REGULAR EXPRESSIONS AND AUTOMATA

Lecture 3: REGULAR EXPRESSIONS AND AUTOMATA

Husni Al-Muhtaseb

بسم الله الرحمن الرحيم

ICS 482: Natural Language Processing

Lecture 3: REGULAR EXPRESSIONS AND AUTOMATA

Husni Al-Muhtaseb

NLP Credits and Acknowledgment

These slides were adapted from presentations of the Authors of the book

SPEECH and LANGUAGE PROCESSING:
An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition

and some modifications from presentations found in the WEB by several scholars including the following

NLP Credits and Acknowledgment

If your name is missing please contact me
muhtaseb
At
Kfupm.
Edu.
sa

NLP Credits and Acknowledgment

Husni Al-Muhtaseb
James Martin
Jim Martin
Dan Jurafsky
Sandiway Fong
Song young in
Paula Matuszek
Mary-Angela Papalaskari
Dick Crouch
Tracy Kin
L. Venkata Subramaniam
Martin Volk
Bruce R. Maxim
Jan Hajič
Srinath Srinivasa
Simeon Ntafos
Paolo Pirjanian
Ricardo Vilalta
Tom Lenaerts
Heshaam Feili
Björn Gambäck
Christian Korthals
Thomas G. Dietterich
Devika Subramanian
Duminda Wijesekera
Lee McCluskey
David J. Kriegman
Kathleen McKeown
Michael J. Ciaraldi
David Finkel
Min-Yen Kan
Andreas Geyer-Schulz
Franz J. Kurfess
Tim Finin
Nadjet Bouayad
Kathy McCoy
Hans Uszkoreit
Khurshid Ahmad
Staffan Larsson
Robert Wilensky
Feiyu Xu
Jakub Piskorski
Rohini Srihari
Mark Sanderson
Andrew Elks
Marc Davis
Ray Larson
Jimmy Lin
Marti Hearst
Andrew McCallum
Nick Kushmerick
Mark Craven
Chia-Hui Chang
Diana Maynard
James Allan
Martha Palmer
julia hirschberg
Elaine Rich
Christof Monz
Bonnie J. Dorr
Nizar Habash
Massimo Poesio
David Goss-Grubbs
Thomas K Harris
John Hutchins
Alexandros Potamianos
Mike Rosner
Latifa Al-Sulaiti
Giorgio Satta
Jerry R. Hobbs
Christopher Manning
Hinrich Schütze
Alexander Gelbukh
Gina-Anne Levow
Guitao Gao
Qing Ma
Zeynep Altan

Agenda: REGULAR EXPRESSIONS AND AUTOMATA

- Why to study it?
  - Talk to ALICE
- Regular expressions
- Finite State Automata
- Assignments

NLP Example: Chat with Alice

- http://www.pandorabots.com/pandora/talk?botid=f5d922d97e345aa1&skin=custom_input
- A.L.I.C.E. (Artificial Linguistic Internet Computer Entity) is an award-winning free natural language artificial intelligence chat robot. The software used to create A.L.I.C.E. is available as free ("open source") Alicebot and AIML software.
- http://www.alicebot.org/about.html

NLP Representations

- State Machines
  - FSAs: Finite State Automata
  - FSTs: Finite State Transducers
  - HMMs: Hidden Markov Model
  - ATNs: Augmented Transition Network
  - RTNs: Recursive Transition Network

NLP Representations

- Rule Systems:
  - CFGs: Context Free Grammar
  - Unification Grammars
  - Probabilistic CFGs
- Logic-based Formalisms
  - 1st Order Predicate Calculus
  - Temporal and other Higher Order Logics
- Models of Uncertainty
  - Bayesian Probability Theory

NLP Algorithms

- Most are transducers: accept or reject input, and construct new structure from input
  - State space search
    - To manage the problem of making choices during processing when we lack the information needed to make the right choice
  - Dynamic programming
    - To avoid having to redo work during the course of a state-space search

State Space Search

- States represent pairings of partially processed inputs with partially constructed answers
- Goals are exhausted inputs paired with complete answers that satisfy some criteria
- The spaces are normally too large to exhaustively explore

Dynamic Programming

- Don’t do the same work over and over
- Avoid this by building and making use of solutions to sub-problems that must be invariant across all parts of the space

Regular Expressions and Text Searching

- Regular expression (RE): A formula (in a special language) for specifying a set of strings
- String: A sequence of alphanumeric characters (letters, numbers, spaces, tabs, and punctuation)

Regular Expression Patterns

- Regular Expression can be considered as a pattern to specify text search strings to search a corpus of texts
- What is Corpus?
- For text search purpose: use Perl syntax
- Show the exact part of the string in a line that first matches a Regular Expression pattern

Regular Expression Patterns

RE	String matched
/woodchucks/	“interesting links to woodchucks and lemurs”
/a/	“Sarah Ali stopped by Mona’s”
/Ali says,/	“My gift please,” Ali says,”
/book/	“all our pretty books”
/!/	“Leave him behind!” said Sami

RE	Match
/[wW]oodchuck/	Woodchuck or woodchuck
/[abc]/	“a”, “b”, or “c”
/[0123456789]/	Any digit

RE	Description
/a*/	Zero or more a’s
/a+/	One or more a’s
/a?/	Zero or one a’s
/cat|dog/	‘cat’ or ‘dog’
/^cat$/	A line containing only ‘cat’
/\bun\B/	Beginnings of longer strings starts by ‘un’

Example

- Find all instances of the word “the” in a text.
  - /the/
    - What About ‘The’
  - /[tT]he/
    - What about ‘Theater”, ‘Another’
  - /\b[tT]he\b/

Sidebar: Errors

- The process we just went through was based on two fixing kinds of errors
  - Matching strings that we should not have matched (there, then, other)
    - False positives
  - Not matching things that we should have matched (The)
    - False negatives

Sidebar: Errors

- Reducing the error rate for an application often involves two efforts
  - Increasing accuracy (minimizing false positives)
  - Increasing coverage (minimizing false negatives)

Regular expressions

- Basic regular expression patterns
- Perl-based syntax (slightly different from other notations for regular expressions)
- Disjunctions **[abc]**
- Ranges **[A-Z]**
- Negations **[^Ss]**
- Optional characters **?** and *****
- Wild cards **.**
- Anchors **^** and **$**, also **\b** and **\B**
- Disjunction, grouping, and precedence **|**

RE	Match	Example Patterns Matched
`woodchucks?`	woodchuck or woodchucks	"woodchuck"
`colou?r`	color or colour	"colour"

RE	Match	Example Patterns
/beg.n/	any character between *beg* and *n*	begin, beg'n, begun

RE	Match (single characters)	Example Patterns Matched
[^A-Z]	not an uppercase letter	"Oyfn pripetchik"
[^Ss]	neither 'S' nor 's'	"I have no exquisite reason for't"
[^\.]	not a period	"our resident Djinn"
[e^]	either 'e' or '^'	"look up ^ now"
a^b	the pattern 'a^b'	"look up a^b now"

Writing correct expressions

- Exercise: write a Perl regular expression to match the English article “the”:

```
/the/
```
missed ‘The’

```
/[tT]he/
```
included ‘the’ in ‘others’

```
/\b[tT]he\b/
```
Missed ‘the25’ ‘the_’

```
/[^a-zA-Z][tT]he[^a-zA-Z]/
```
Missed ‘The’ at the beginning of a line

```
/(^|[^a-zA-Z])[tT]he[^a-zA-Z]/
```

A more complex example

- Exercise: Write a regular expression that will match “any PC with more than 500MHz and 32 Gb of disk space for less than $1000”:

Example

- Price
  - /$[0-9]+/ # whole dollars
  - /$[0-9]+\.[0-9][0-9]/ # dollars and cents
  - /$[0-9]+(\.[0-9][0-9])?/ #cents optional
  - /\b$[0-9]+(\.[0-9][0-9])?\b/ #word boundaries
- Specifications for processor speed
  - /\b[0-9]+ *(MHz|[Mm]egahertz|Ghz|[Gg]igahertz)\b/
- Memory size
  - /\b[0-9]+ *(Mb|[Mm]egabytes?)\b/
  - /\b[0-9](\.[0-9]+) *(Gb|[Gg]igabytes?)\b/
- Vendors
  - /\b(Win95|WIN98|WINNT|WINXP *(NT|95|98|2000|XP)?)\b/
  - /\b(Mac|Macintosh|Apple)\b/

Advanced Operators

RE	Expansion	Match	Example Patterns
\d	[0-9]	any digit	Party␣of␣5
\D	[^0-9]	any non-digit	Blue␣moon
\w	[a-zA-Z0-9_]	any alphanumeric or underscore	Daiyu
\W	[^\w]	a non-alphanumeric	!!!!
\s	[␣\r\t\n\f]	whitespace (space, tab)	
\S	[^\s]	Non-whitespace	in␣Concord

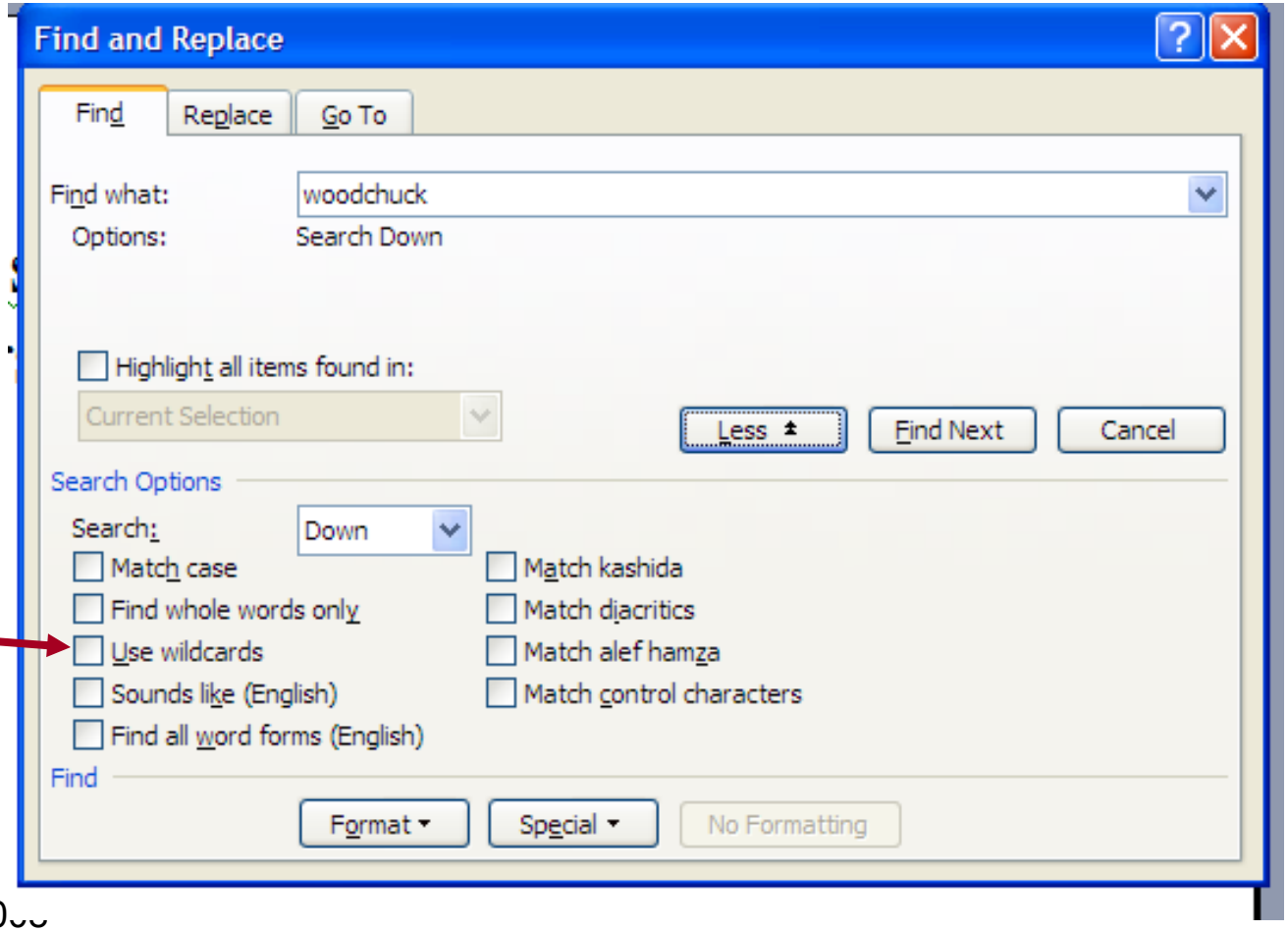
Figure 2.6 Aliases for common sets of characters.

Underscore:
Correct figure 2.6

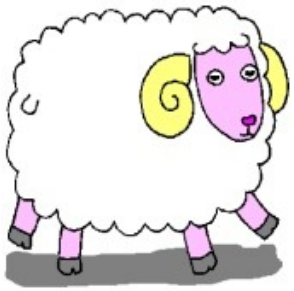
RE	Match	Example Patterns Matched
*	an asterisk "*"	"K*A*P*L*A*N"
\.	a period "."	"Dr. Livingston, I presume"
\?	a question mark	"Would you light my candle?"
\n	a newline	
\t	a tab	

RE	Match
*	zero or more occurrences of the previous char or expression
+	one or more occurrences of the previous char or expression
?	exactly zero or one occurrence of the previous char or expression
{n}	*n* occurrences of the previous char or expression
{n,m}	from *n* to *m* occurrences of the previous char or expression
{n,}	at least *n* occurrences of the previous char or expression

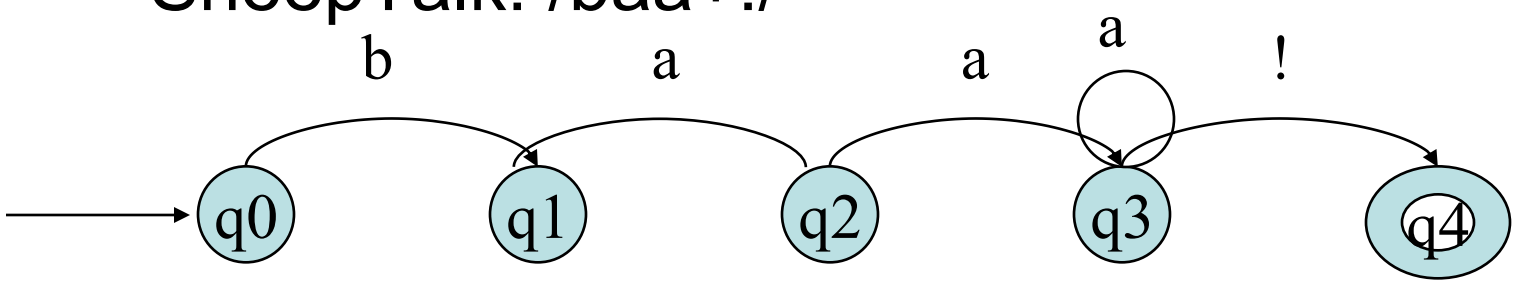
Assignment: Try regular expressions in MS WORD in both Arabic & English



Finite State Automata



- **FSA**s recognize the regular languages represented by regular expressions
  - SheepTalk: /baa+!/

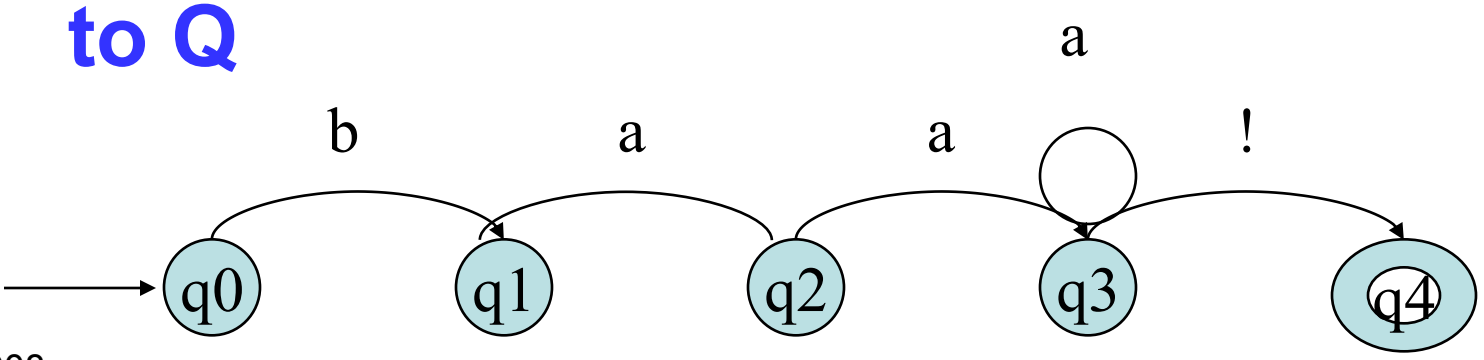


- **Directed graph with labeled nodes and arc transitions**

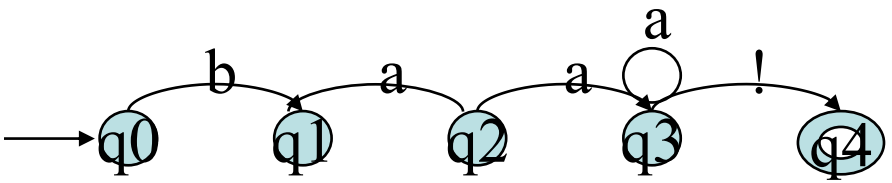
•Five **states:** q0 the **start** state, q4 the **final** state, 5 transitions

Formally

- **FSA** is a 5-tuple consisting of
  - **Q**: set of states {q0,q1,q2,q3,q4}
  - **Σ**: an alphabet of symbols {a,b,!}
  - **q0**: A start state
  - **F**: a set of final states in Q {q4}
  - **δ(q,i)**: a transition function mapping **Q x Σ to Q**



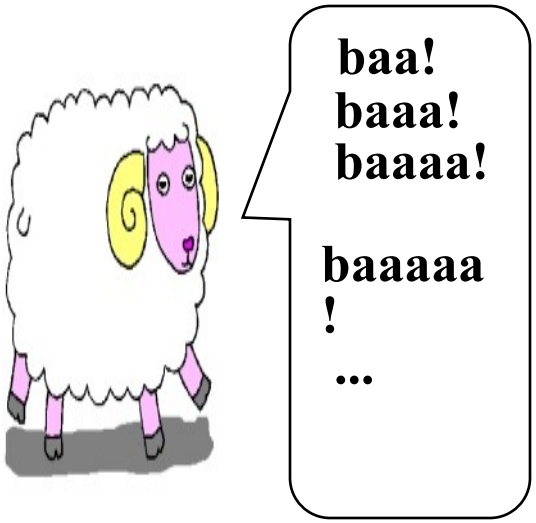
- FSA recognizes (**accepts**) strings of a regular language
  - baa!
  - baaa!
  - baaaa!
  - …

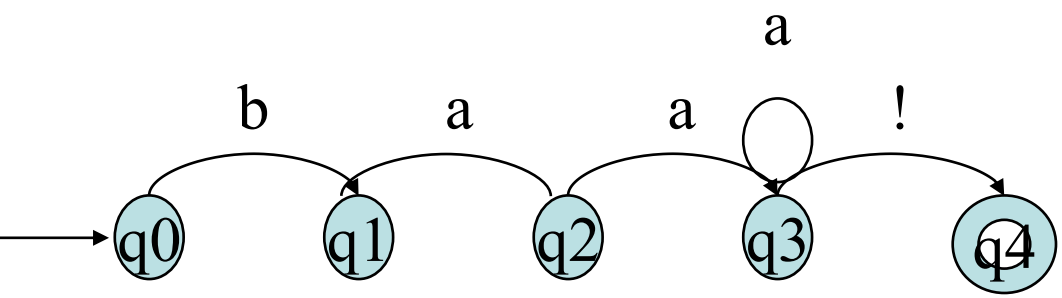


- Tape Input: a rejected input

	a	b	a	!	b	

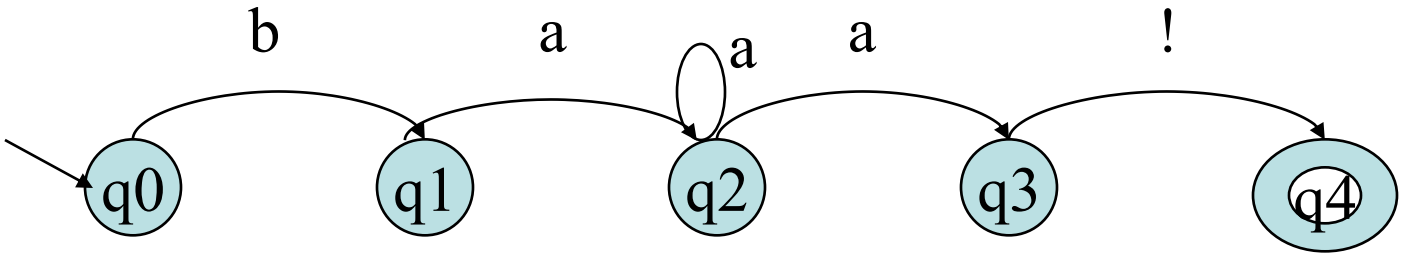
State Transition Table for SheepTalk

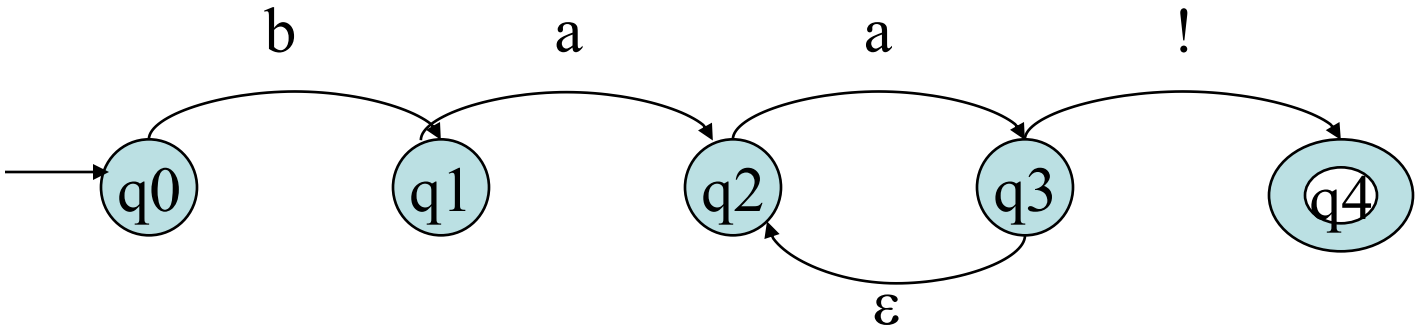




State	Input		
	b	a	!
0	1	Ø	Ø
1	Ø	2	Ø
2	Ø	3	Ø
3	Ø	3	4
4	Ø	Ø	Ø

Non-Deterministic FSAs for SheepTalk





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

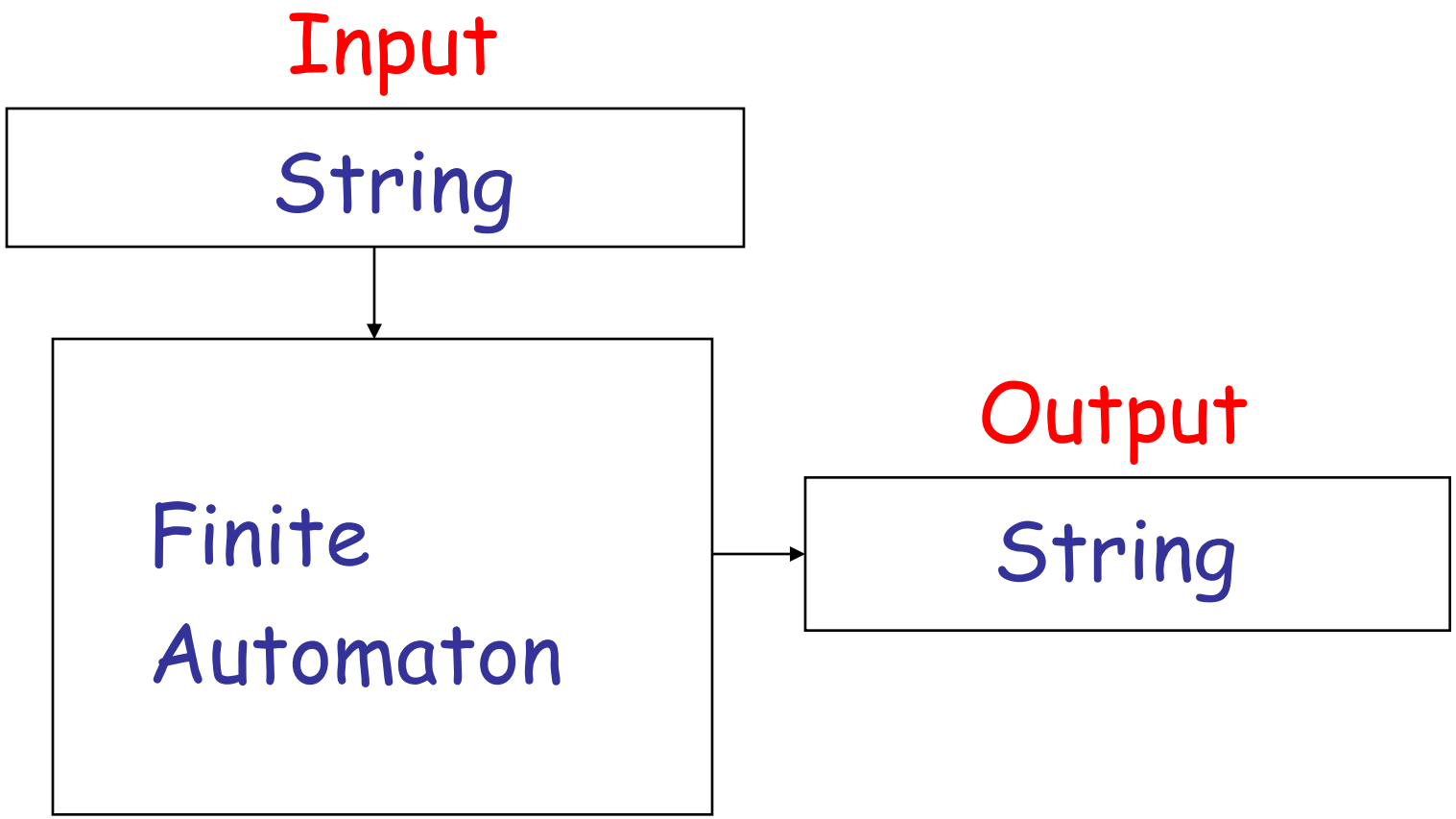
ab $\quad u = ab$

$abba$ $\quad v = bbbaaa$

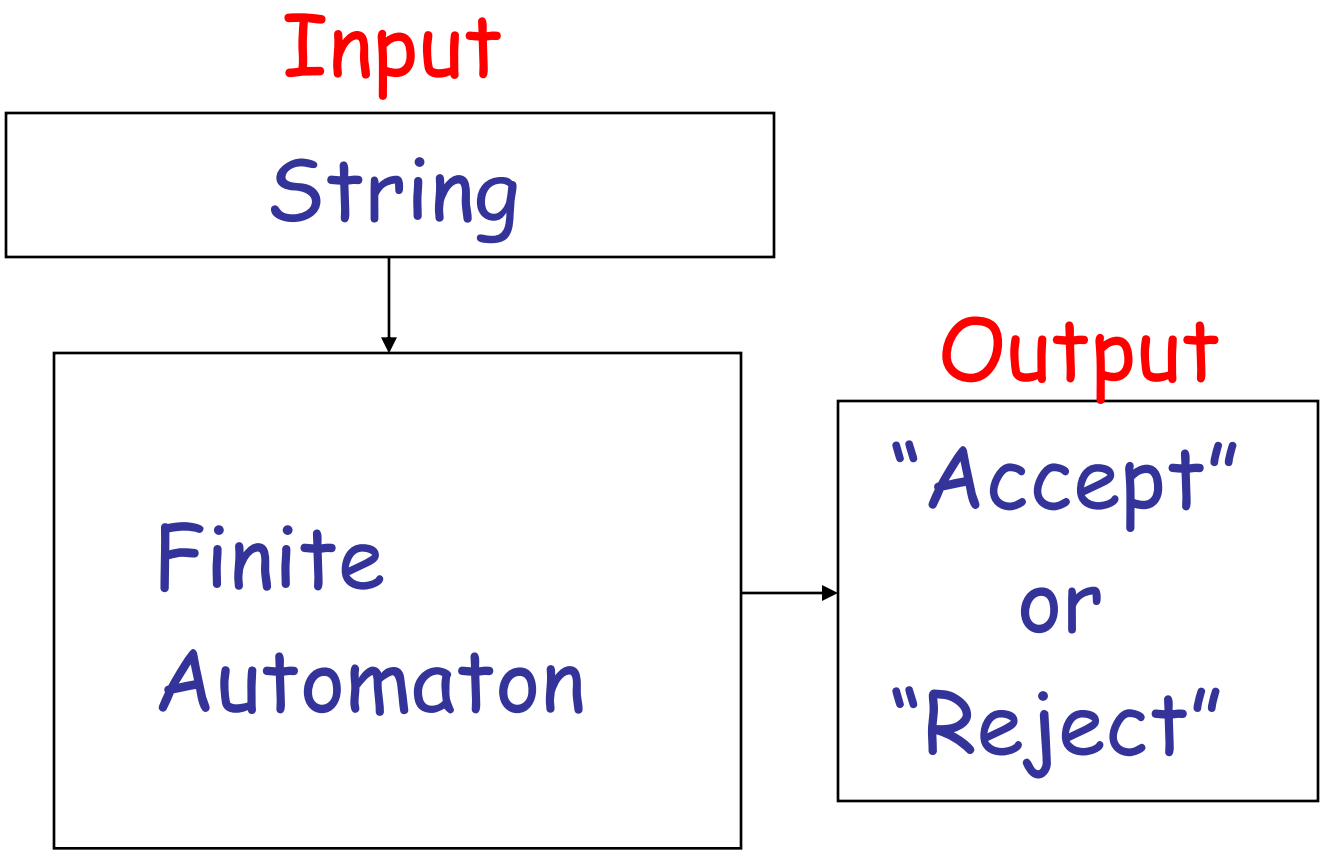
$baba$ $\quad w = abba$

$aaabbbaabab$

Finite Automaton

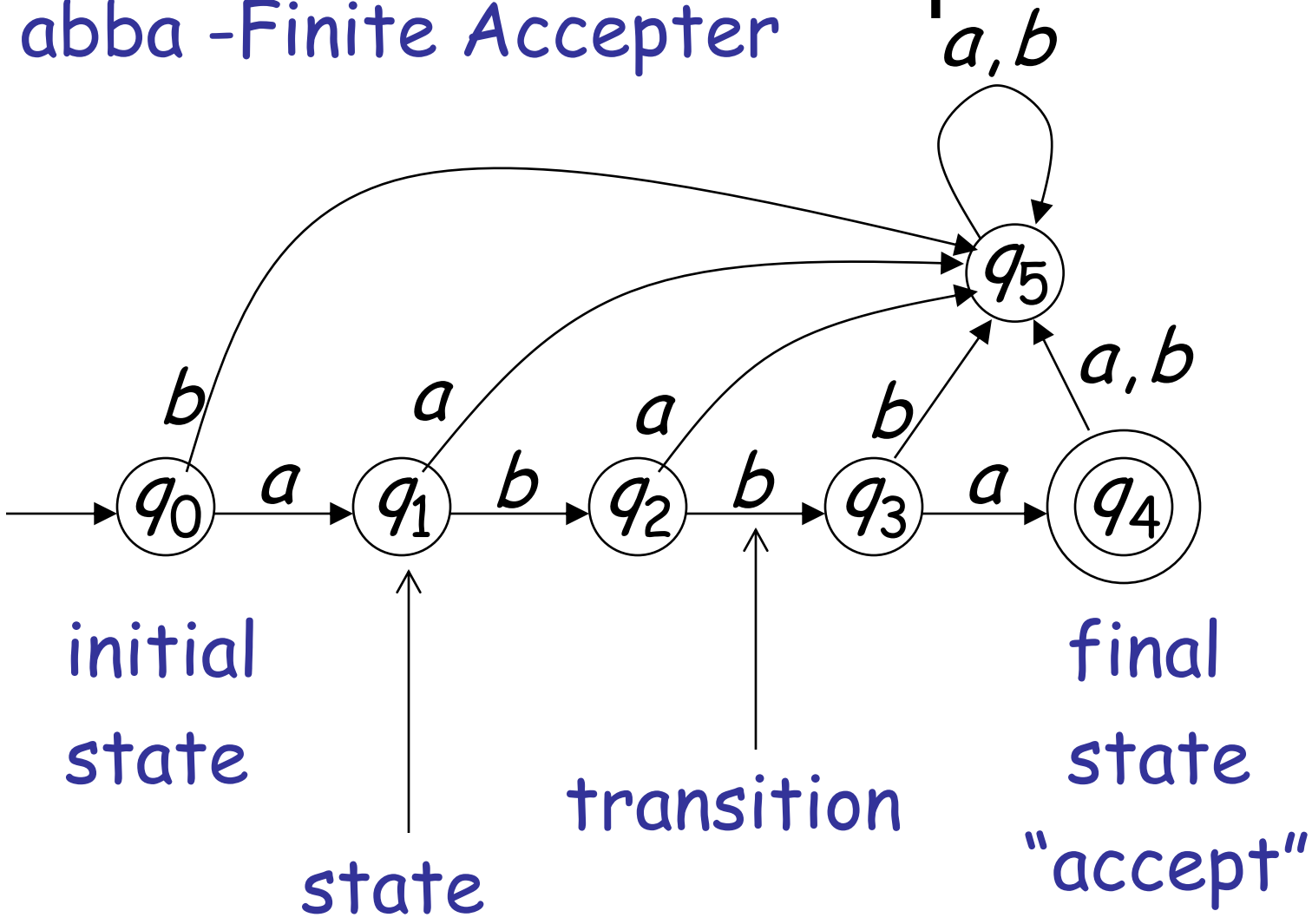
- 

Finite Accepter

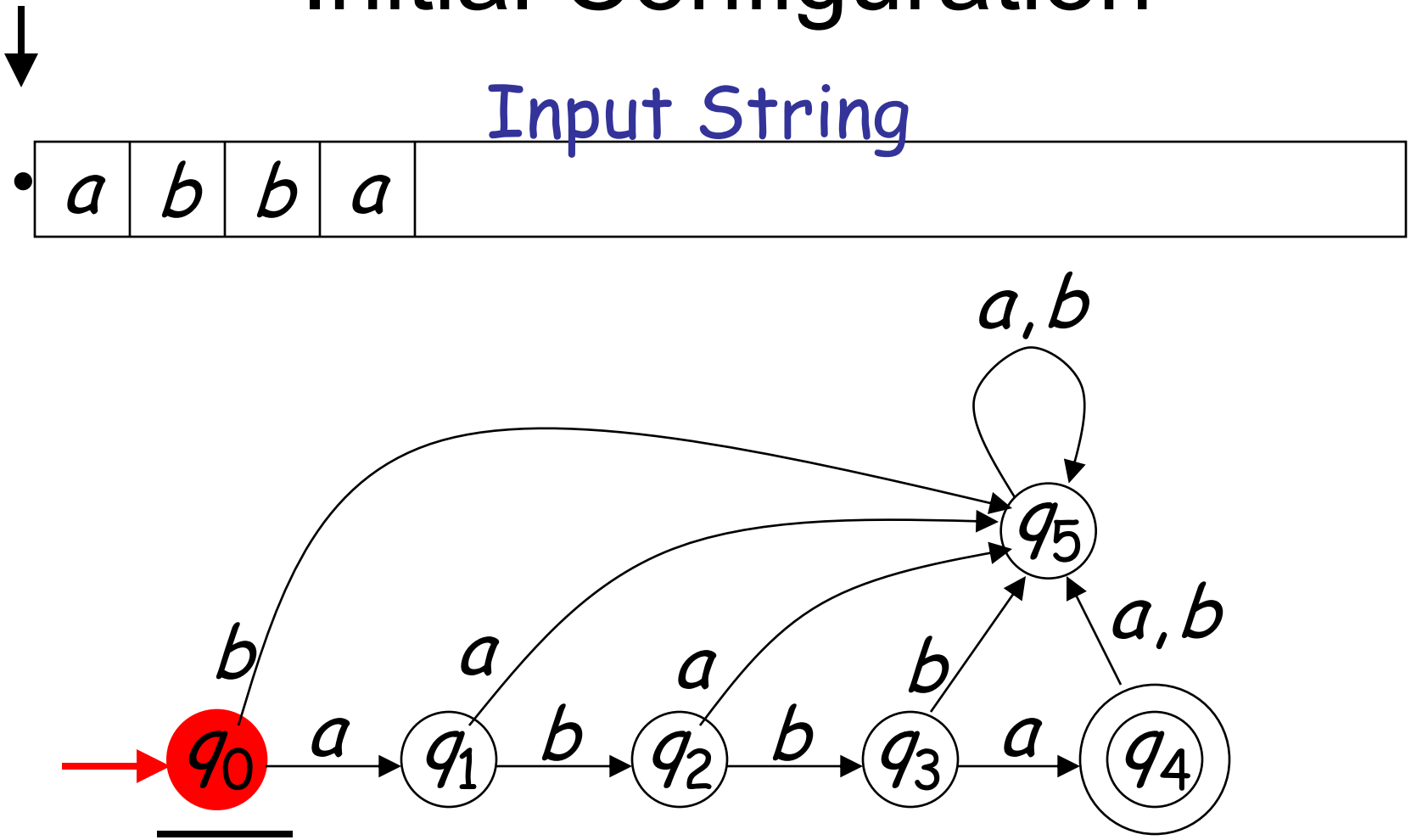


Transition Graph

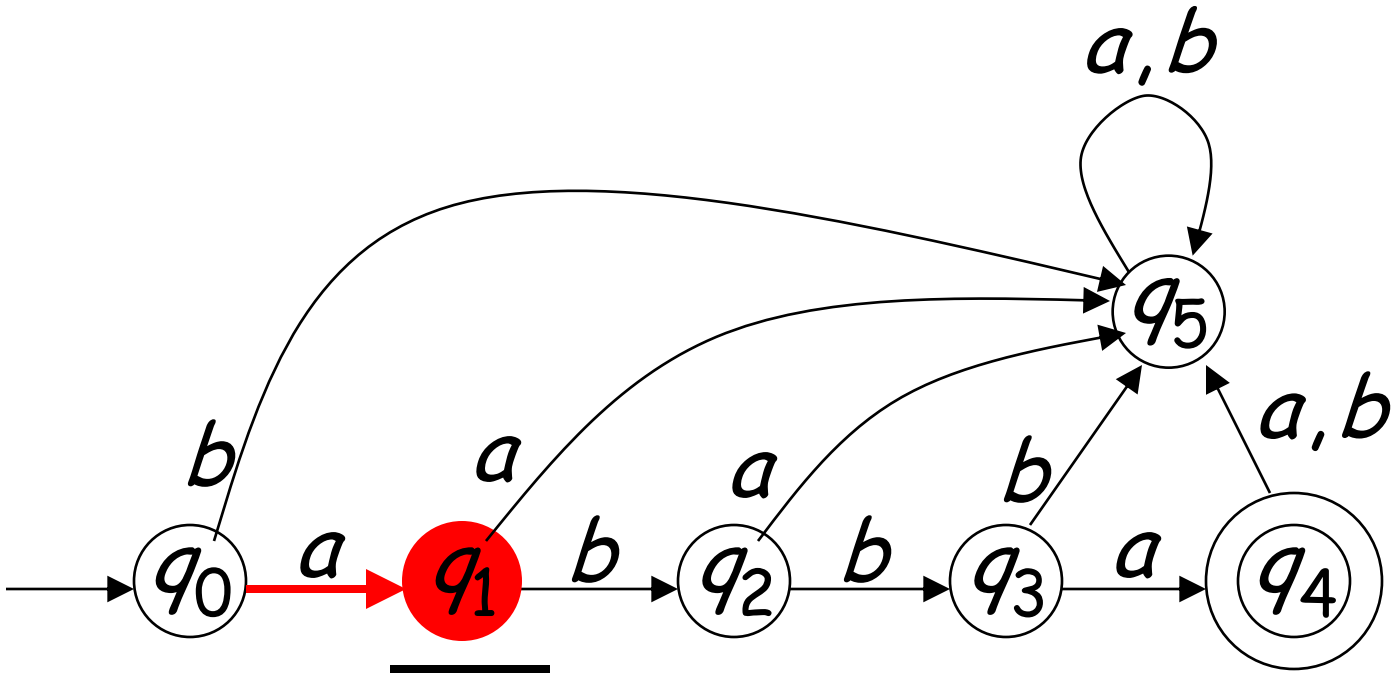
abba -Finite Accepter

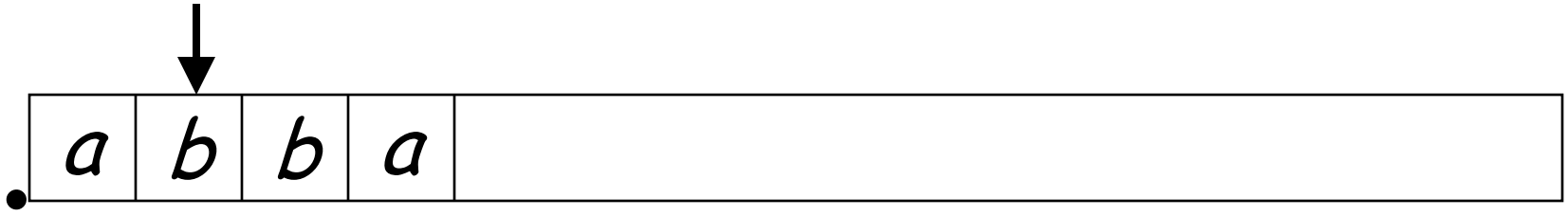


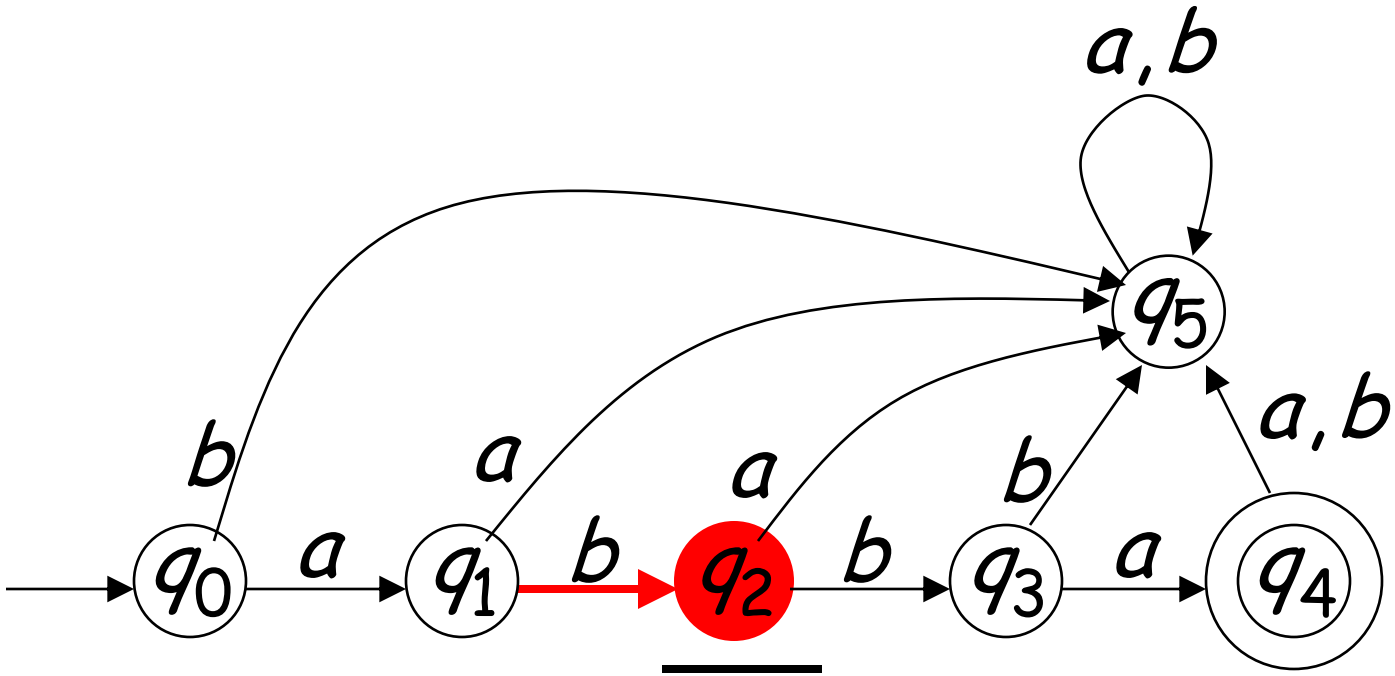
Initial Configuration

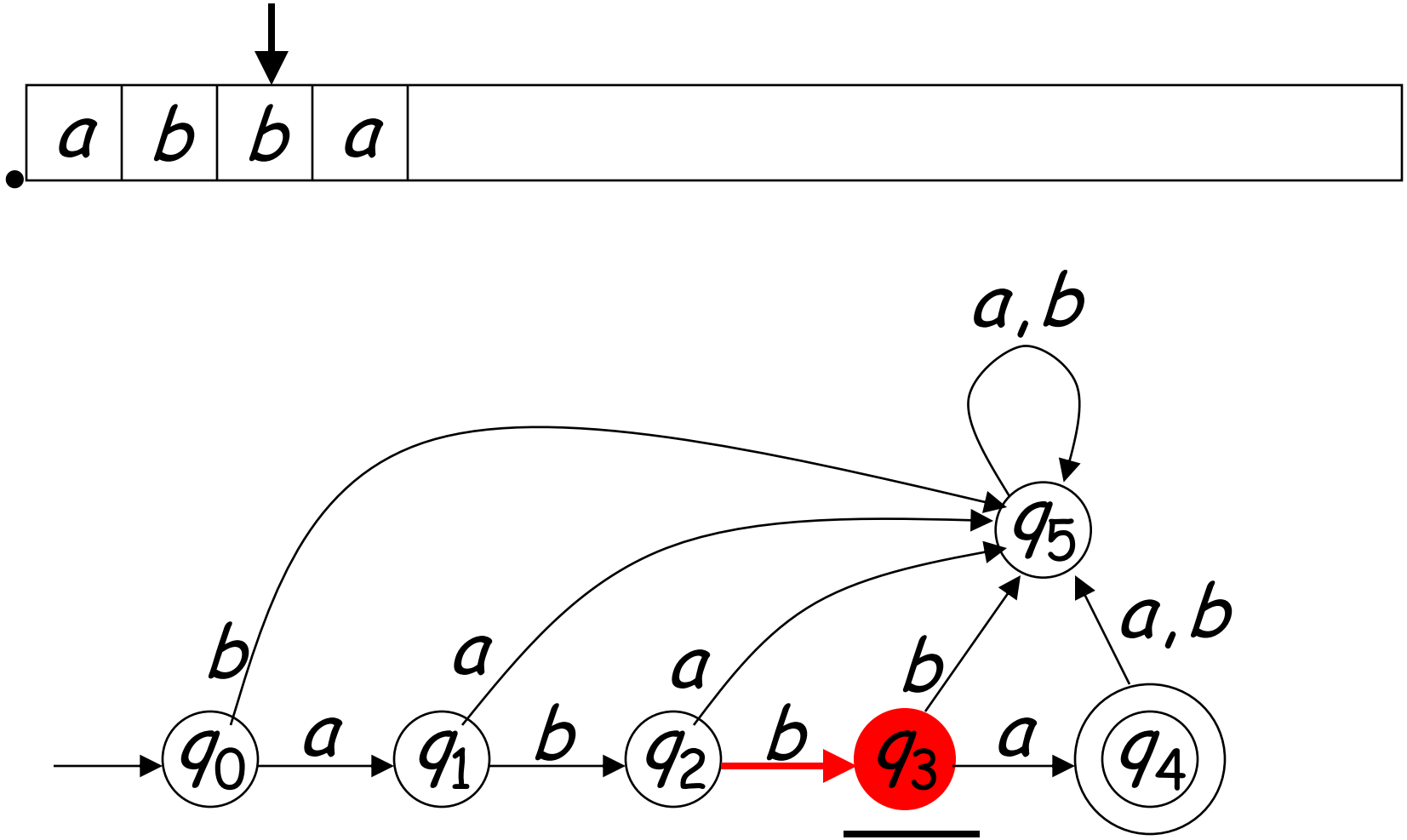


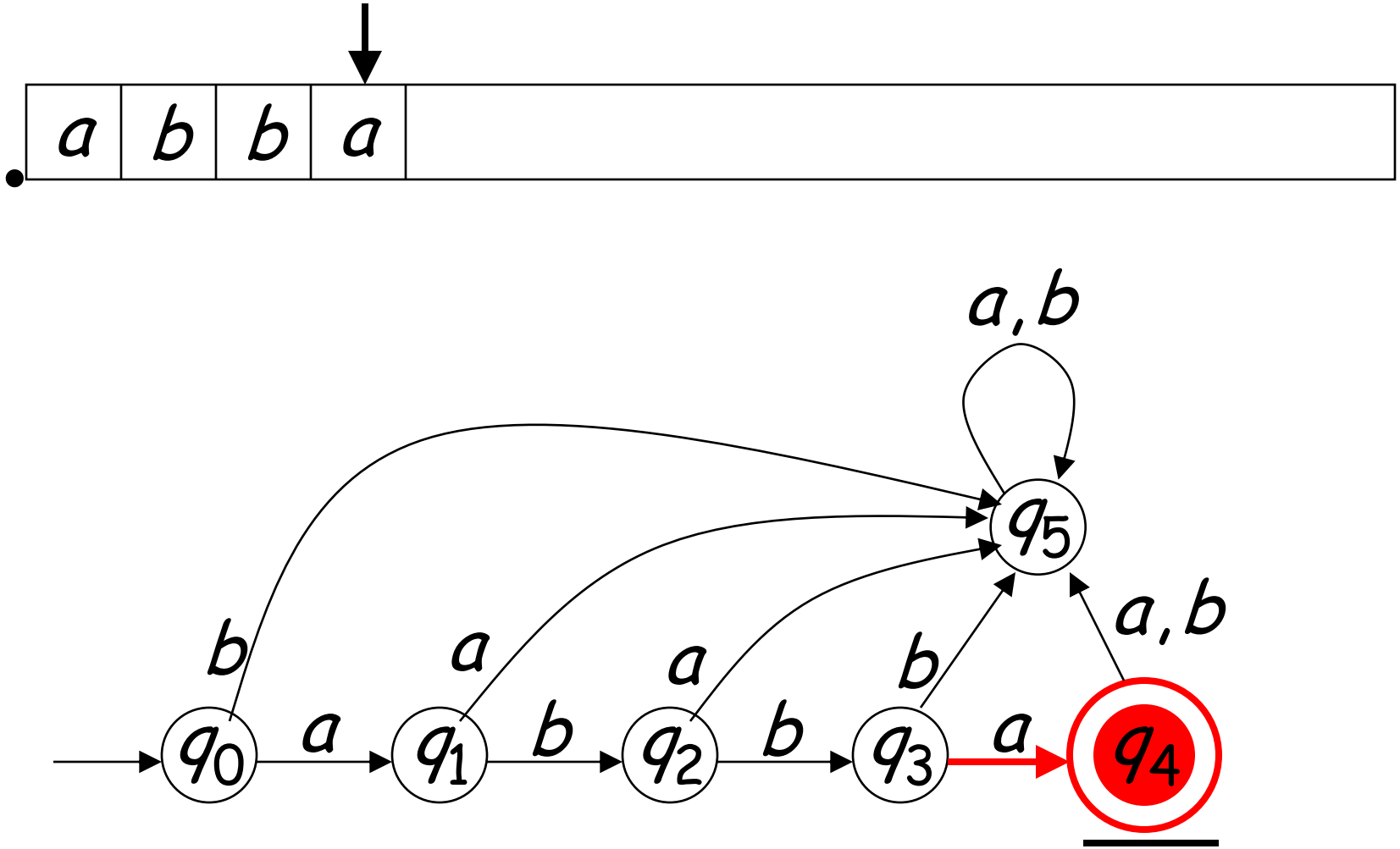
Reading the Input

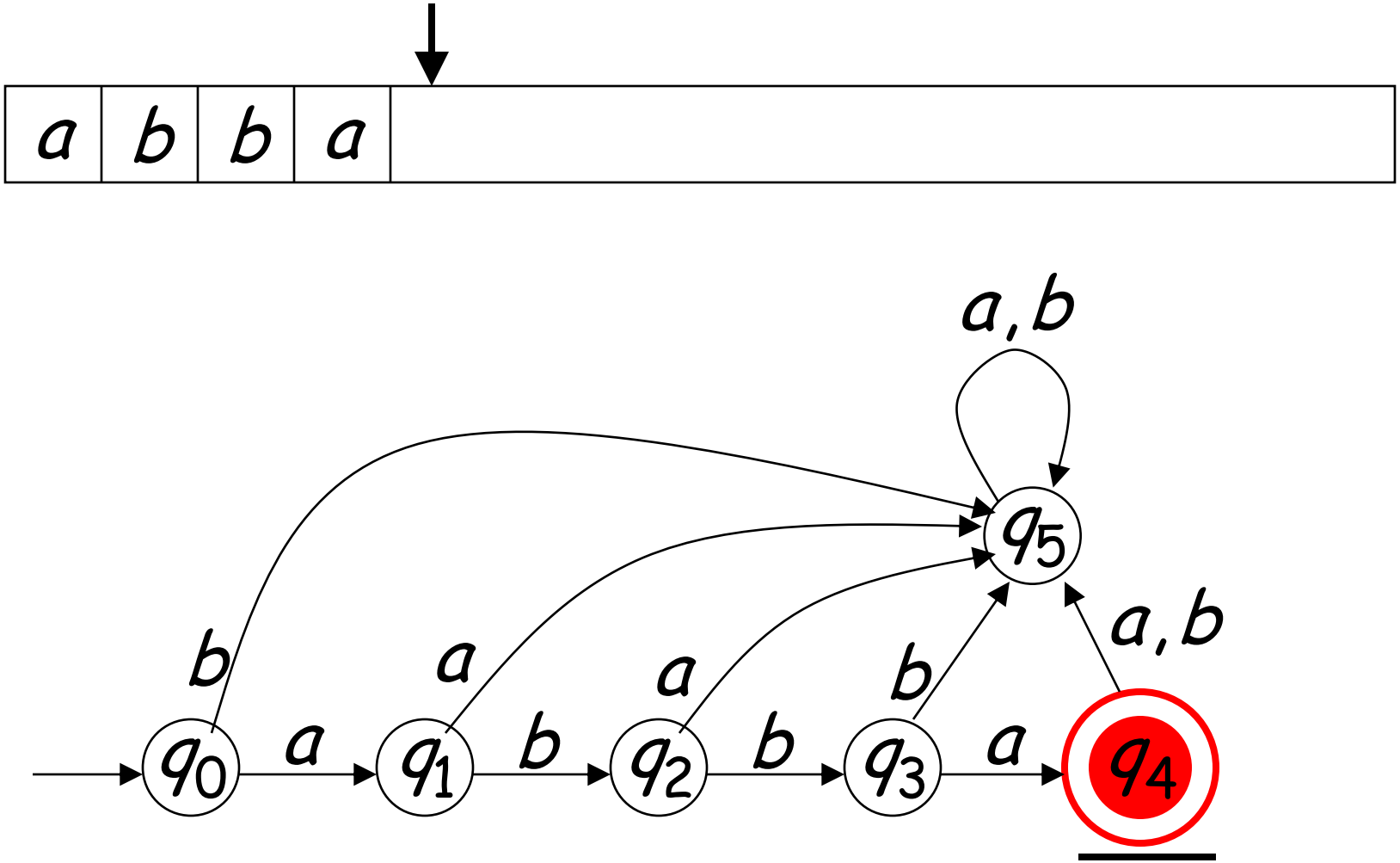



a
b
b
a


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

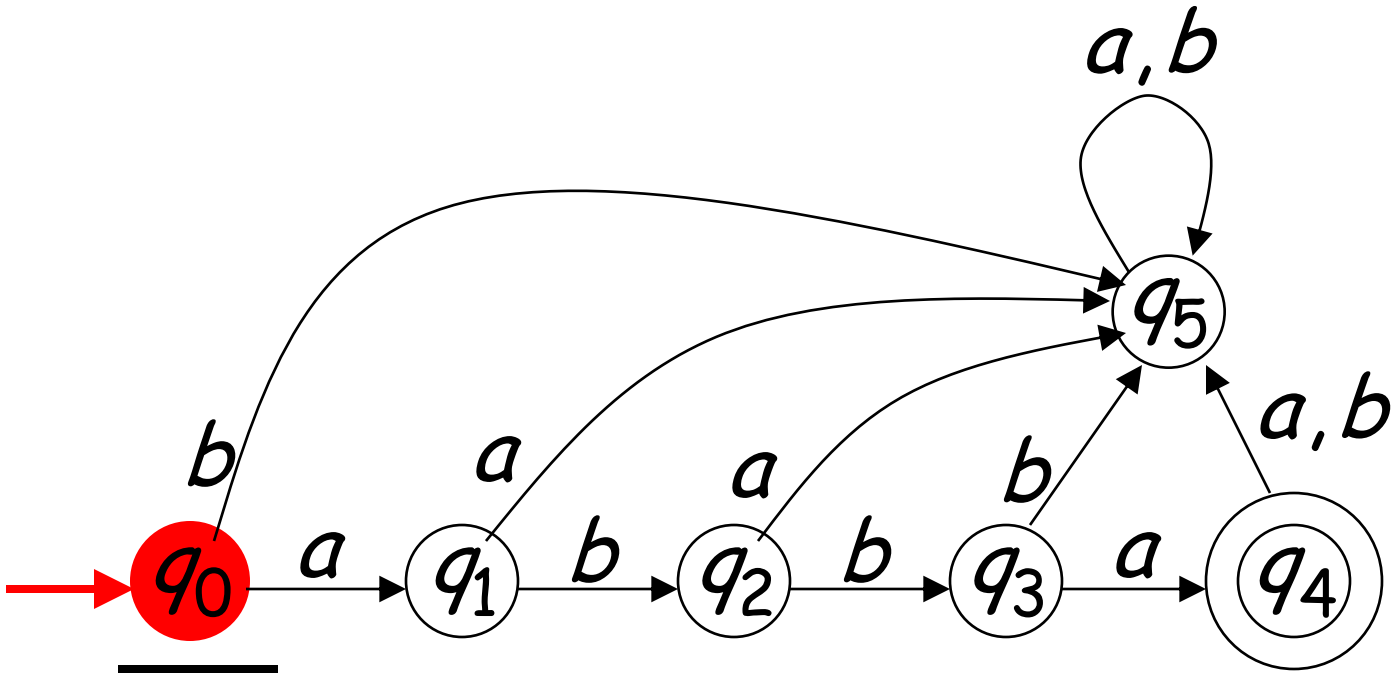

a b b a
a, b
q5
b
a
a
b
a, b
q0
a
q1
b
q2
b
q3
a
q4

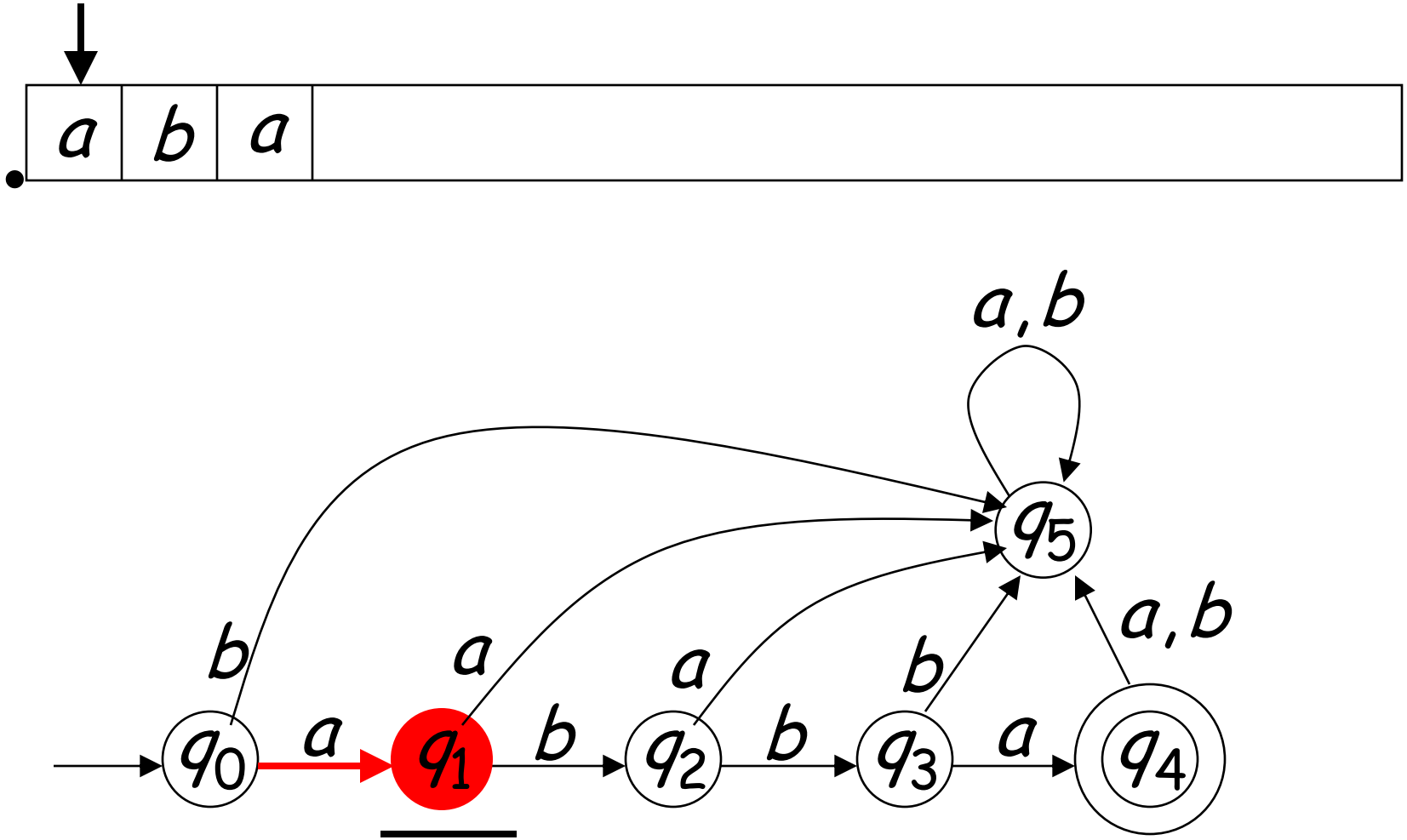

a b b a
a, b
q5
b
a
a
b
a, b
q0
a
q1
b
q2
b
q3
a
q4

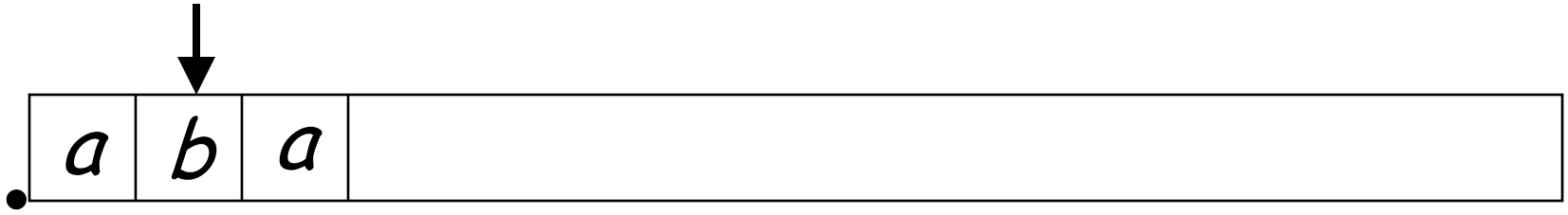


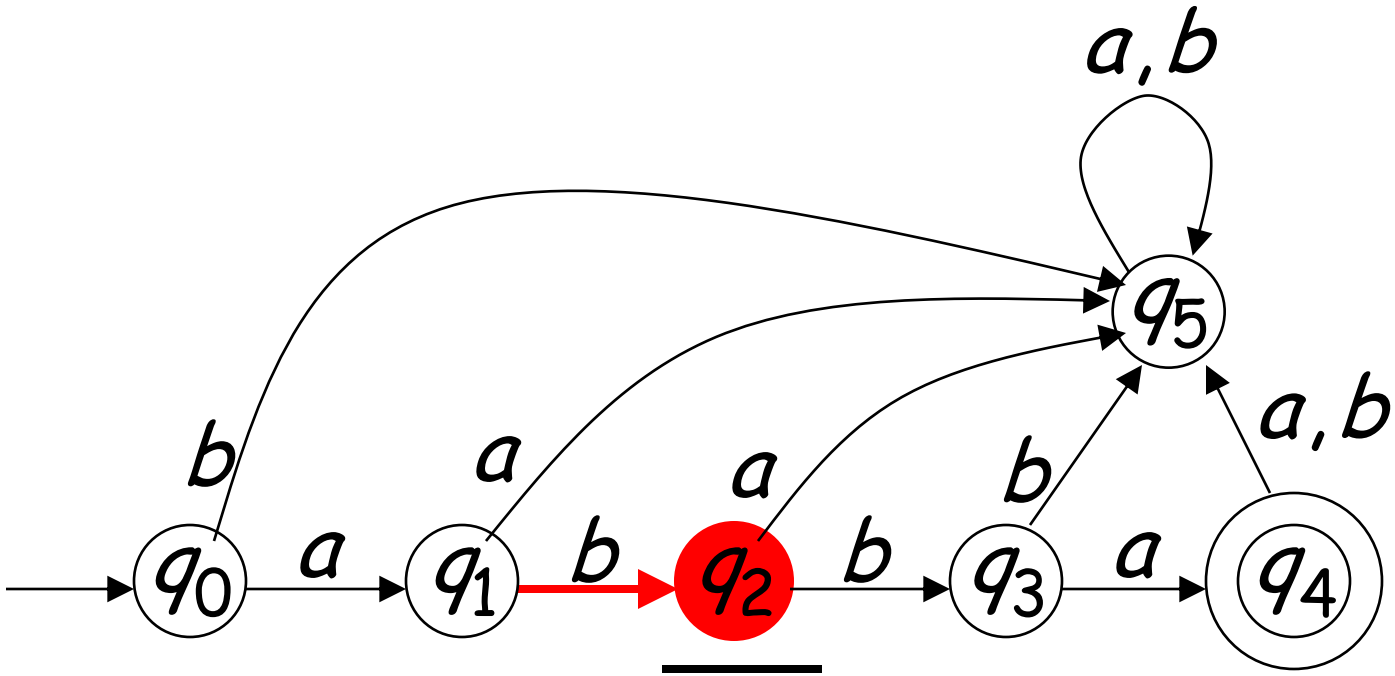
Output: "accept"

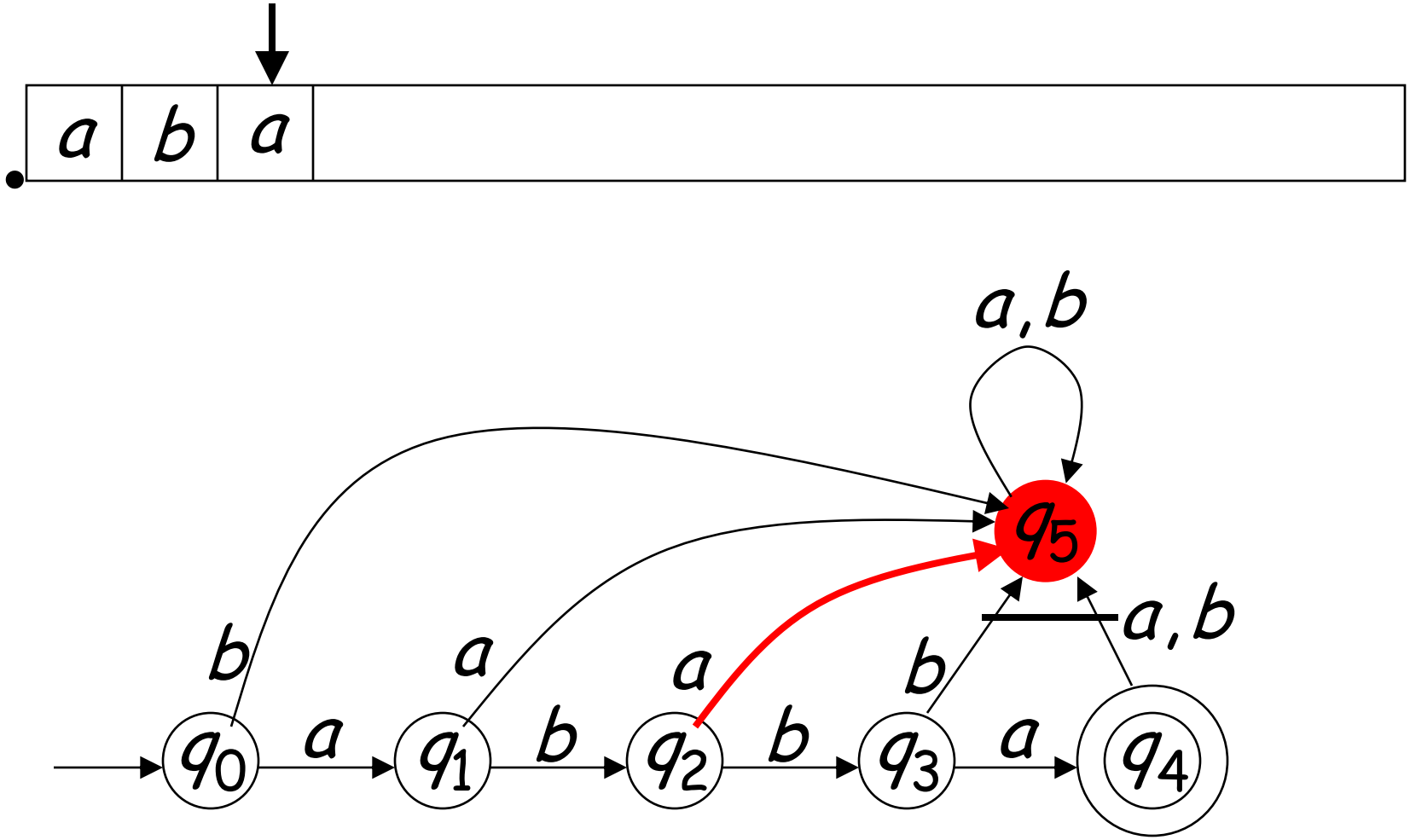
Rejection

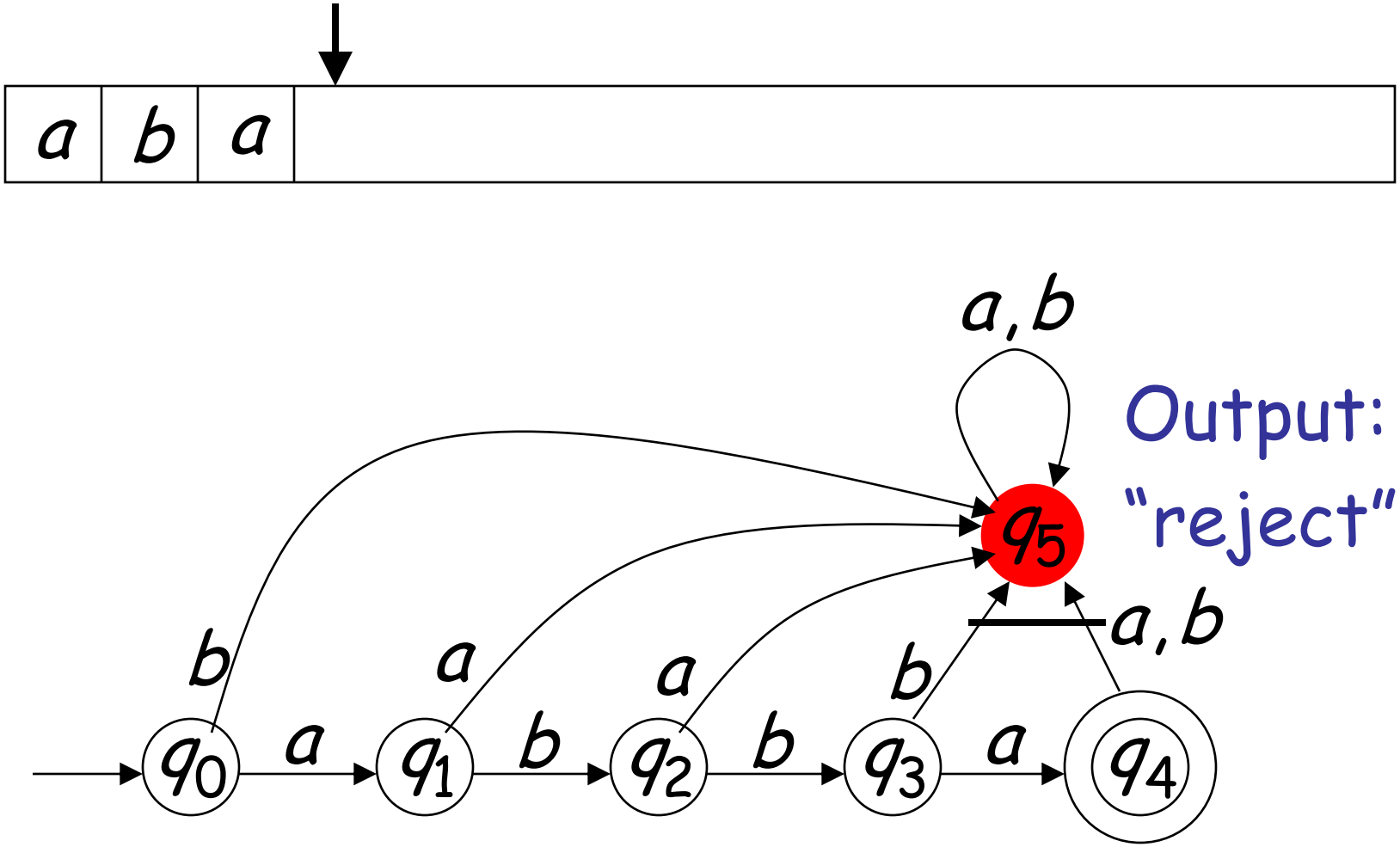



a b a
a, b
q5
b
a
a
b
a, b
q0
a
q1
b
q2
b
q3
a
q4

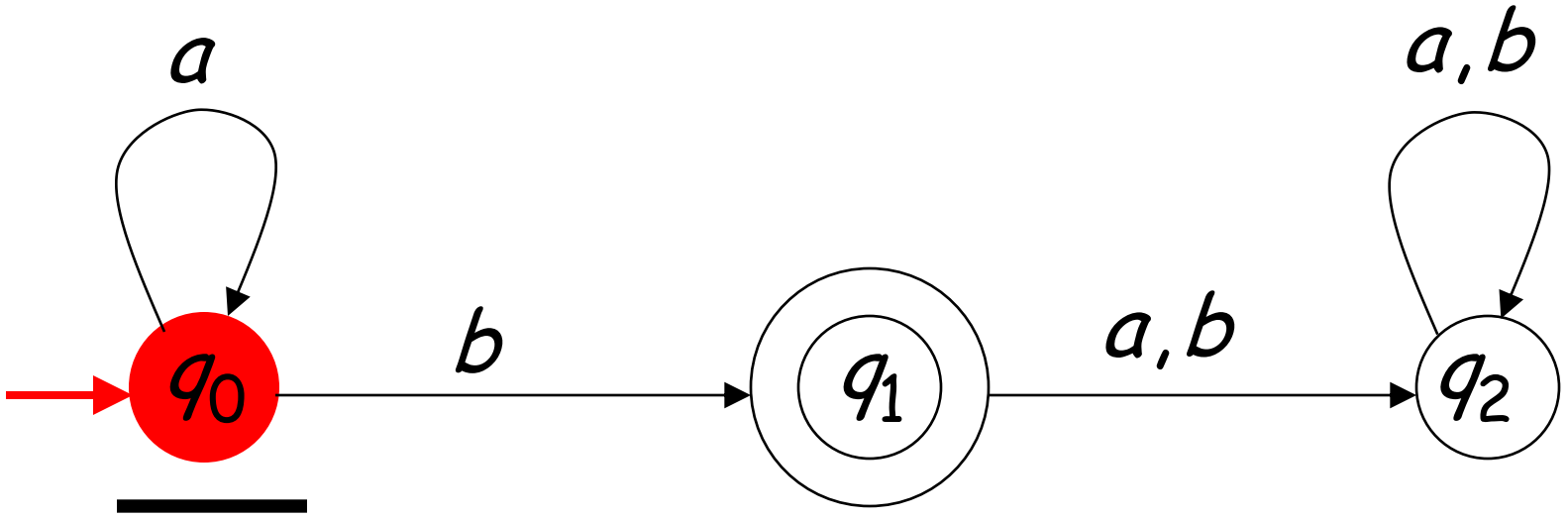

a
b
a

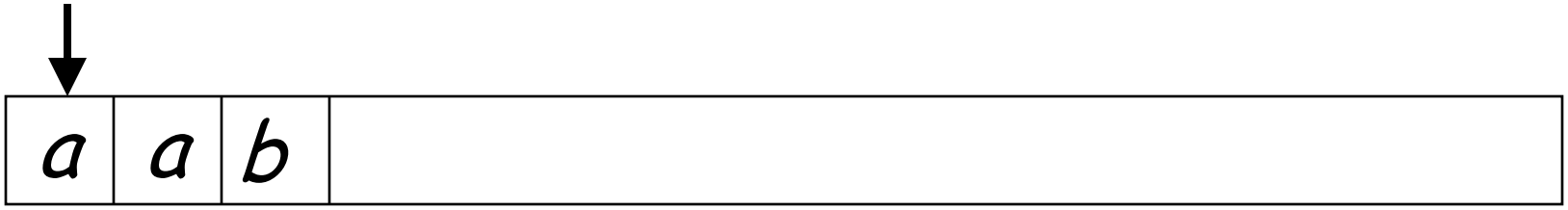

a, b
q5
b
a
a
b
a, b
q0
a
q1
b
q2
b
q3
a
q4

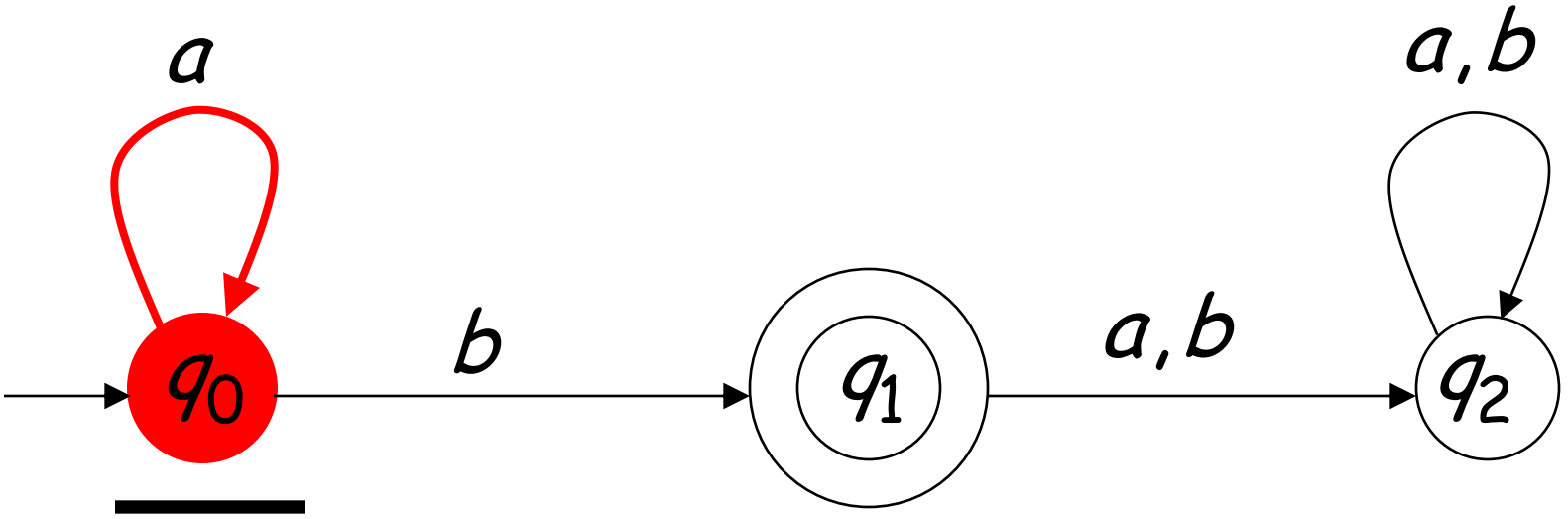

a b a
a, b
q5
b
a
a
b
a, b
q0
a
q1
b
q2
b
q3
a
q4

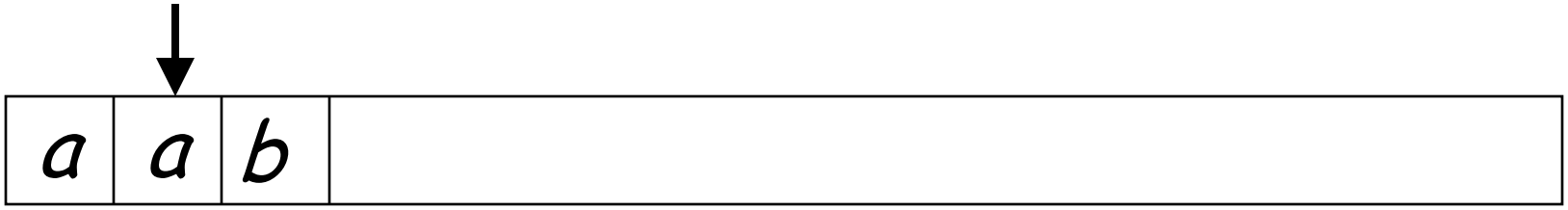

a b a
a, b
Output:
"reject"
q5
a, b
b
a
a
b
q0
a
q1
b
q2
b
q3
a
q4

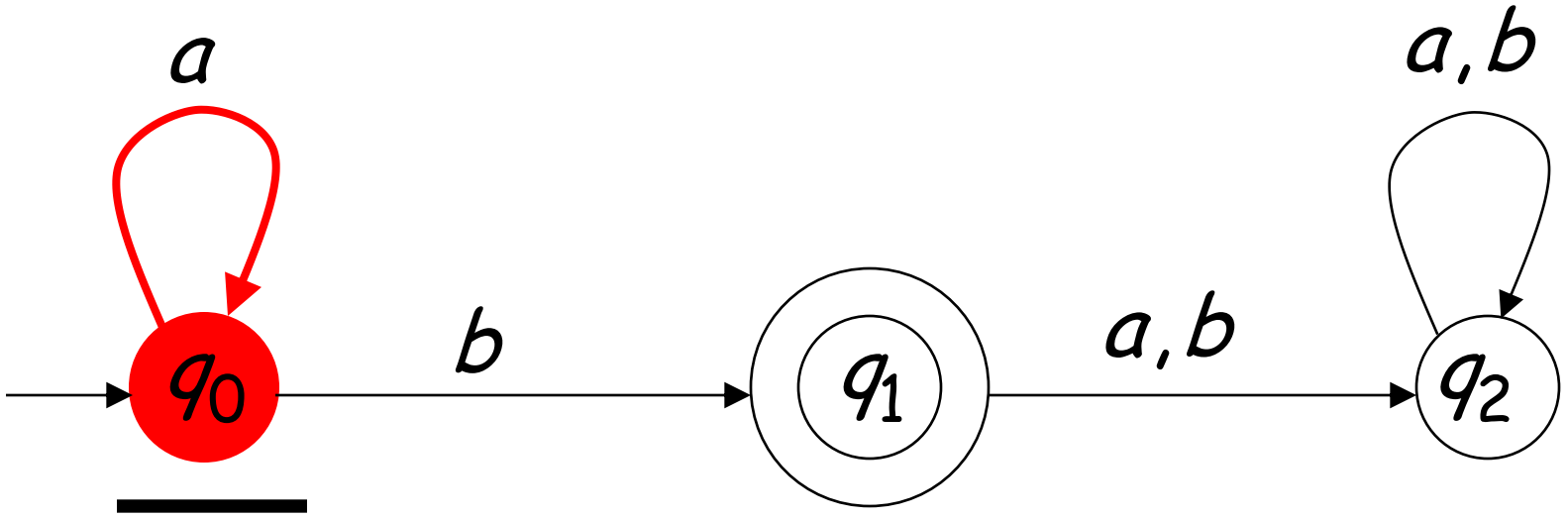
Another Example

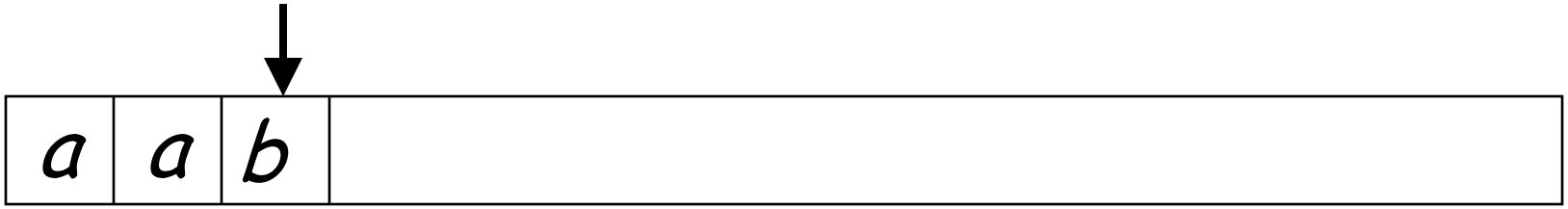


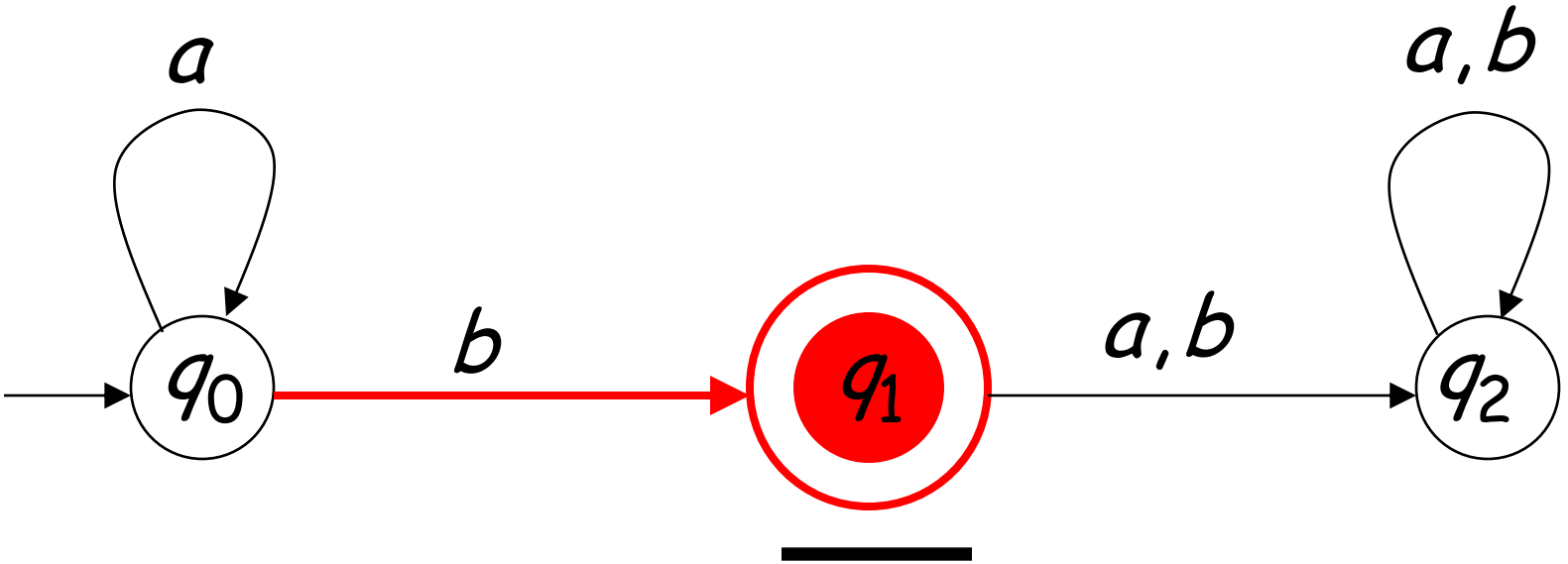

a
a
b

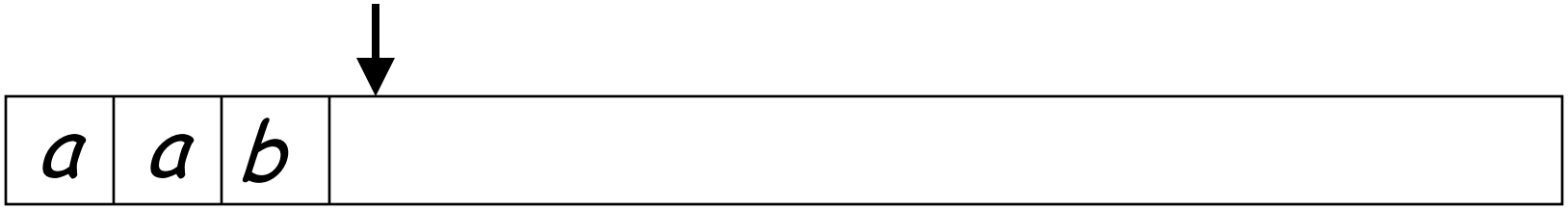

a
a, b
b
a, b
q0
q1
q2

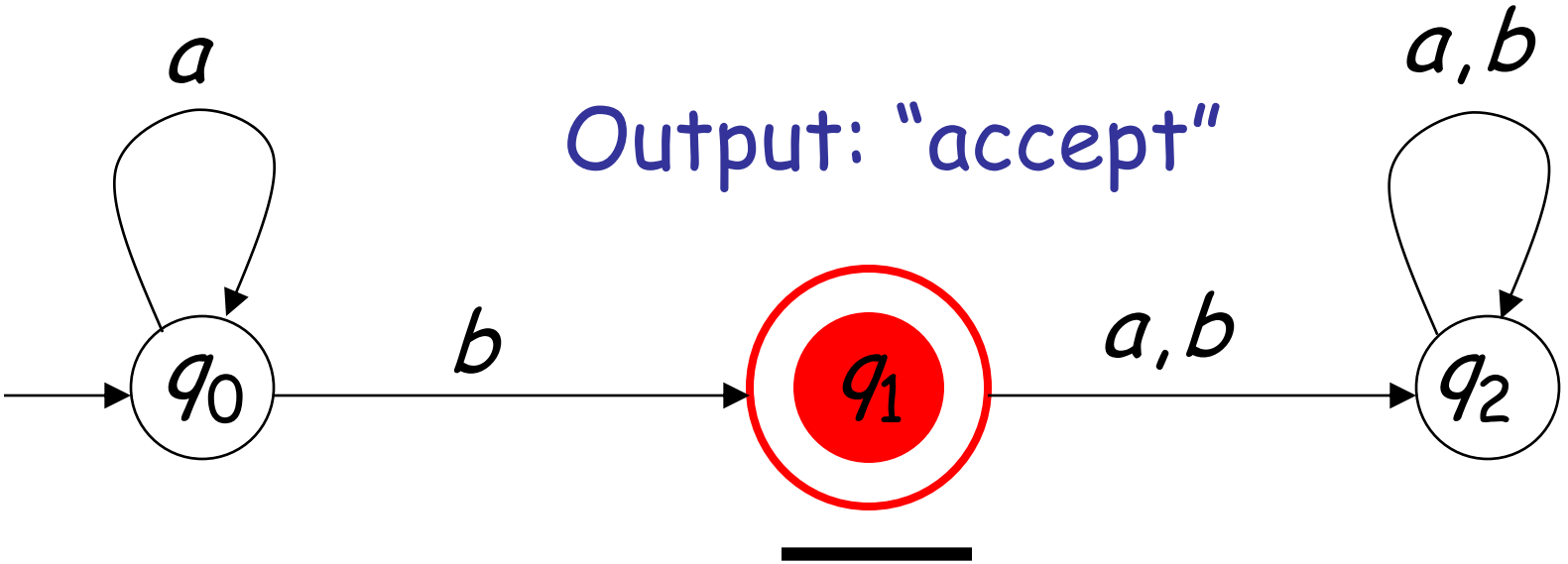

a
a
b


a
a, b
b
a, b
q0
q1
q2

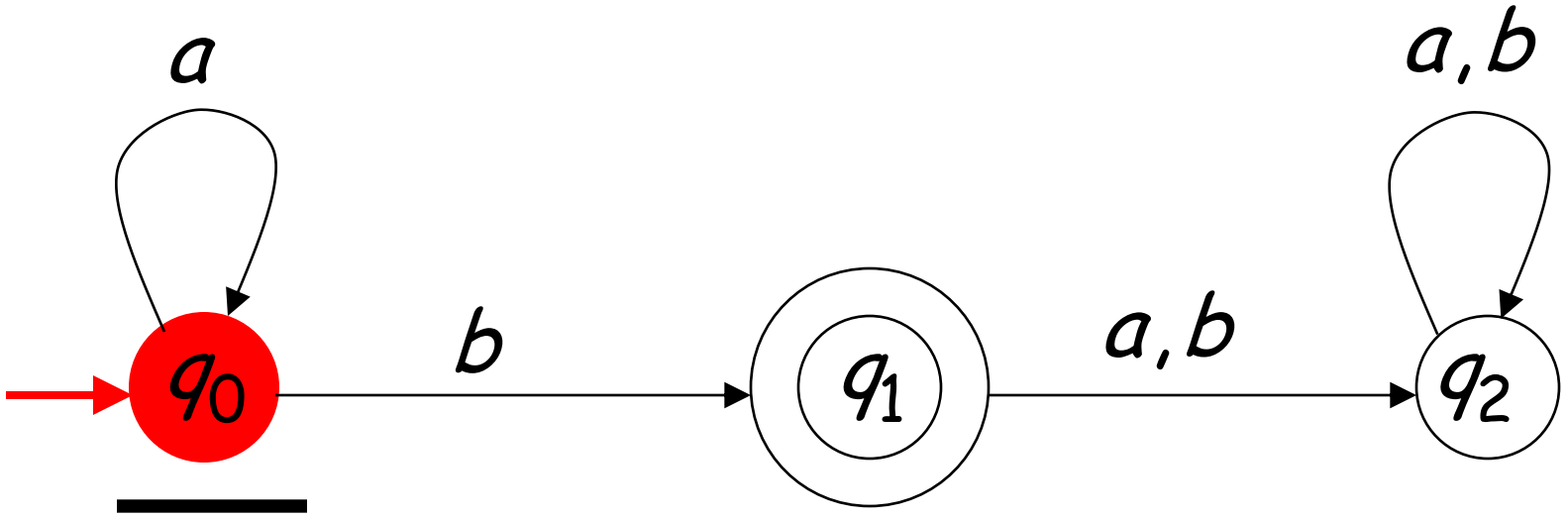

a
a
b

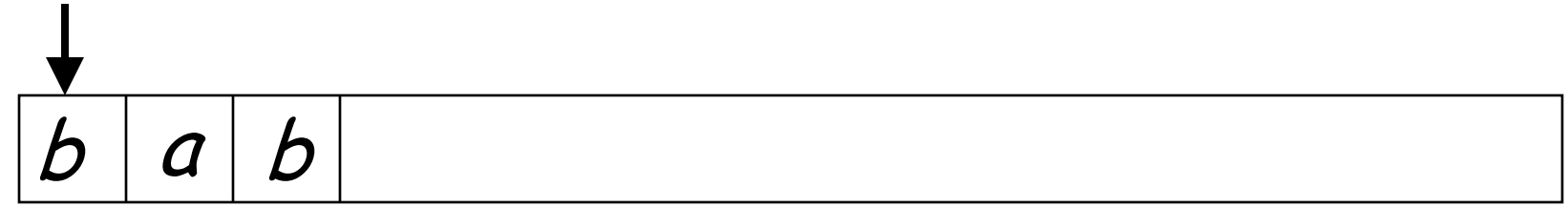

a
a, b
b
a, b
q0
q1
q2

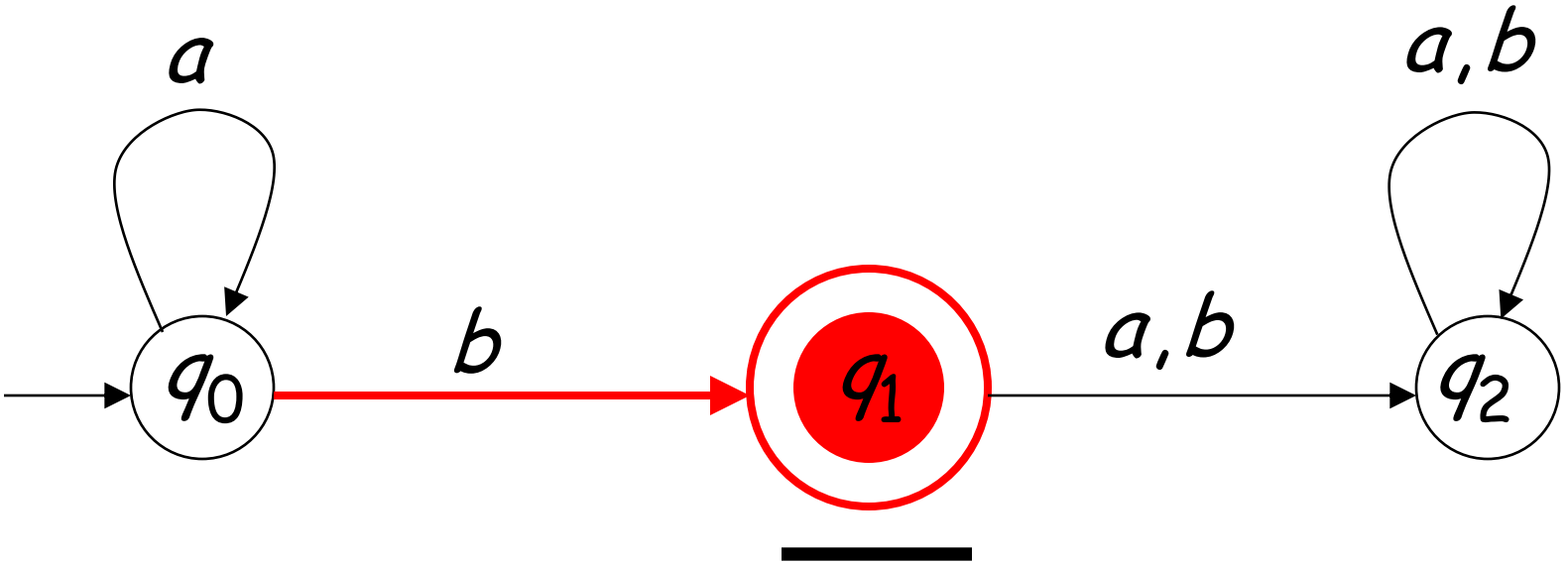

a
a
b

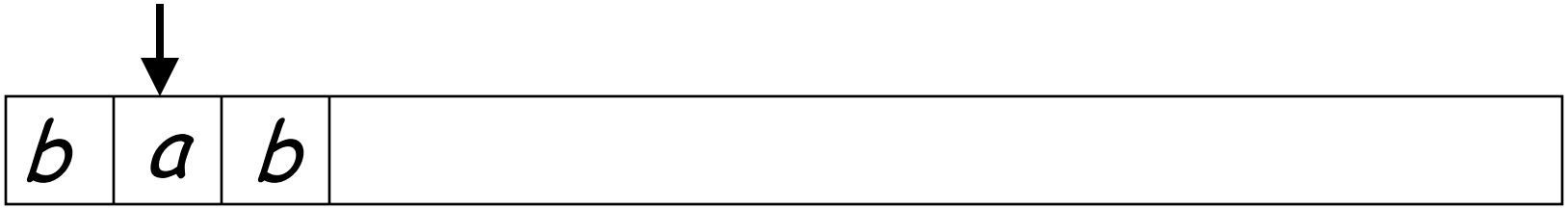

a
a, b
Output: "accept"
b
a, b
q0
q1
q2

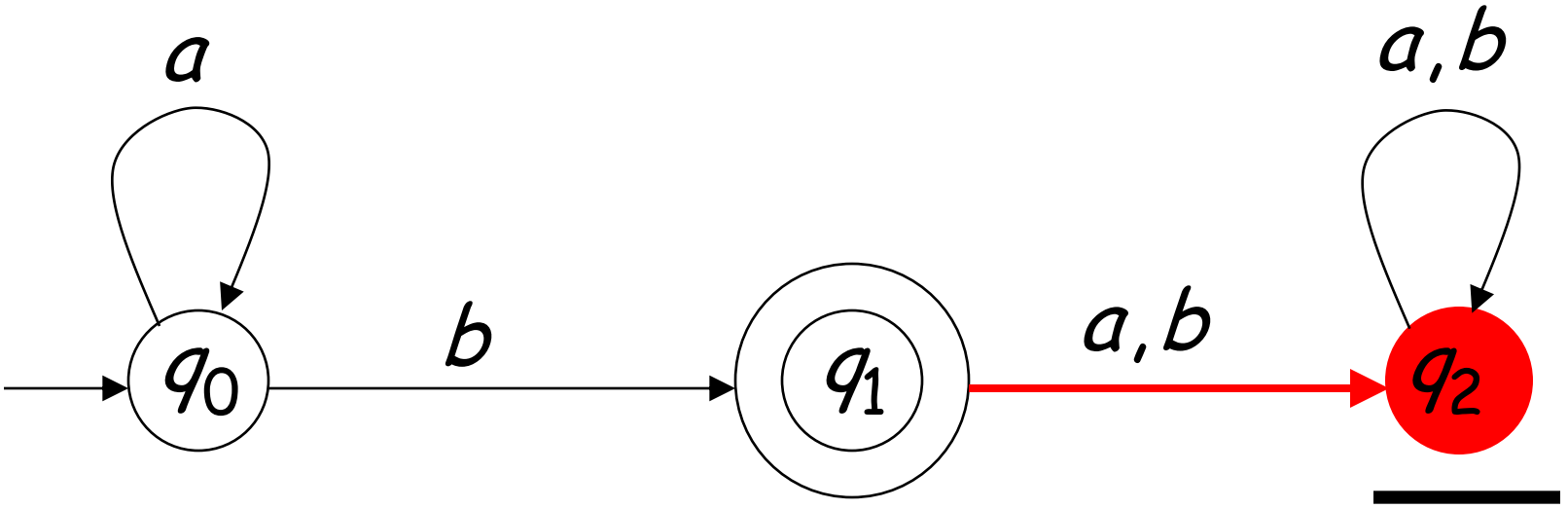
Rejection

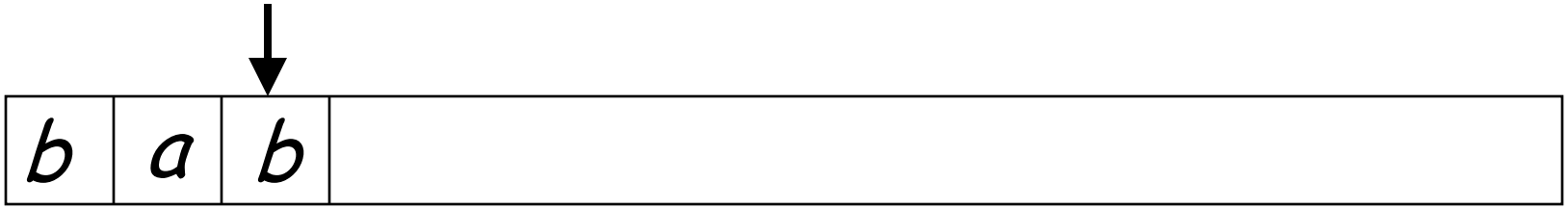


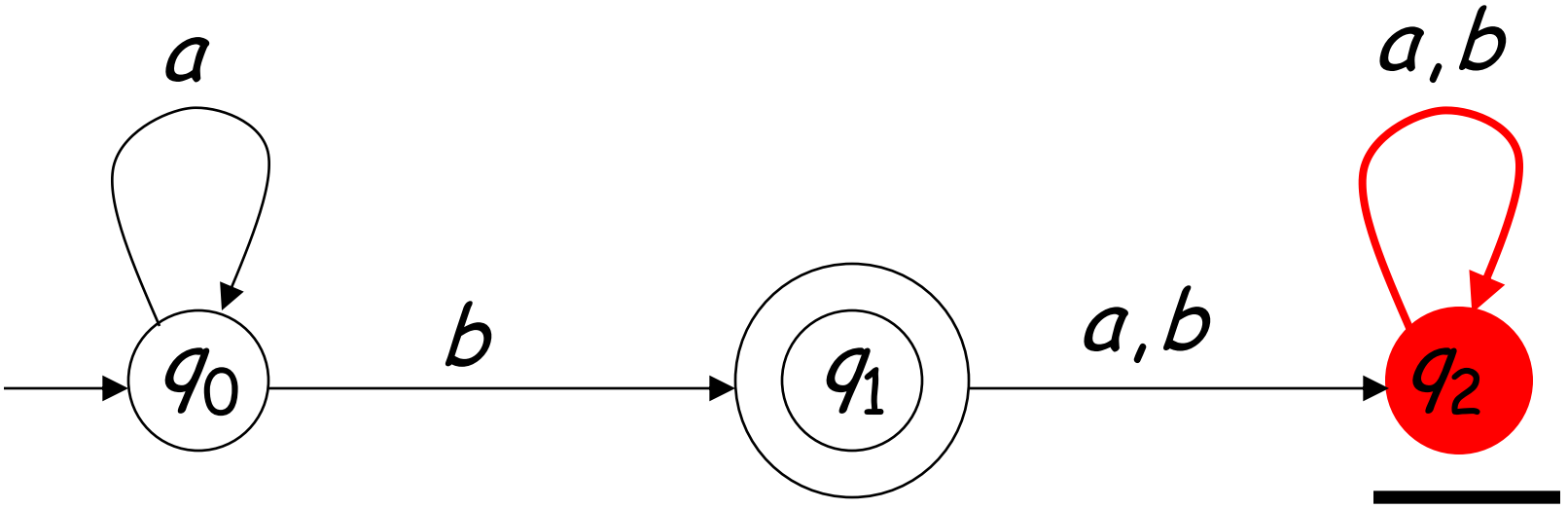

b
a
b

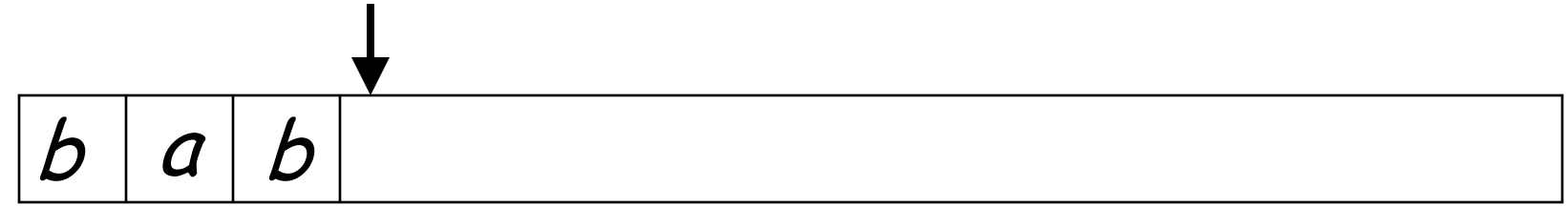

a
a, b
b
a, b
q_0
q_1
q_2

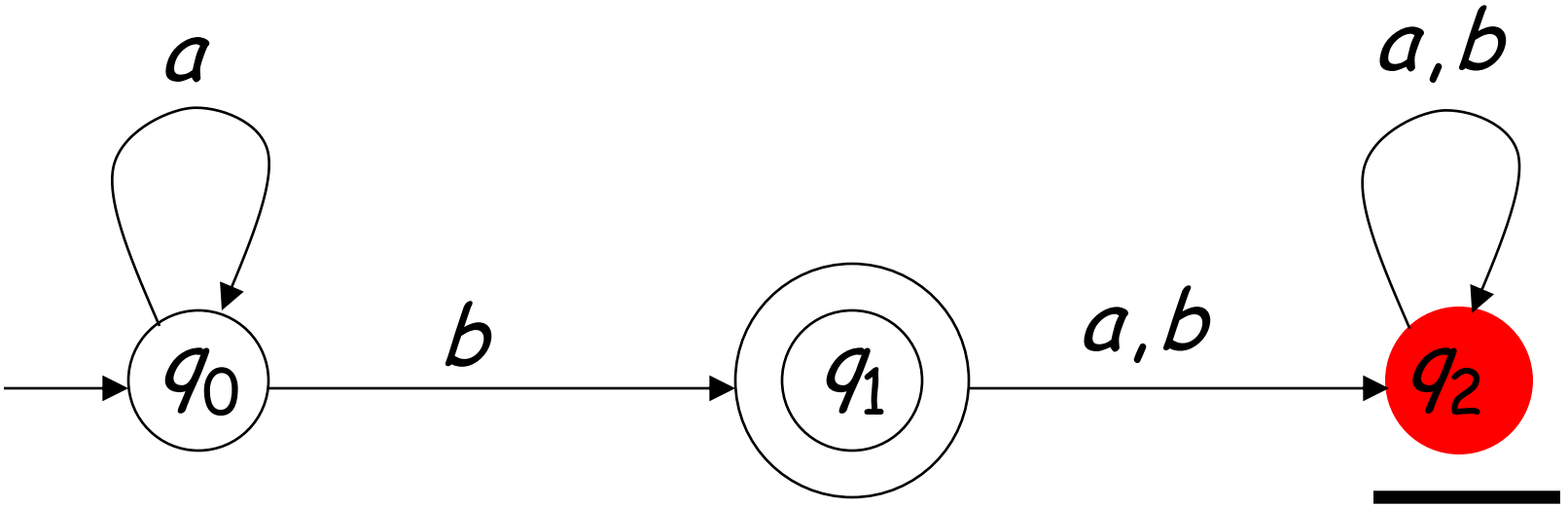

b
a
b


a
a, b
b
a, b
q_0
q_1
q_2


b
a
b


a
a, b
b
a, b
q0
q1
q2





Output: "reject"

Formalities

- Deterministic Finite Accepter (DFA)

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

Q : set of states

Σ : input alphabet

δ : transition function

q_0 : initial state

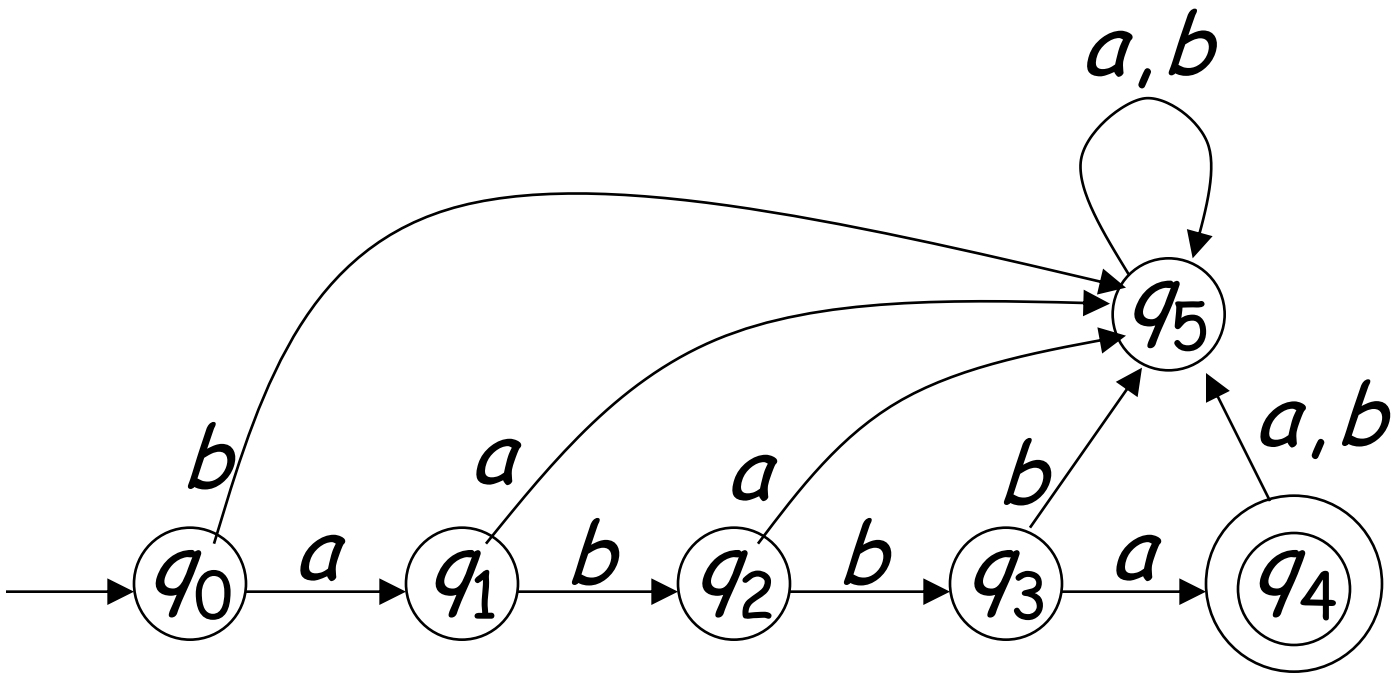
F : set of final states

About Alphabets

- Alphabets means we need a finite set of symbols in the input.
- These symbols can and will stand for bigger objects that can have internal structure.

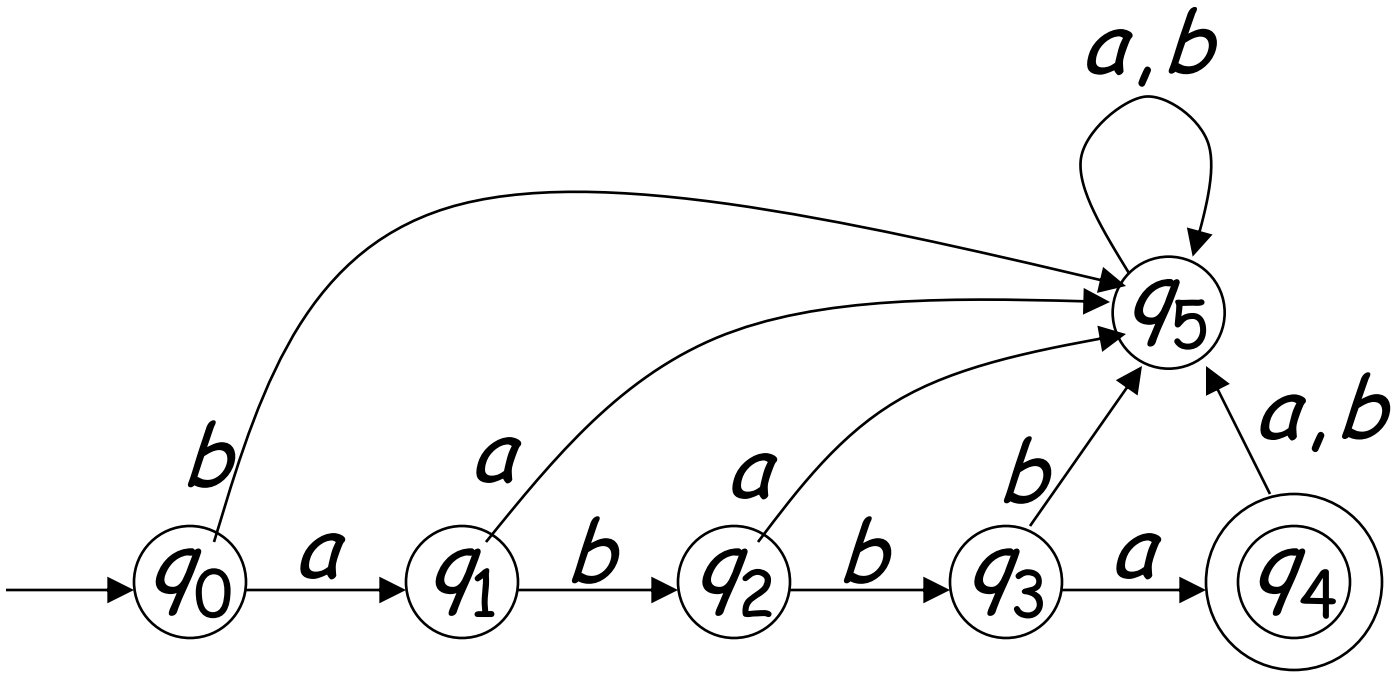
Input Aplhabet Σ

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

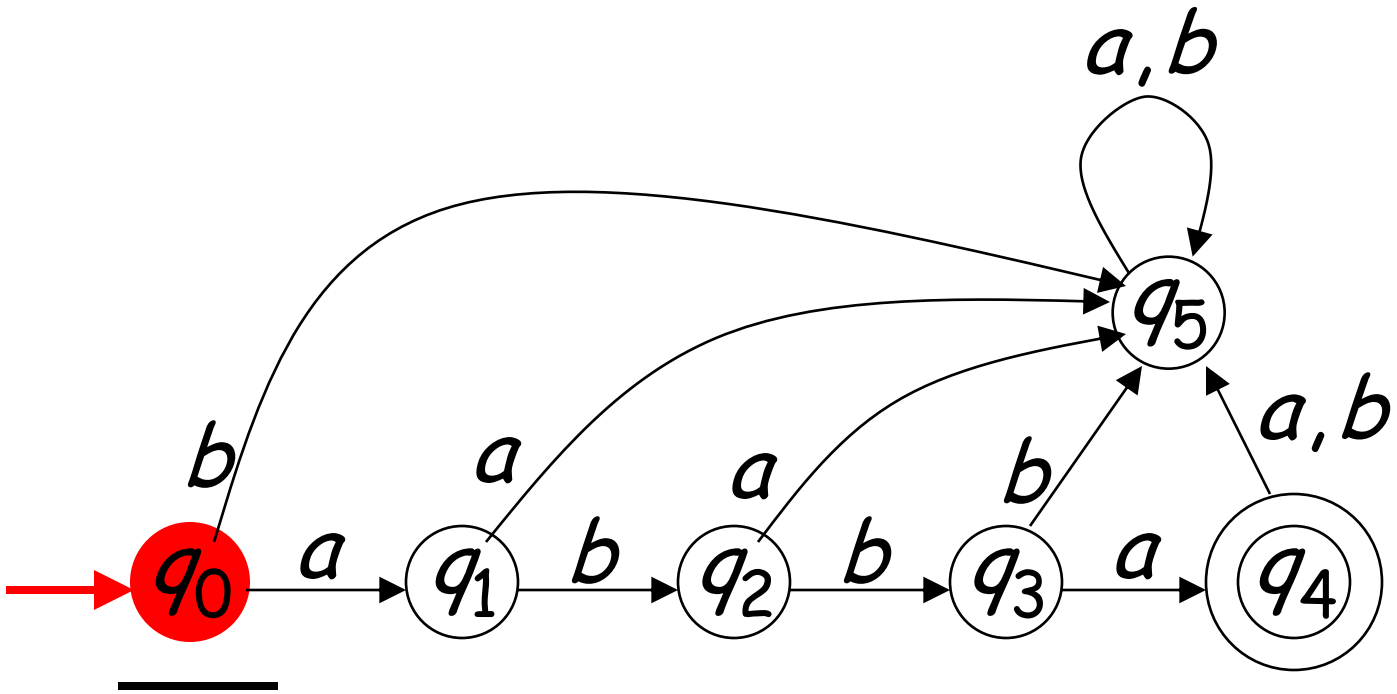


Set of States Q

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

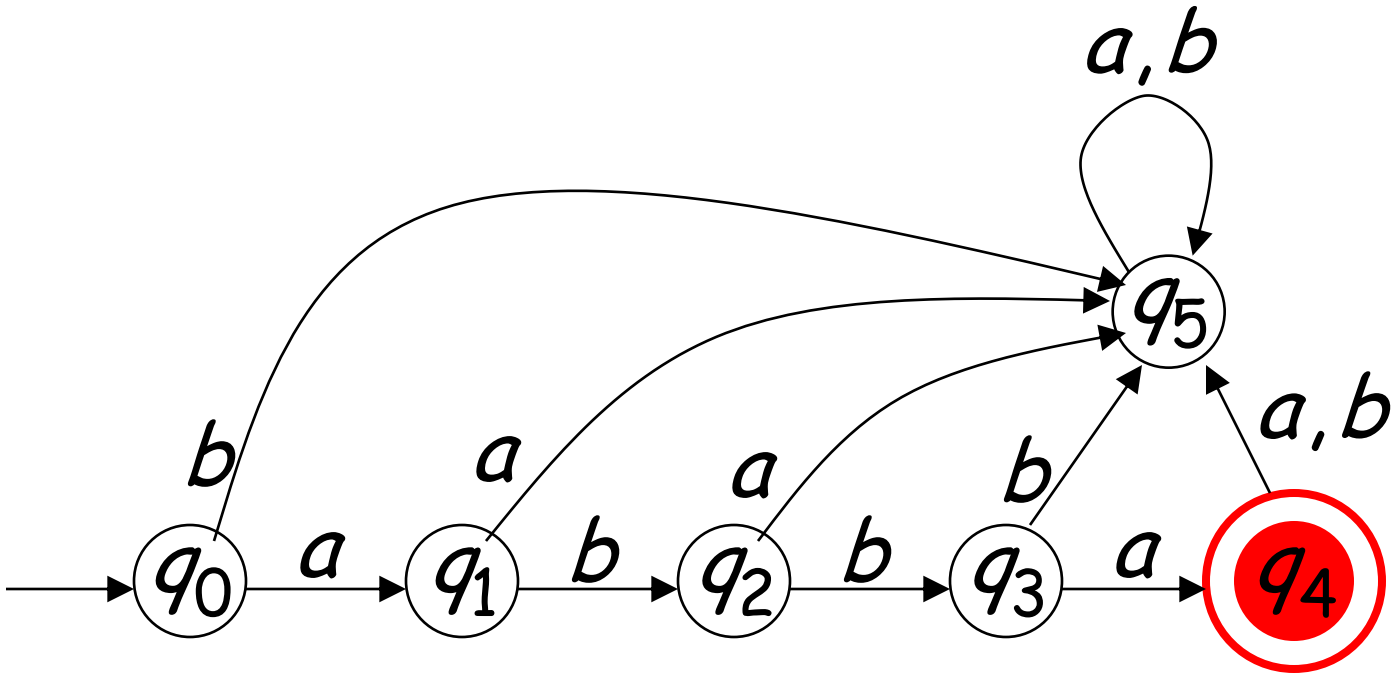


Initial State q_0



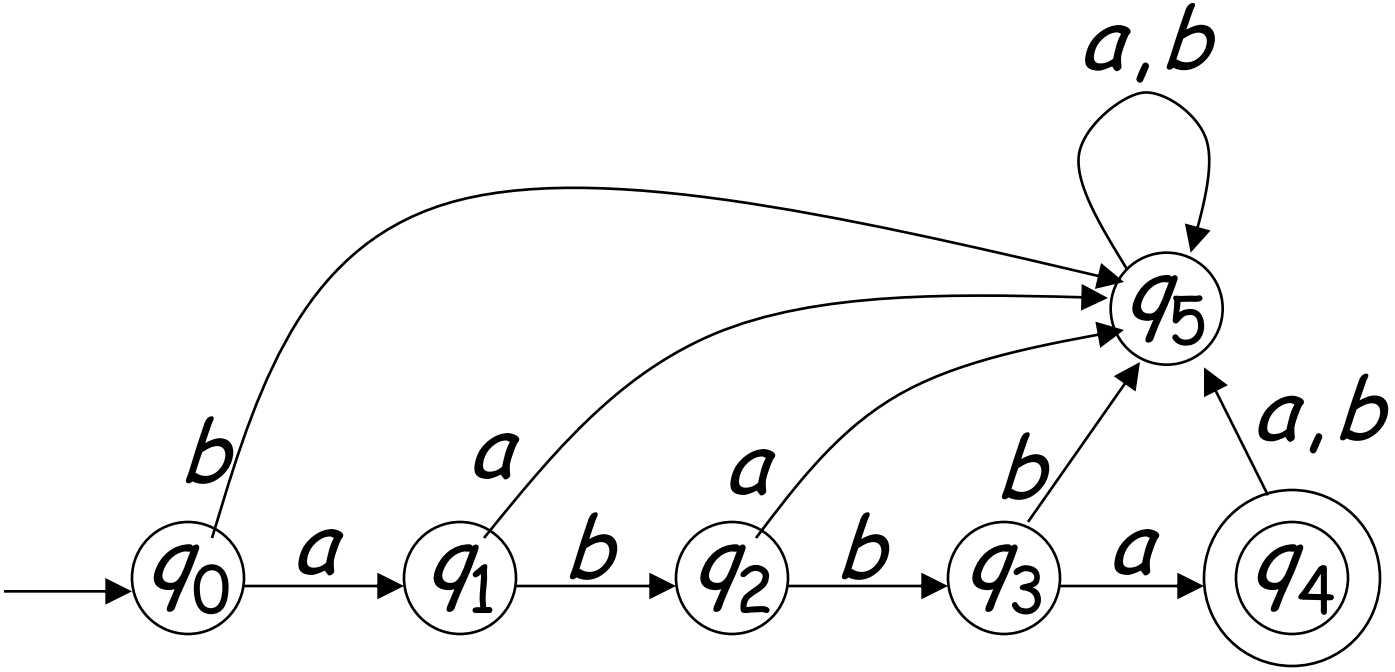
Set of Final States F

$$F = \{q_4\}$$

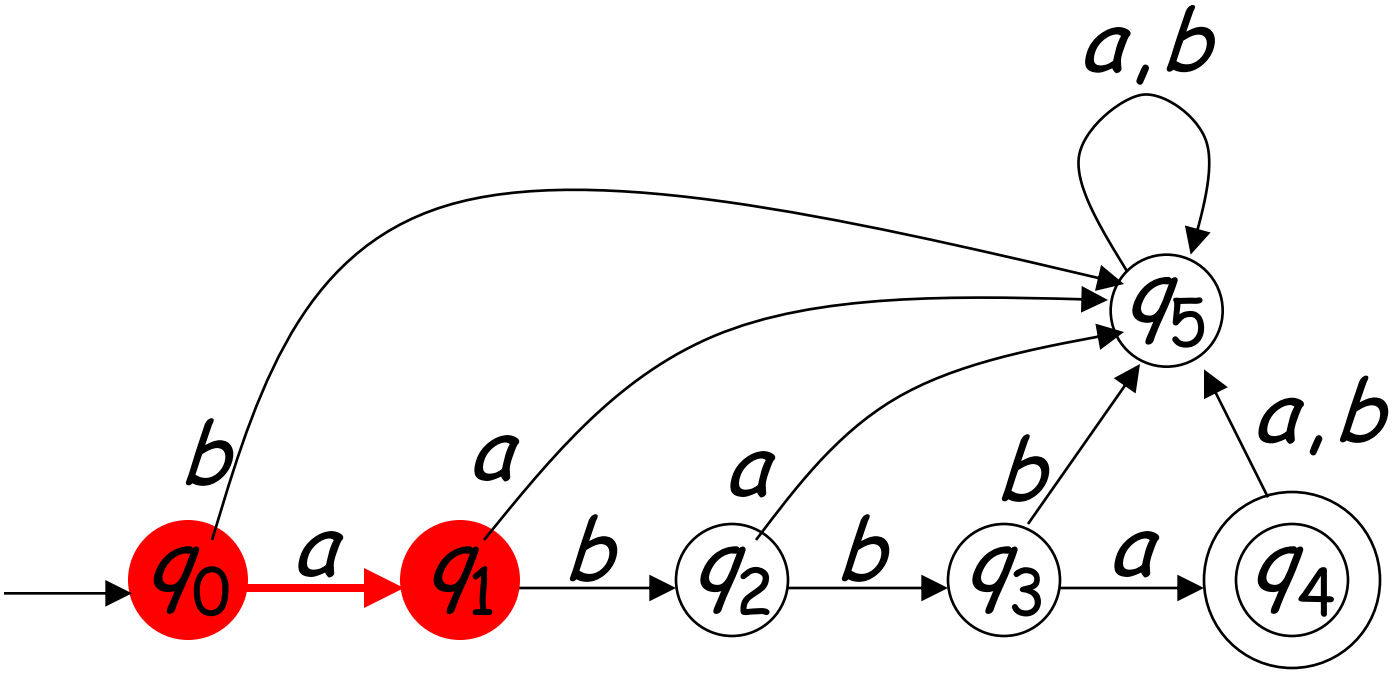


Transition Function δ

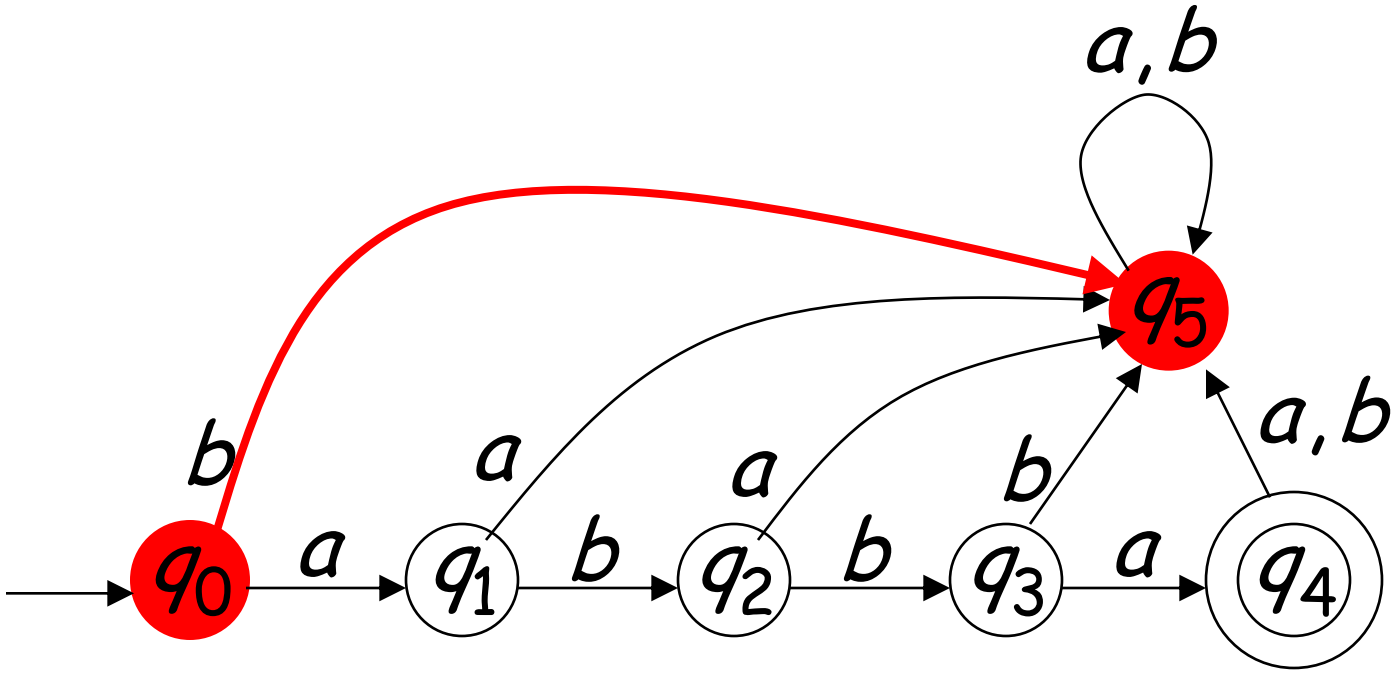
$$\delta : Q \times \Sigma \to Q$$



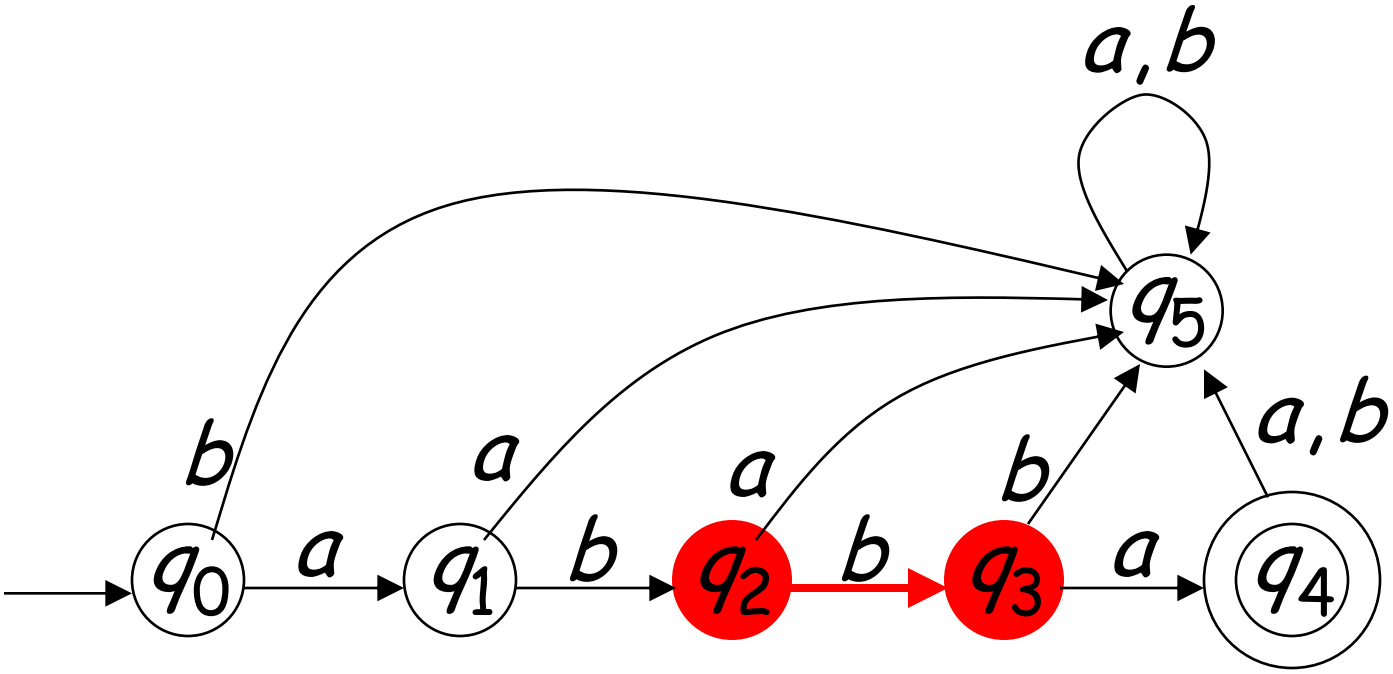
$$\delta(q_0, a) = q_1$$



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

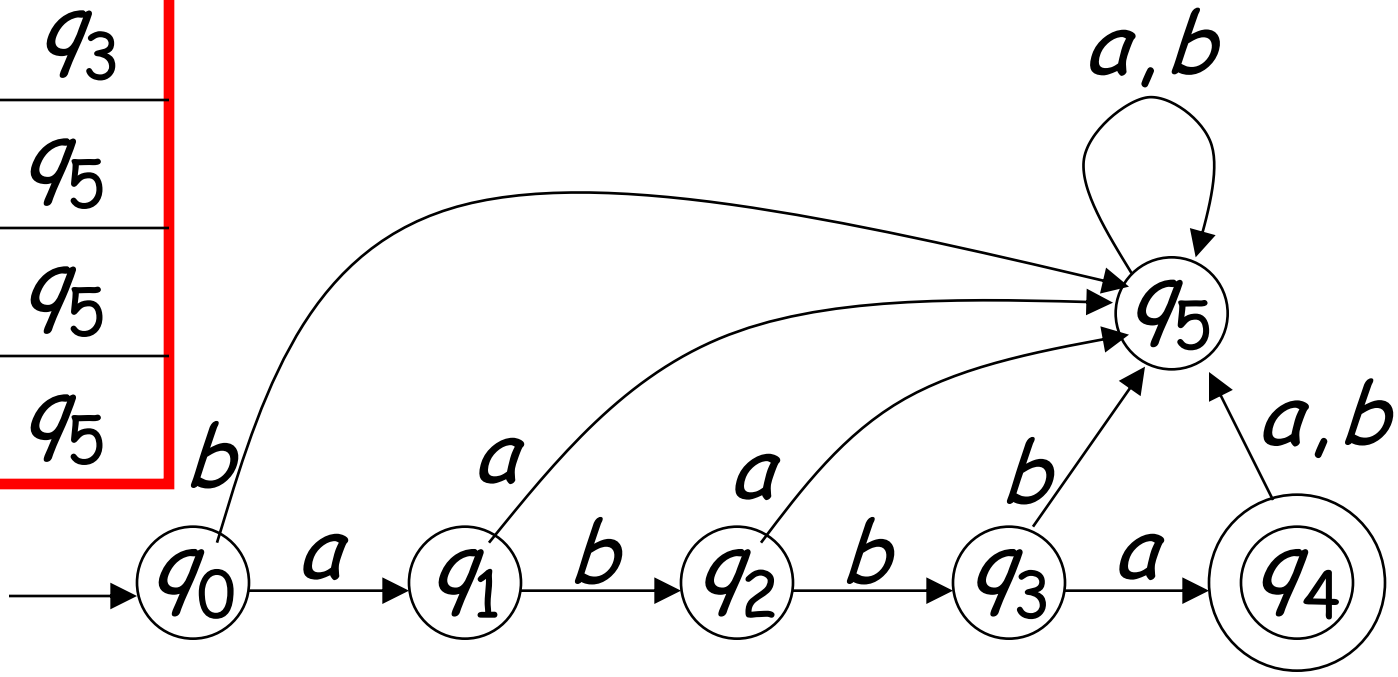


$$\delta(q_2, b) = q_3$$



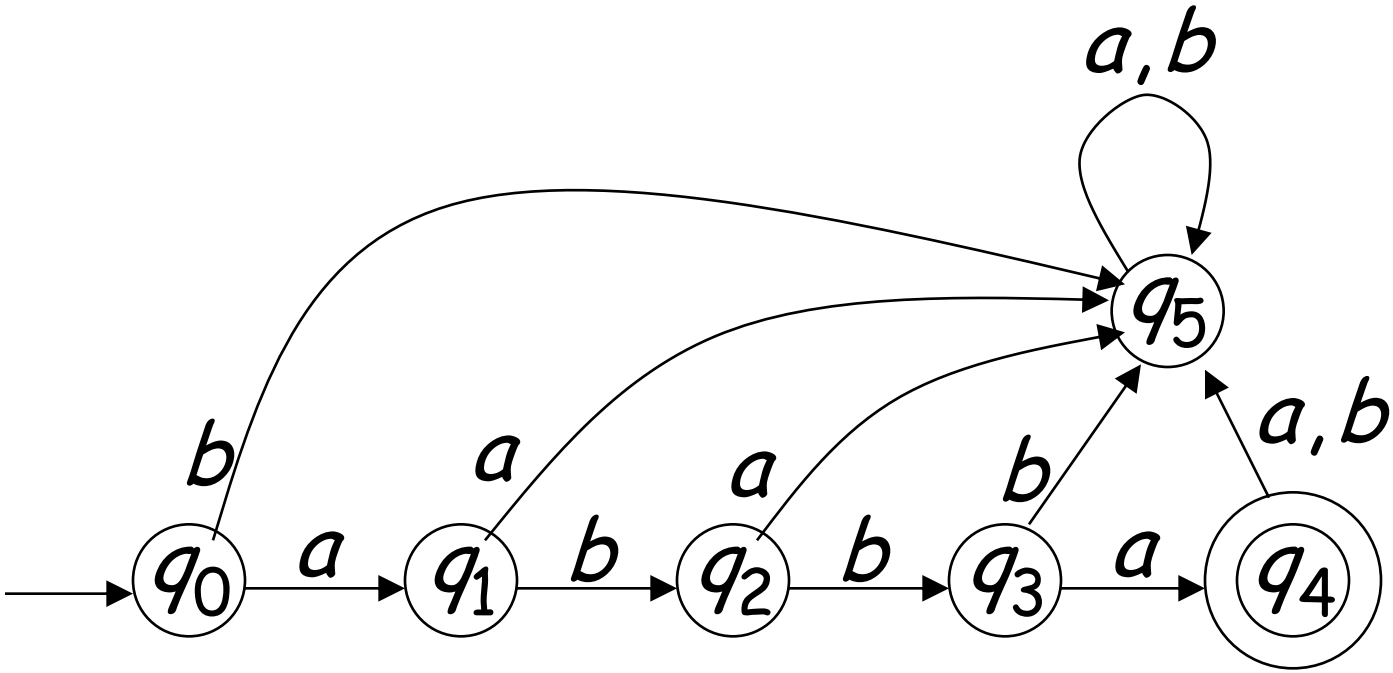
Transition Function δ

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

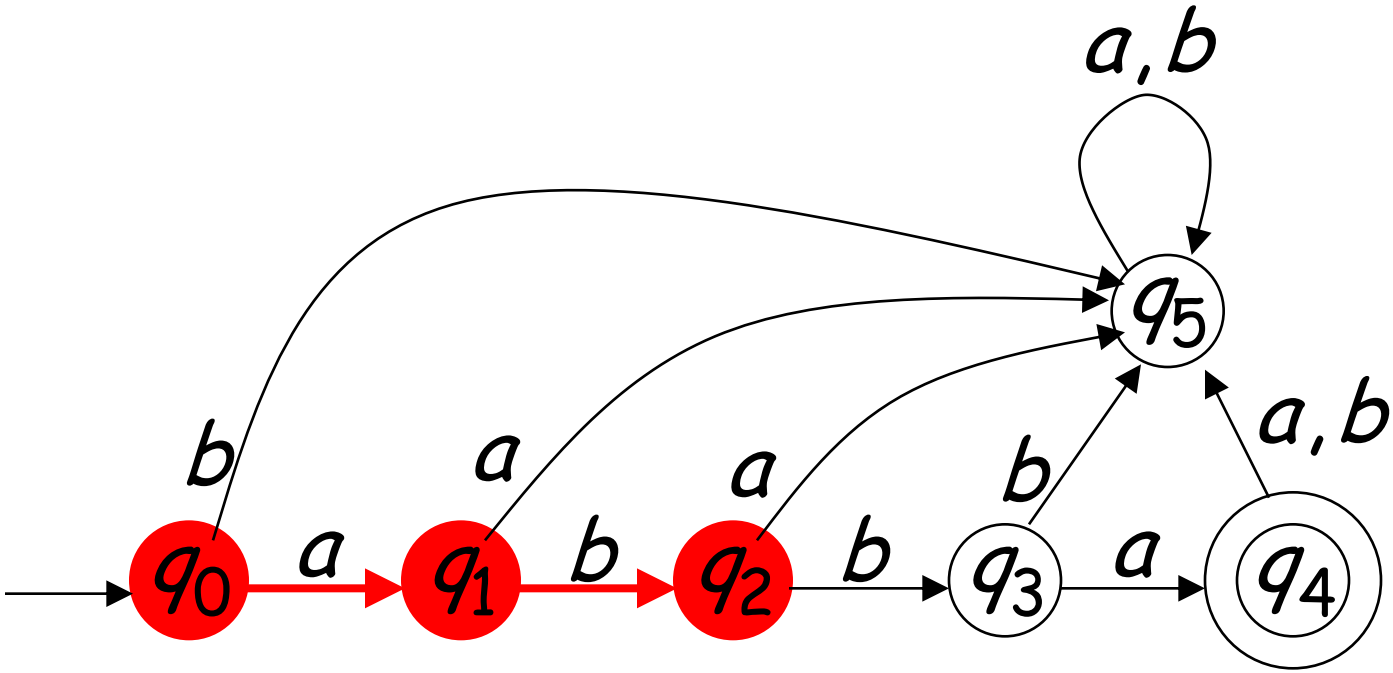


Extended Transition Function δ^* (Reads the entire string)

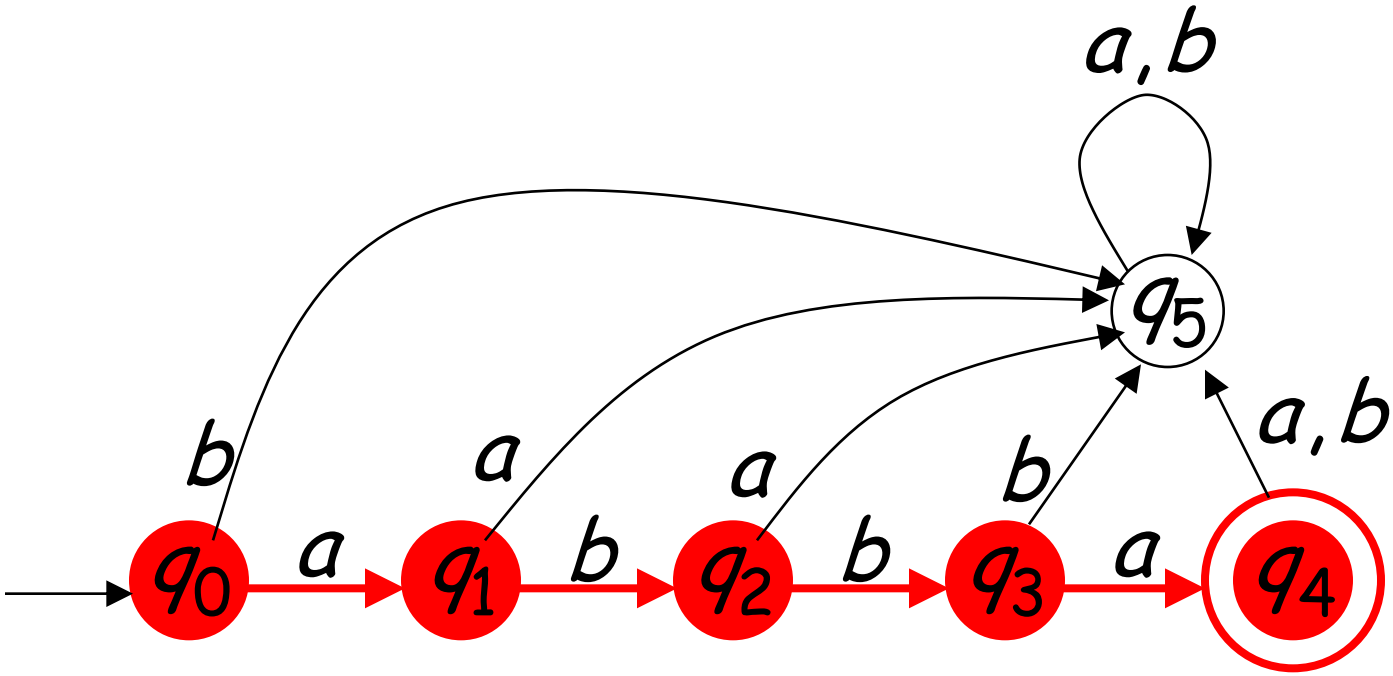
$$\delta^* : Q \times \Sigma^* \to Q$$



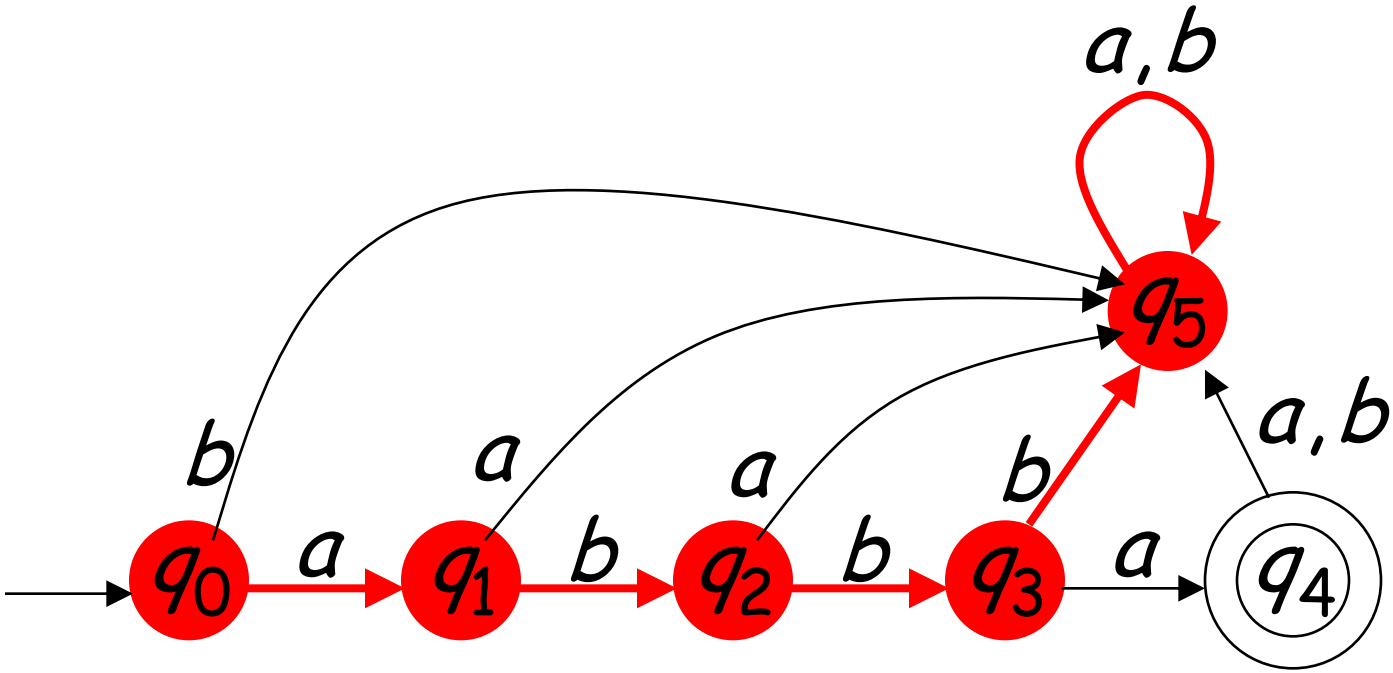
$$\delta^*(q_0, ab) = q_2$$



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

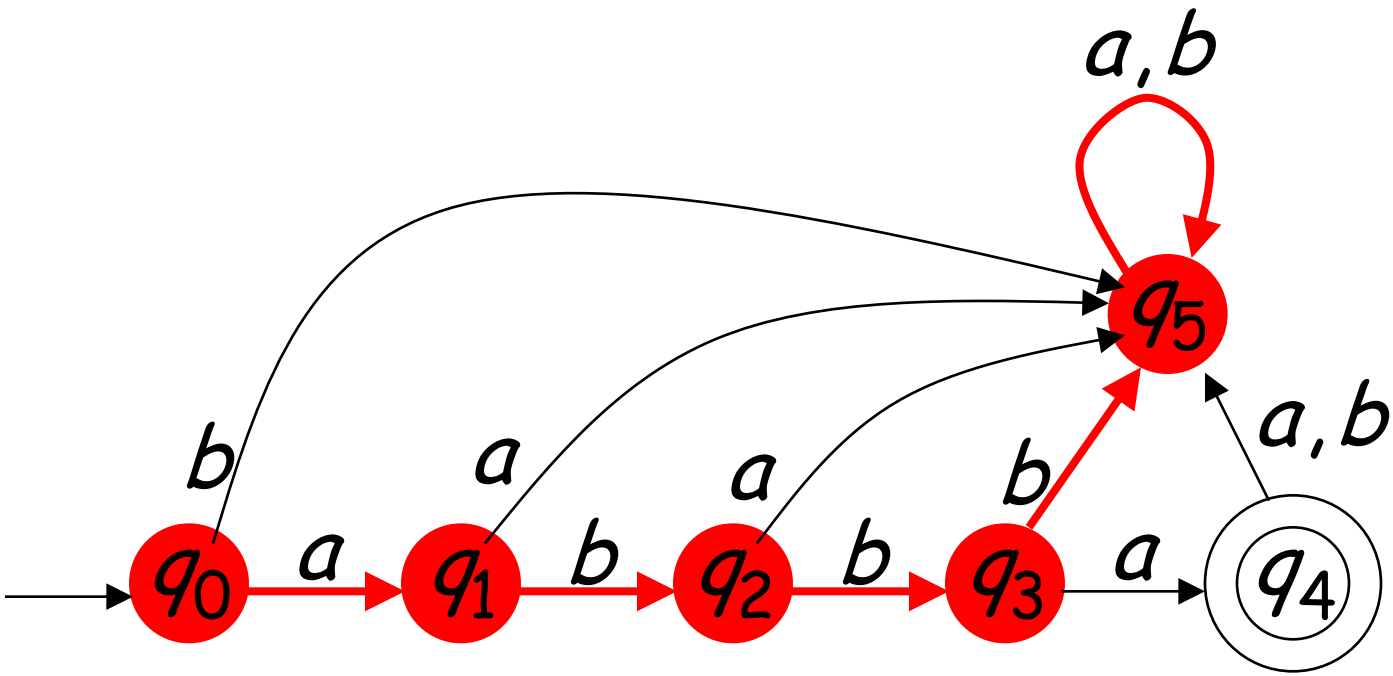


$$\delta^*(q_0, abbbaa) = q_5$$



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

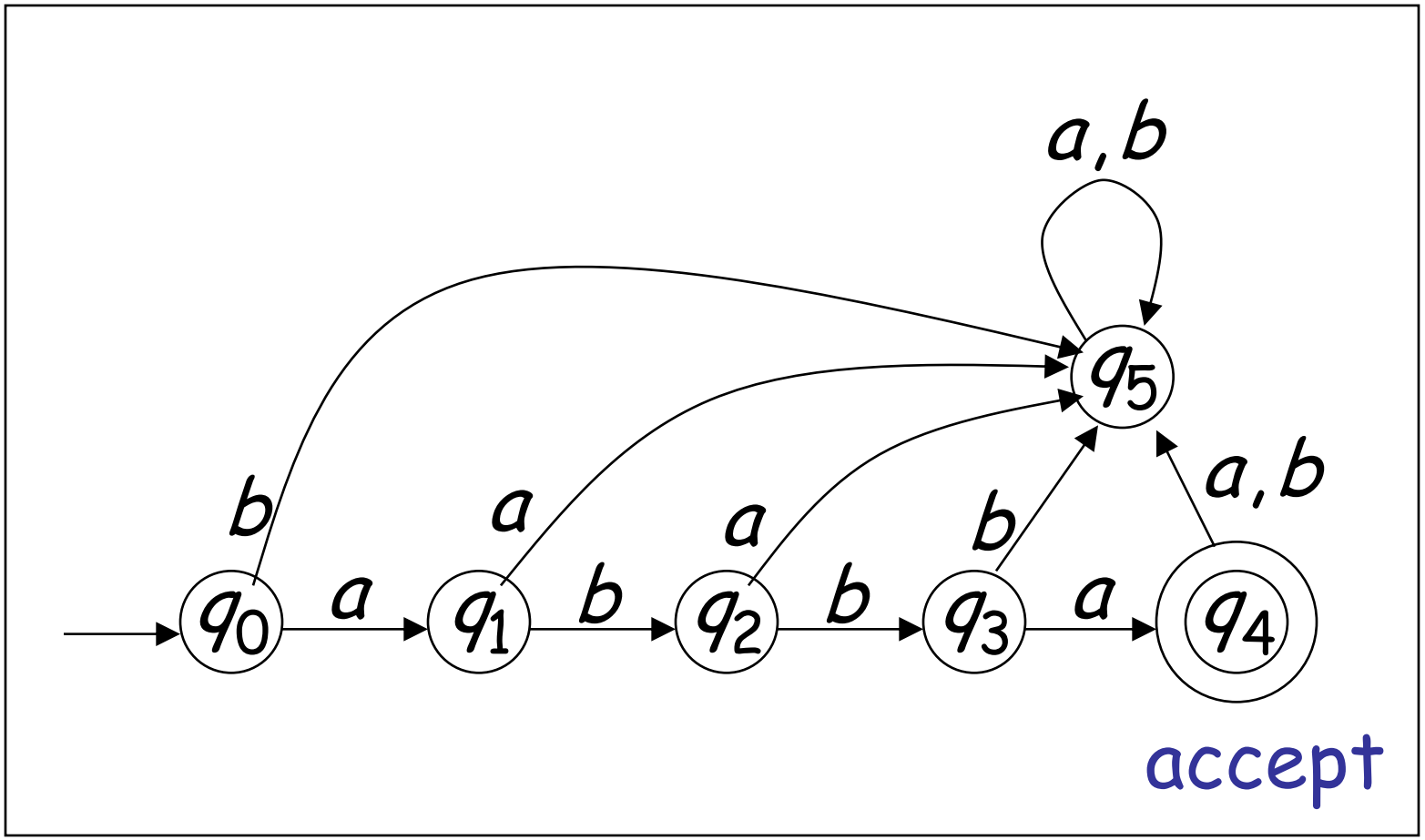
$$\delta^*(q_0, abbbaa) = q_5$$



Example

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

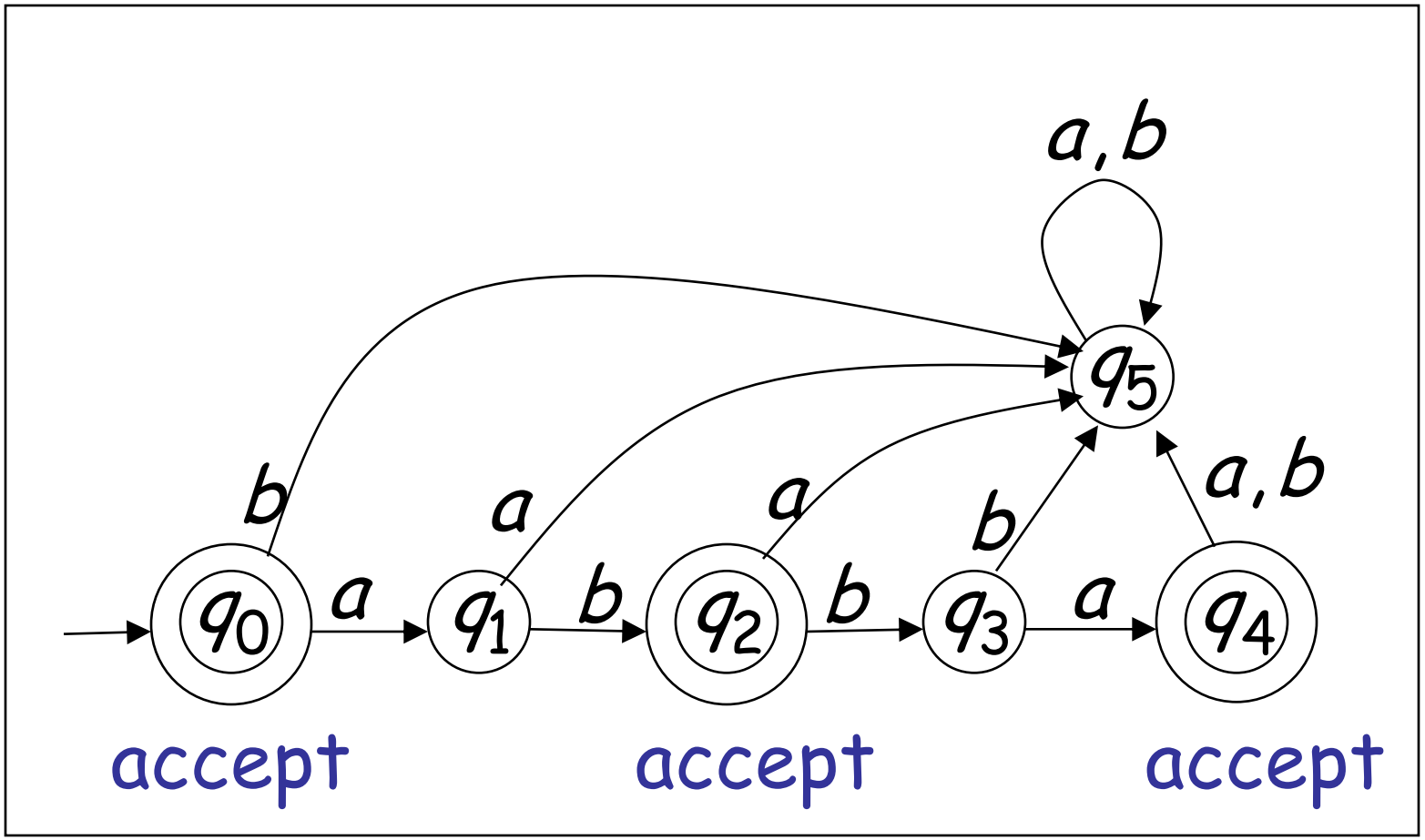
M



Another Example

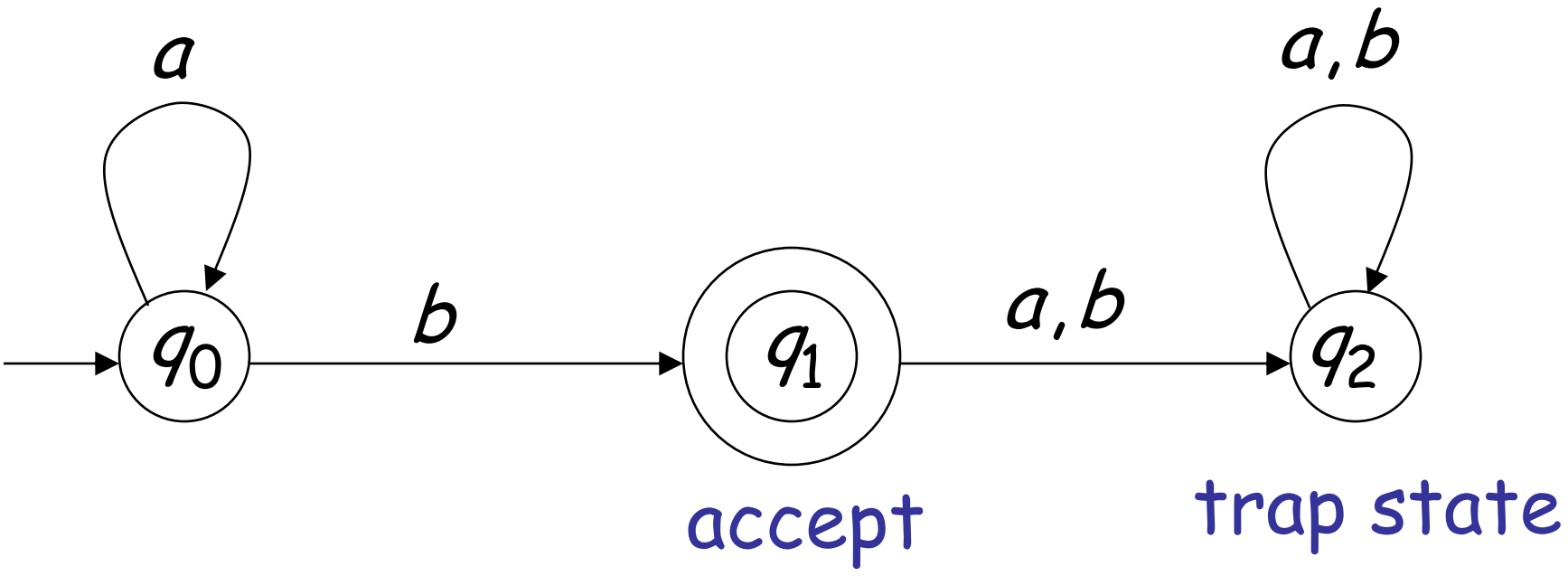
$L(M) = \{\lambda, ab, abba\}$

M

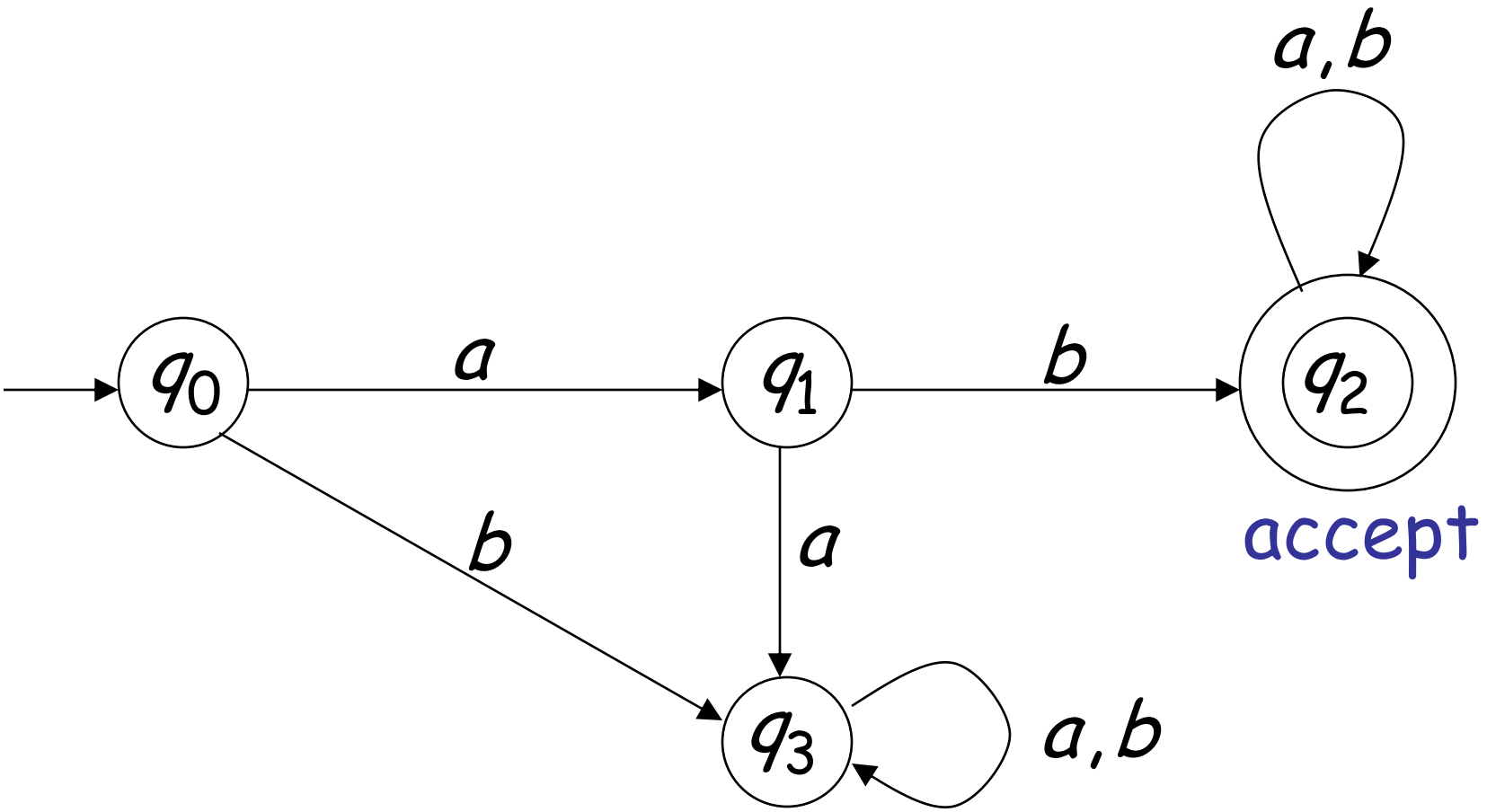


More Examples

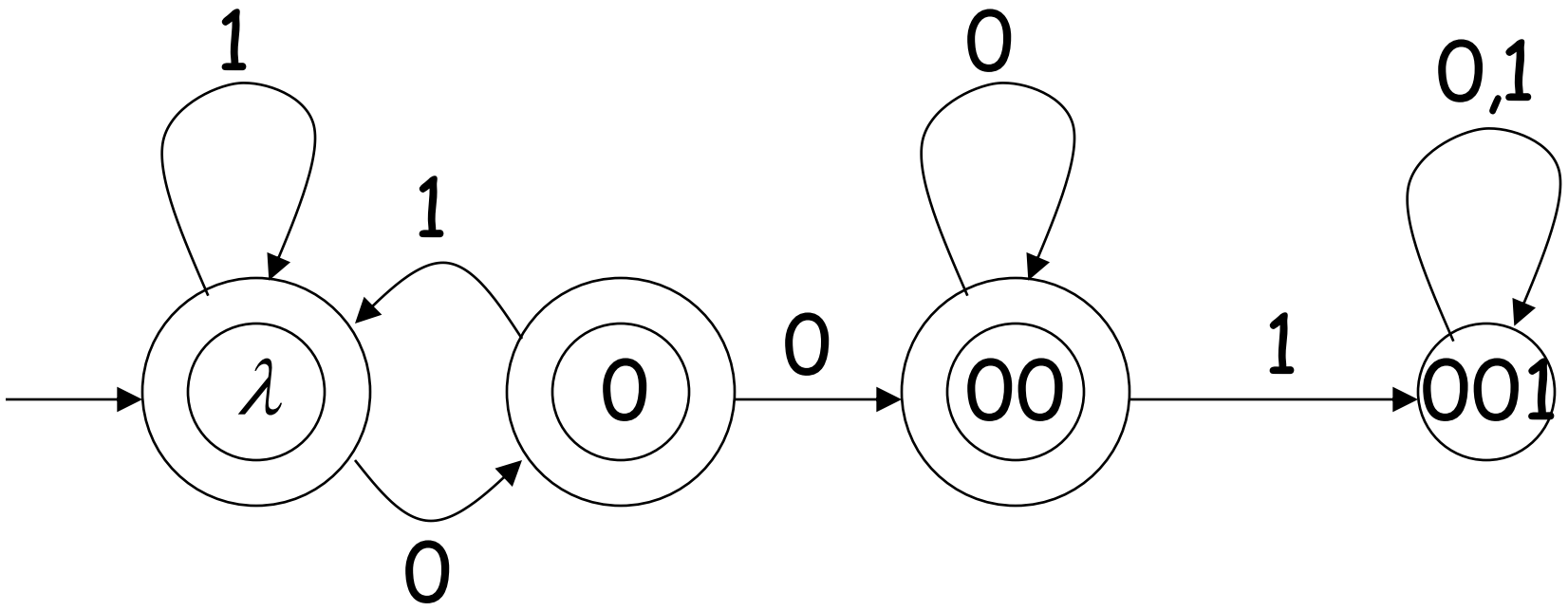
$$L(M) = \{a^n b : n \geq 0\}$$



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



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



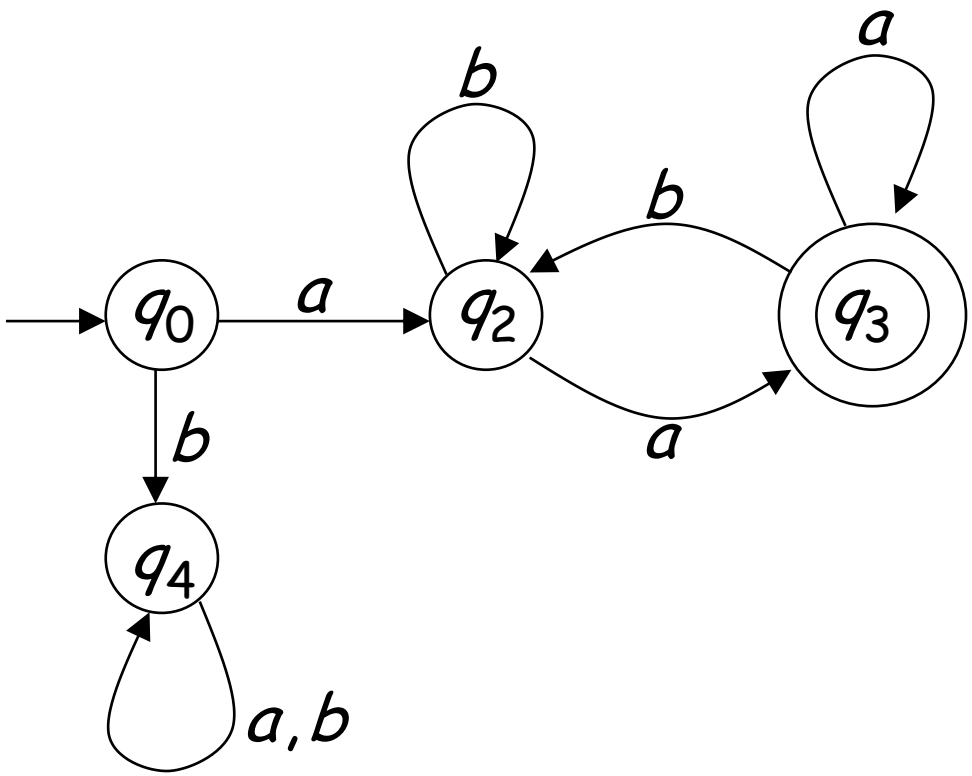
Regular Languages

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

- All regular languages form a language family

Example

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



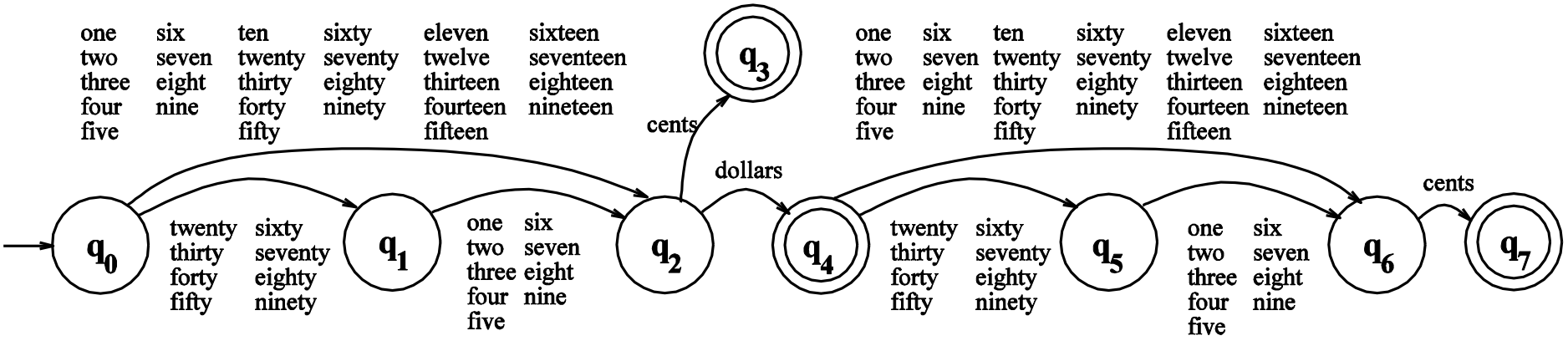
Finite State Automata

- Regular expressions can be viewed as a textual way of specifying the structure of finite-state automata.

More Formally

- You can specify an FSA by enumerating the following things.
  - The set of states: Q
  - A finite alphabet: Σ
  - A start state
  - A set of accept/final states
  - A transition function that maps QxΣ to Q

Dollars and Cents



Assignment 2 - Part 1

- A windows-based version of Python interpreter is available at the supplementary material section of the course website. Please download the interpreter and practice it. Use the help, tutorials and available documentation to investigate the possibility of using Arabic text. summarize your findings.

Assignment 2 - Part 2

- Practice search in Ms Word using regular expressions (Wildcards) for both Arabic and English. Submit at least 5 nontrivial examples.

Assignment 2 - Part 3

- You have been asked to participate in writing an exam about chapter 2 of the textbook. Write one question to check student understanding of chapter two material. Include the answer in your submission.

Thank you

السلام عليكم ورحمة الله