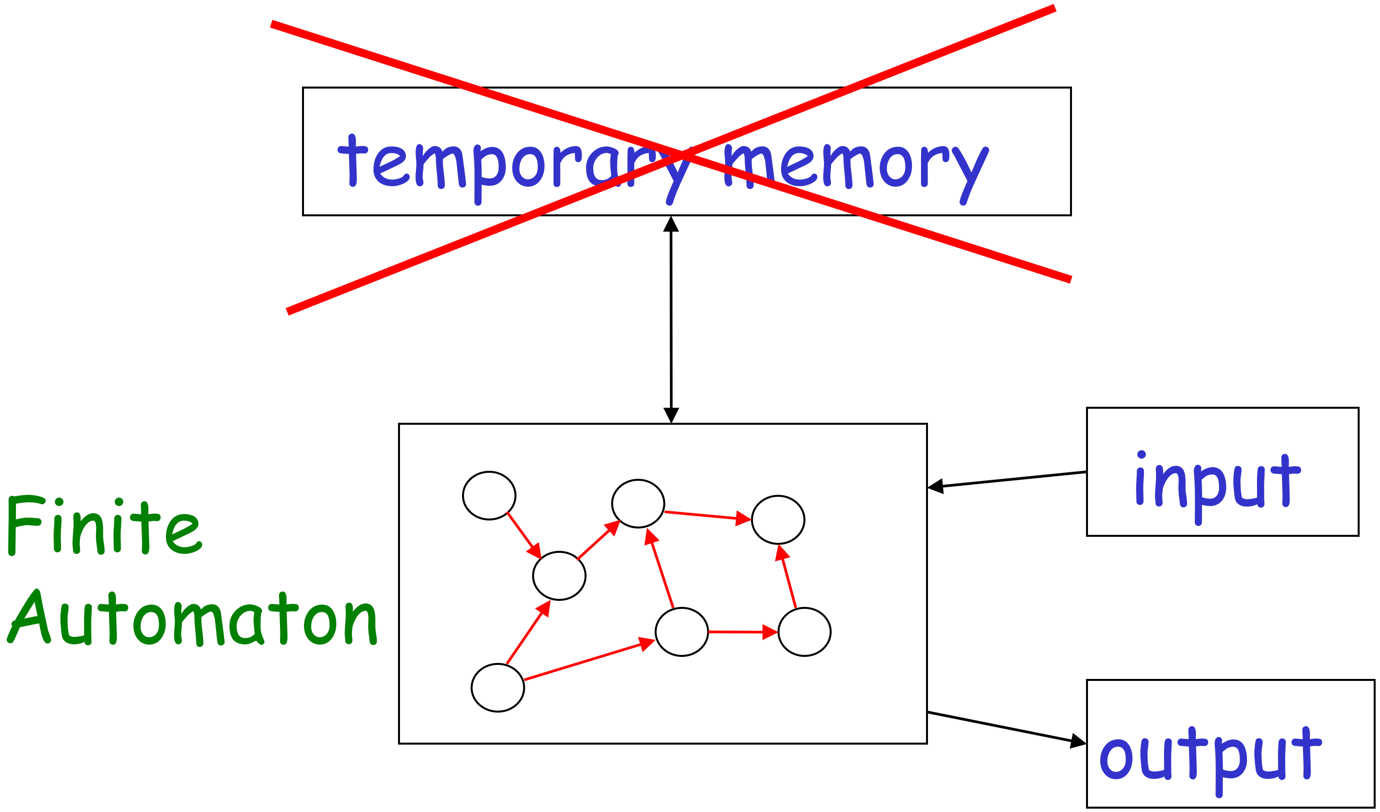
Memory affects computational power:

More flexible memory

results to

The solution of more computational problems

Finite Automaton

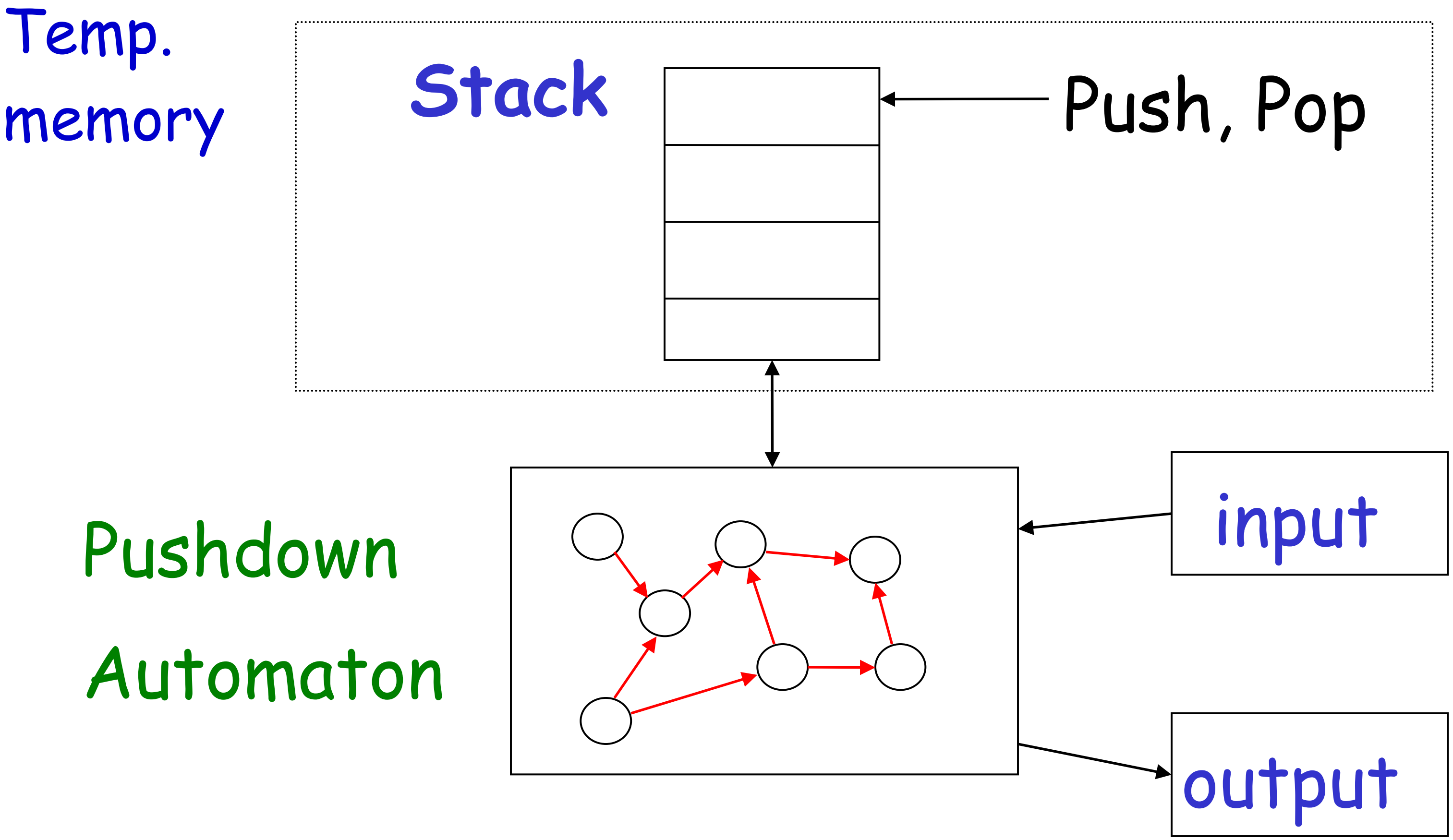


xample: Elevators, Vending Machines,

Lexical Analyzers

(small computing power)

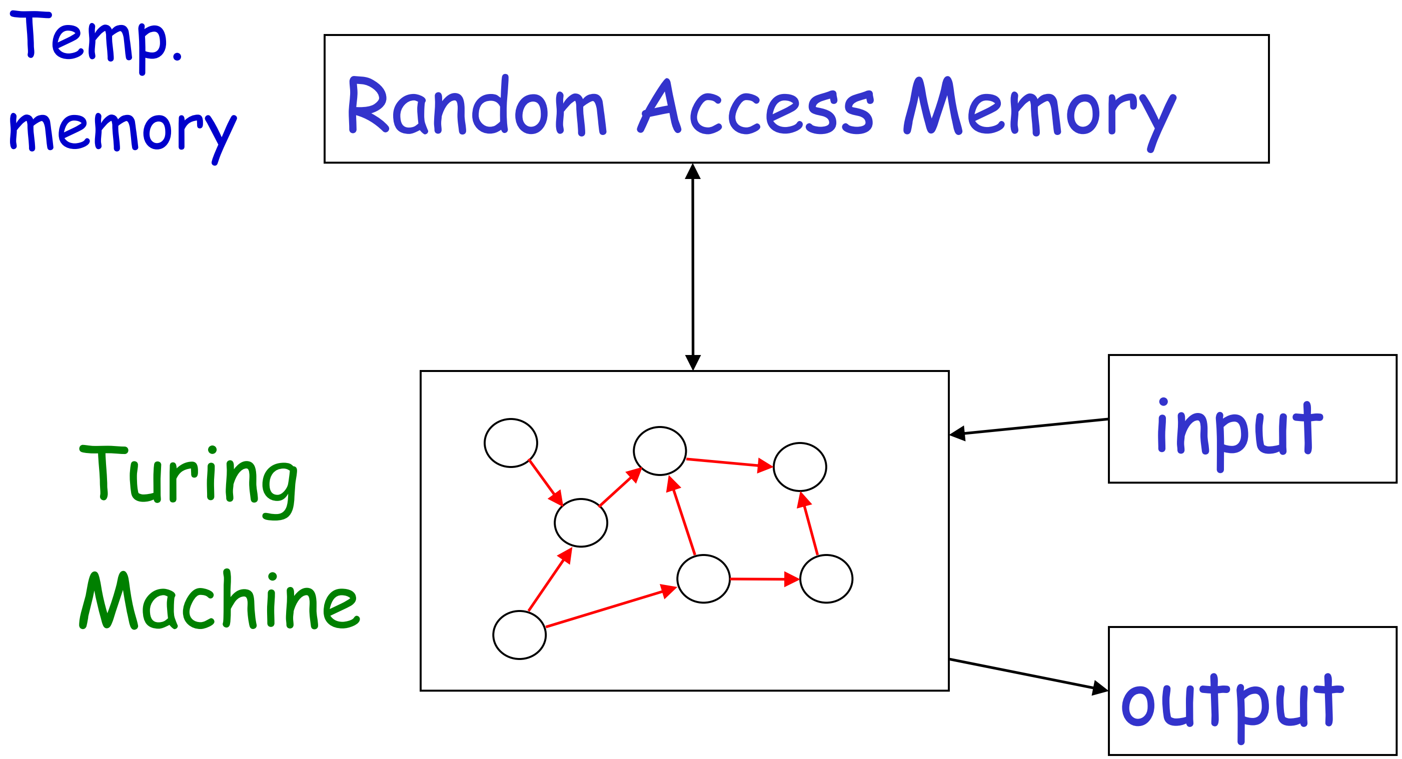
Pushdown Automaton



Example: Parsers for Programming Languages

(medium computing power)

Turing Machine



Examples: Any Algorithm

(highest known computing power)

Power of Automata

Simple problems	More complex problems	Hardest problems
Finite Automata	$<$ Pushdown Automata	$<$ Turing Machine

Less power $\longrightarrow$ More power

Solve more computational problems

Turing Machine is the most powerful known computational model

Question: can Turing Machines solve all computational problems?

Answer: NO

(there are unsolvable problems)

Time Complexity of Computational Problems:

P problems:

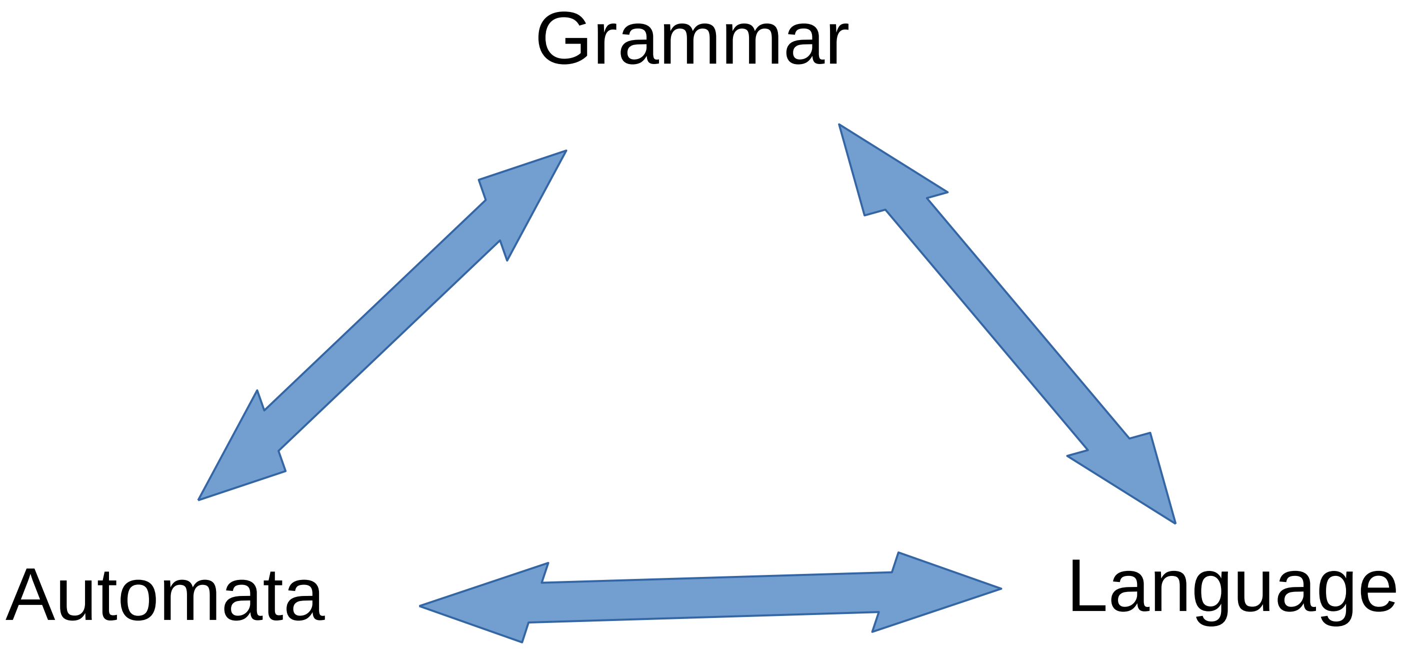
(Polynomial time problems)

Solved in polynomial time

NP-complete problems:

(Non-deterministic Polynomial time problems)

Believed to take exponential time to be solved



Grammar
Automata
Language

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 $\qquad u = ab$

$abba$ $\qquad v = bbbaaa$

$baba$ $\qquad w = abba$

$aaabbbaabab$

Alphabets and Strings

Alphabets: Finite Non-empty set of symbols

$$\Sigma = \{a, b\} \quad \Sigma = \{0,1\} \quad \Sigma = \{a, b, \ldots, z\}$$

String: Finite sequence of alphabets from set of symbols

- Empty/Null string will be default in any set of alphabets

$\in$ λ

String Operations

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

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

Concatenation

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

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

Reverse

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

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: $|\lambda| = 0$

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

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

Substring

Substring of string:
a subsequence of consecutive characters

String	Substring
$\underline{ab}bab$	ab
$\underline{abba}b$	$abba$
$ab\underline{b}ab$	b
$a\underline{bbab}$	$bbab$

Prefix and Suffix

$abbab$

Prefixes	Suffixes
λ	$abbab$
a	$bbab$
ab	bab
abb	ab
$abba$	b
$abbab$	λ

$w = uv$

prefix

suffix

Another Operation

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

Example: $(abba)^2 = abbaabba$

Definition: $w^0 = \lambda$

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

The * Operation

$\Sigma *$: the set of all possible strings from alphabet Σ

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

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

The + Operation

Σ^+ : the set of all possible strings from alphabet Σ except λ

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

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

$$\Sigma^+ = \Sigma^* - \lambda$$

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

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 $\varnothing = \{\ \} \neq \{\lambda\}$

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

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

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

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

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

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

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: $L^* = L^0 \cup L^1 \cup L^2 \cdots$

Example:

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

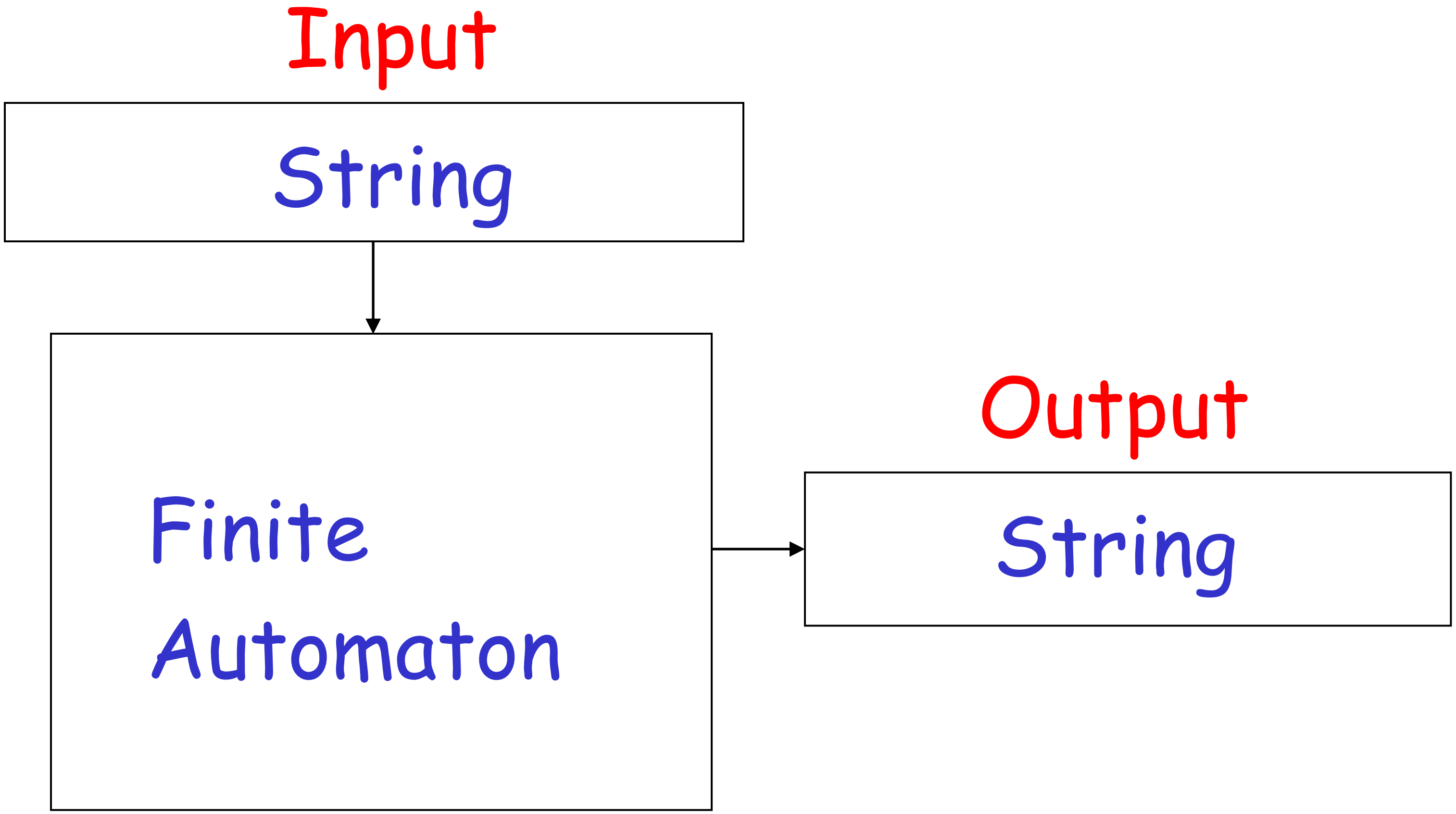
Positive Closure

Definition: $$L^{+} = L^{1} \cup L^{2} \cup \cdots$$
$$= L^{*} - \{\lambda\}$$

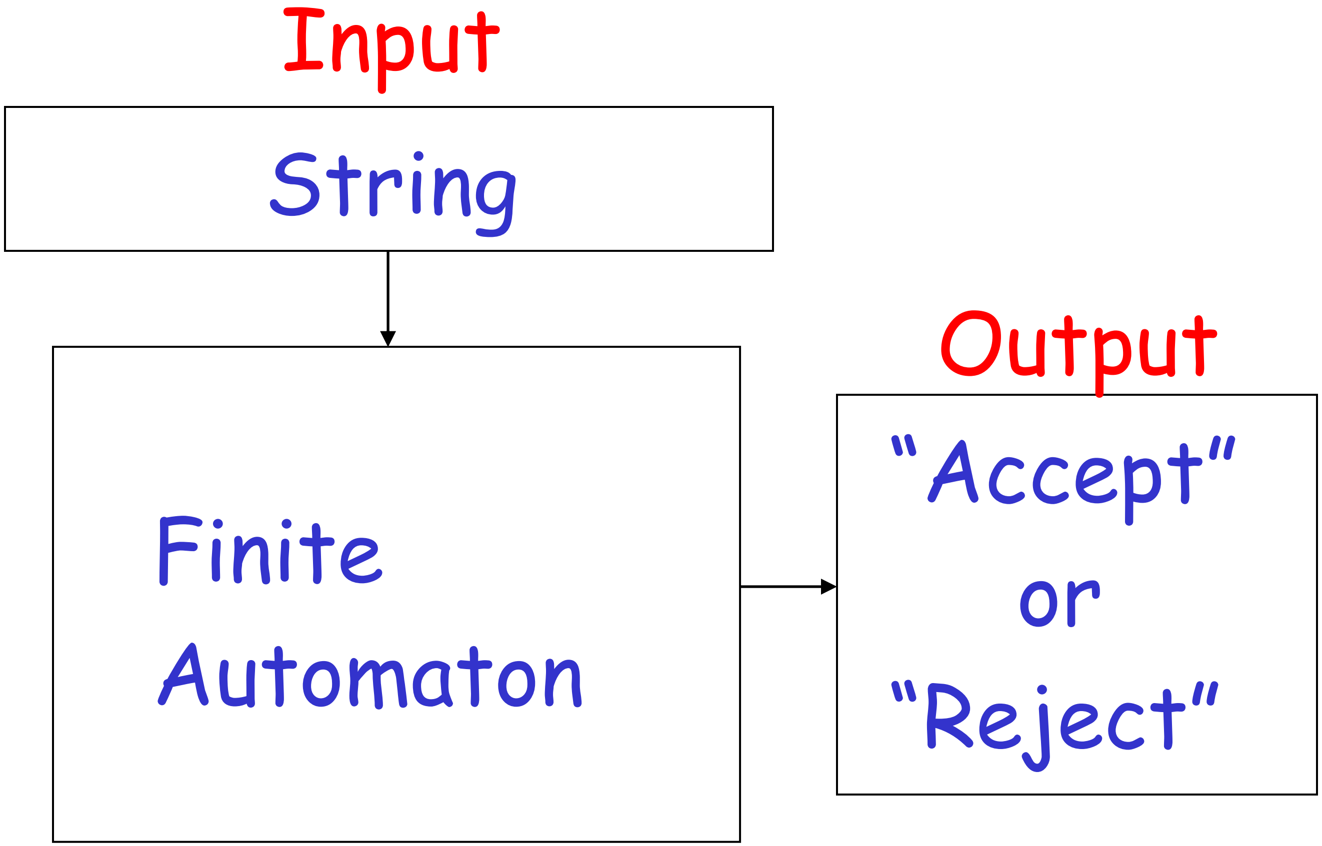
$$\{a,bb\}^{+} = \left\{ \begin{array}{l} a, bb, \\ aa, abb, bba, bbbb, \\ aaa, aabb, abba, abbbb, \ldots \end{array} \right\}$$

Finite Automata

Finite Automaton

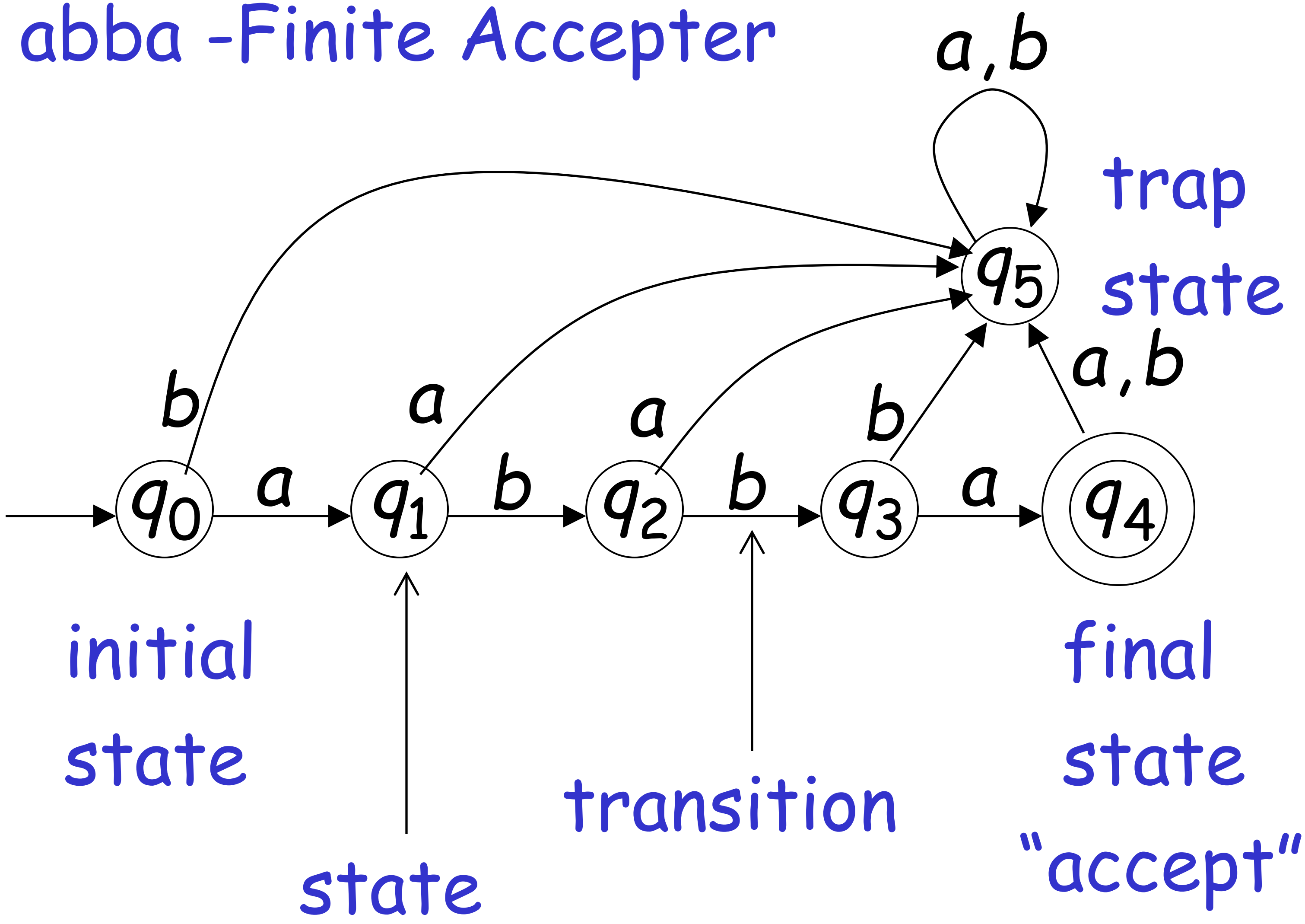


Finite Accepter



Transition Graph

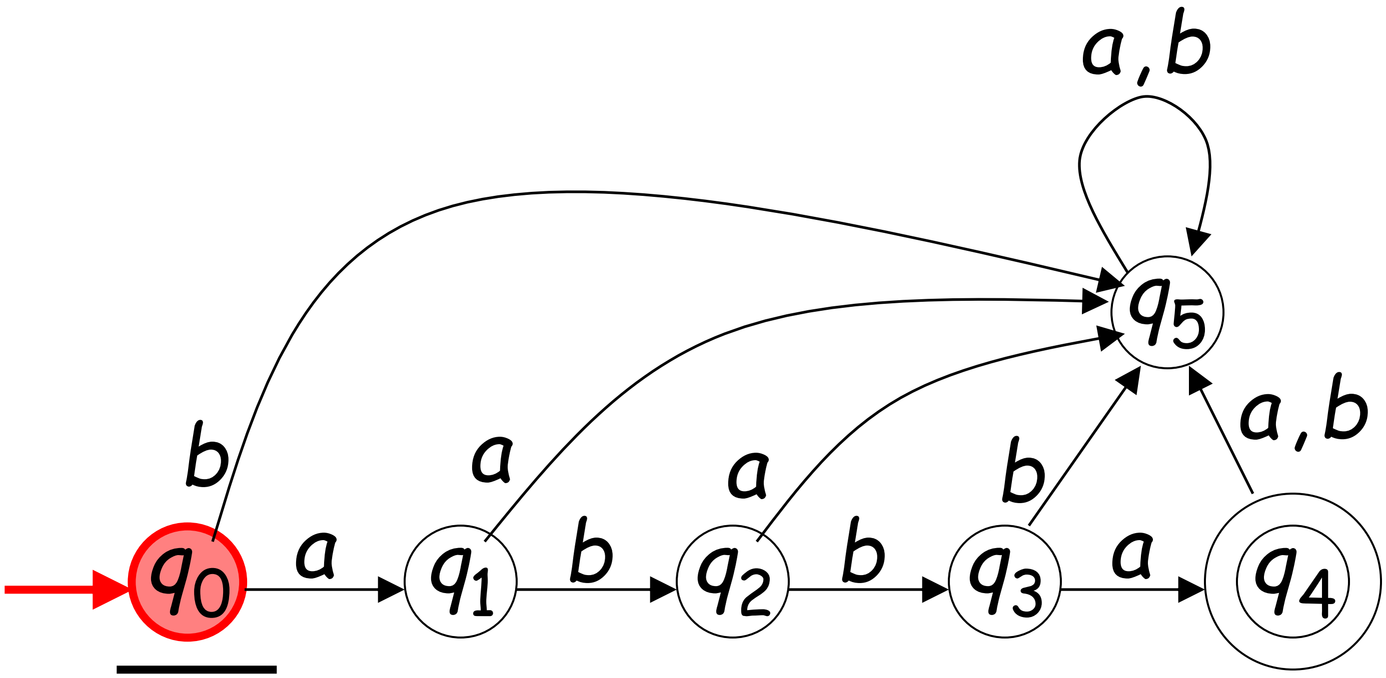
abba -Finite Accepter



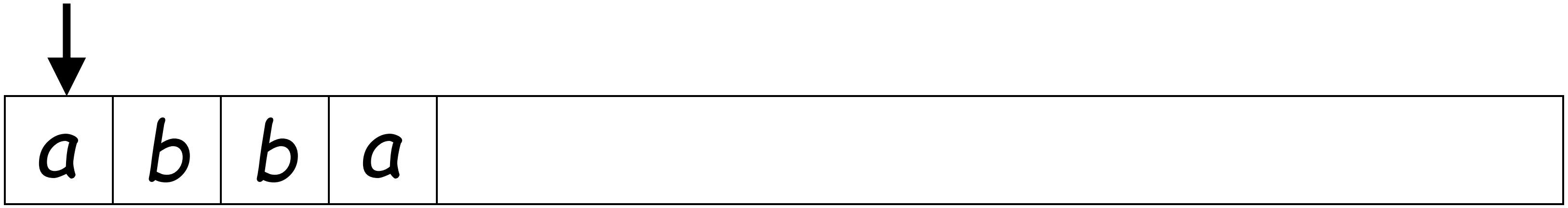
Initial Configuration

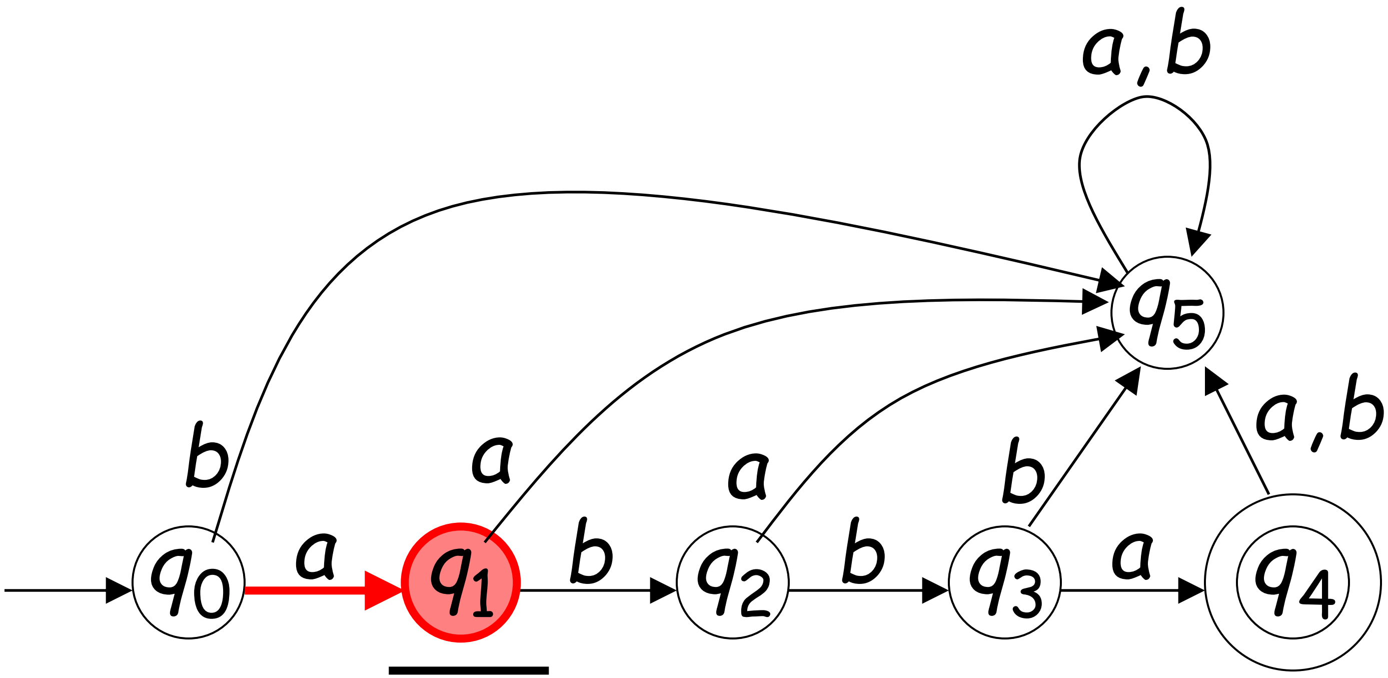
Input String

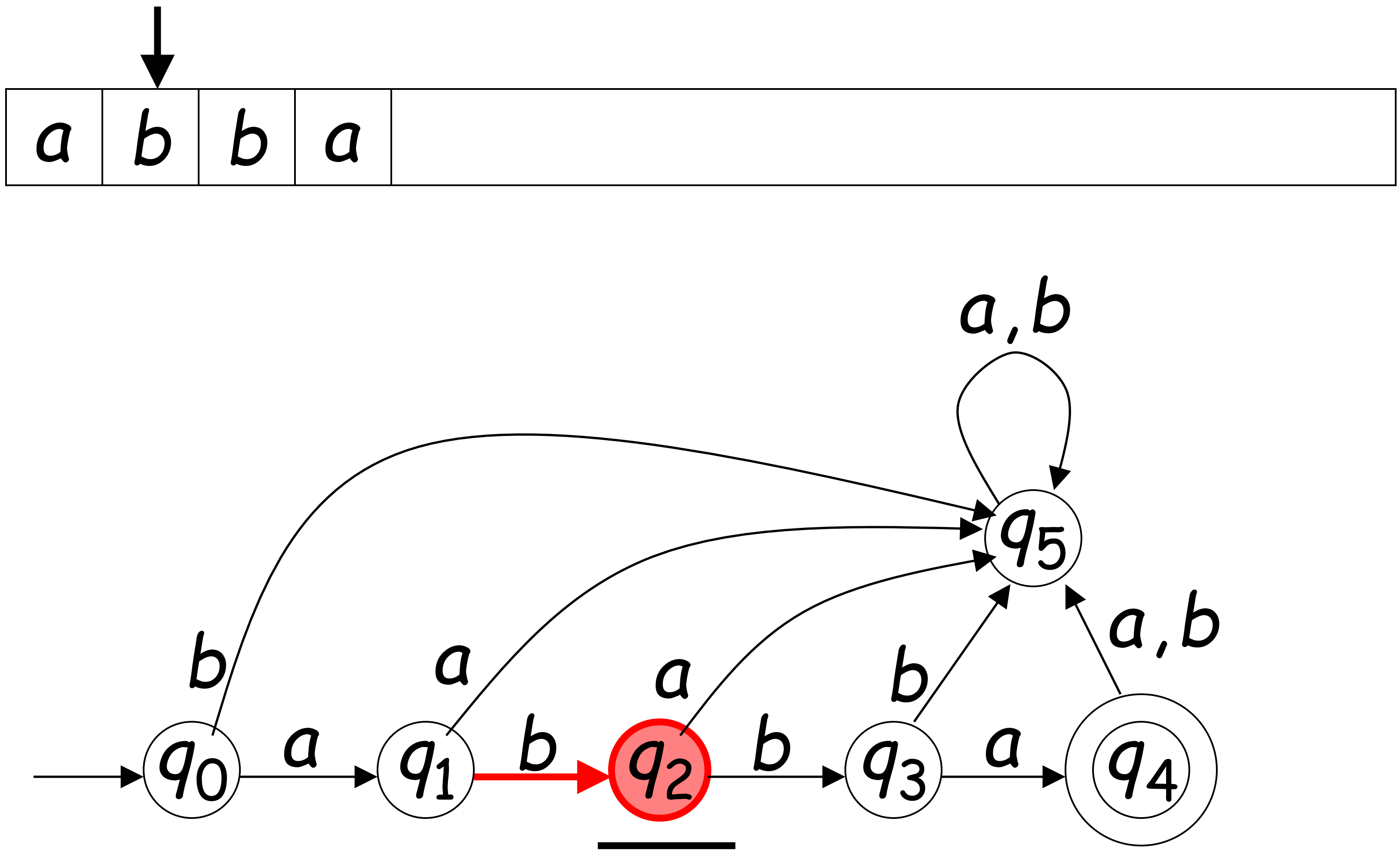
a	b	b	a	

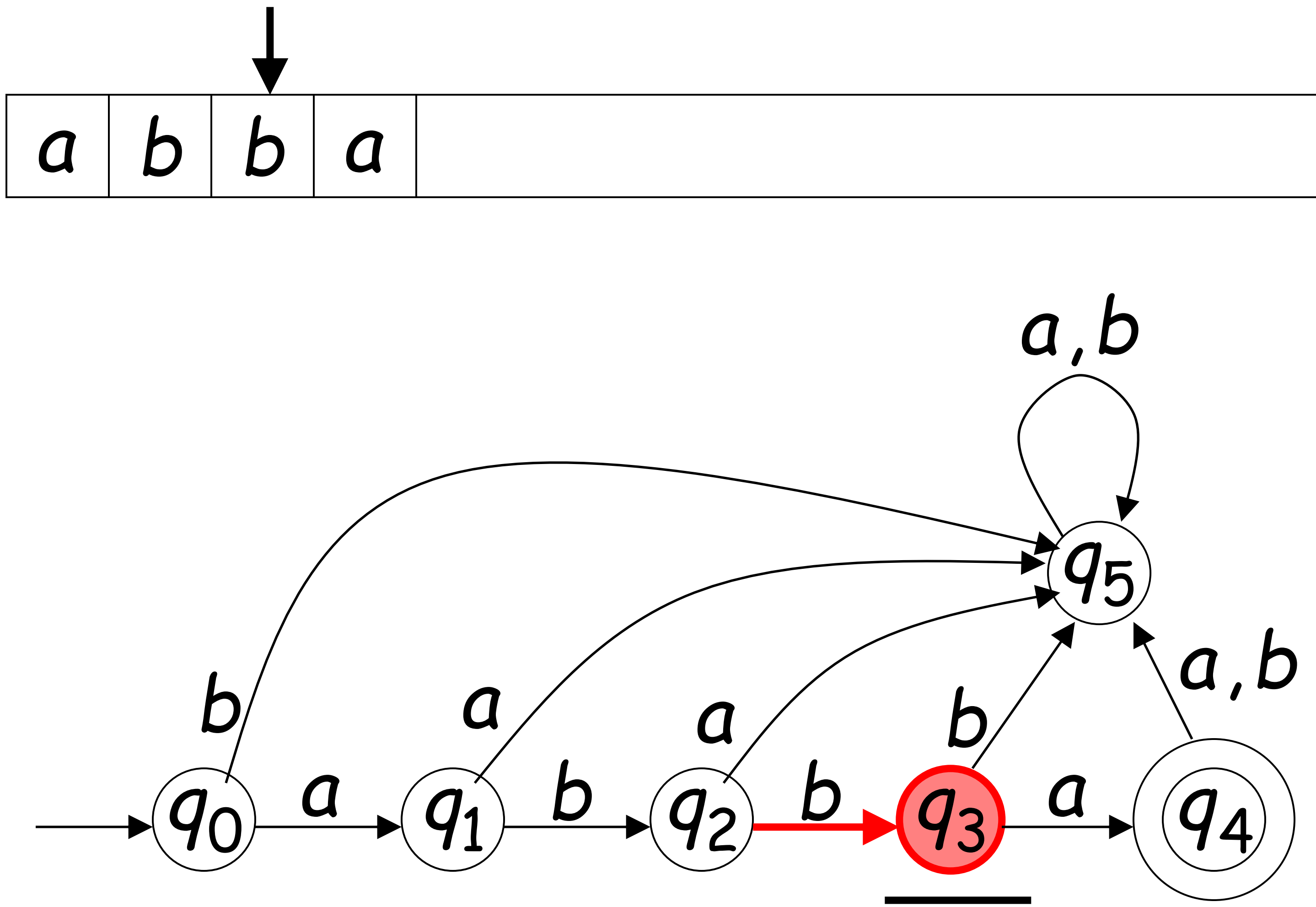


Reading the Input

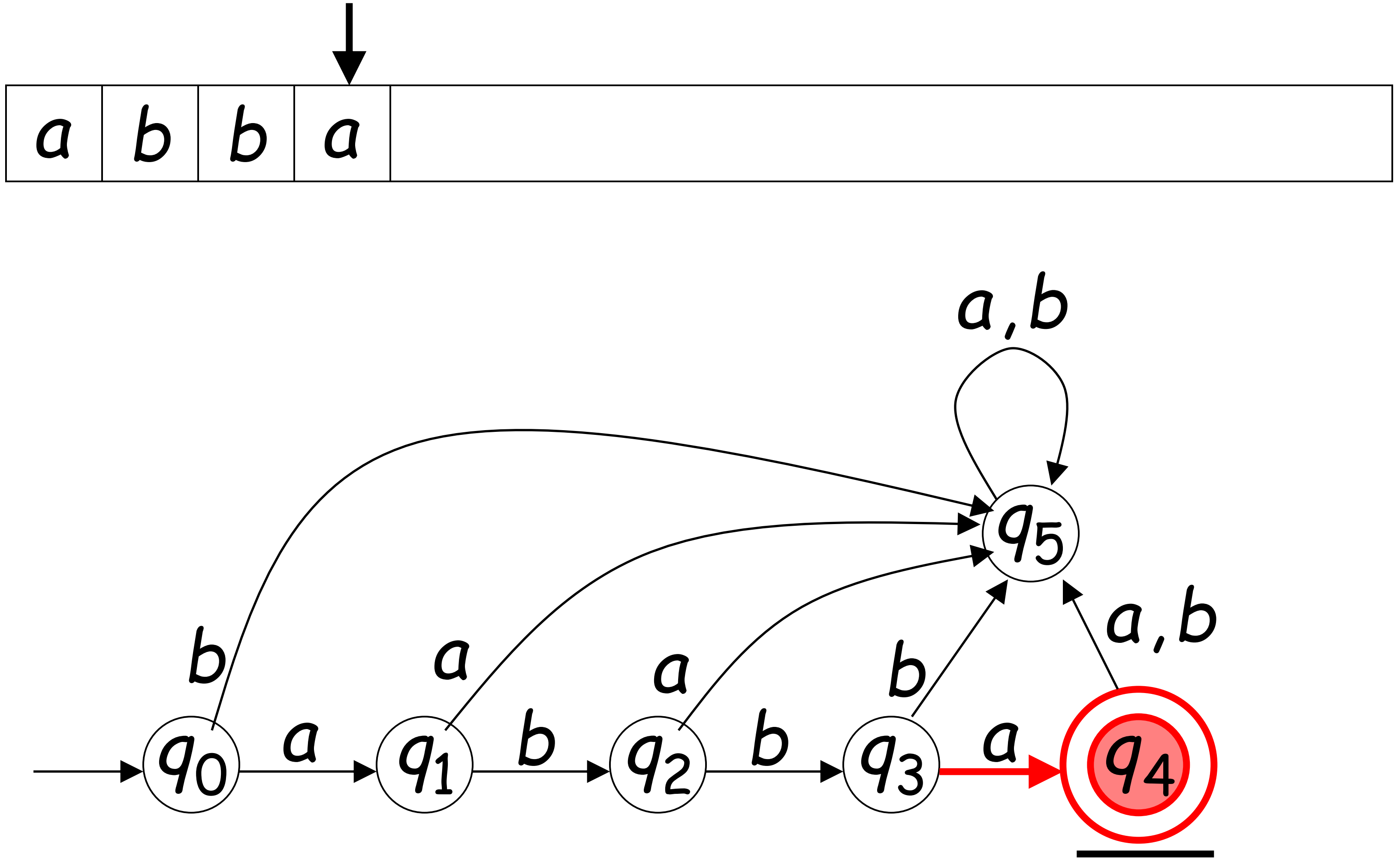





a
b
b
a
a,b
b
a
a
b
a
b
a
a,b
q0
q1
q2
q3
q4
q5

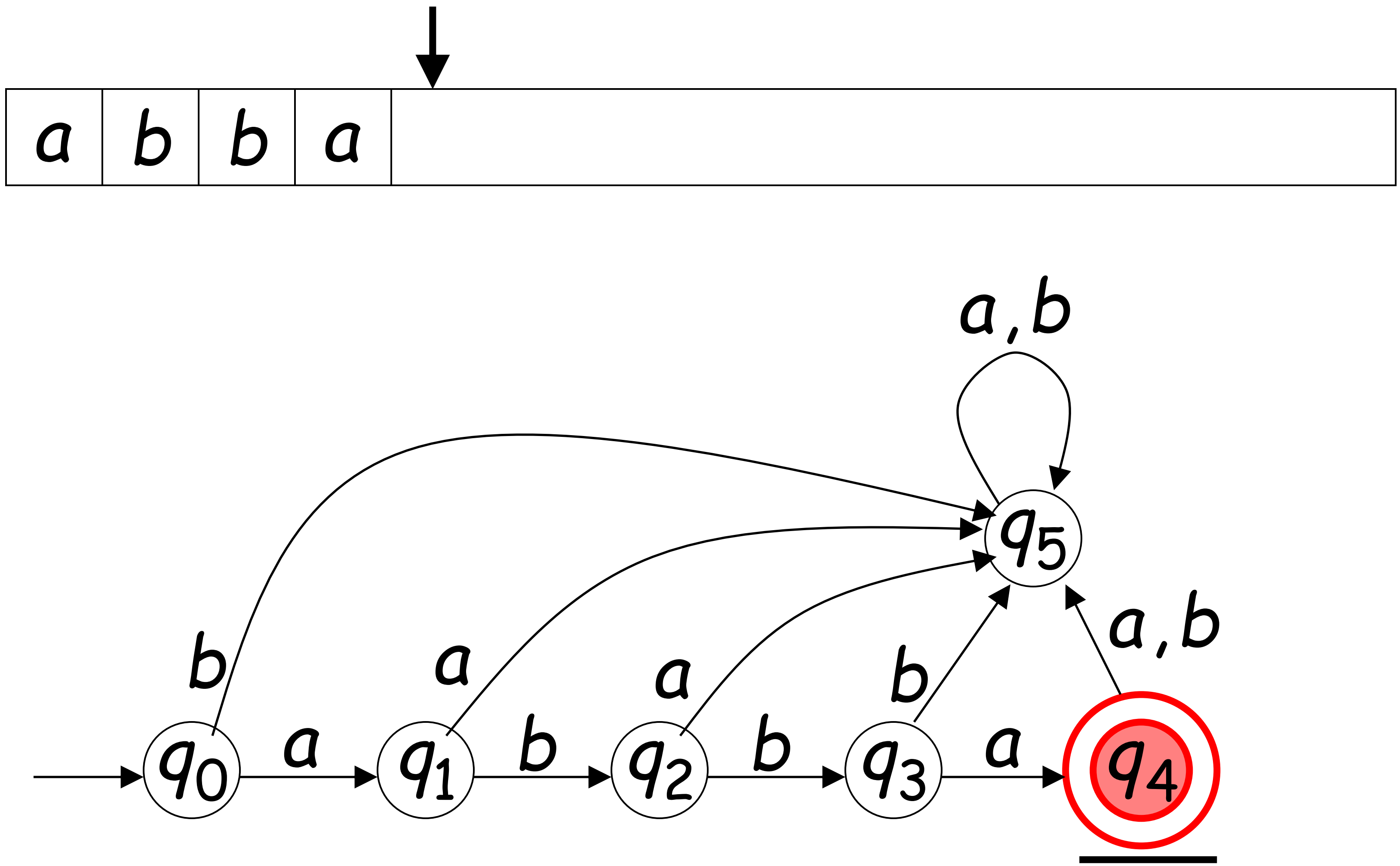


a
b
b
a
a,b
q_5
a,b
b
a
a
b
q_0
a
q_1
b
q_2
b
q_3
a
q_4



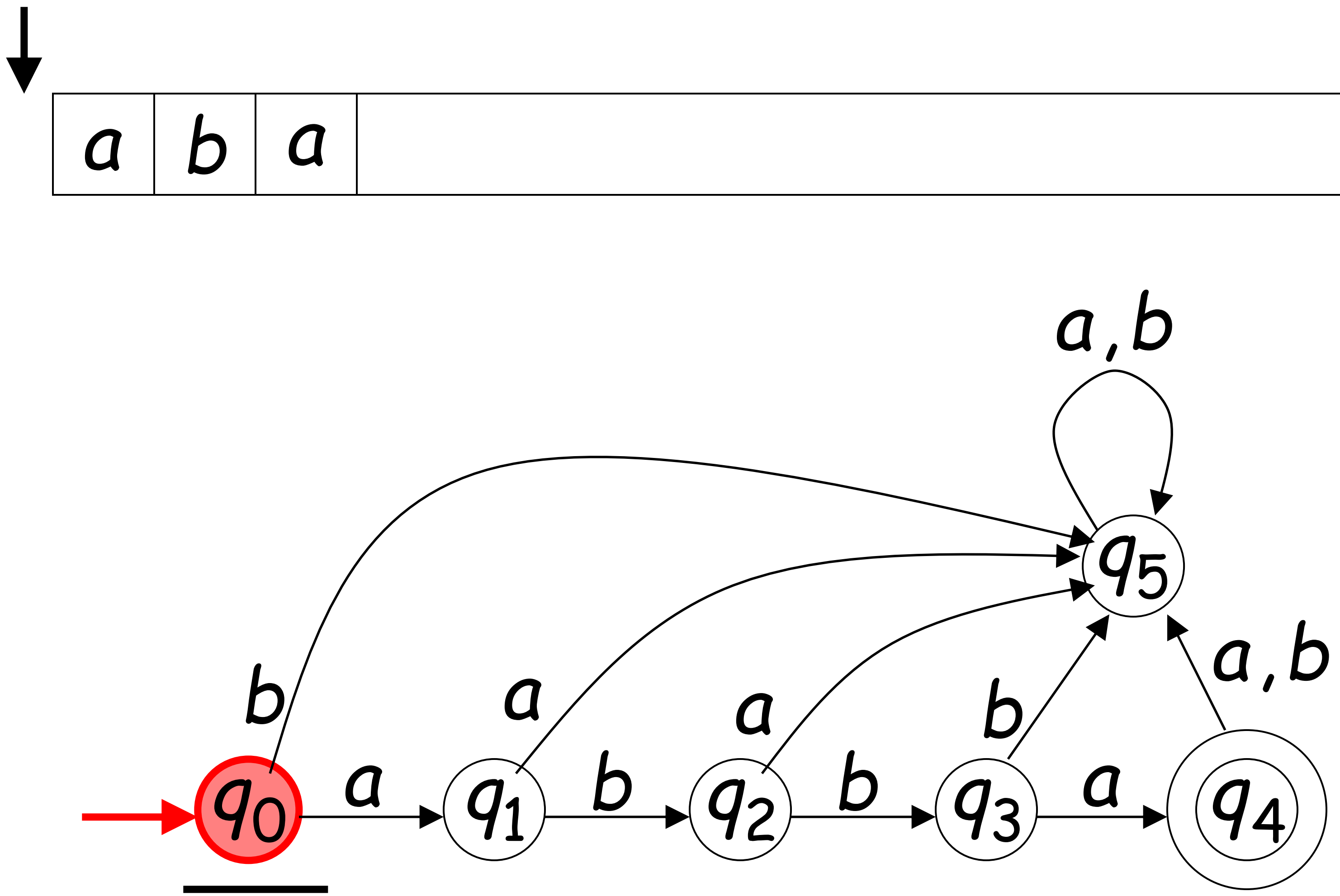
a
b
b
a
a, b
b
a
a
b
a, b
q_0
a
q_1
b
q_2
b
q_3
a
q_4
q_5

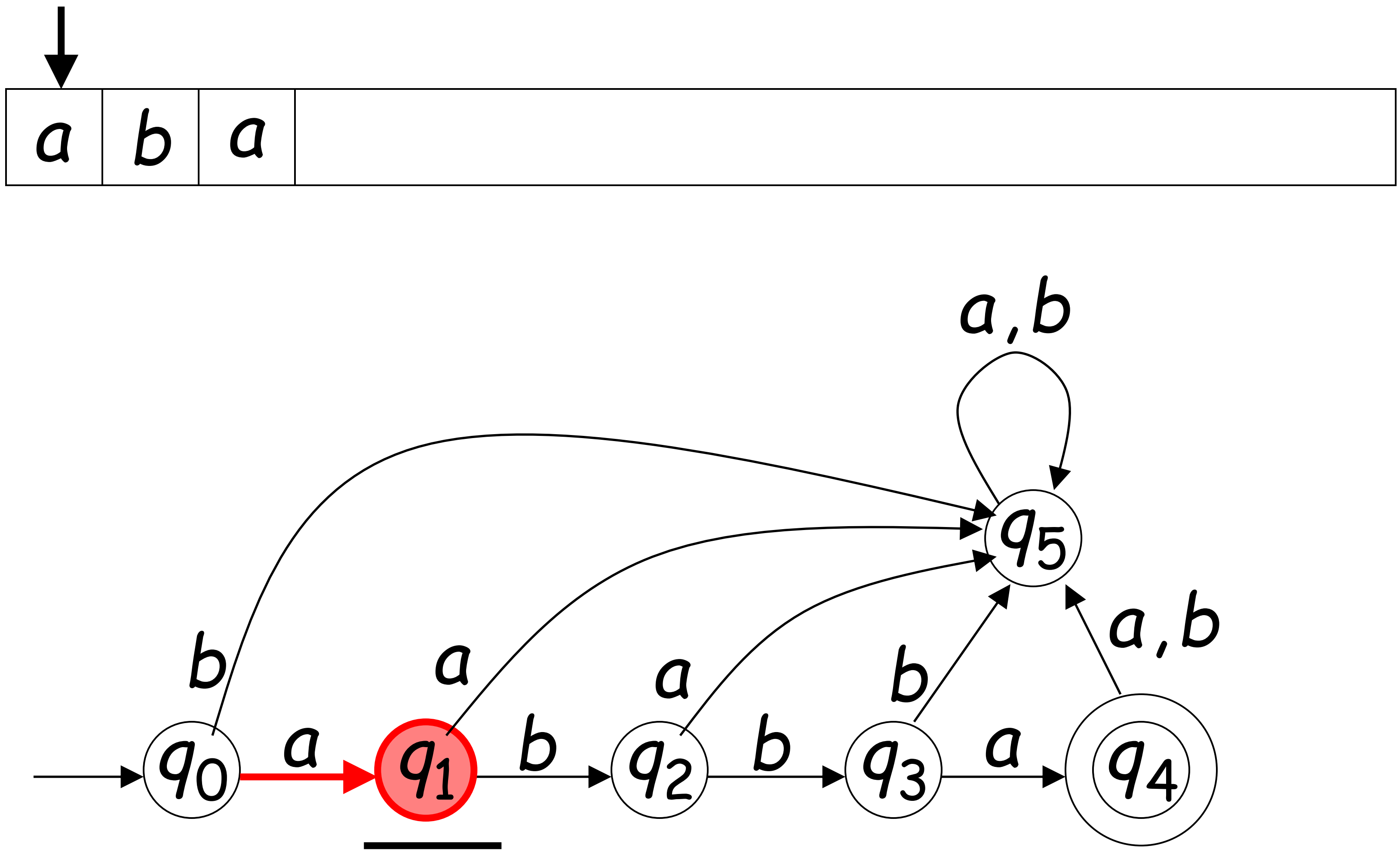
Input finished

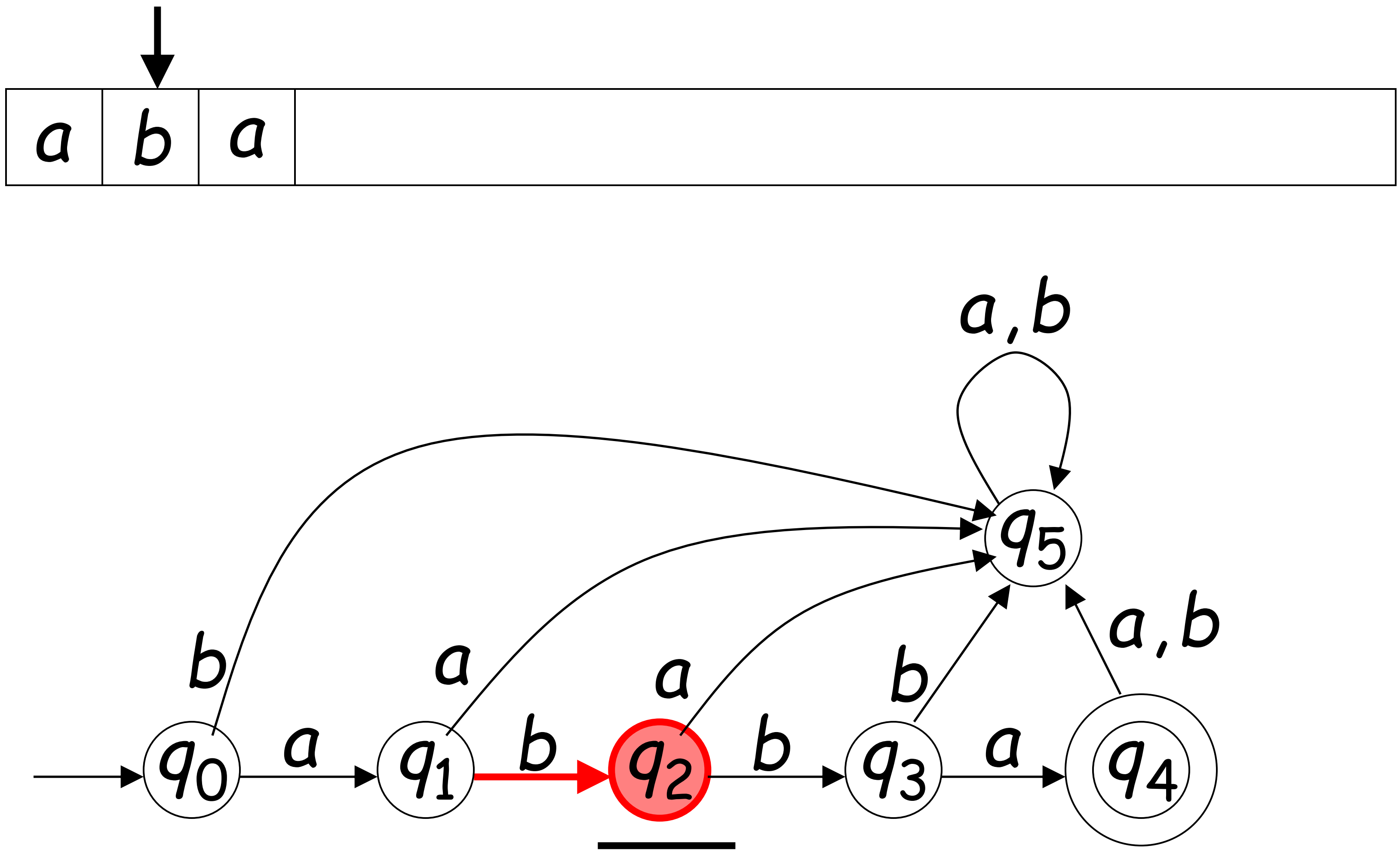


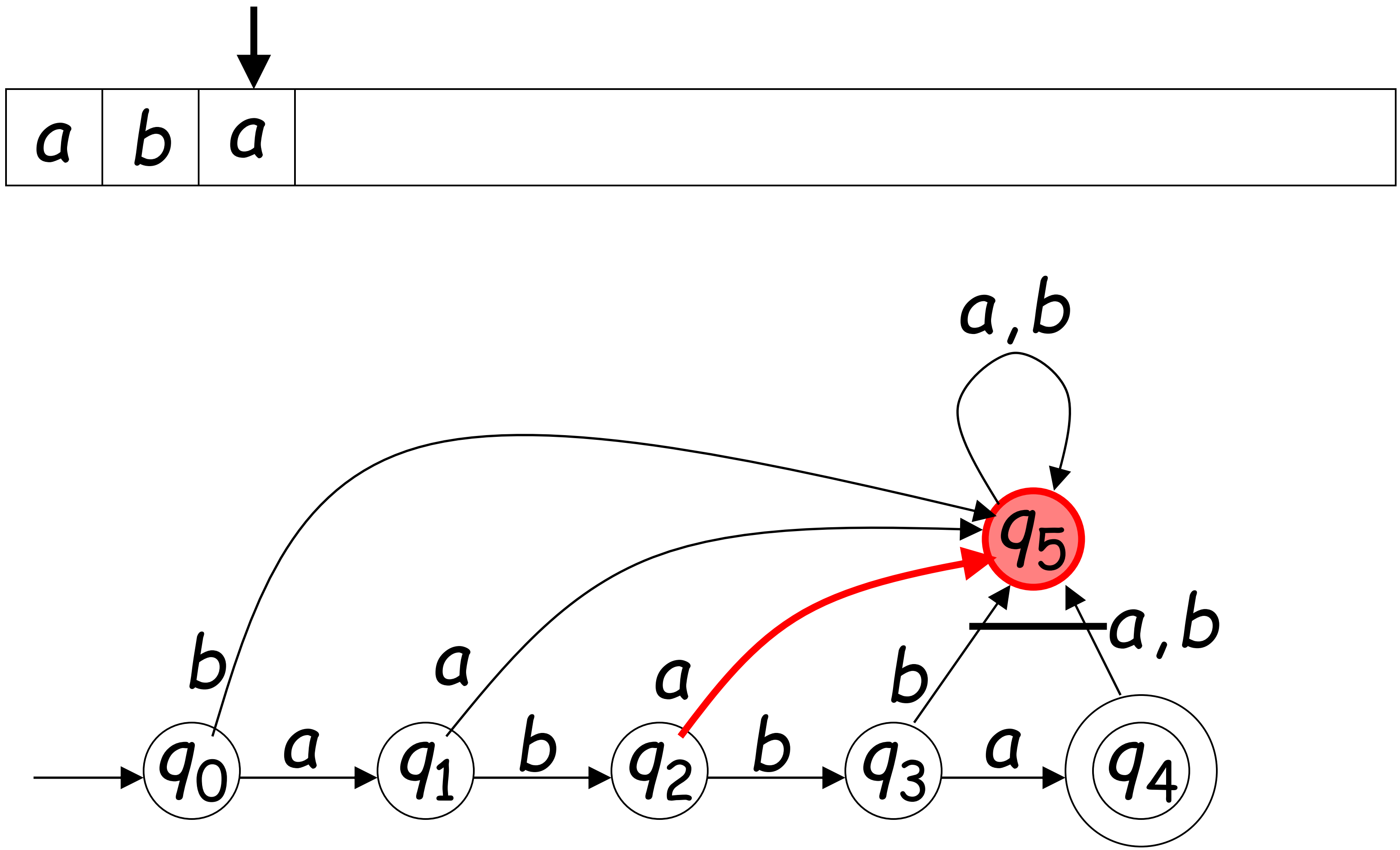
Output: “accept”

Rejection



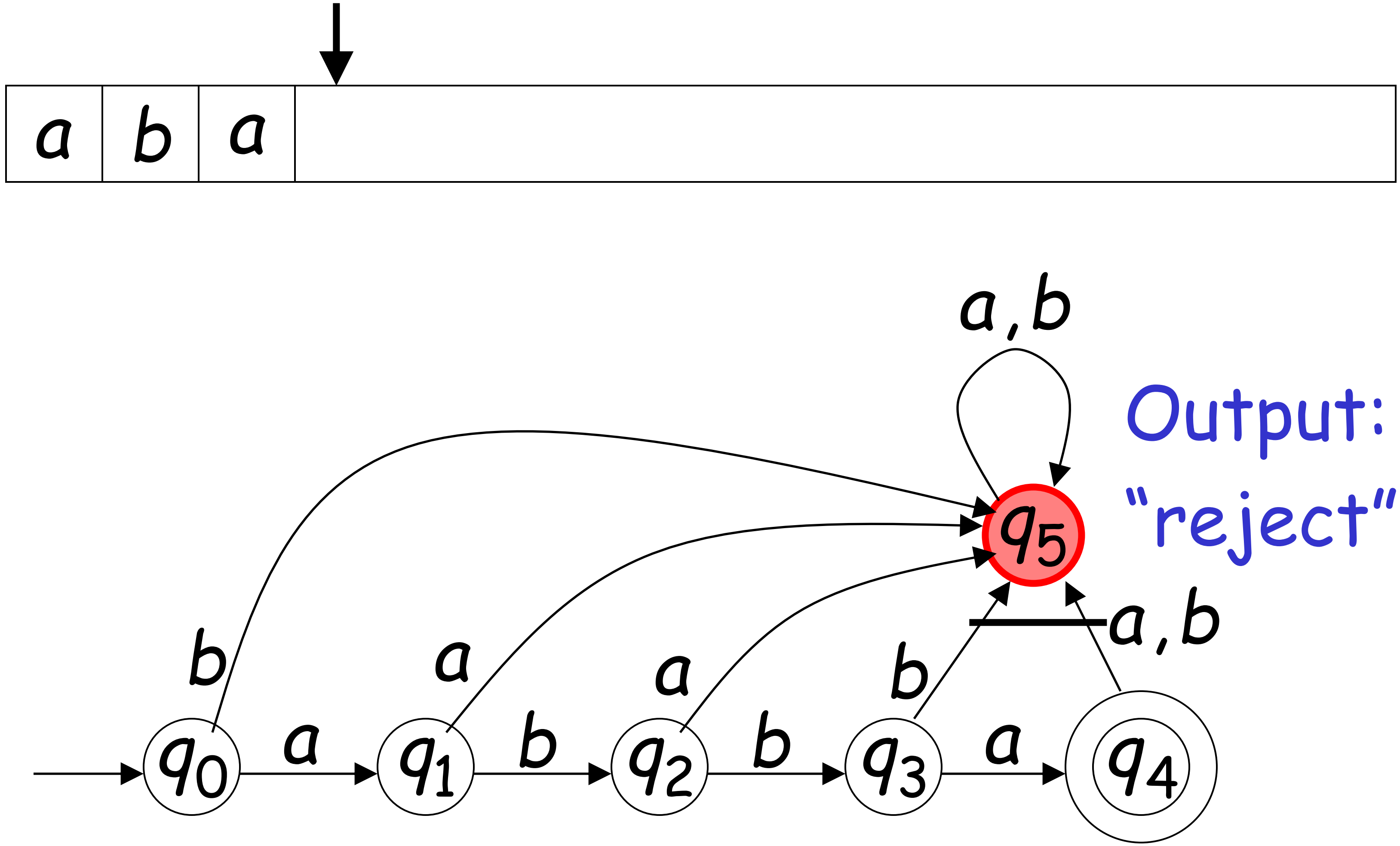

a
b
a
a,b
q5
b
a
a
b
a,b
q0
a
q1
b
q2
b
q3
a
q4


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

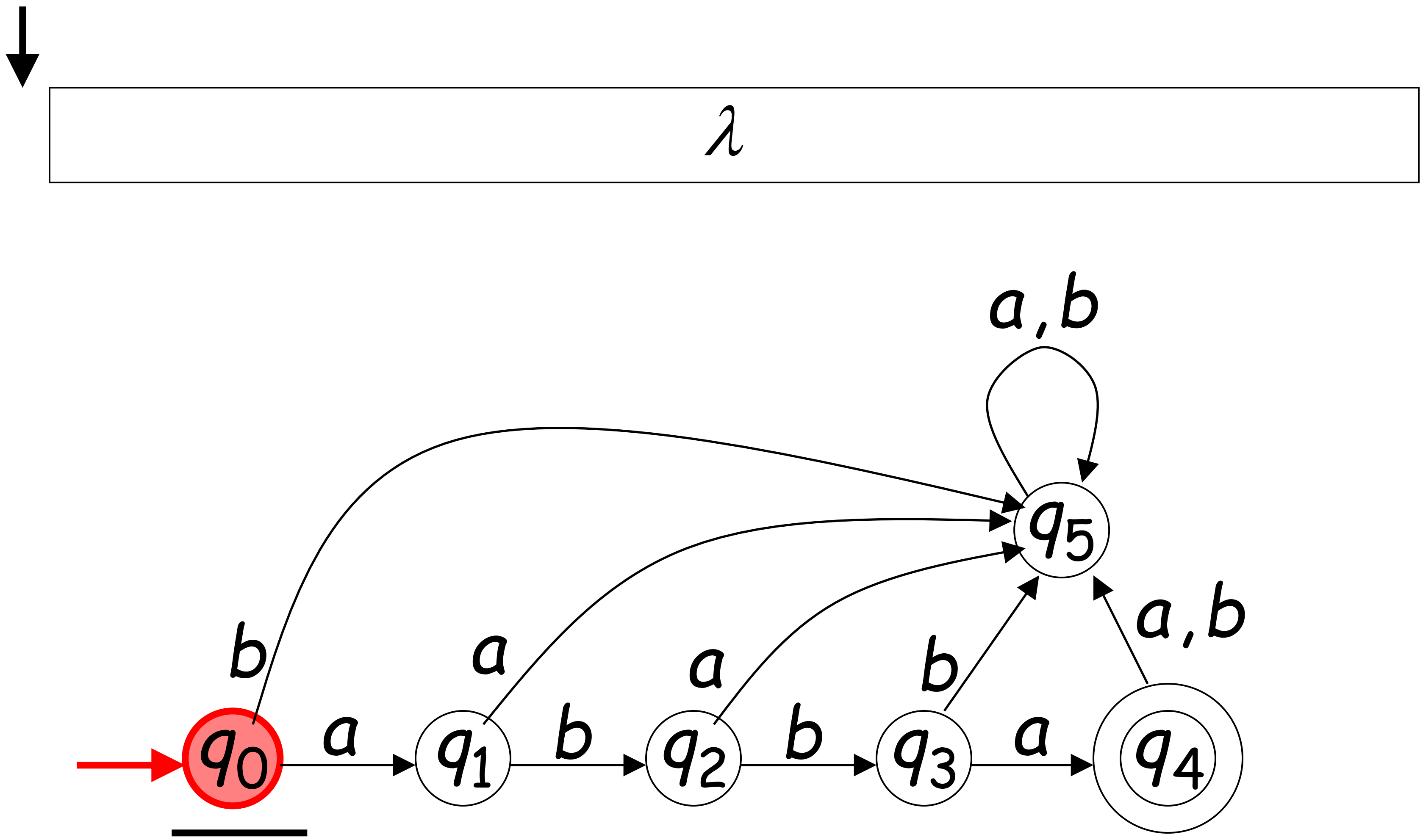


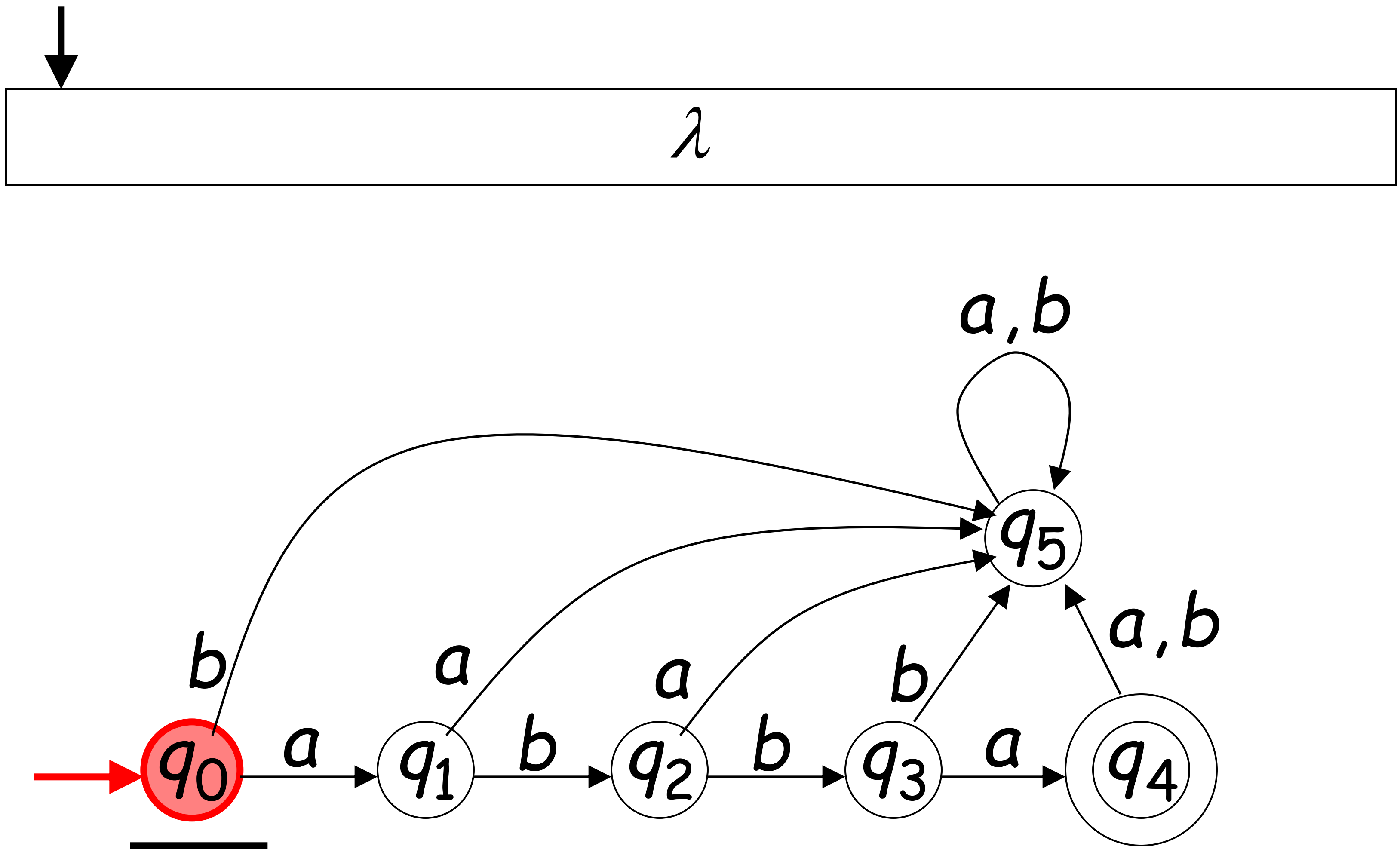
a
b
a
a,b
b
a
a
b
a,b
q_0
a
q_1
b
q_2
b
q_3
a
q_4
q_5

Input finished



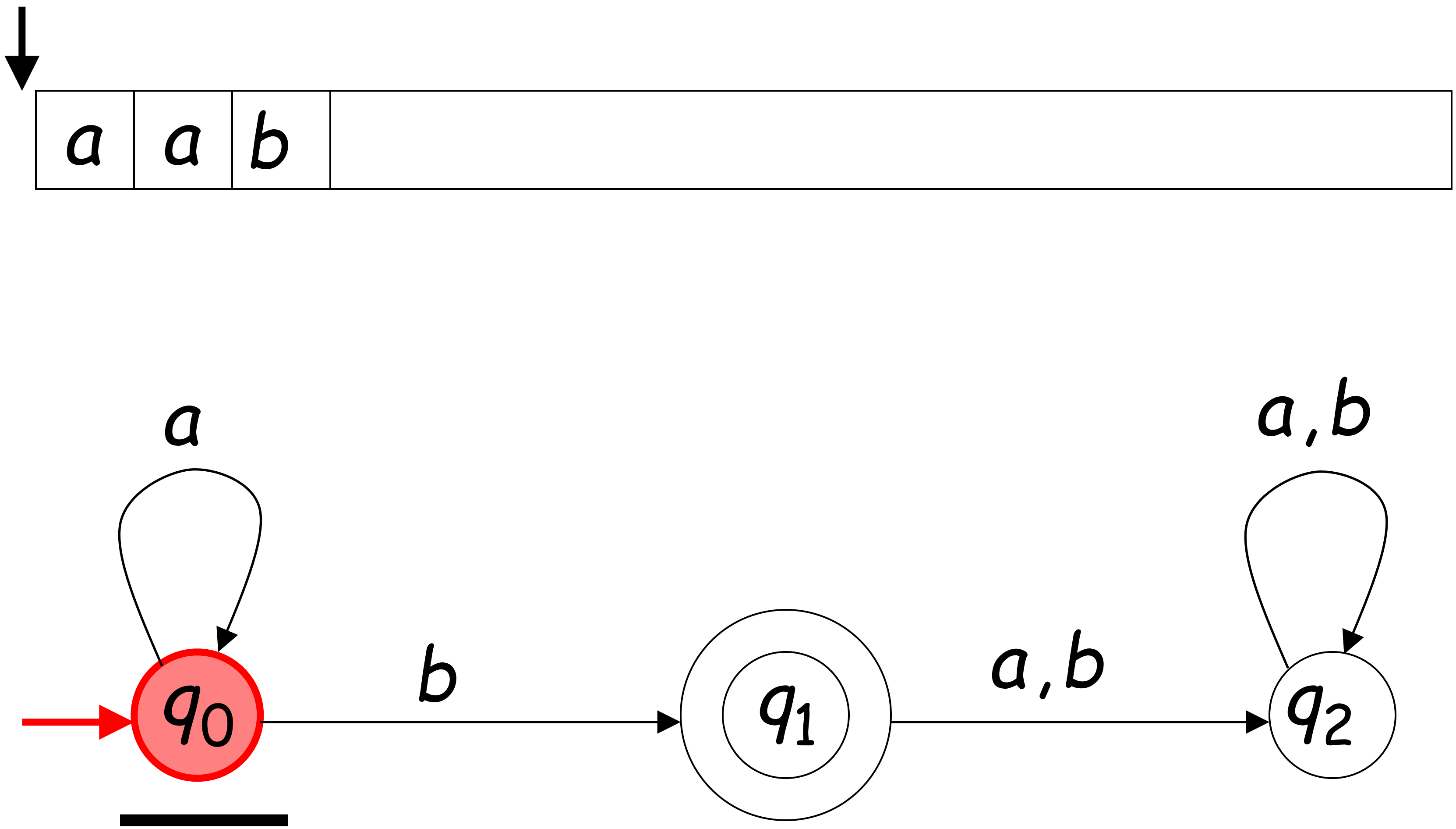
Another Rejection



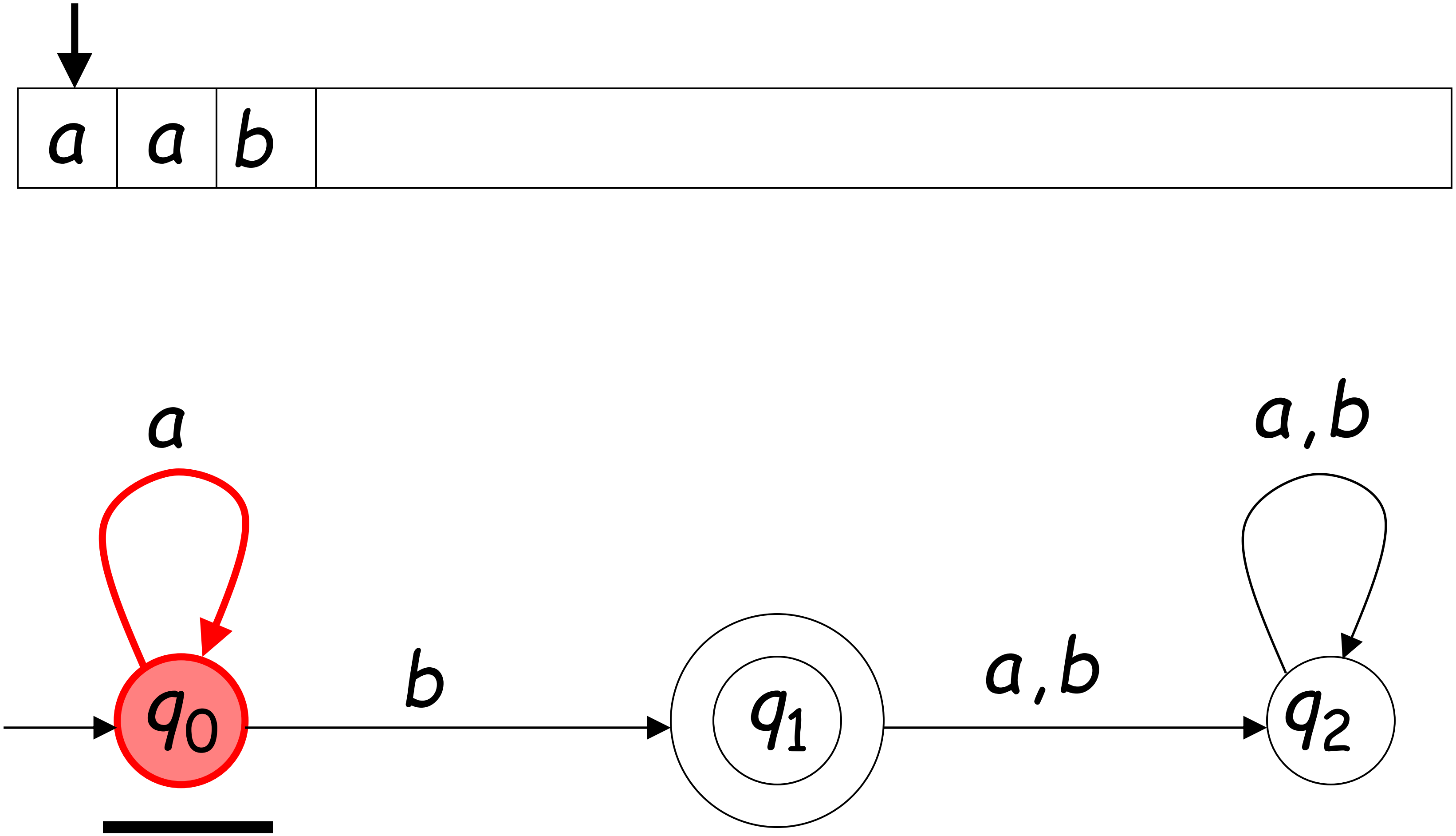


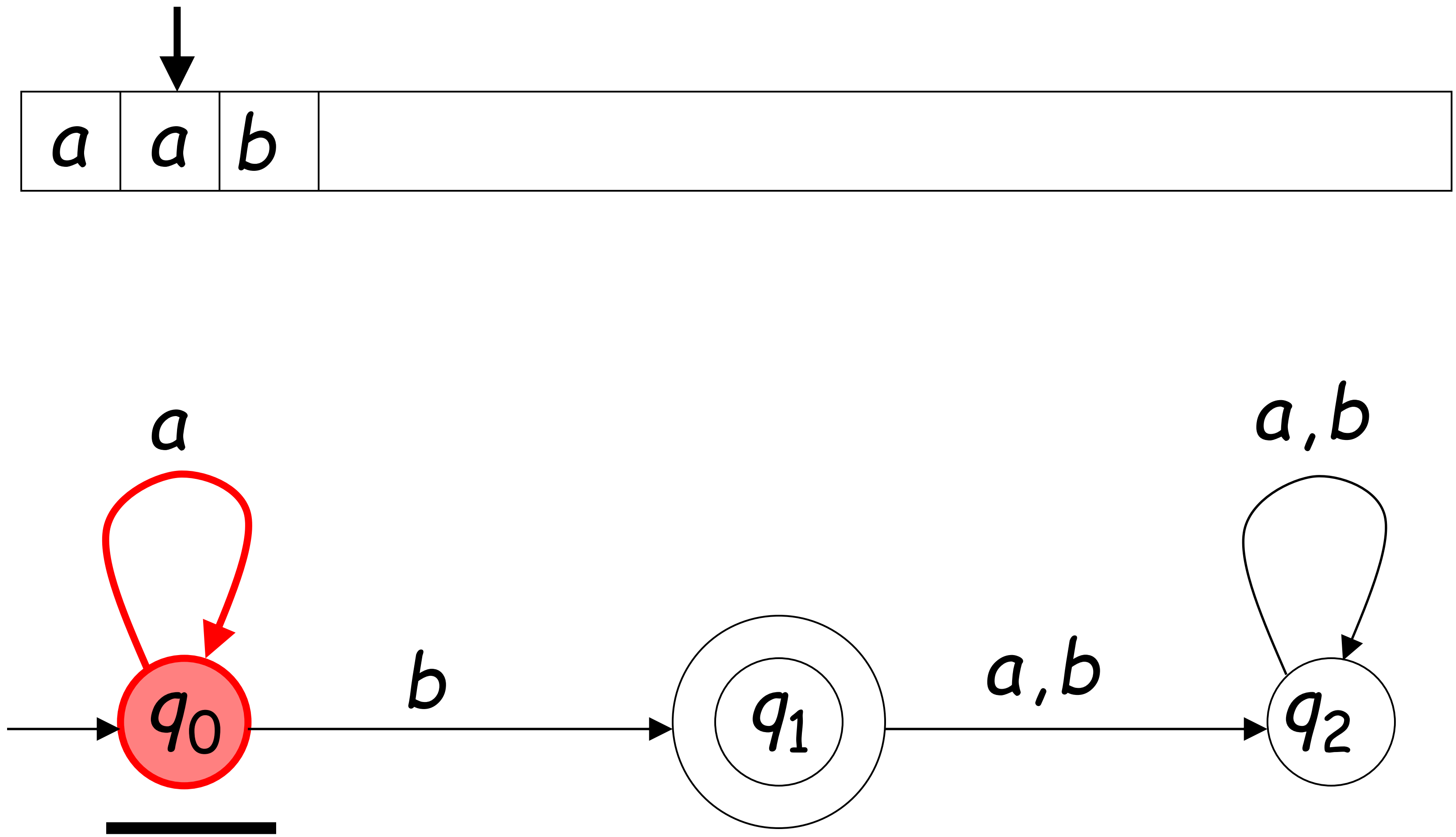
Output:
"reject"

Another Example

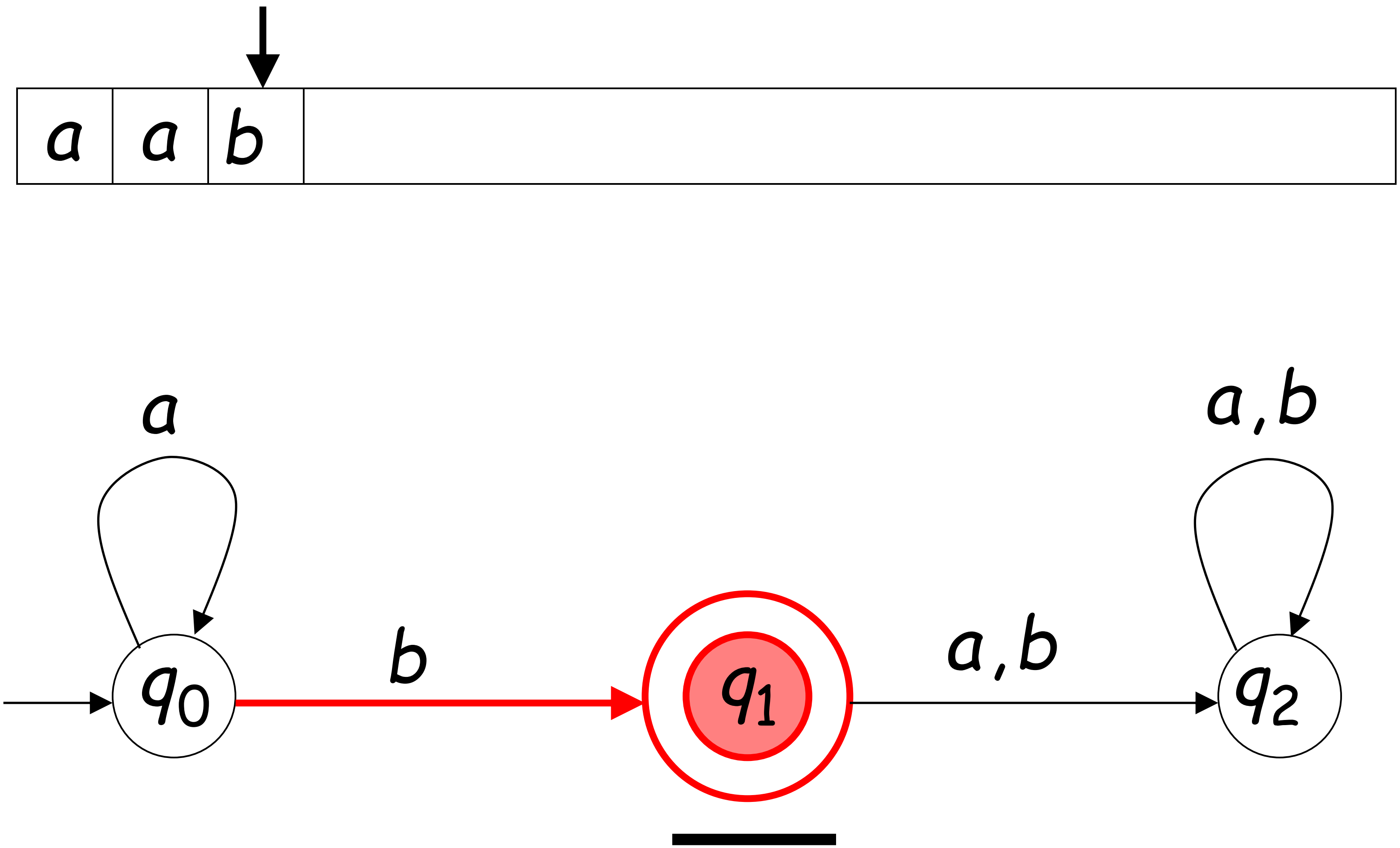


$L = a^* b$


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

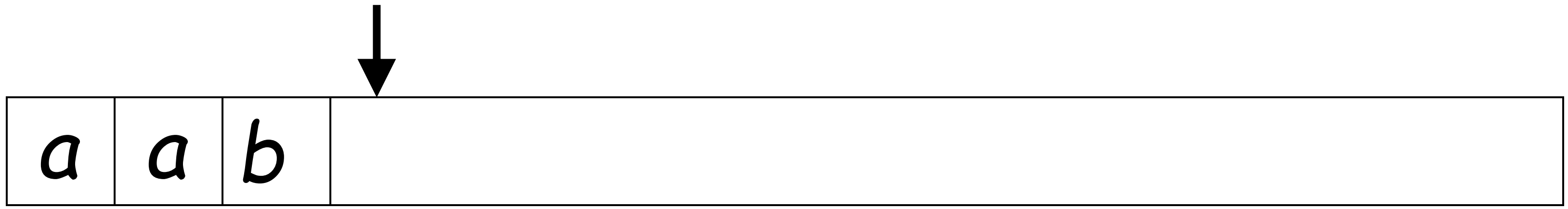


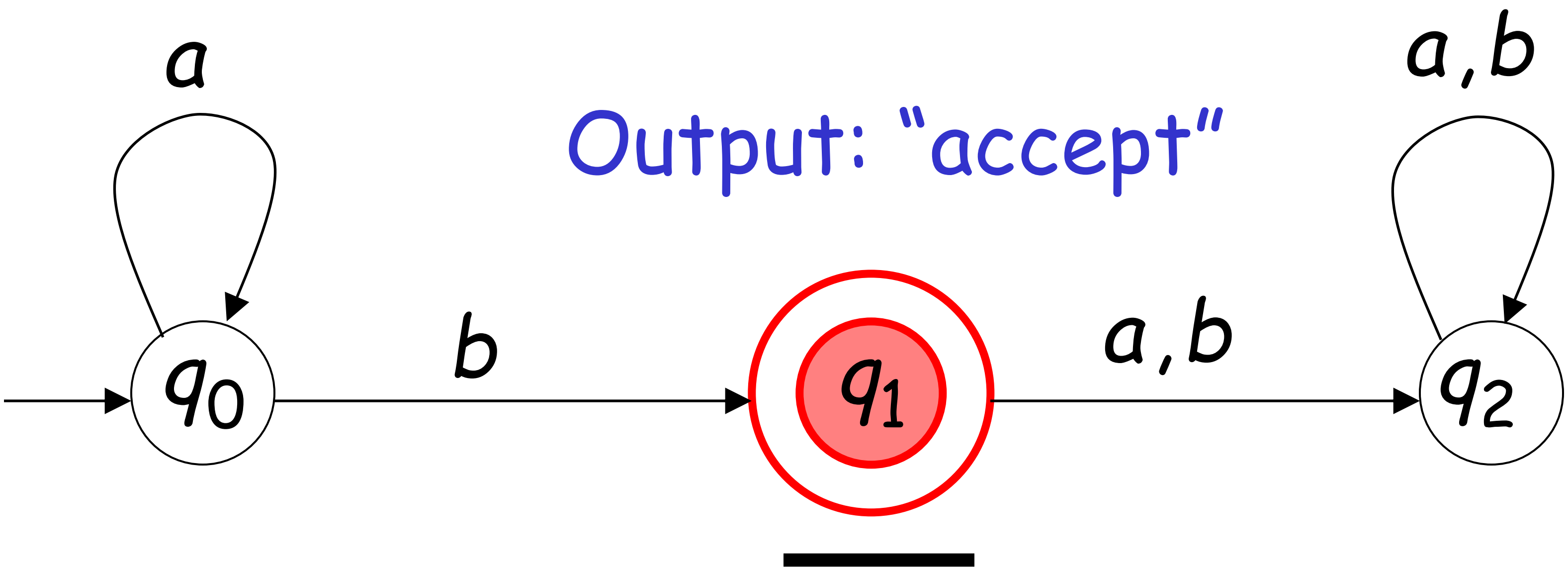
a
a
b
a
a,b
b
a,b
q0
q1
q2



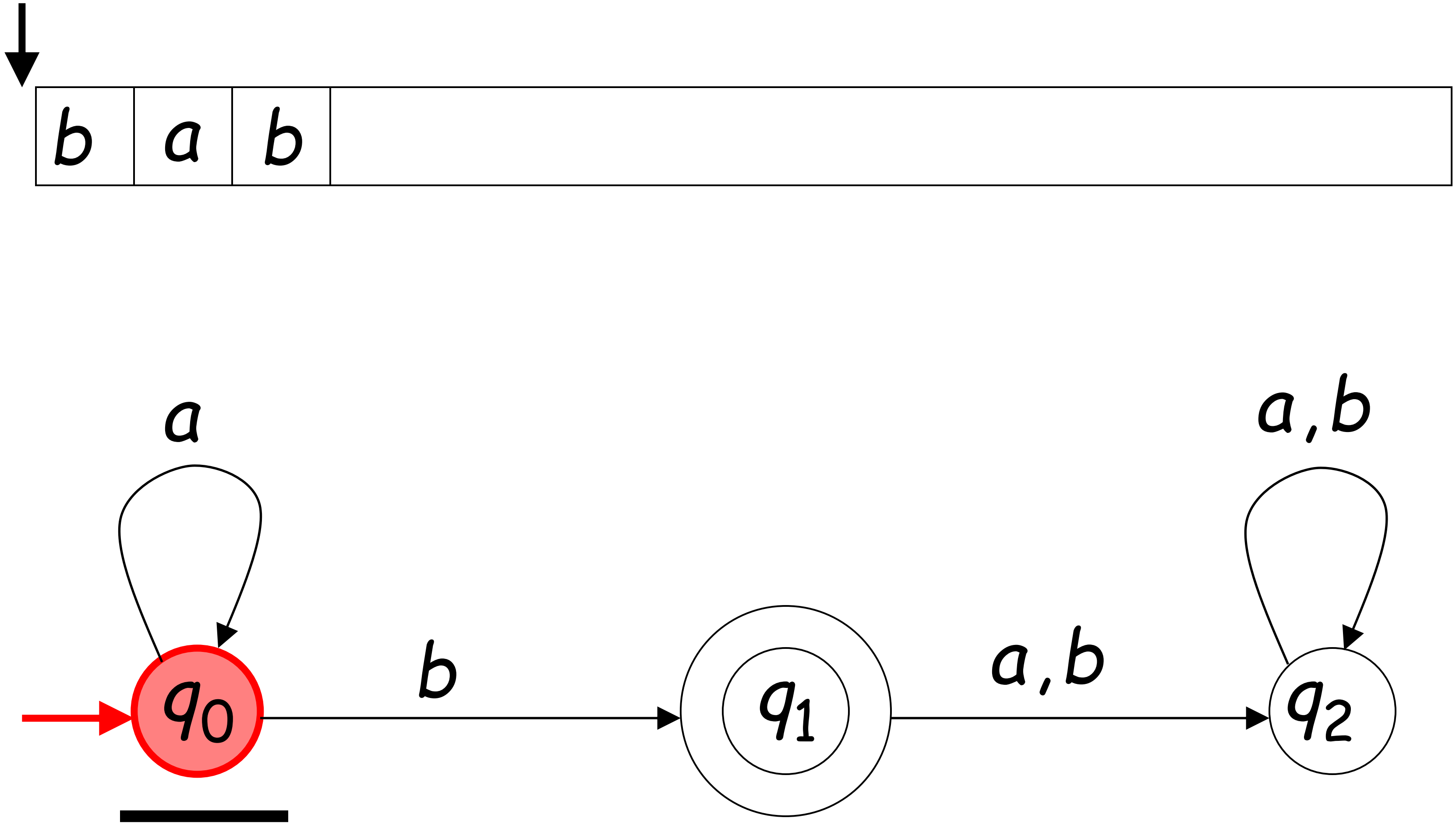
a
a
b
a
a,b
b
a,b
q_0
q_1
q_2

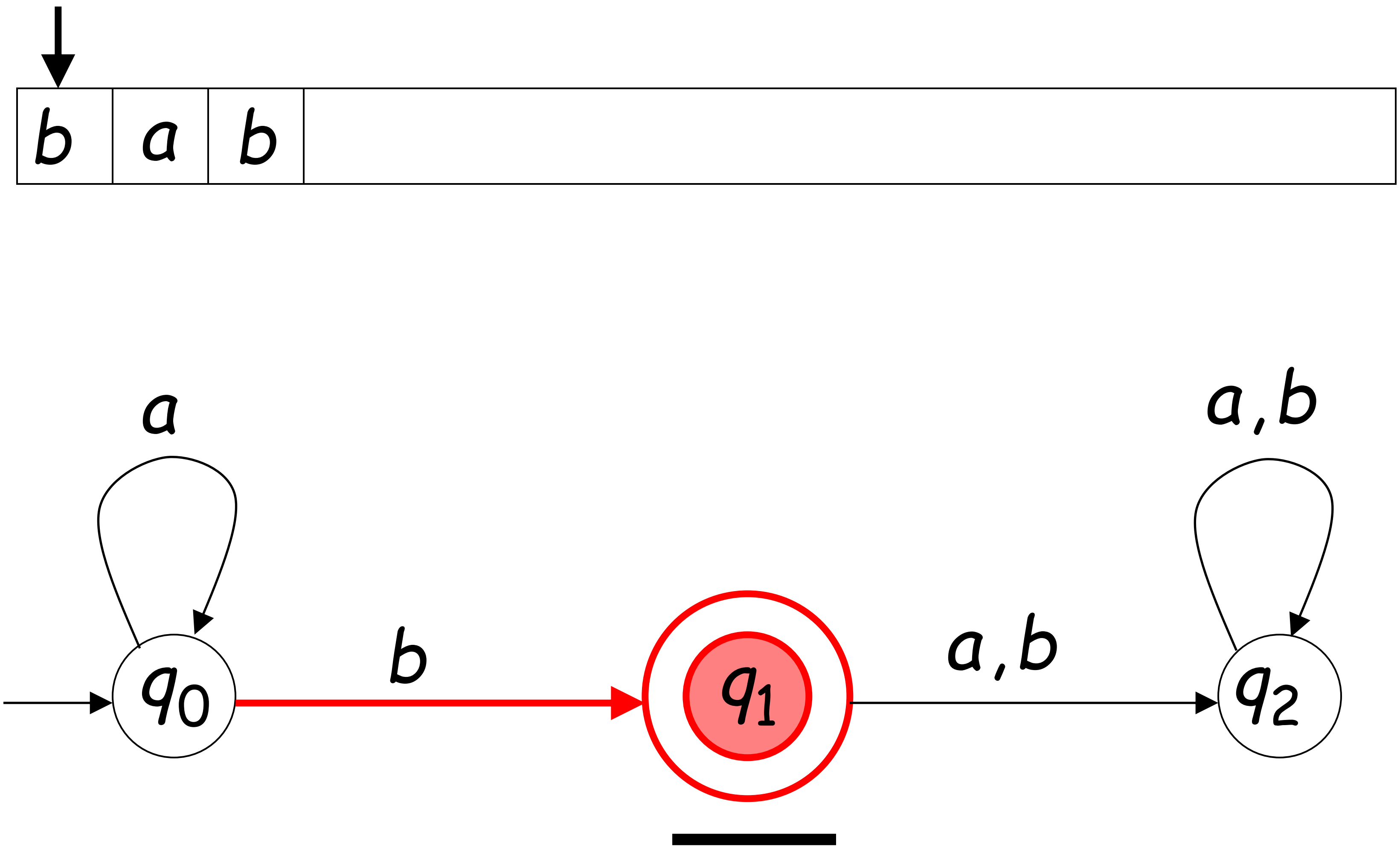
Input finished

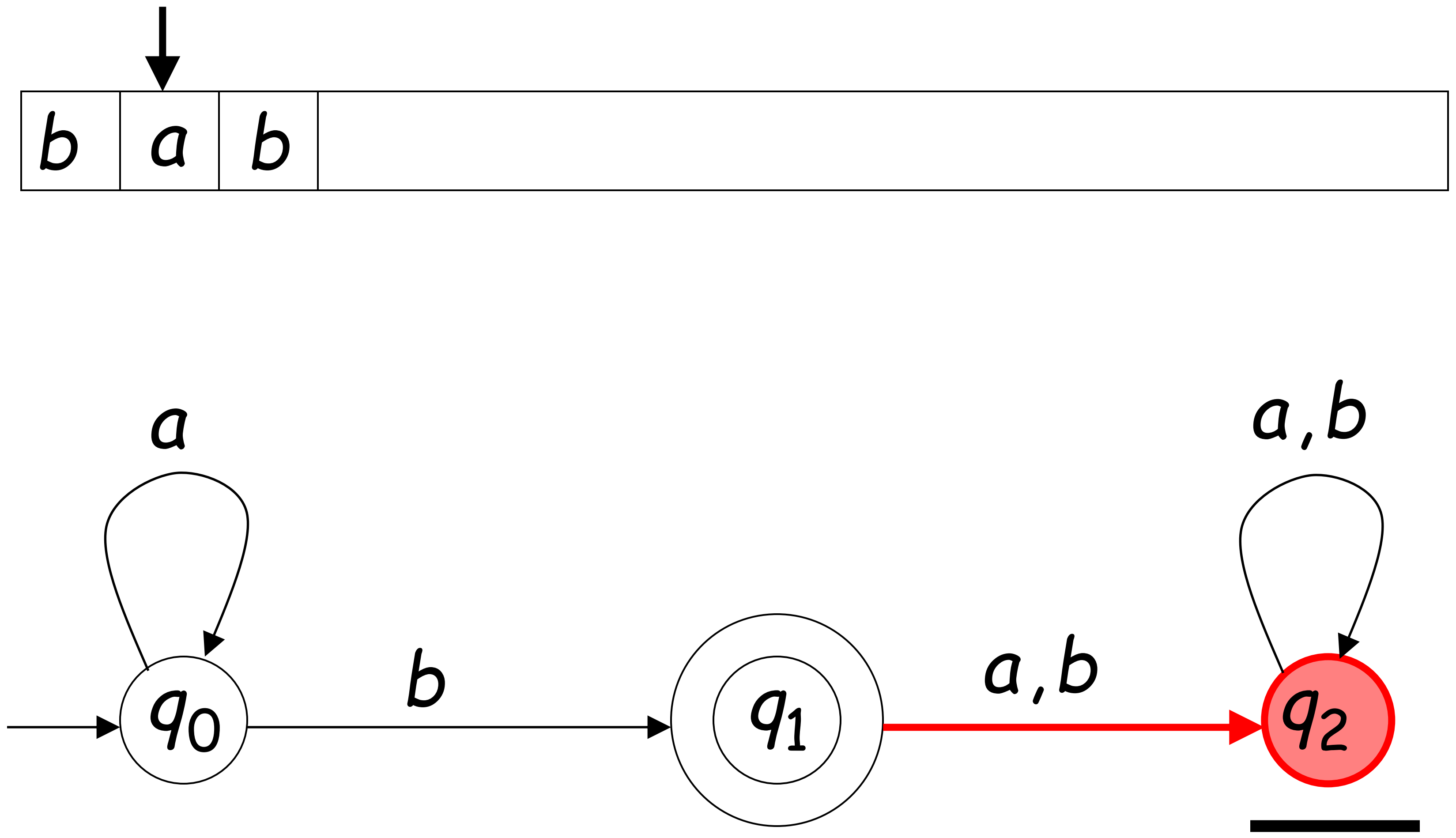




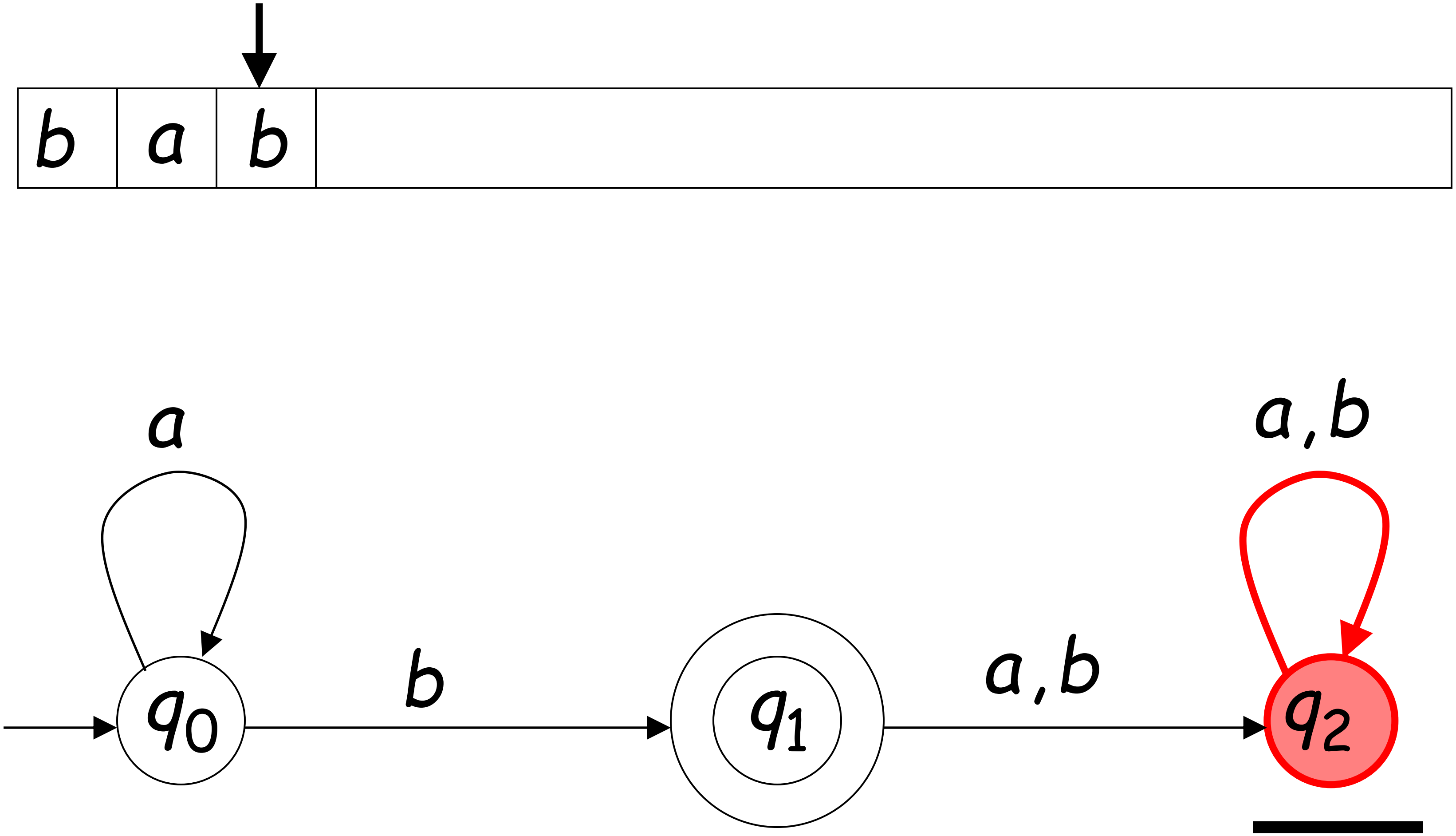
Rejection




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

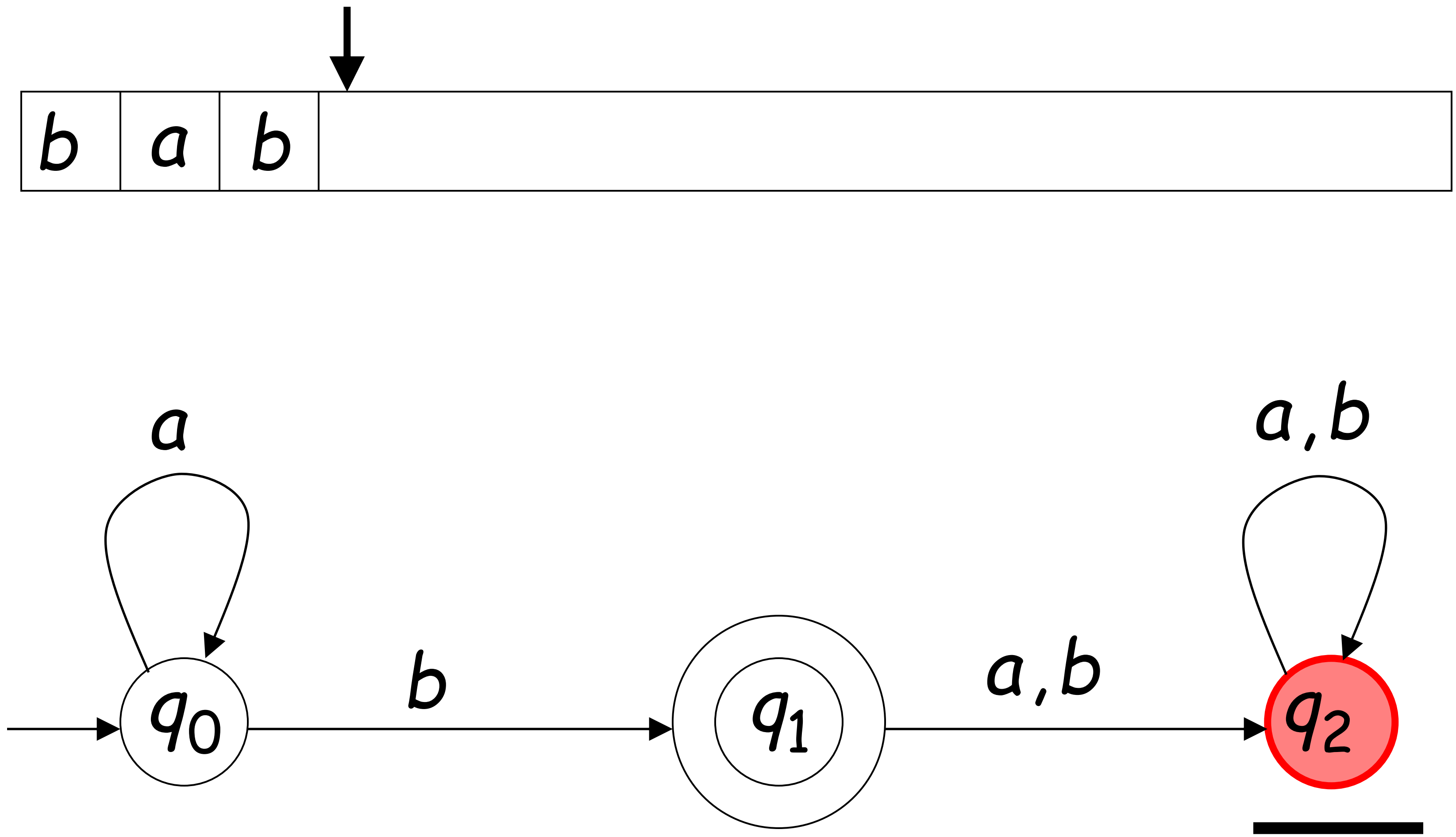


b
a
b
a
a,b
b
a,b
q_0
q_1
q_2



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

Input finished



Output: "reject"

Formalities

Deterministic Finite Accepter (DFA)

$$M = (Q, \Sigma, \delta, q_0, F)$$ 5 Tuple

Q : set of states

Σ : input alphabet

δ : transition function

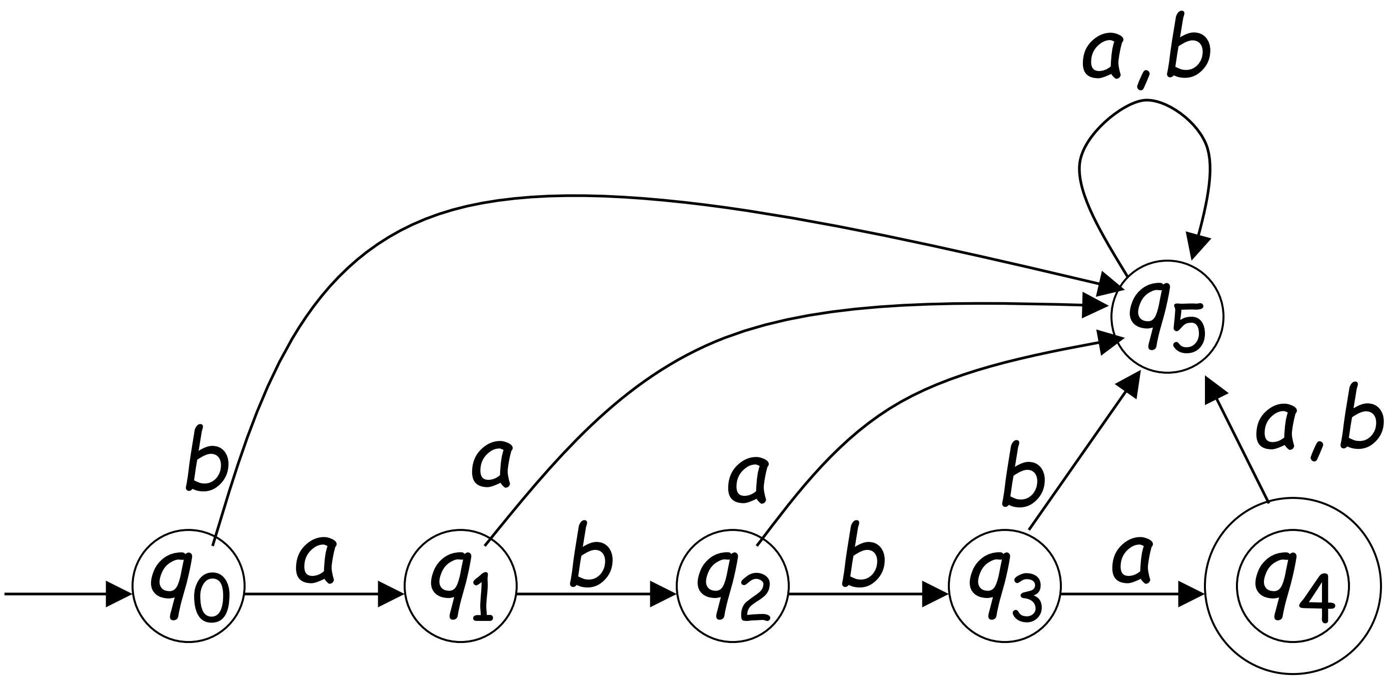
q_0 : initial state

F : set of final states

$F \subseteq Q$

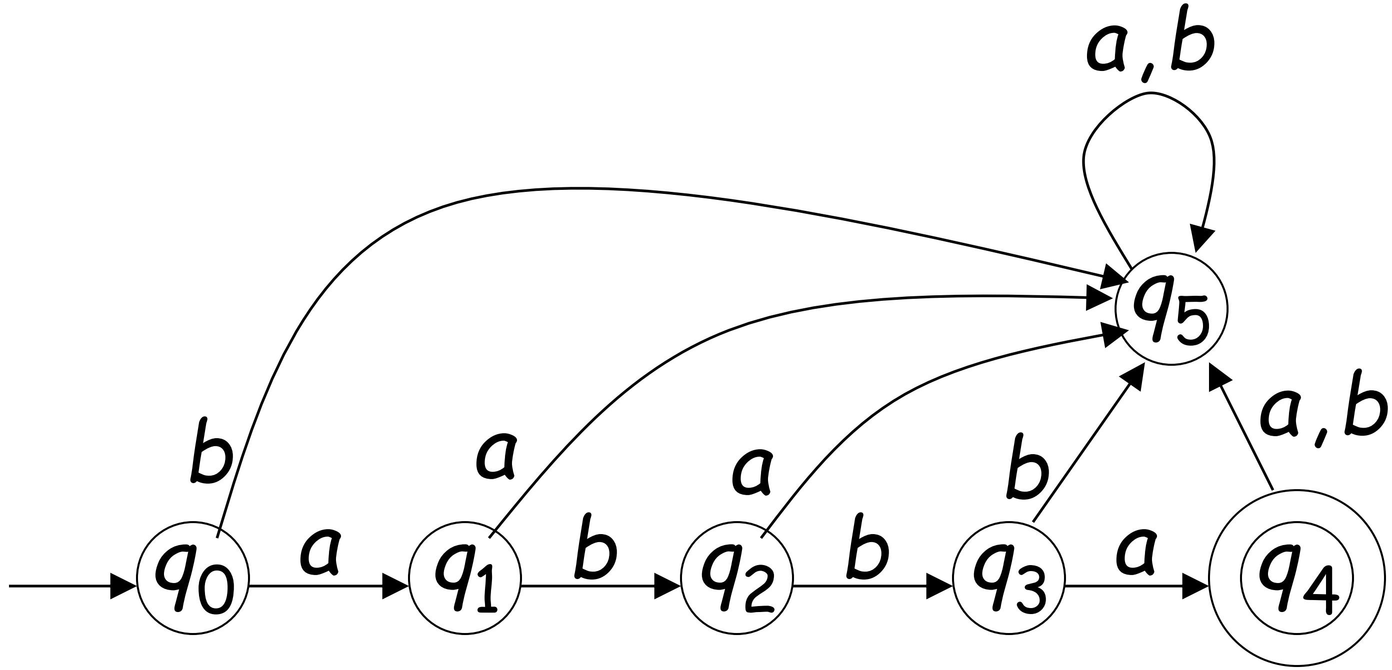
Input Alphabet Σ

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

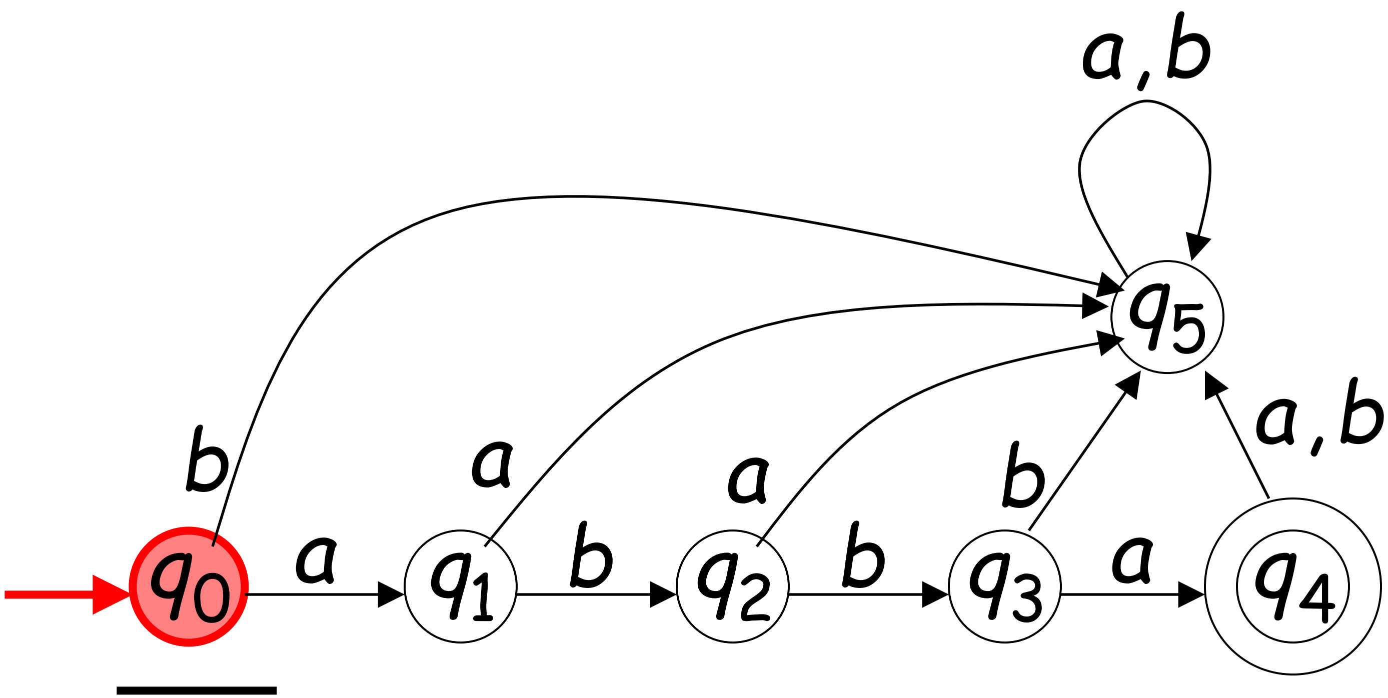


Set of States Q

$$Q = \{q_0, q_1, q_2, q_3, q_4, q_5\}$$

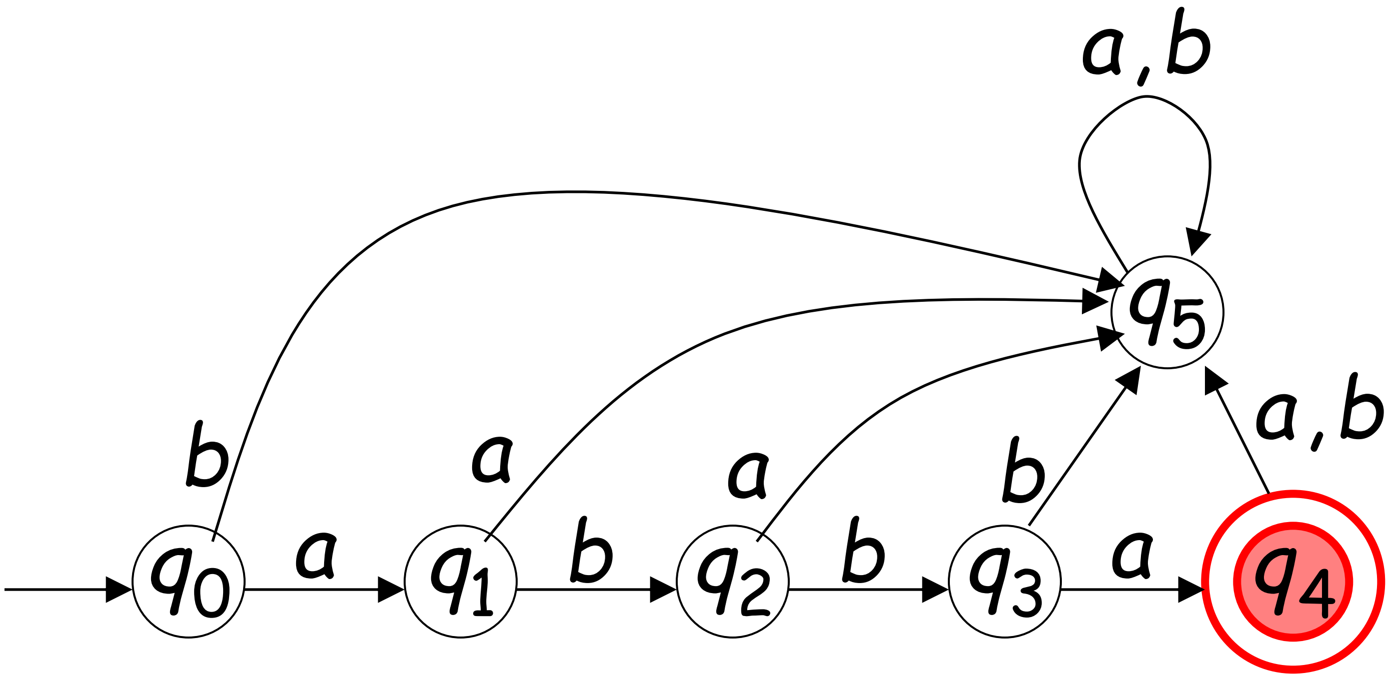


Initial State q_0



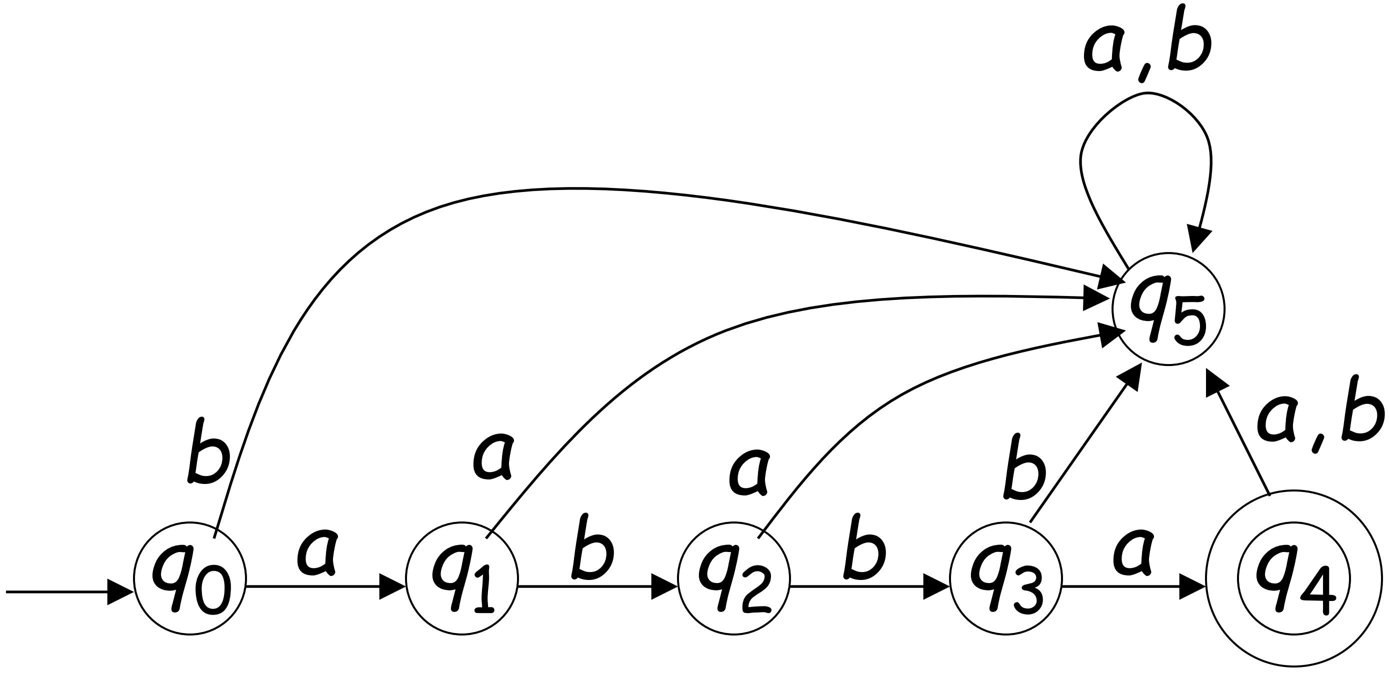
Set of Final States F

$$F = \{q_4\}$$

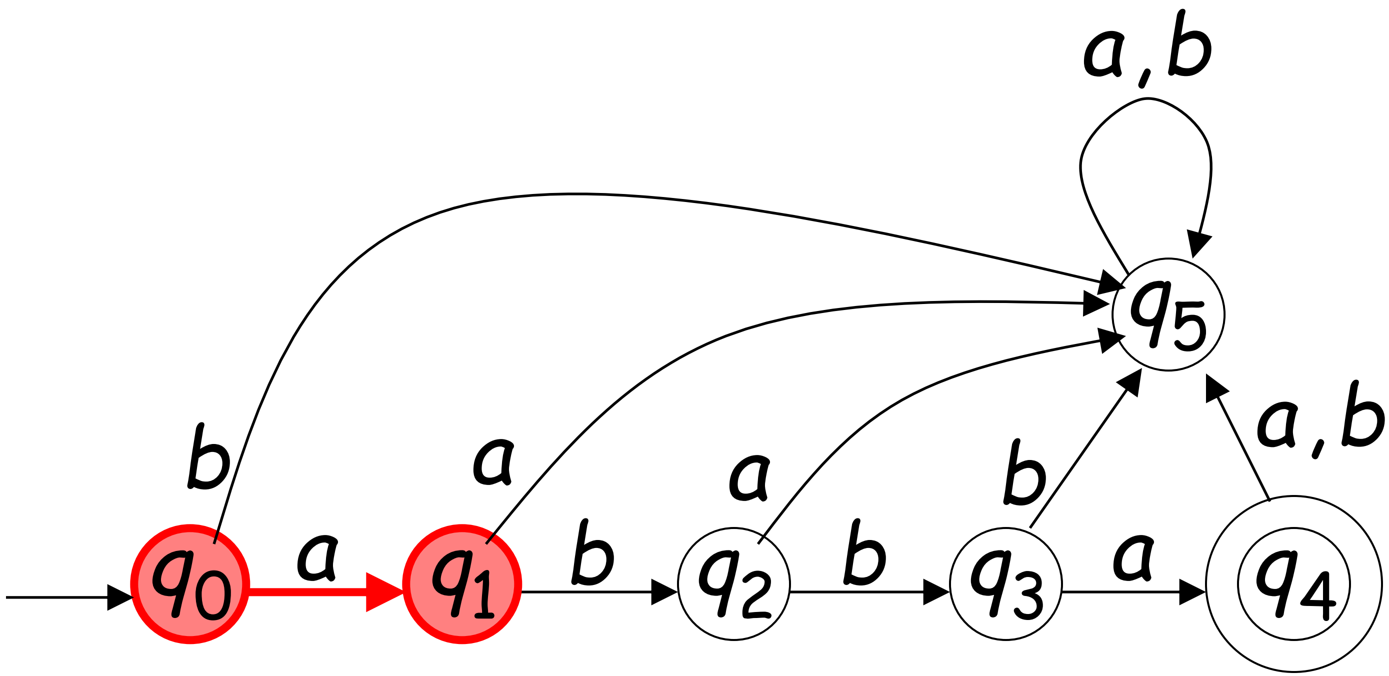


Transition Function δ

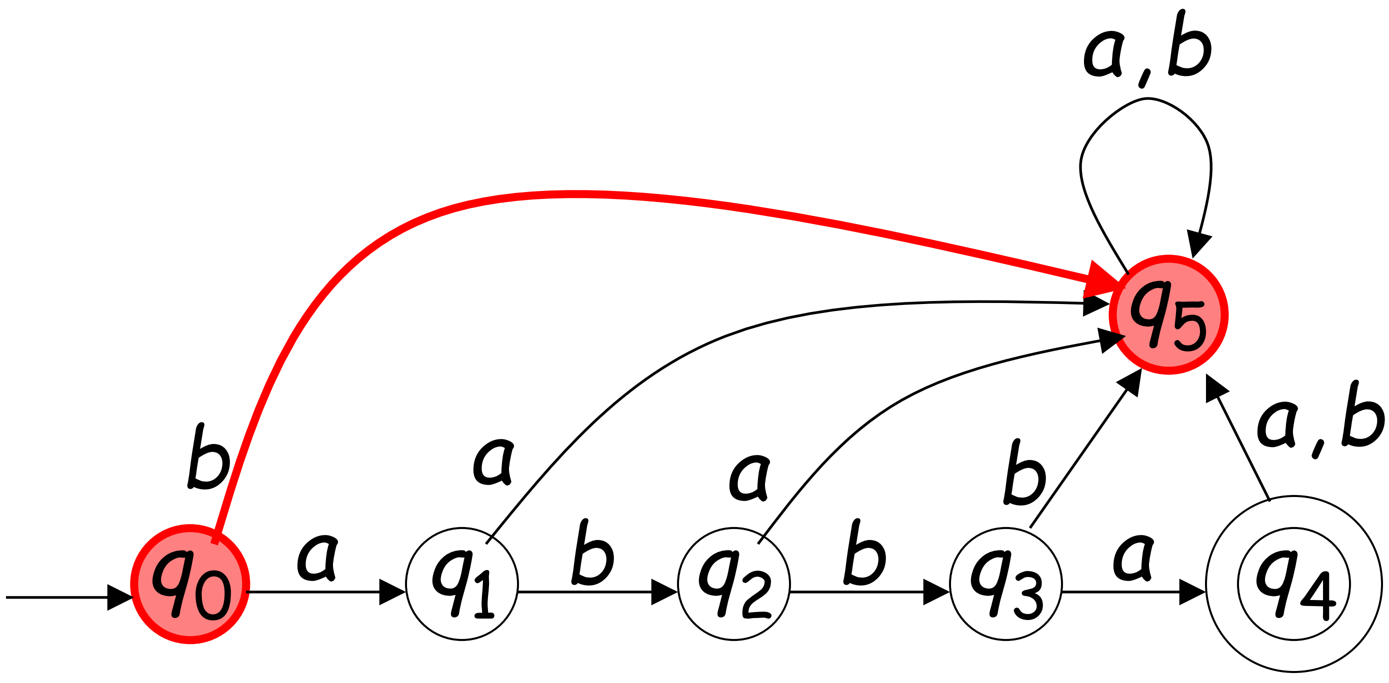
$$\delta : Q \times \Sigma \rightarrow Q$$



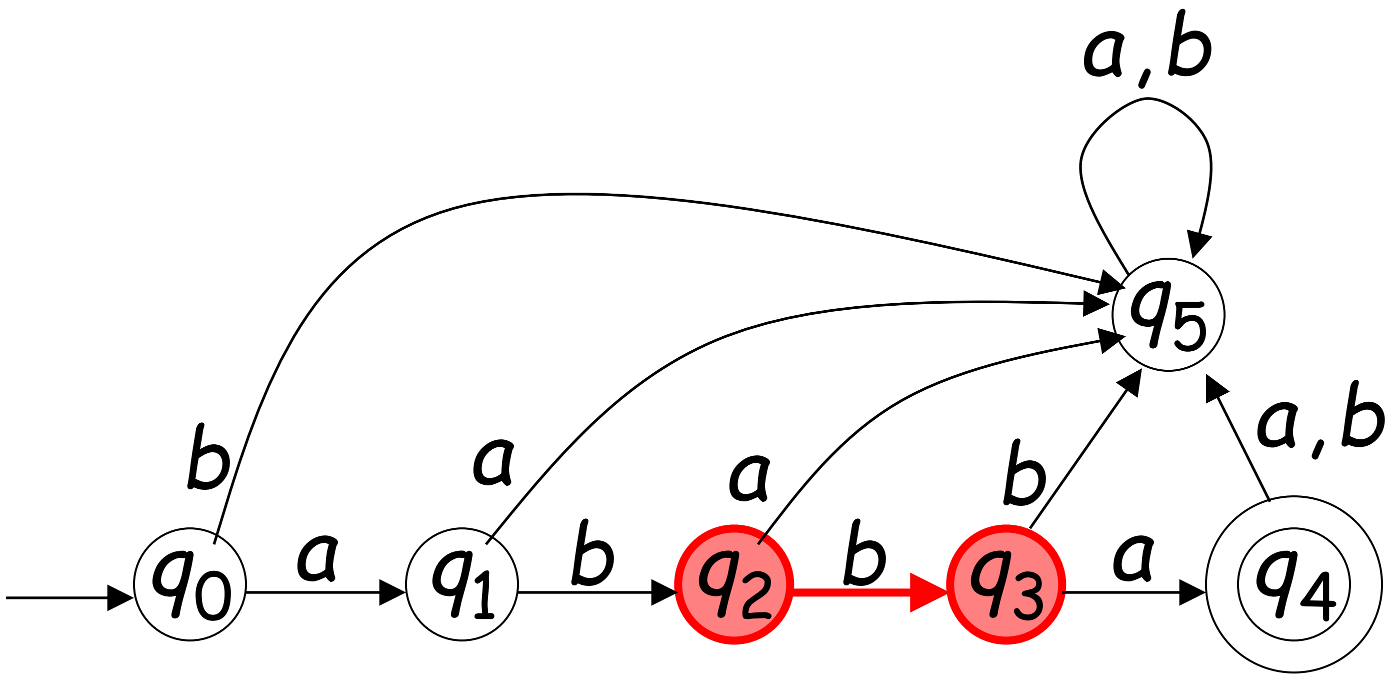
$$\delta(q_0, a) = q_1$$



$$\delta(q_0, b) = q_5$$

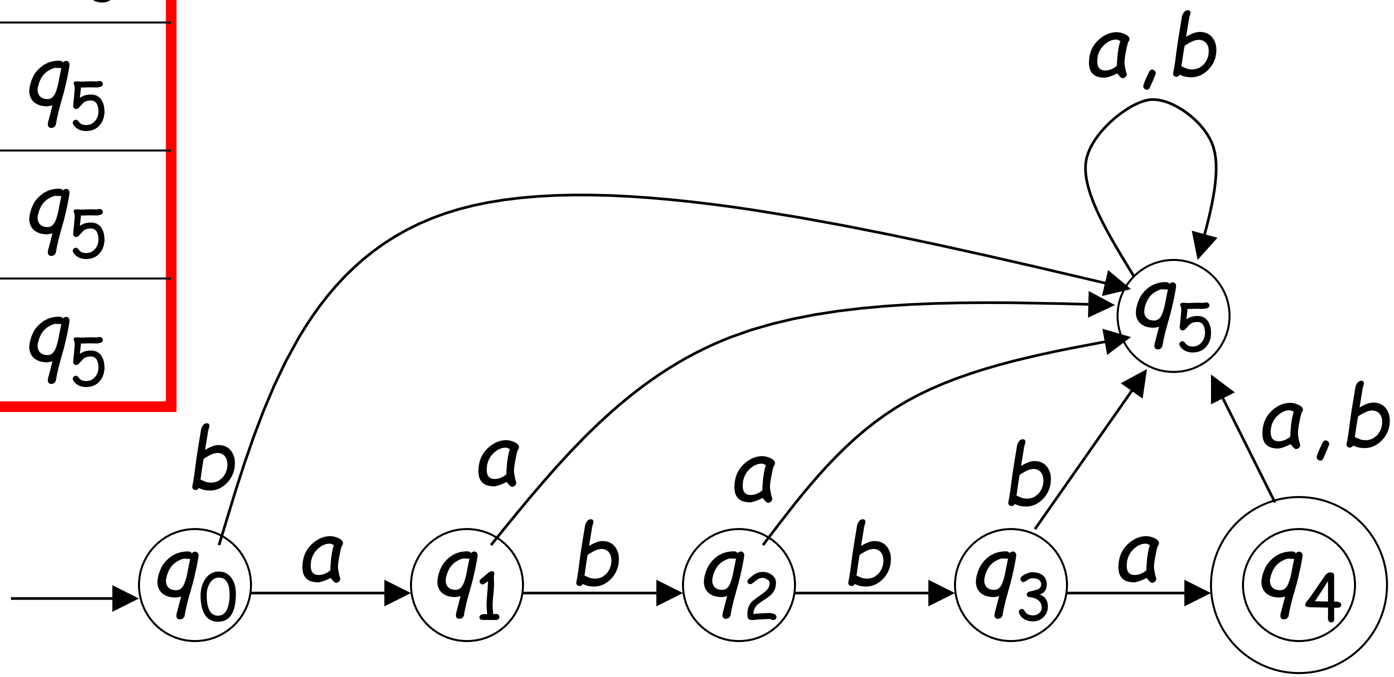


$$\delta(q_2, b) = q_3$$



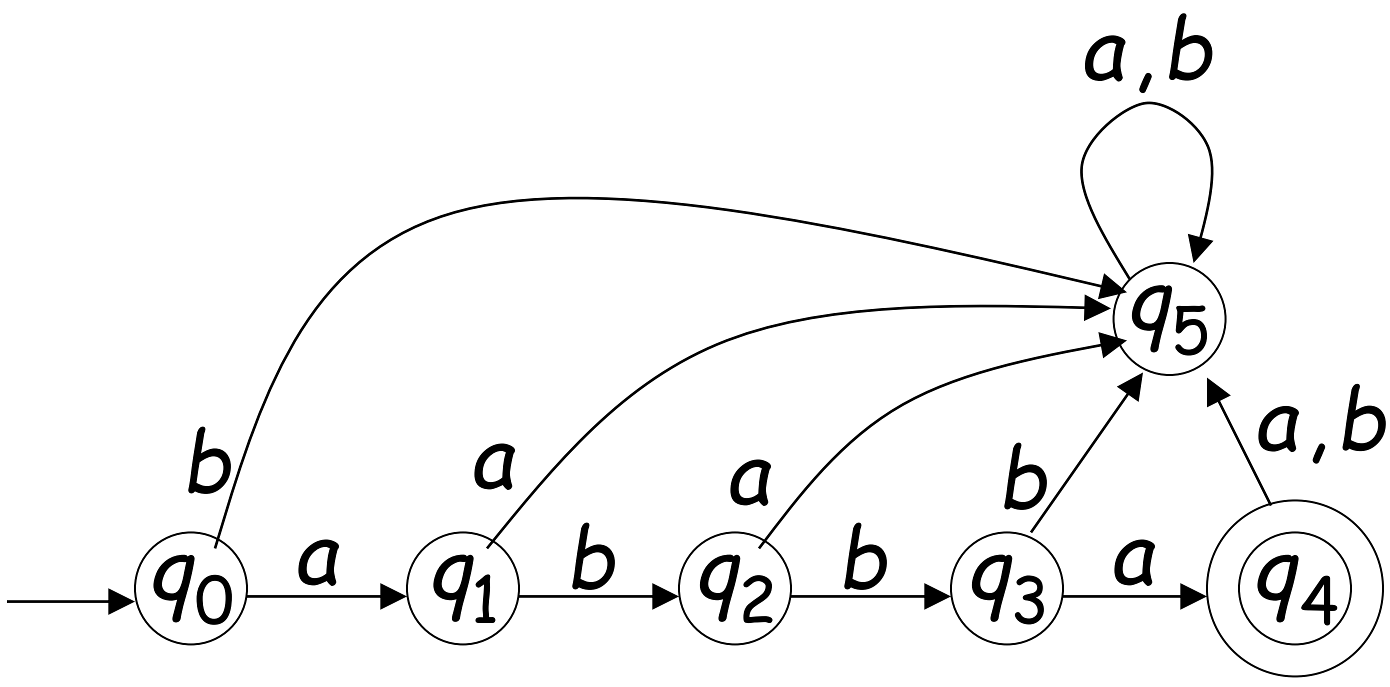
Transition Function δ

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

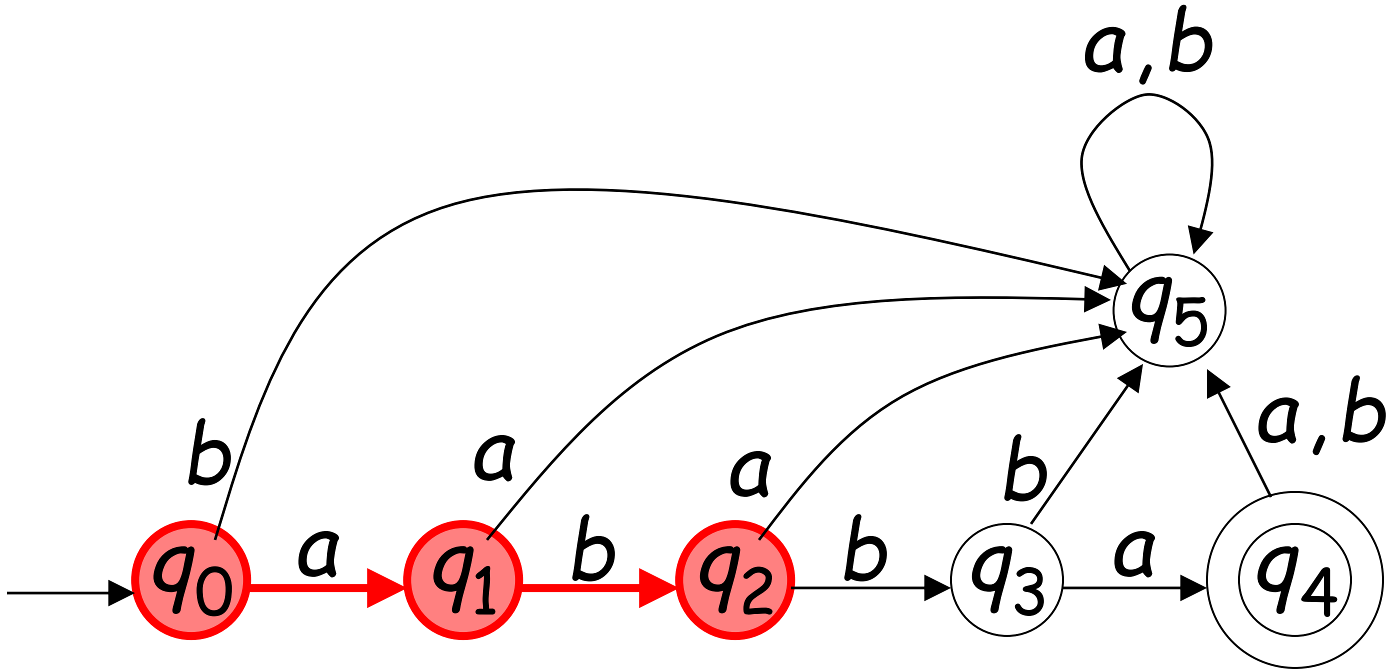


Extended Transition Function $\delta *$

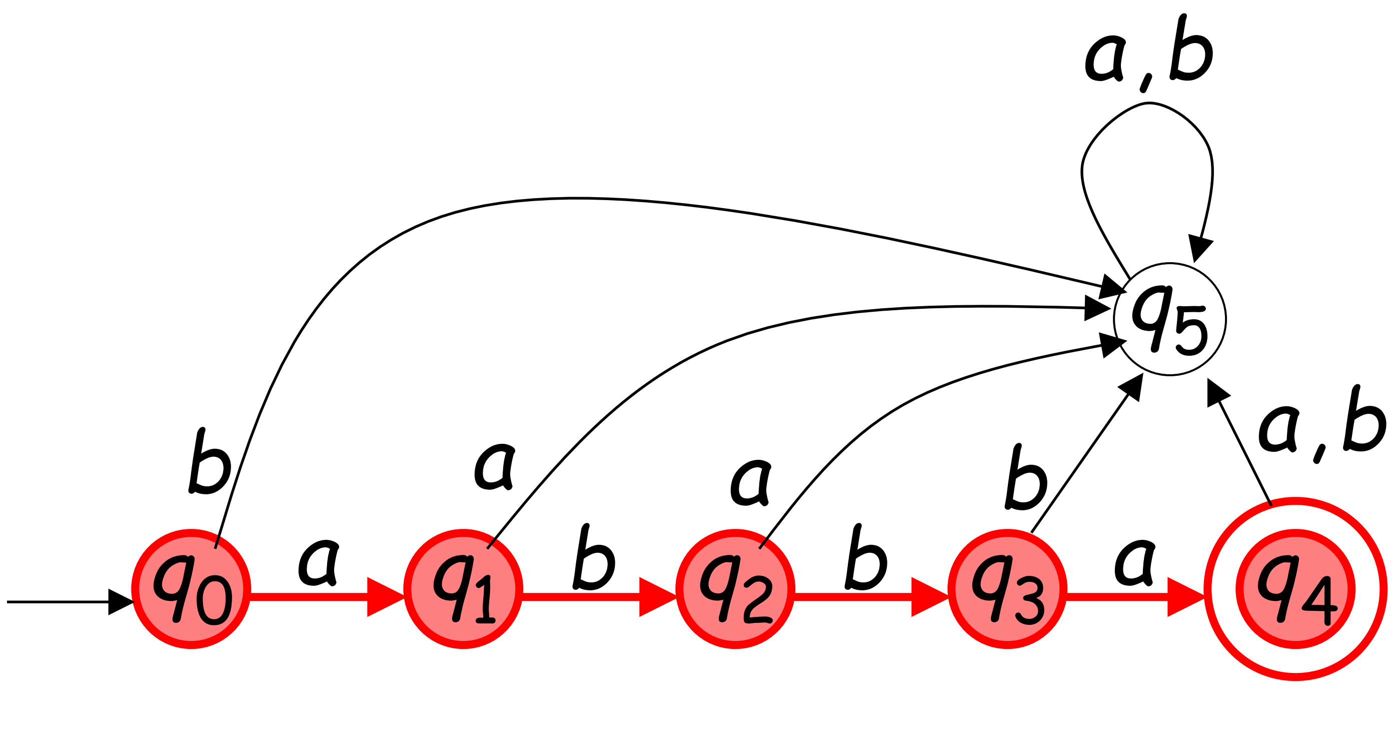
$$\delta * : Q \times \Sigma * \to Q$$



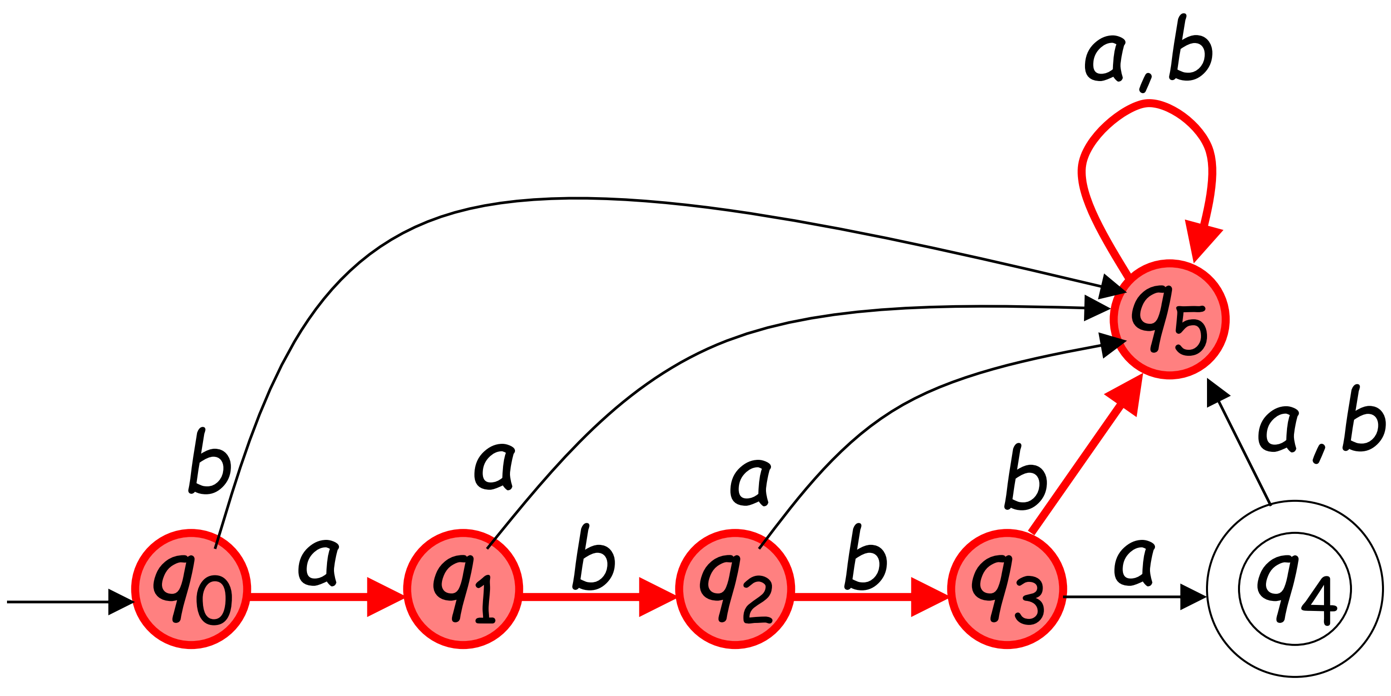
$$\delta^*(q_0, ab) = q_2$$



$$\delta^*(q_0, abba) = q_4$$

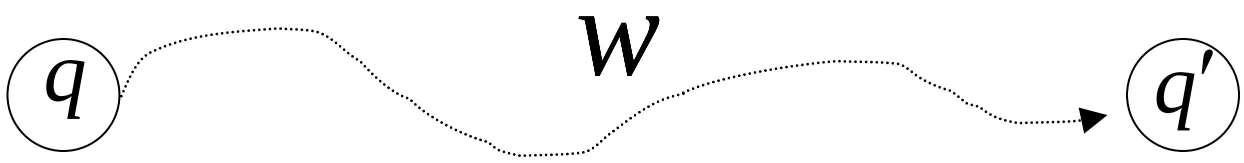


$$\delta^*(q_0, abbbaa) = q_5$$

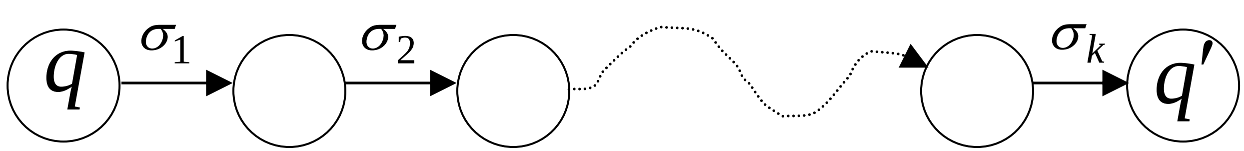


Observation: There is a walk from q to q' with label w

$$\delta^*(q,w) = q'$$

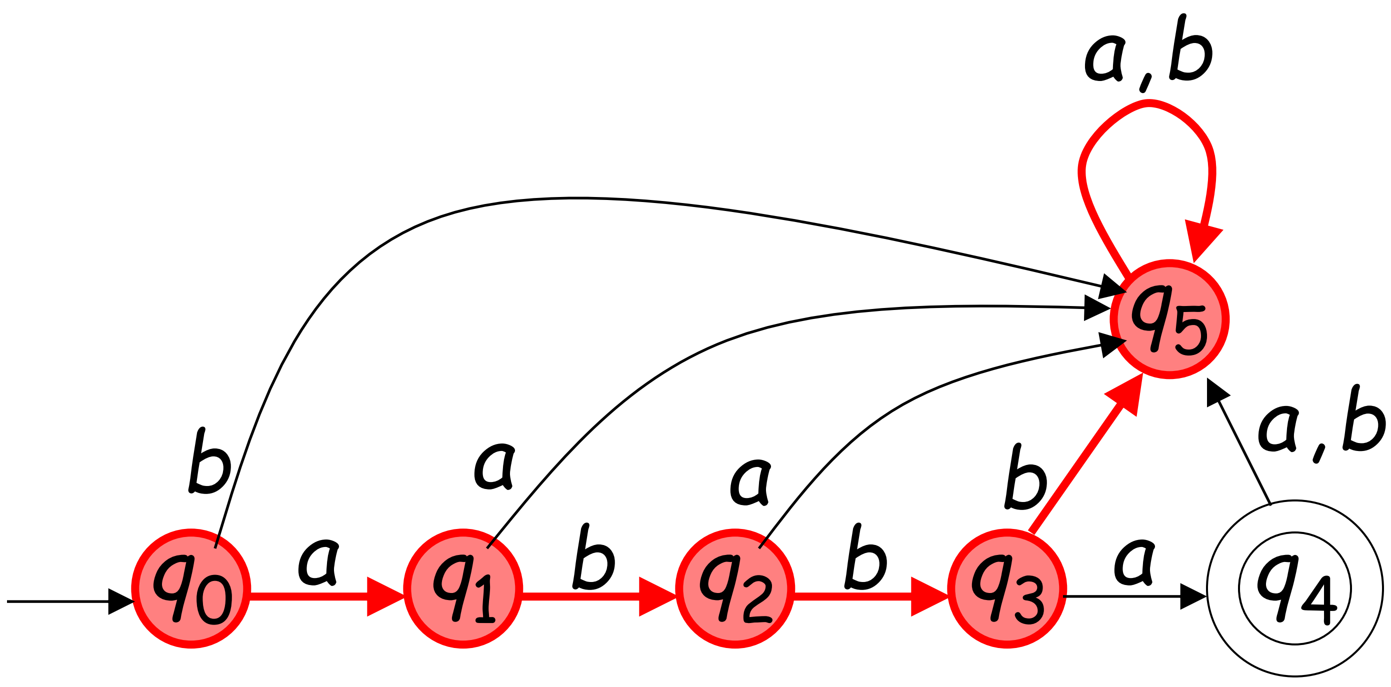


$$w = \sigma_1 \sigma_2 \cdots \sigma_k$$



Example: There is a walk from q_0 to q_5 with label $abbbaa$

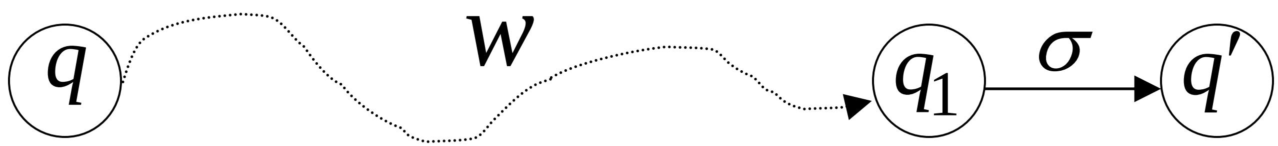
$$\delta^*(q_0, abbbaa) = q_5$$



Recursive Definition

$$\delta^*(q,\lambda)=q$$

$$\delta^*(q,w\sigma)=\delta(\delta^*(q,w),\sigma)$$



$$\left.\begin{array}{l}\delta^*(q,w\sigma)=q' \\ \\ \delta(q_1,\sigma)=q'\end{array}\right\} \Rightarrow \delta^*(q,w\sigma)=\delta(q_1,\sigma)$$

$$\left.\begin{array}{l}\delta^*(q,w\sigma)=\delta(q_1,\sigma) \\ \\ \delta^*(q,w)=q_1\end{array}\right\} \Rightarrow \delta^*(q,w\sigma)=\delta(\delta^*(q,w),\sigma)$$

$$\delta^*(q_0, ab) =$$

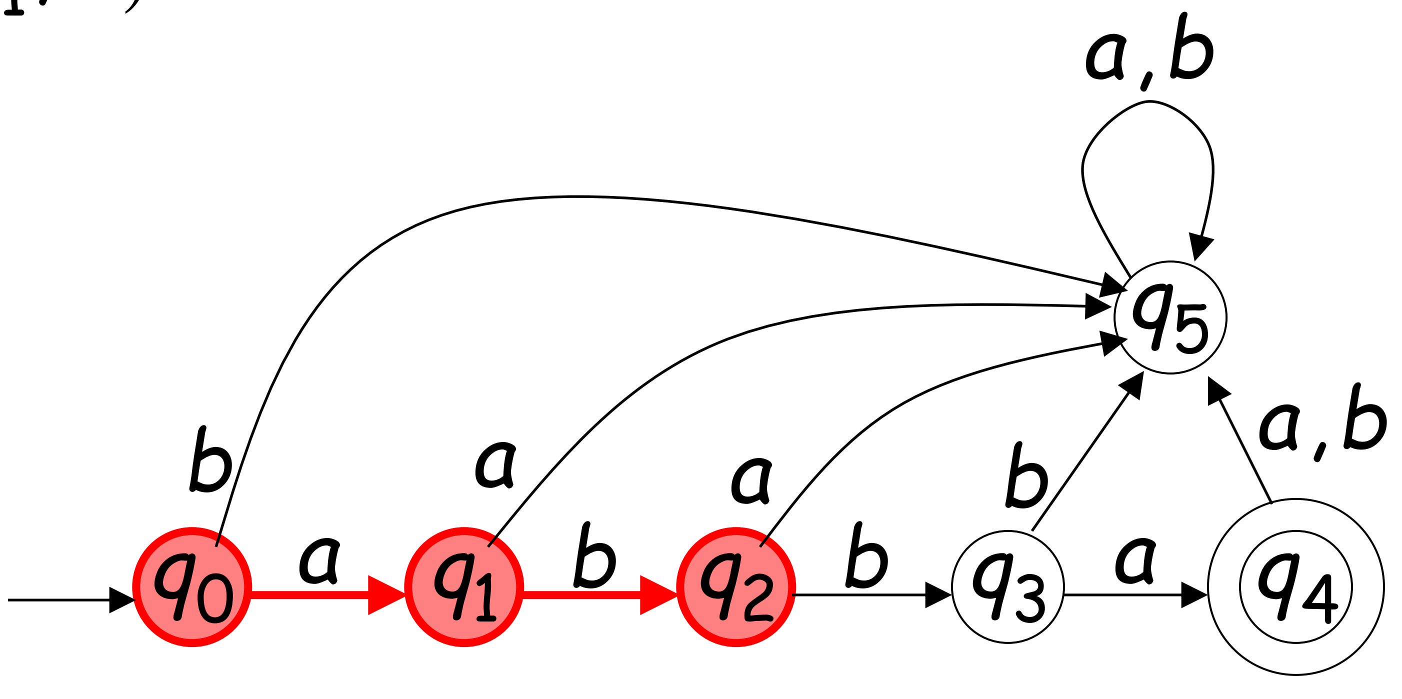
$$\delta(\delta^*(q_0, a), b) =$$

$$\delta(\delta(\delta^*(q_0, \lambda), a), b) =$$

$$\delta(\delta(q_0, a), b) =$$

$$\delta(q_1, b) =$$

$$q_2$$



Languages Accepted by DFAs

Take DFA M

Definition:

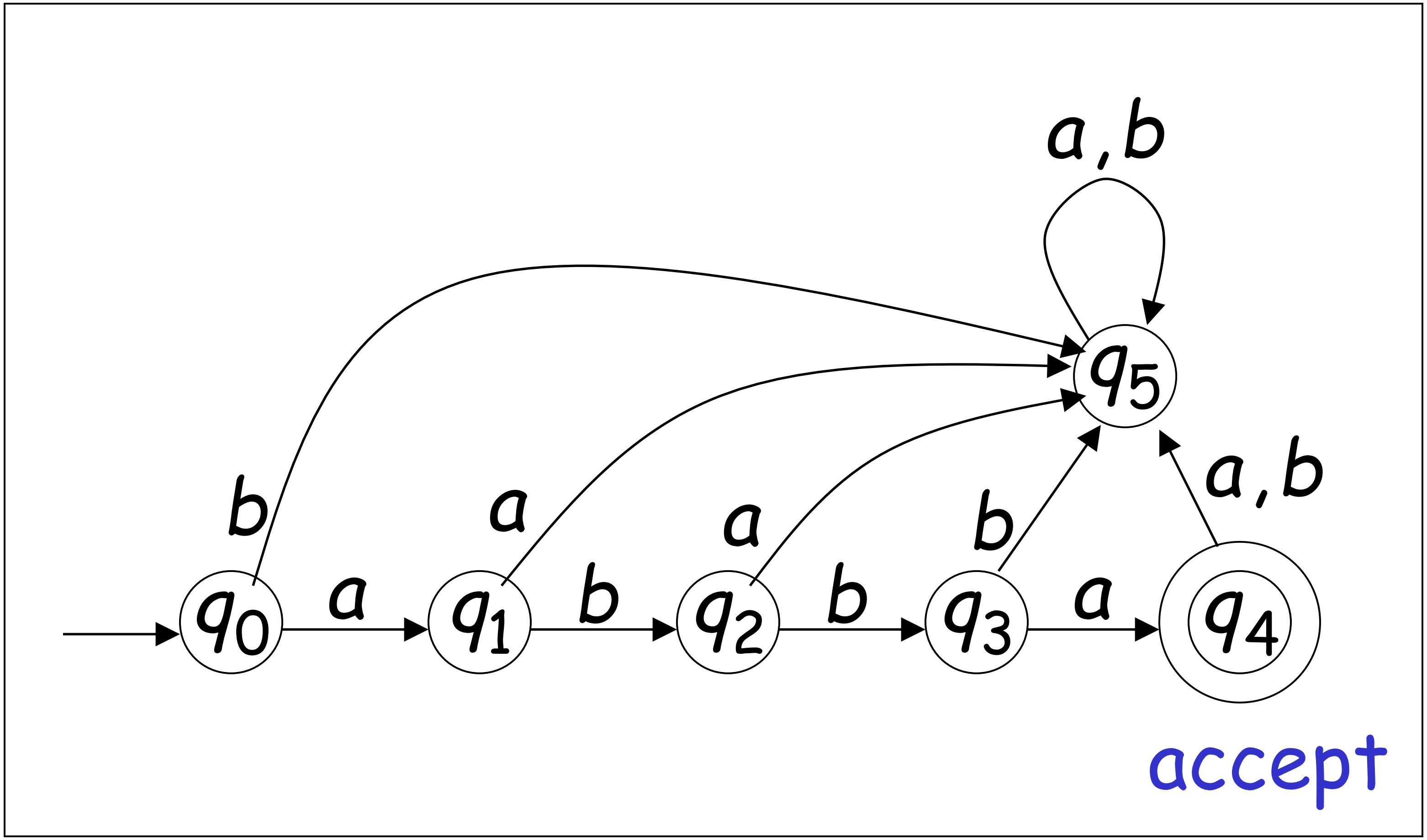
The language $L(M)$ contains all input strings accepted by M

$L(M)$ = { strings that drive M to a final state}

Example

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

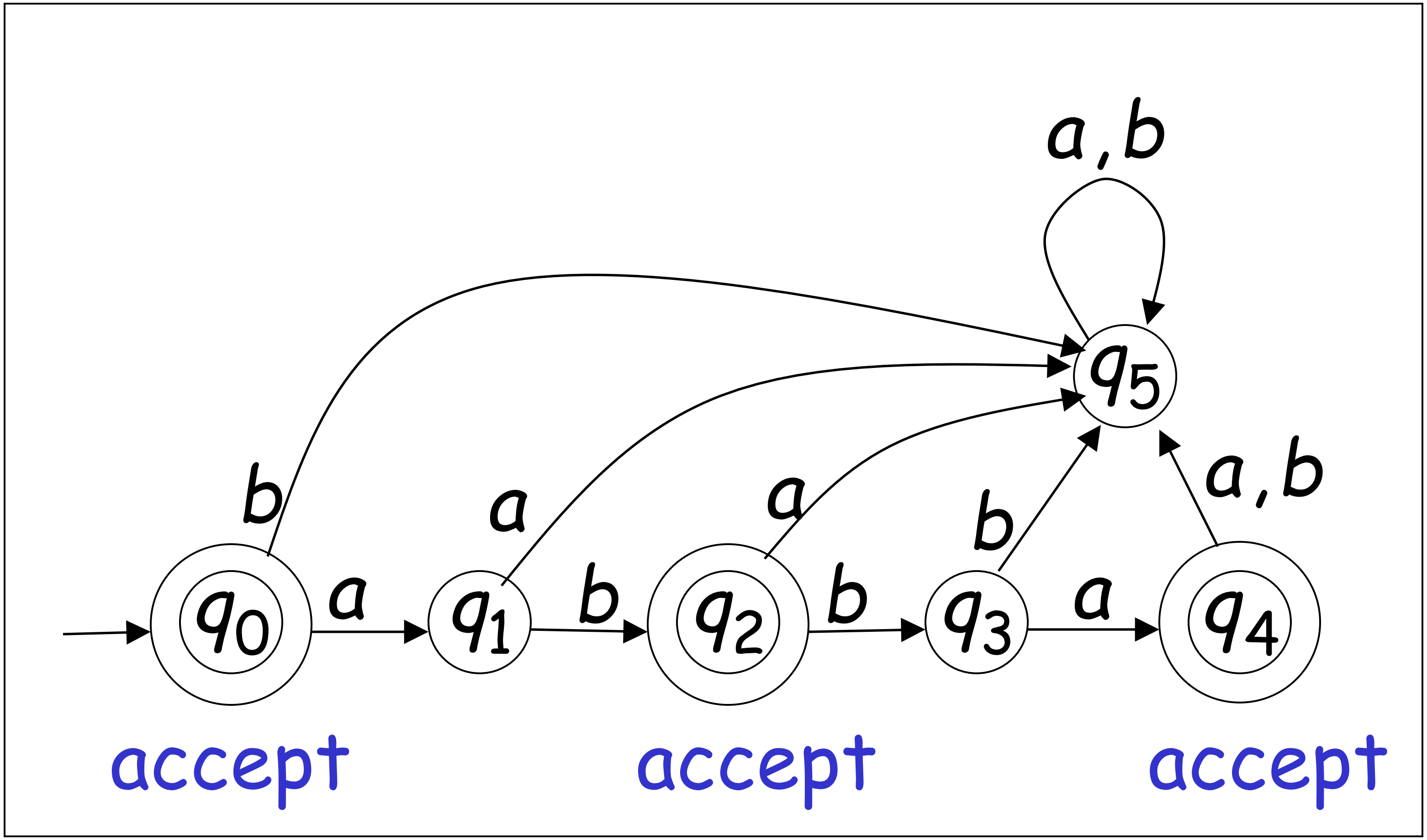
M



Another Example

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

M

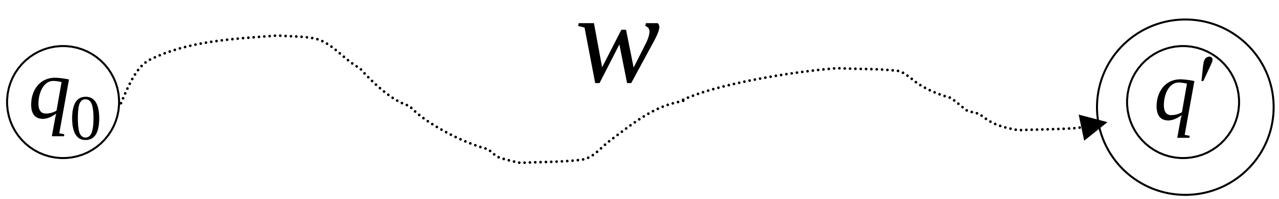


Formally

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

Language accepted by M :

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



$q' \in F$

Observation

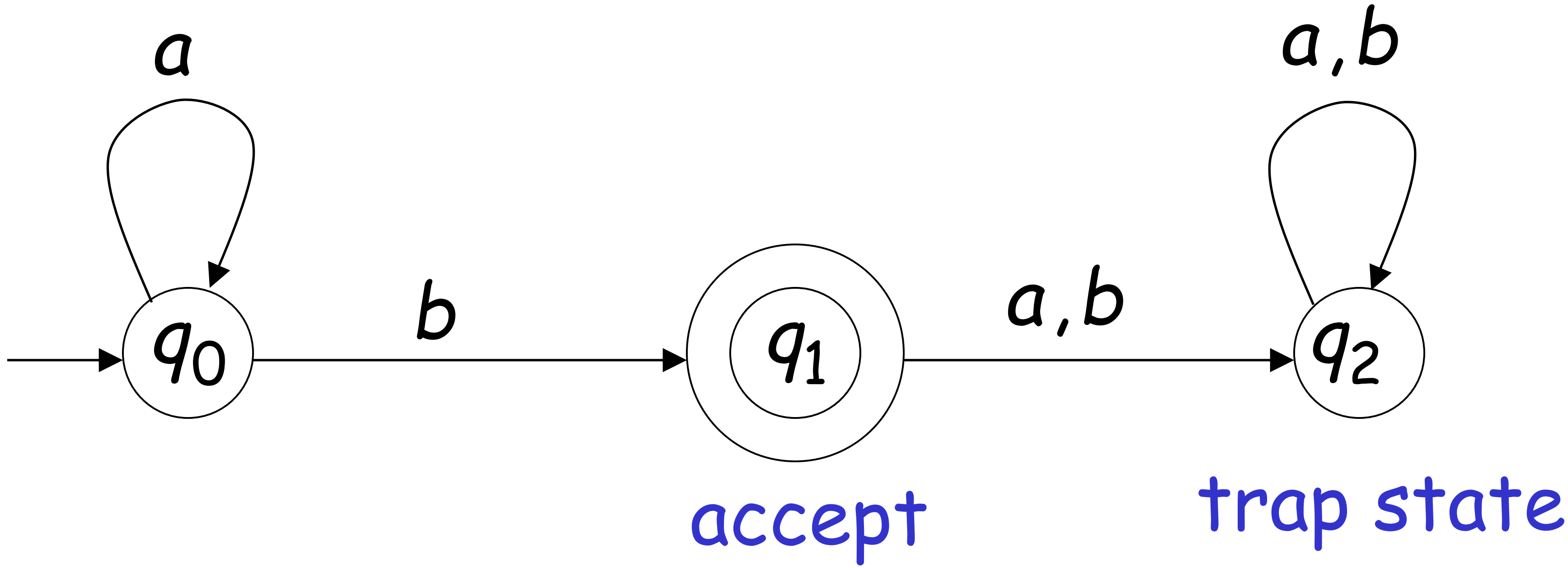
Language rejected by M :

$$\overline{L(M)} = \{ w \in \Sigma^* : \delta^*(q_0, w) \notin F \}$$

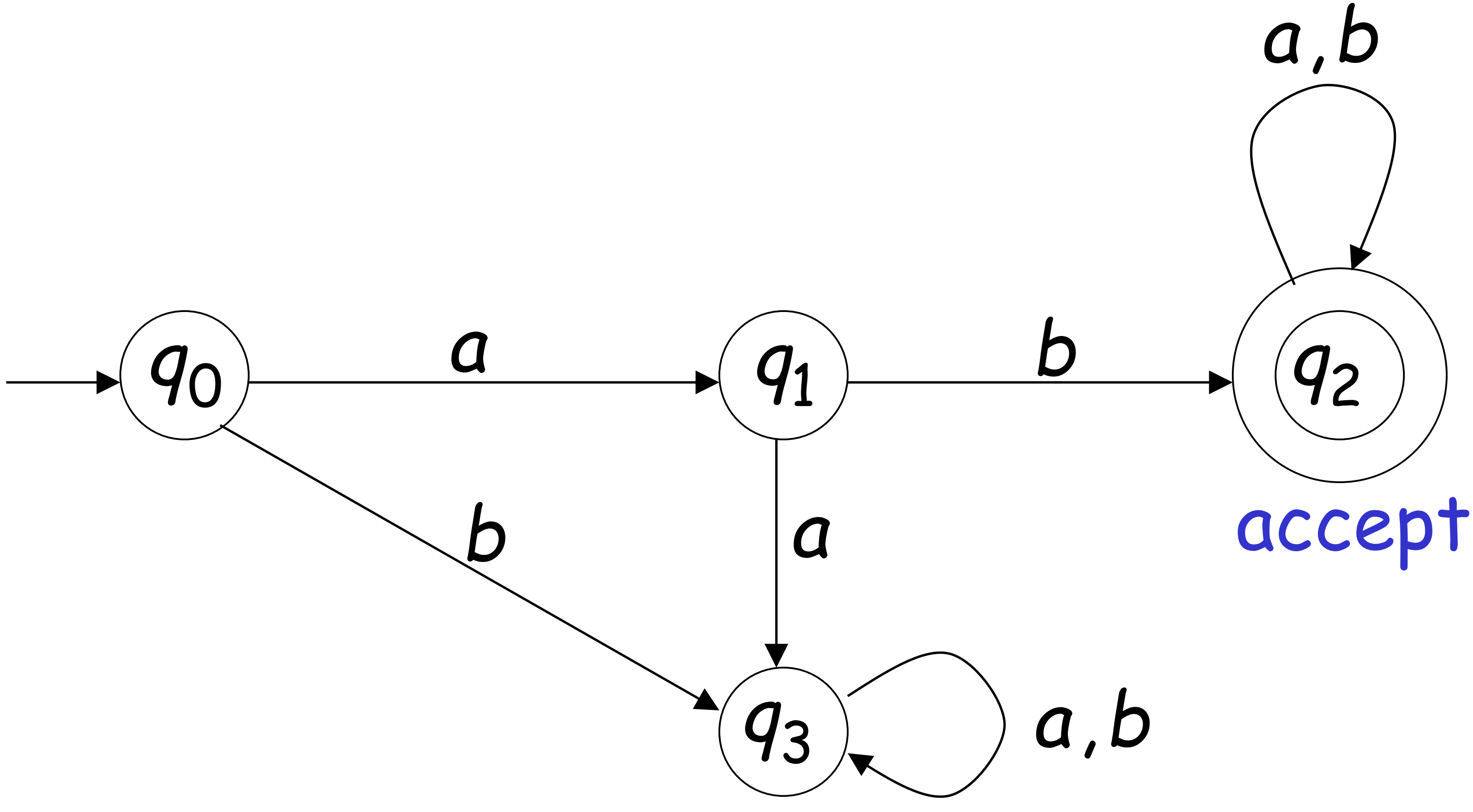
$$q_0 \xrightarrow{w} q' \qquad q' \notin F$$

More Examples

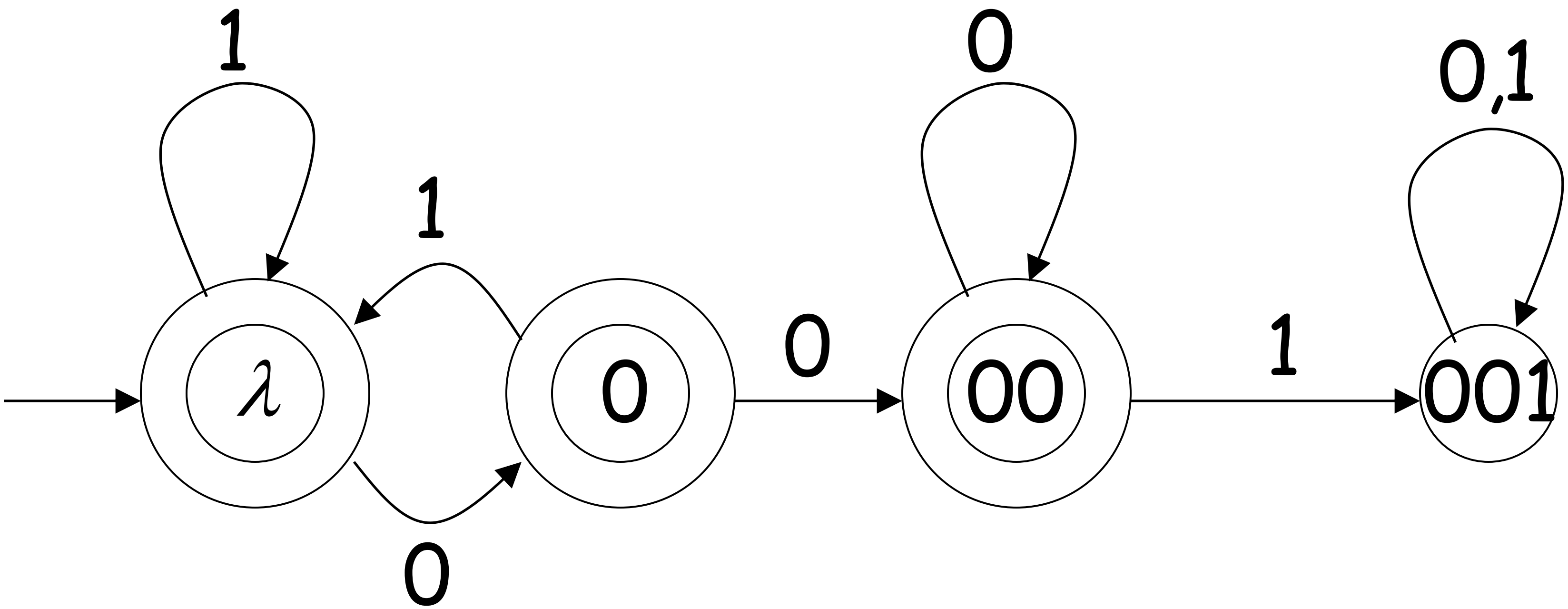
$$L(M) = \{a^n b : n \geq 0\}$$



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



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



Regular Languages

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

All regular languages form a language family

Examples of regular languages:

$\{abba\}$ $\{\lambda, ab, abba\}$ $\{a^n b : n \geq 0\}$

{ all strings with prefix ab }

{ all strings with prefix ab }

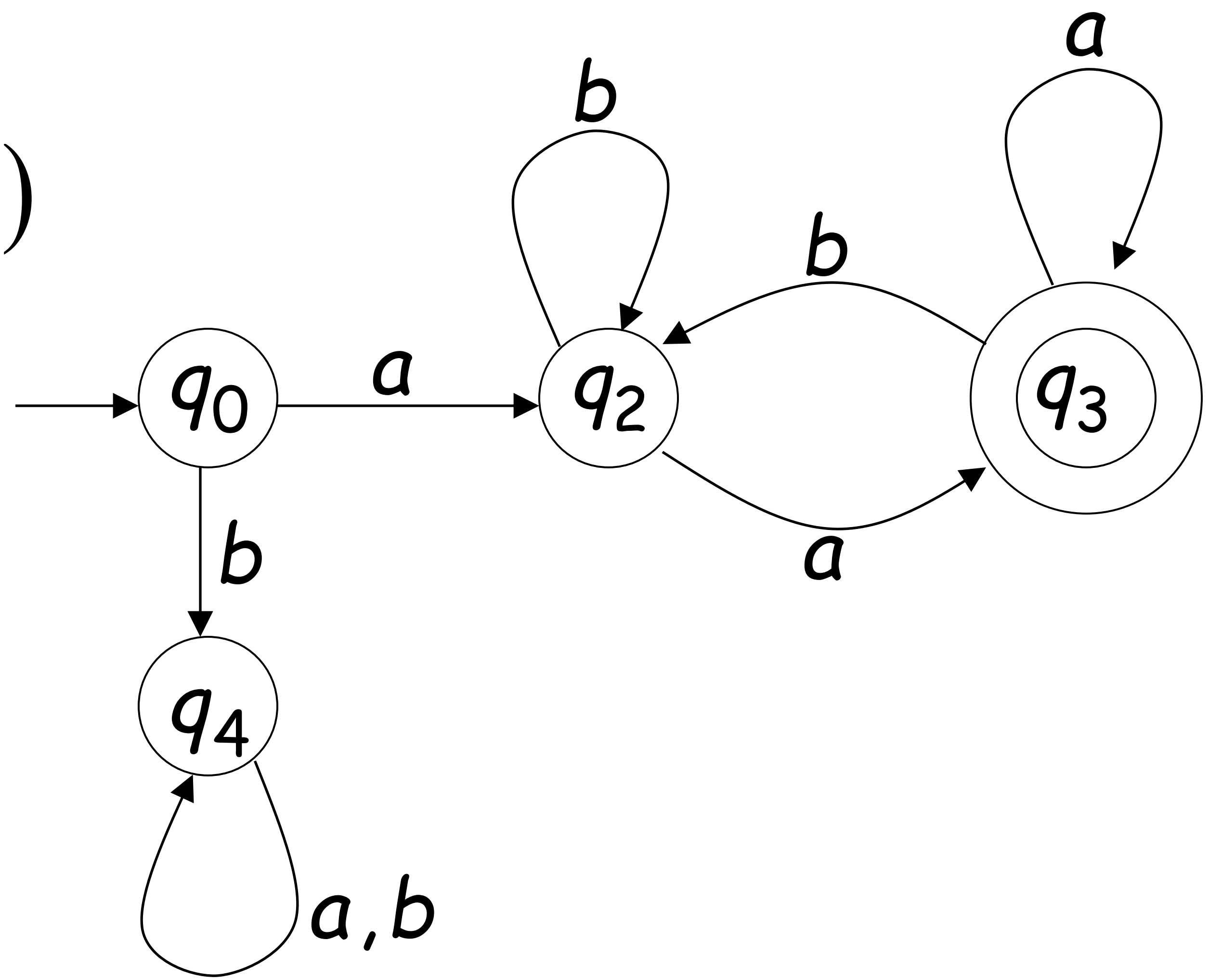
{ all strings without substring 001 }

There exist automata that accept these Languages (see previous slides).

Another Example

The language $L = \{awa : w \in \{a,b\}^*\}$
is regular:

$$L = L(M)$$



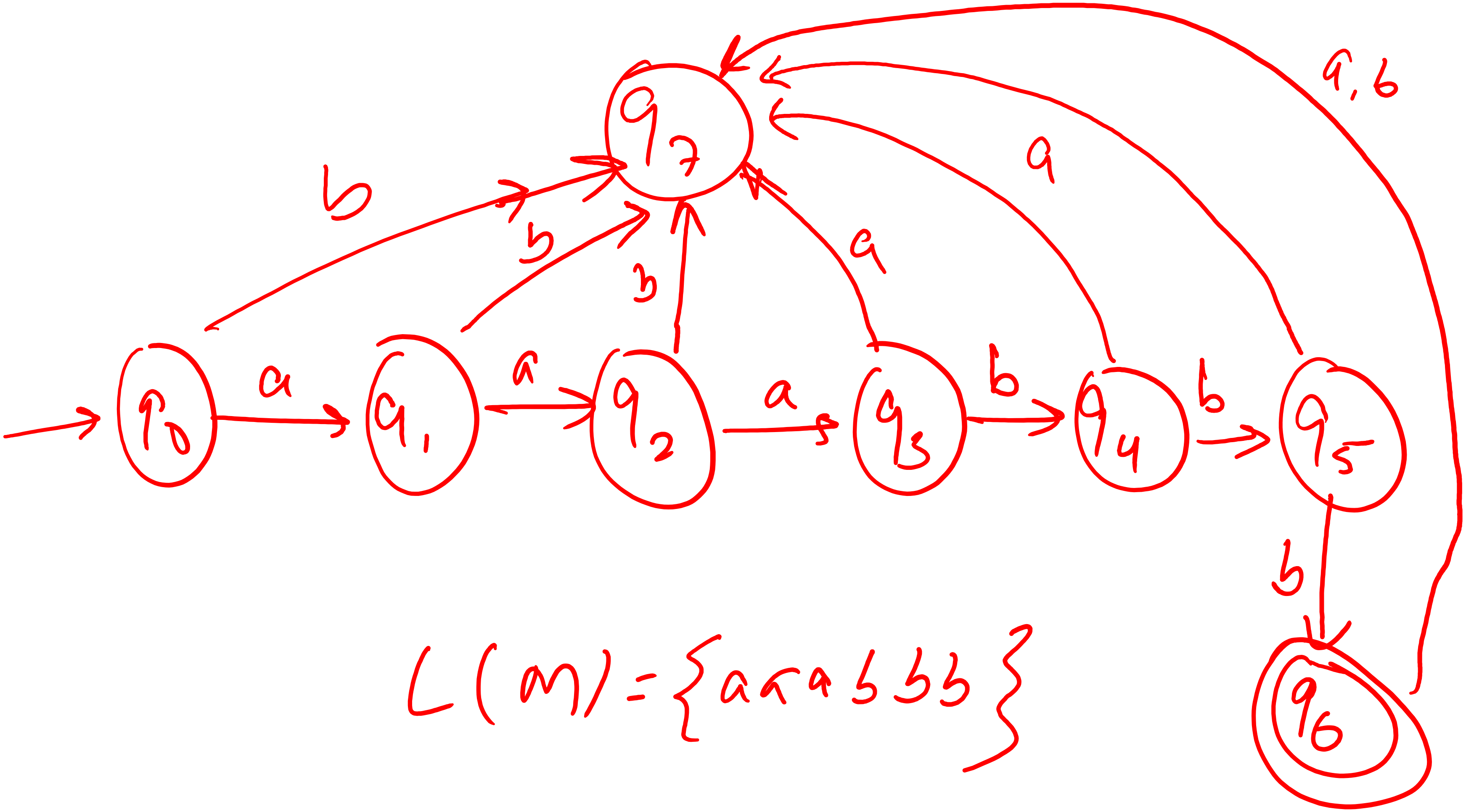
There exist languages which are <u>not</u> Regular:

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

There is no DFA that accepts such a language

(we will prove this later in the class)

aaabbb



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

$\boxed{\text{Design } L = \{aaaabbbb\}}$