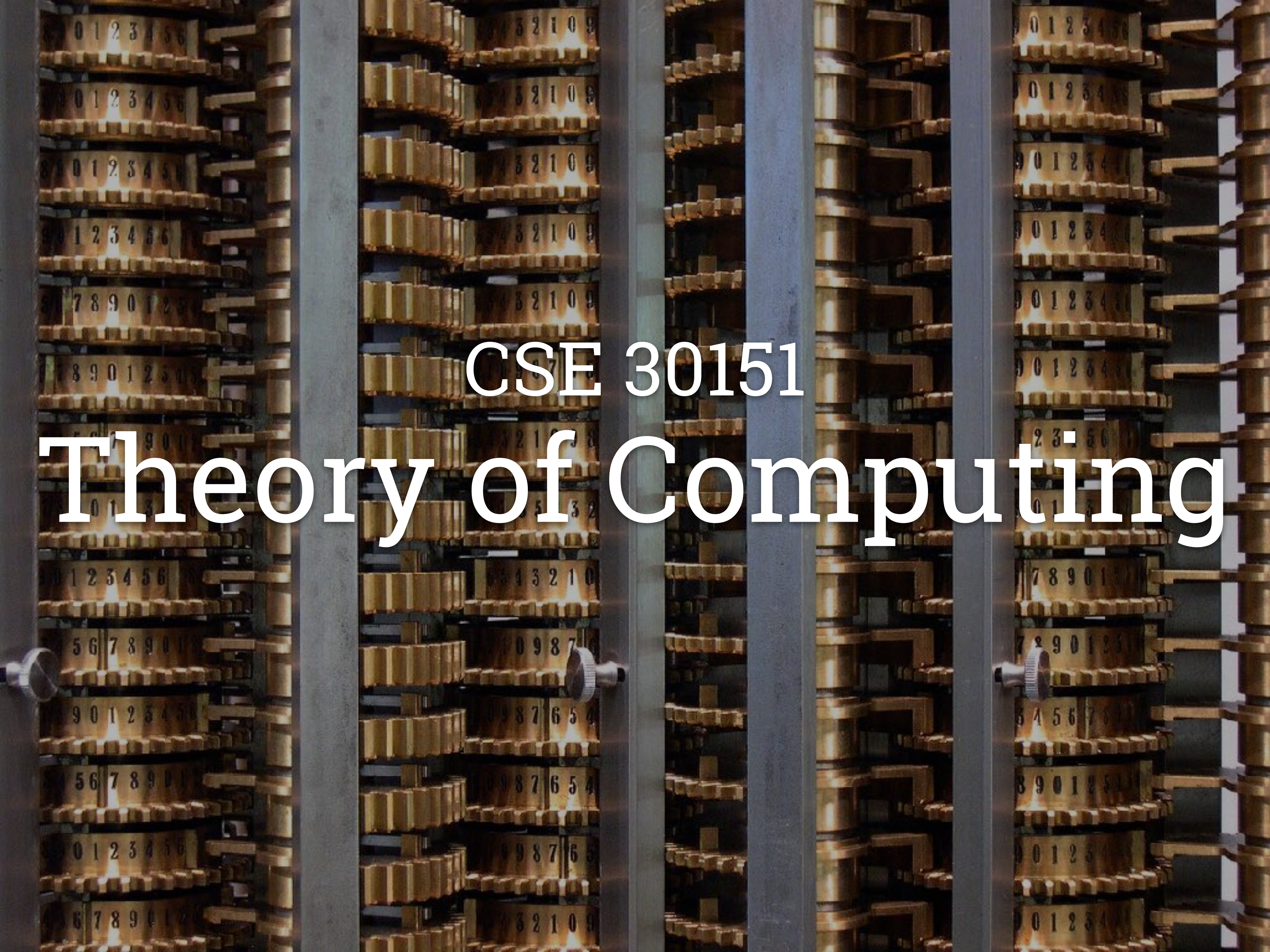


CSE 30151

Theory of Computing

TUESDAY, 2018/03/20
READING: SIPSER 3.1

THE BIG PICTURE

CHOMSKY HIERARCHY

Turing machines

are more powerful than

CFGs and PDAs

are more powerful than

DFAs, NFAs, and regular expressions

CHURCH-TURING THESIS

IN MODERN LANGUAGE

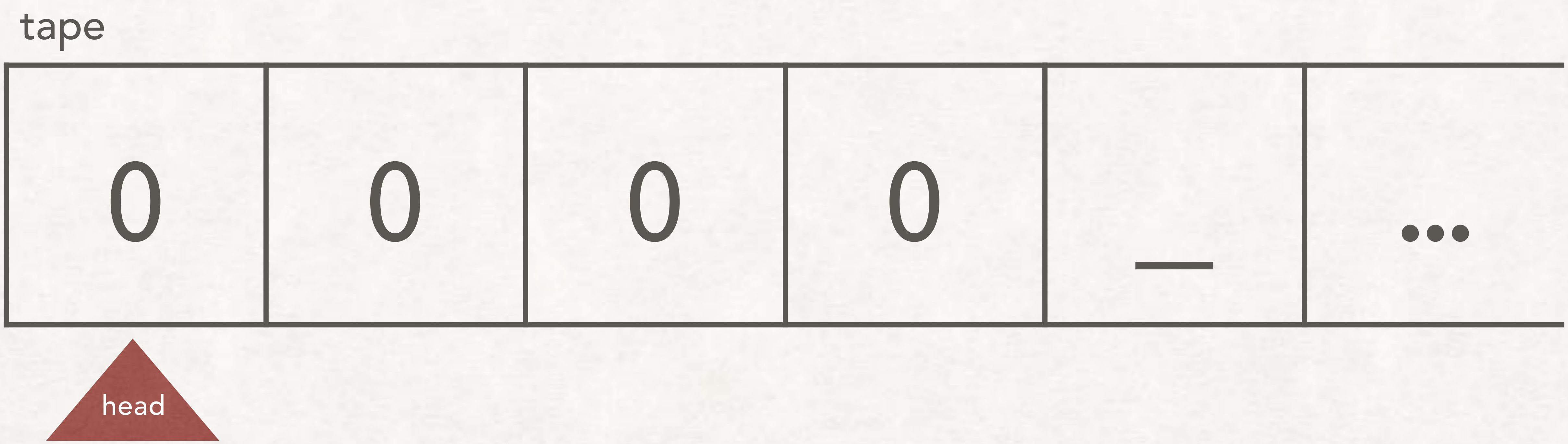
Intuitive notion of algorithm

=

Turing machine algorithm

TURING MACHINES

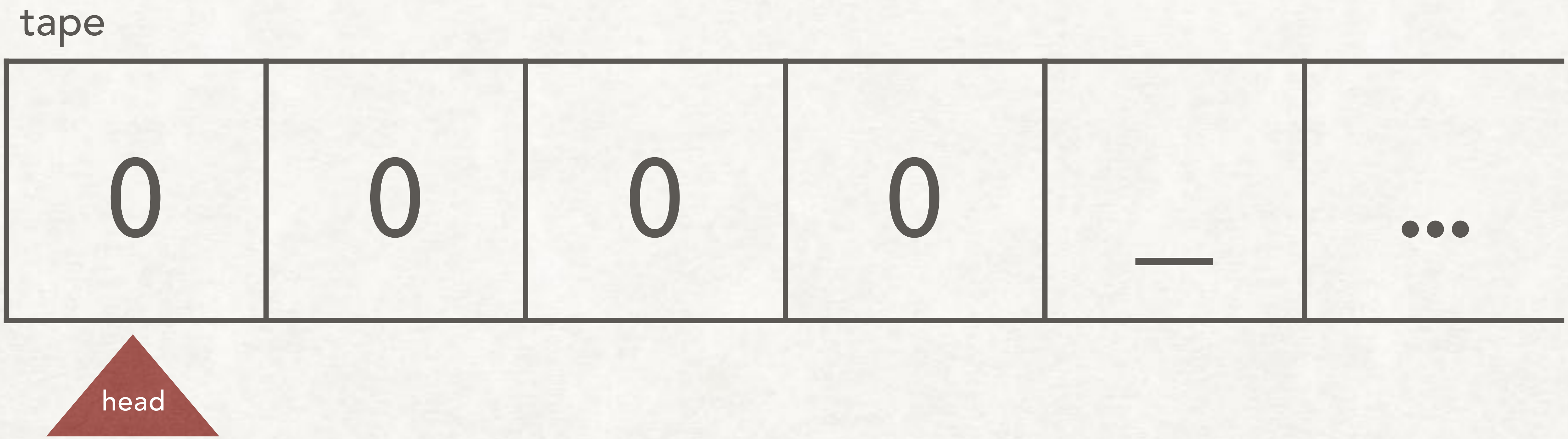
OVERVIEW



- **Tape** that has a left end and extends infinitely to the right
- **Head** that moves across the cells of the tape
- **State** (just like finite and pushdown automata)

TURING MACHINES

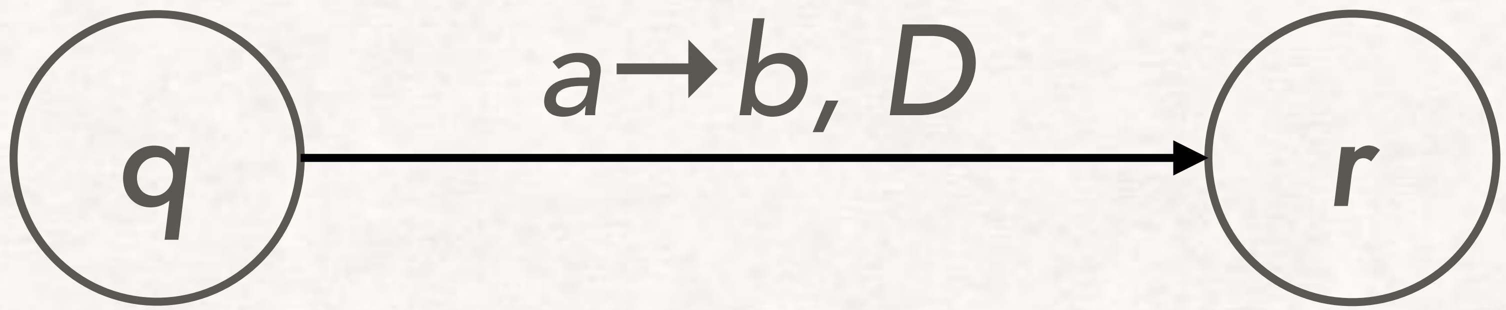
INITIAL CONFIGURATION



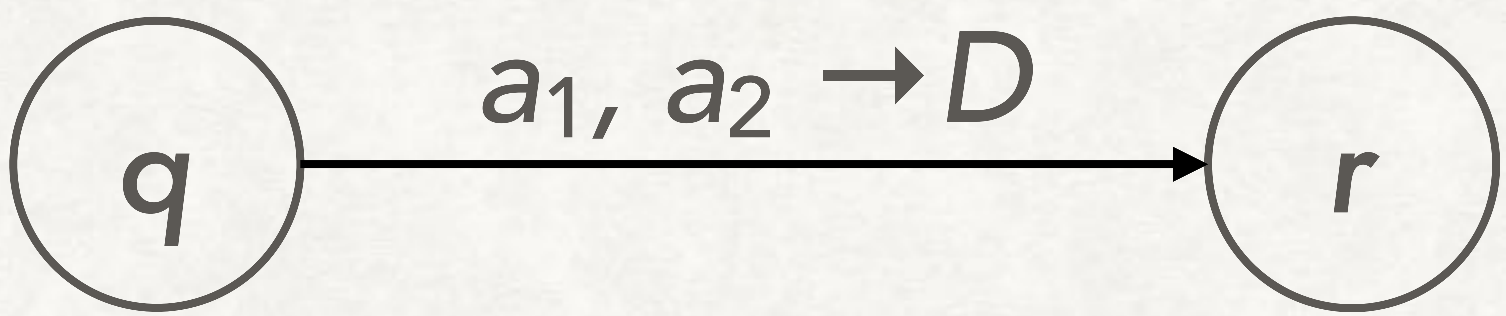
- **Tape** initialized to input string followed by blanks (_)
- **Head** starts at first cell of state
- **State** is the start state (q_0)

TURING MACHINES

TRANSITIONS



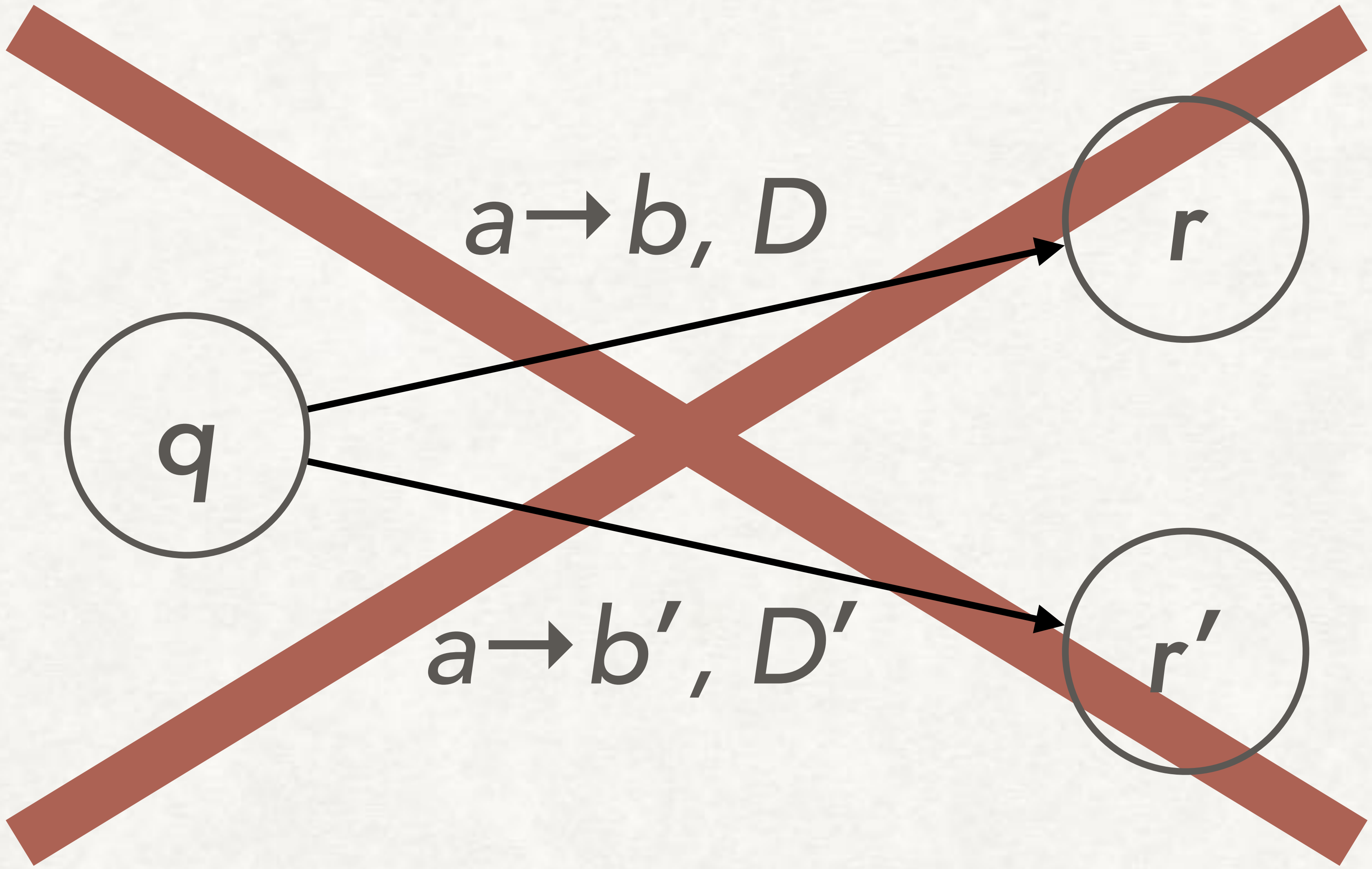
If in state q and read symbol a
then write b, move in direction D, and go to state r
where D can be L (left), S (stay), or R (right)



If in state q and read symbol a_1 or a_2
then move in direction D and go to state r

TURING MACHINES

TRANSITIONS



If a state has *no* transition for a symbol, assume there is an implicit transition to the reject state.

TURING MACHINES

THREE POSSIBLE OUTCOMES

accept and halt	by entering q_{accept}
reject and halt	by entering q_{reject}
loop	otherwise

TURING MACHINES

RECOGNIZING AND DECIDING LANGUAGES

Turing-recognizable:

If the string is in *L*, then accept and halt

Otherwise, reject and halt, **or loop**

(Turing-)decidable:

If the string is in *L*, then accept and halt

Otherwise, reject and halt

TURING MACHINES

THREE WAYS OF WRITING

- Formal description: tuple and table, or state diagram
- Implementation description: pseudocode
  - Describes exact contents of tape and motion of head
  - Arithmetic, etc. not allowed
  - Should enable the reader to reimplement the machine
- High-level description:
  - Should convince the reader that the machine exists

TURING MACHINES

EXAMPLE IMPLEMENTATION DESCRIPTION

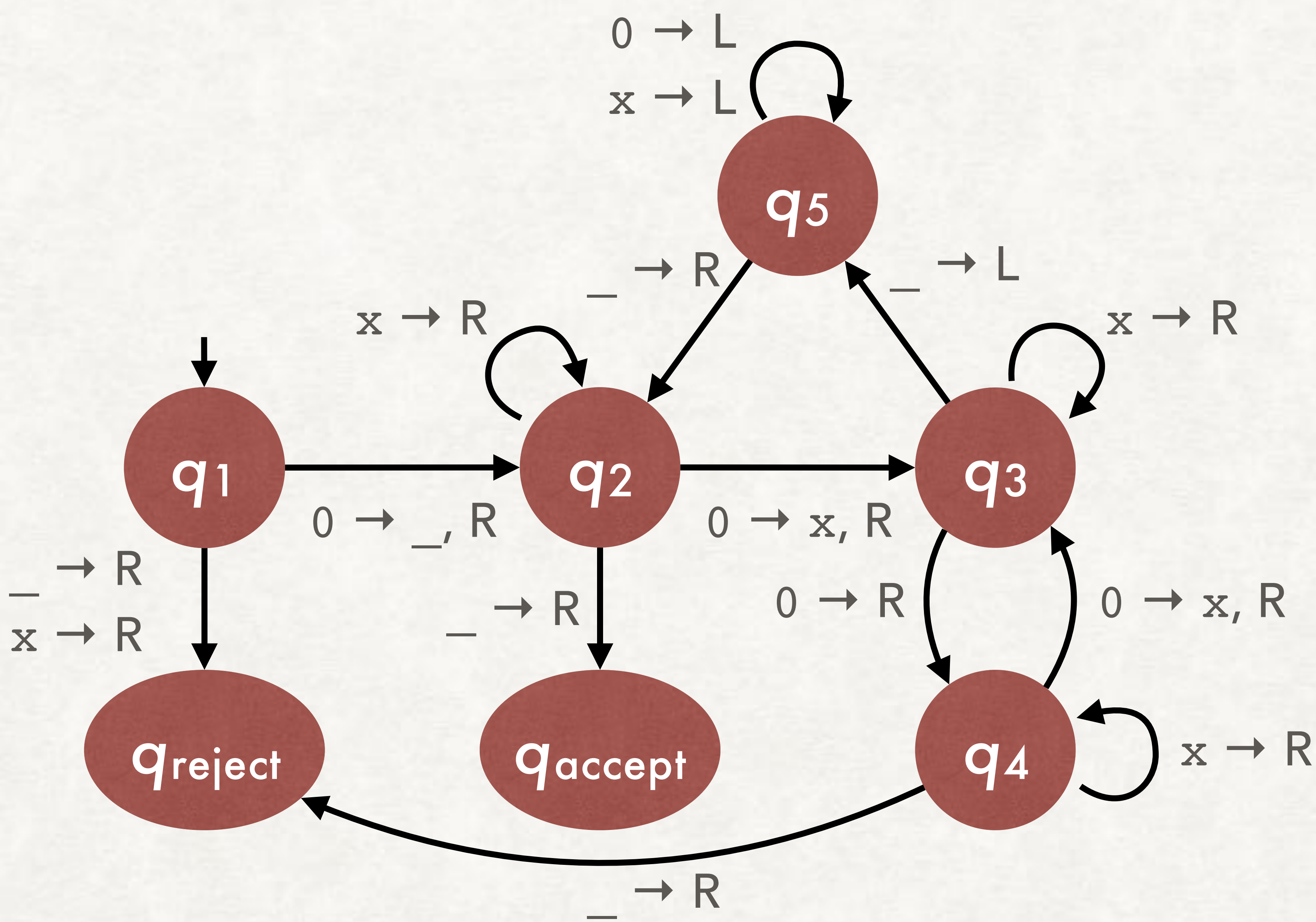
$A = \{0^n \mid n \text{ is a power of 2}\}$

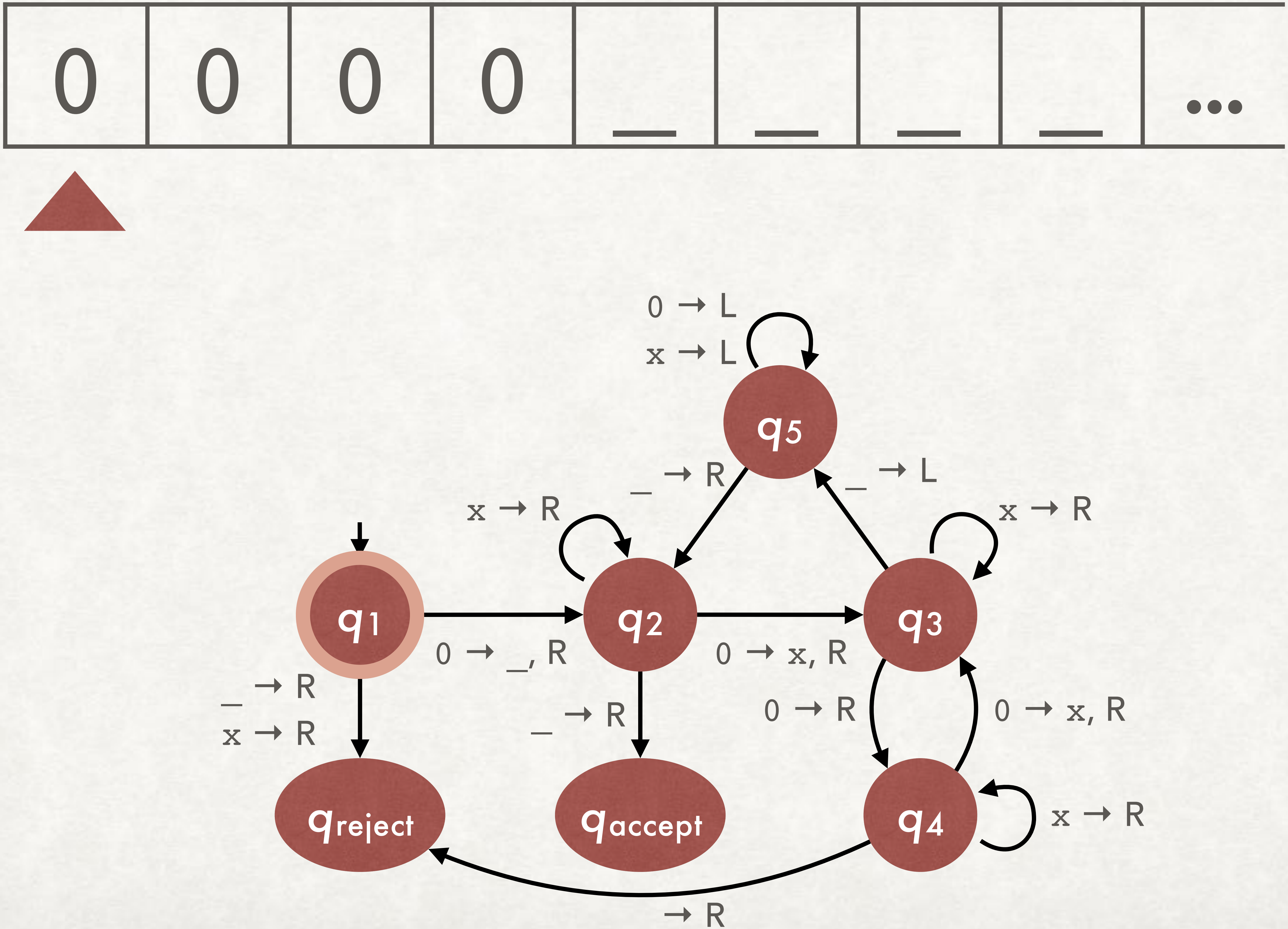
M_2 = "On input string *w*:

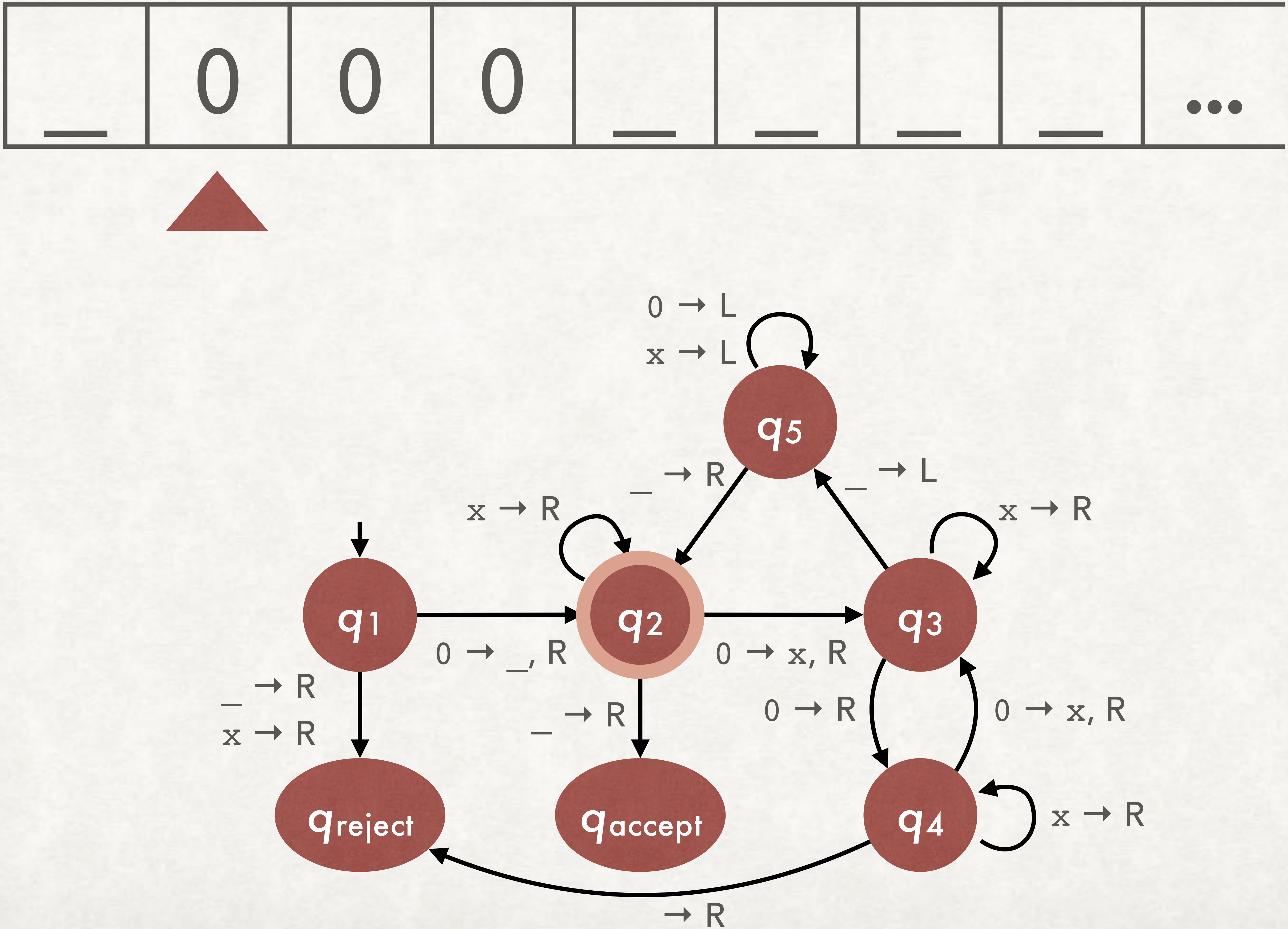
1. Sweep left to right across the tape, crossing off every other `0`.
2. If in stage 1 the tape contained a single `0`, *accept*.
3. If in stage 1 the contained more than a single `0` and the number of `0`s was odd, *reject*.
4. Return the head to the left-hand end of the tape.
5. Go to stage 1."

TURING MACHINES

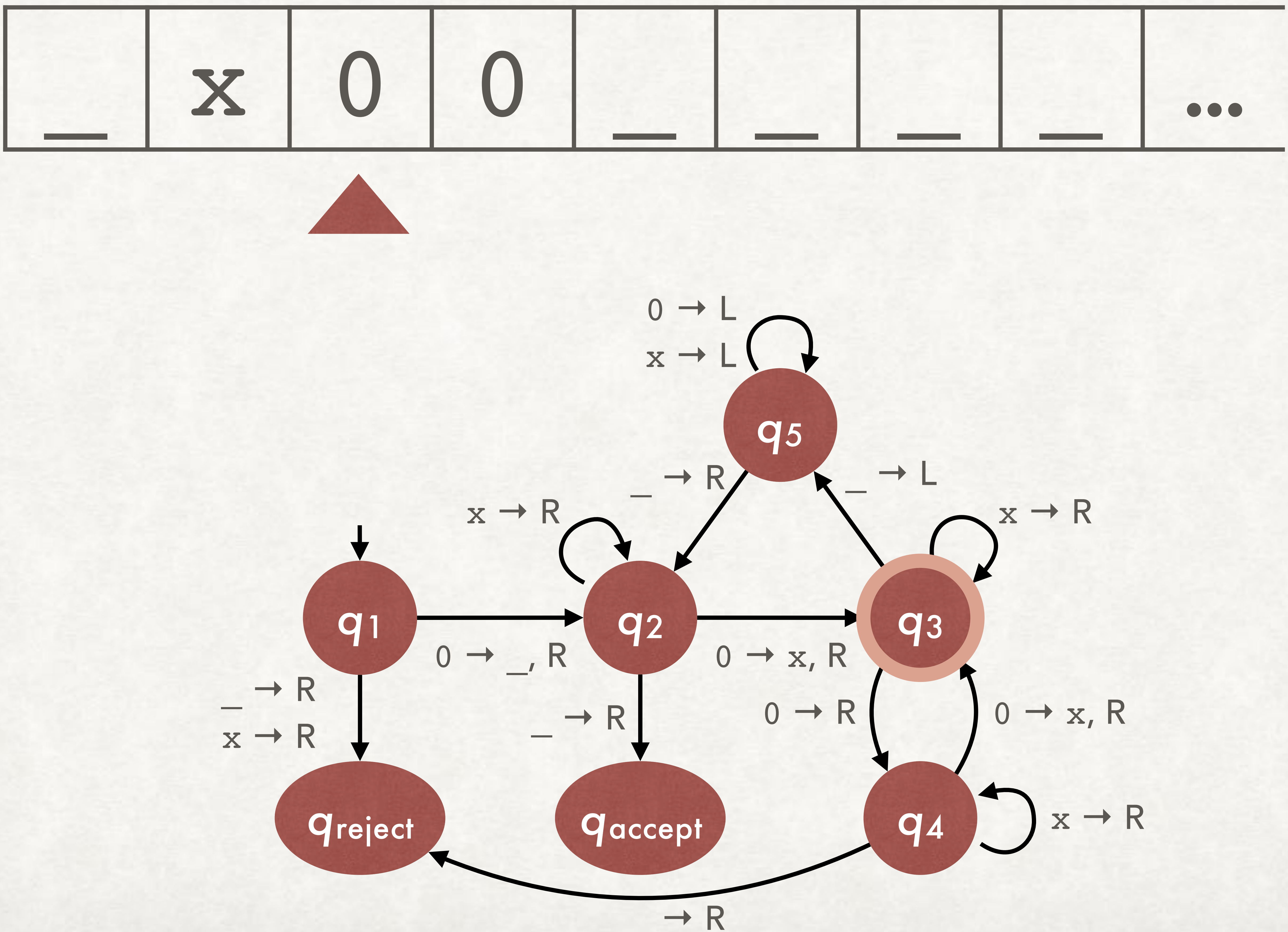
EXAMPLE FORMAL DESCRIPTION (STATE DIAGRAM)



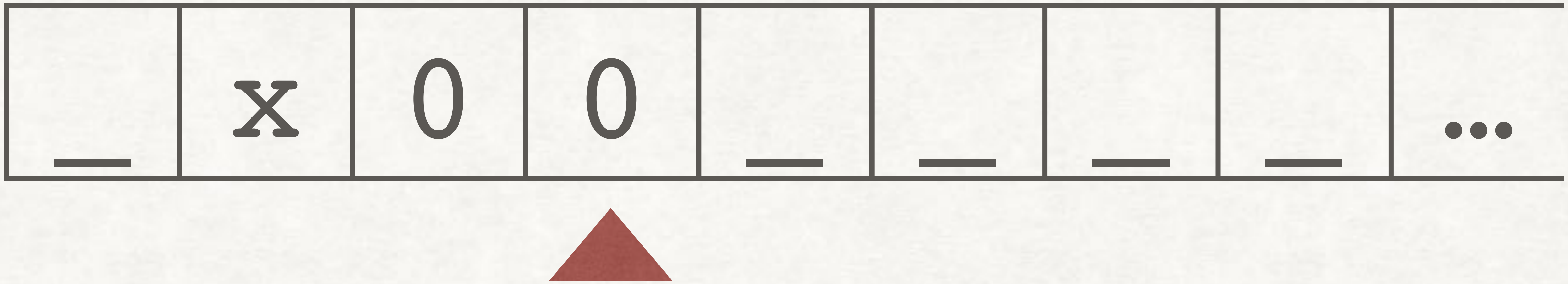

0 0 0 0 _ _ _ _ …
0 → L
x → L
q5
_ → R
_ → L
x → R
x → R
q1
q2
q3
0 → _, R
0 → x, R
_ → R
x → R
_ → R
0 → R
0 → x, R
qreject
qaccept
q4
x → R
_ → R



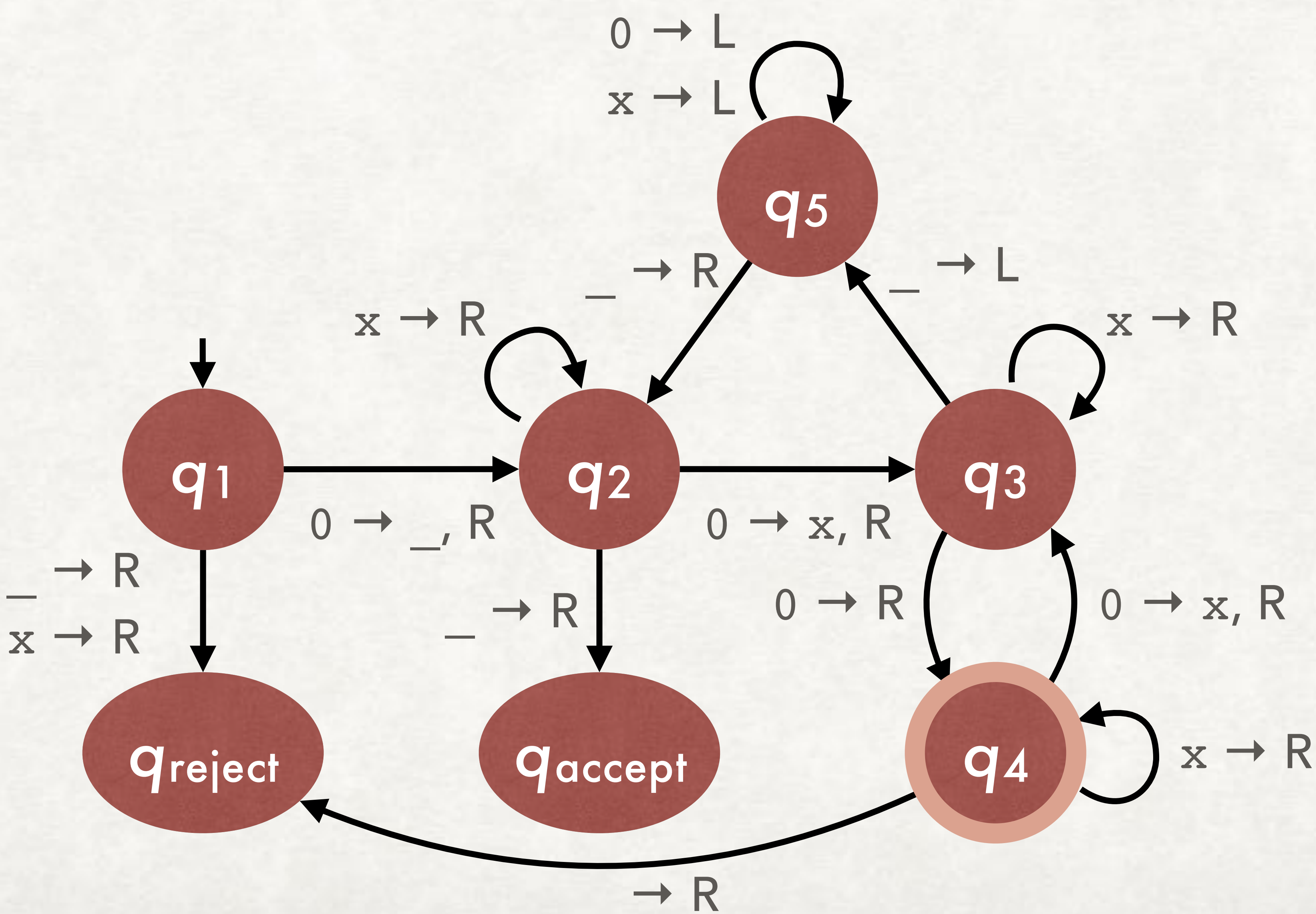
_
0
0
0
_
_
_
_
...
0 → L
x → L
q5
_ → R
_ → L
x → R
x → R
q1
q2
q3
0 → _, R
0 → x, R
_ → R
x → R
_ → R
0 → R
0 → x, R
qreject
qaccept
q4
x → R
_ → R



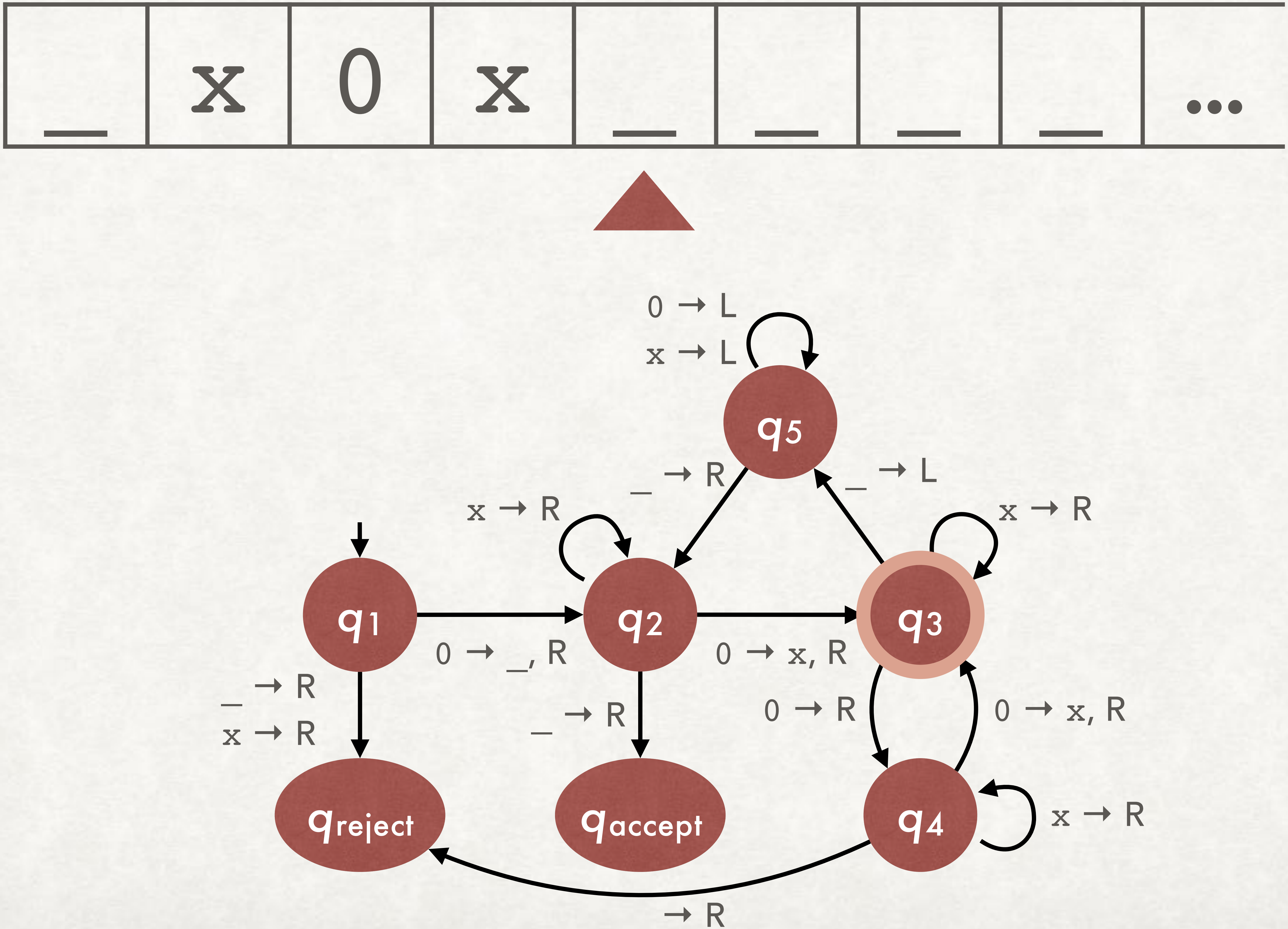
_
x
0
0
_
_
_
_
...
0 → L
x → L
q5
_ → R
_ → L
x → R
x → R
q1
q2
q3
0 → _, R
0 → x, R
_ → R
x → R
_ → R
0 → R
0 → x, R
qreject
qaccept
q4
x → R
_ → R



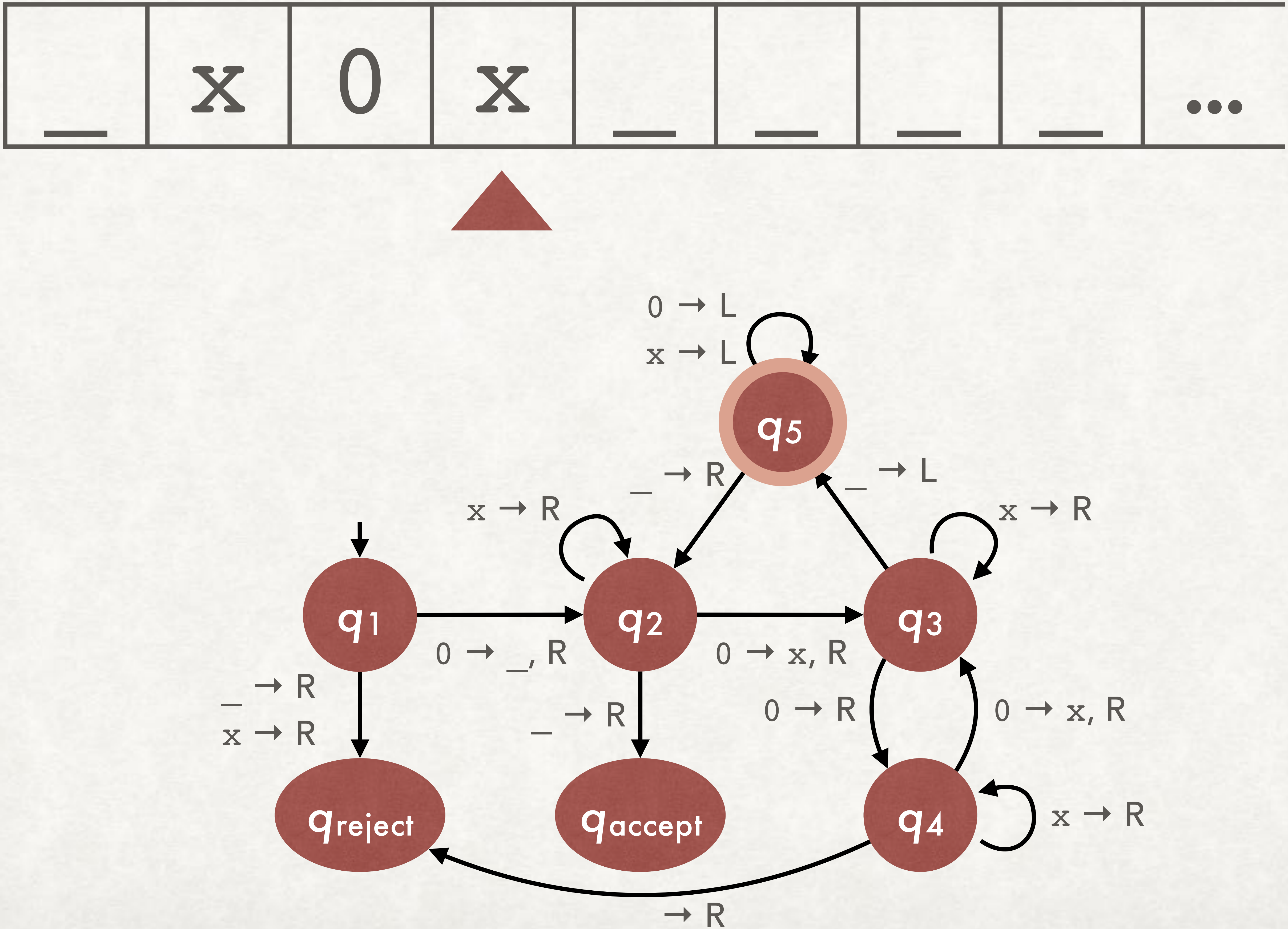
_
x
0
0
_
_
_
_
…

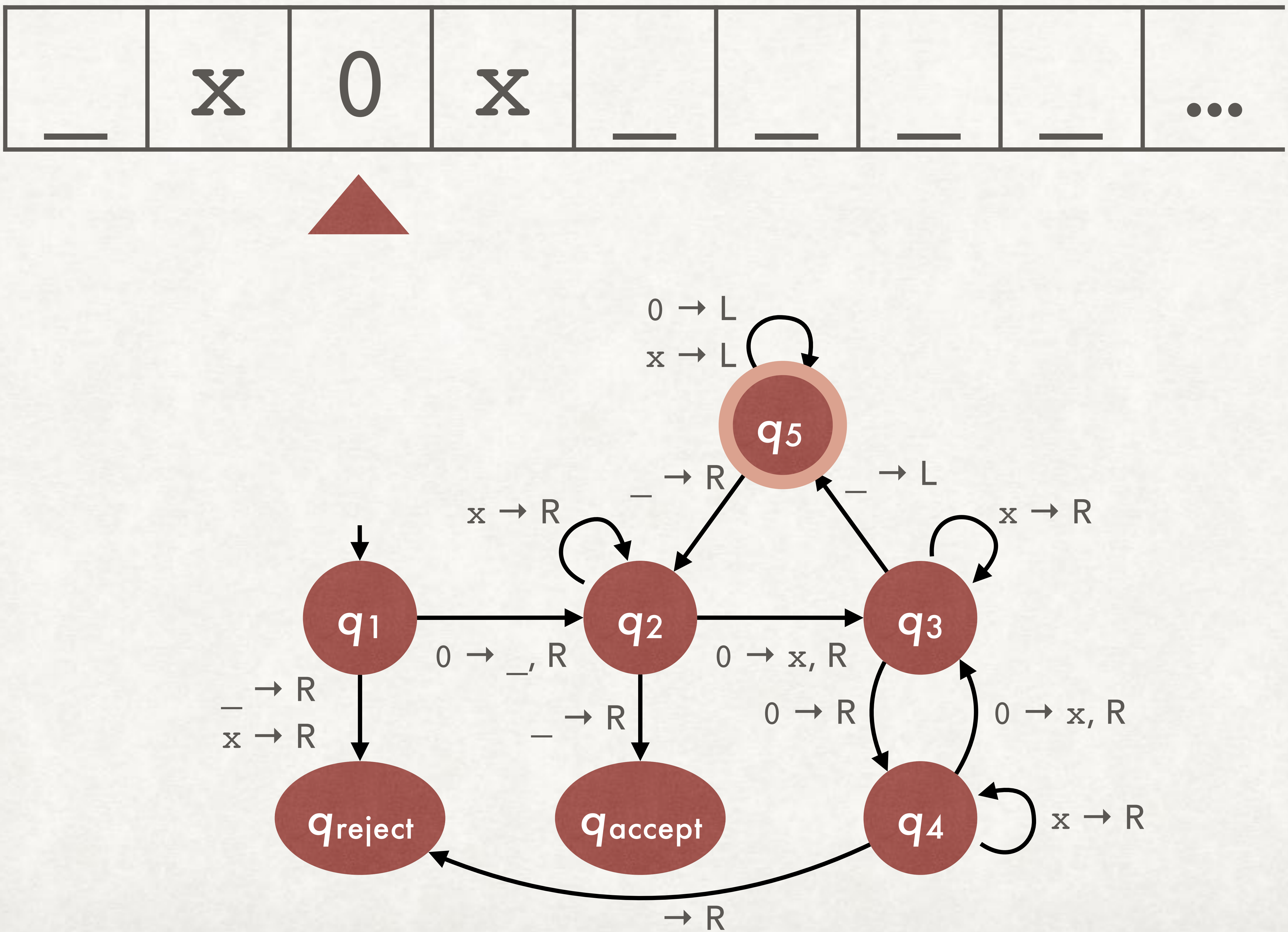


0 → L
x → L
q5
_ → R
_ → L
x → R
x → R
q1
q2
q3
0 → _, R
0 → x, R
_ → R
x → R
_ → R
0 → R
0 → x, R
qreject
qaccept
q4
x → R
_ → R



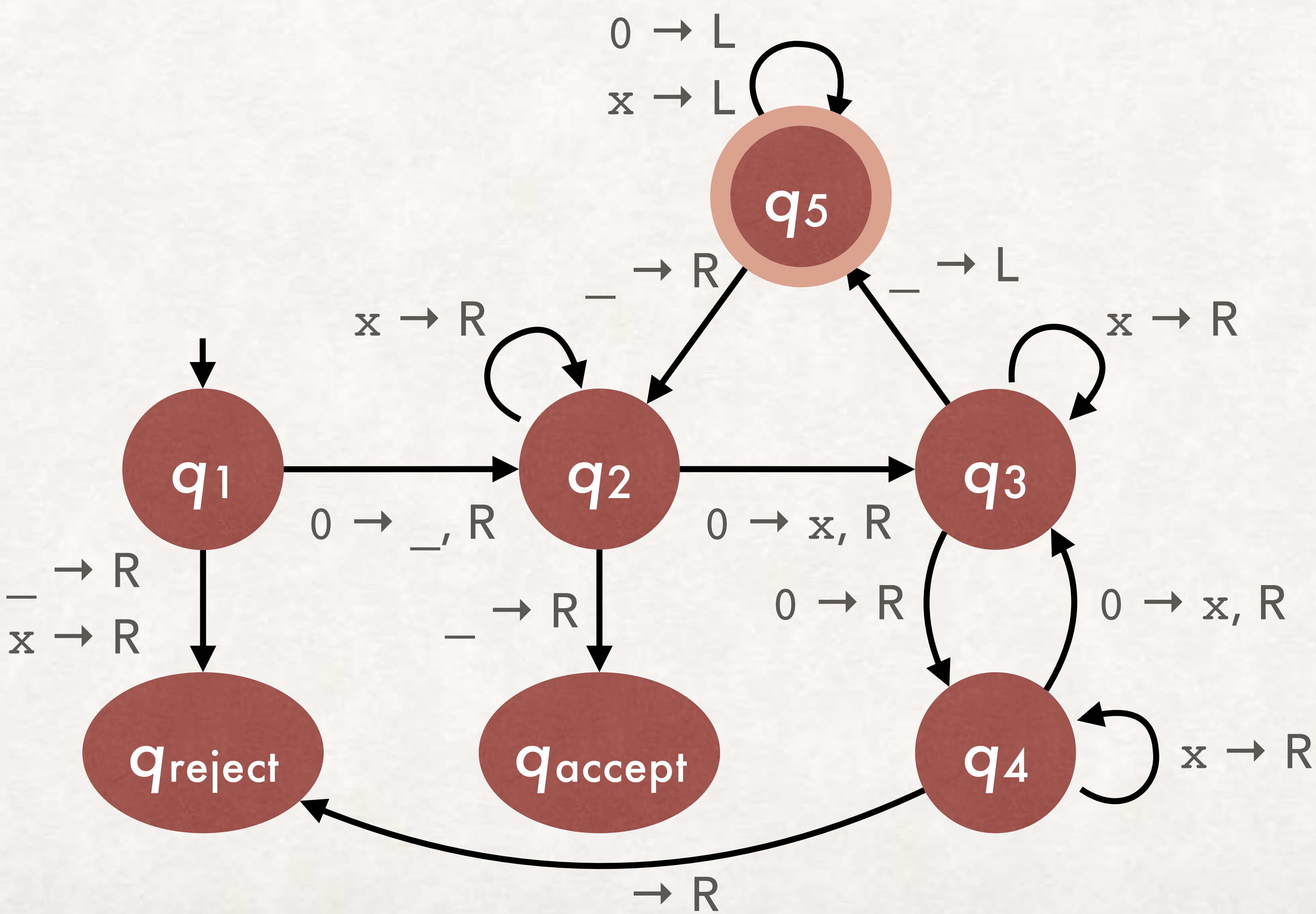
_
x
0
x
_
_
_
_
...
0 → L
x → L
q5
_ → R
_ → L
x → R
x → R
q1
q2
q3
0 → _, R
0 → x, R
_ → R
x → R
_ → R
0 → R
0 → x, R
qreject
qaccept
q4
x → R
_ → R

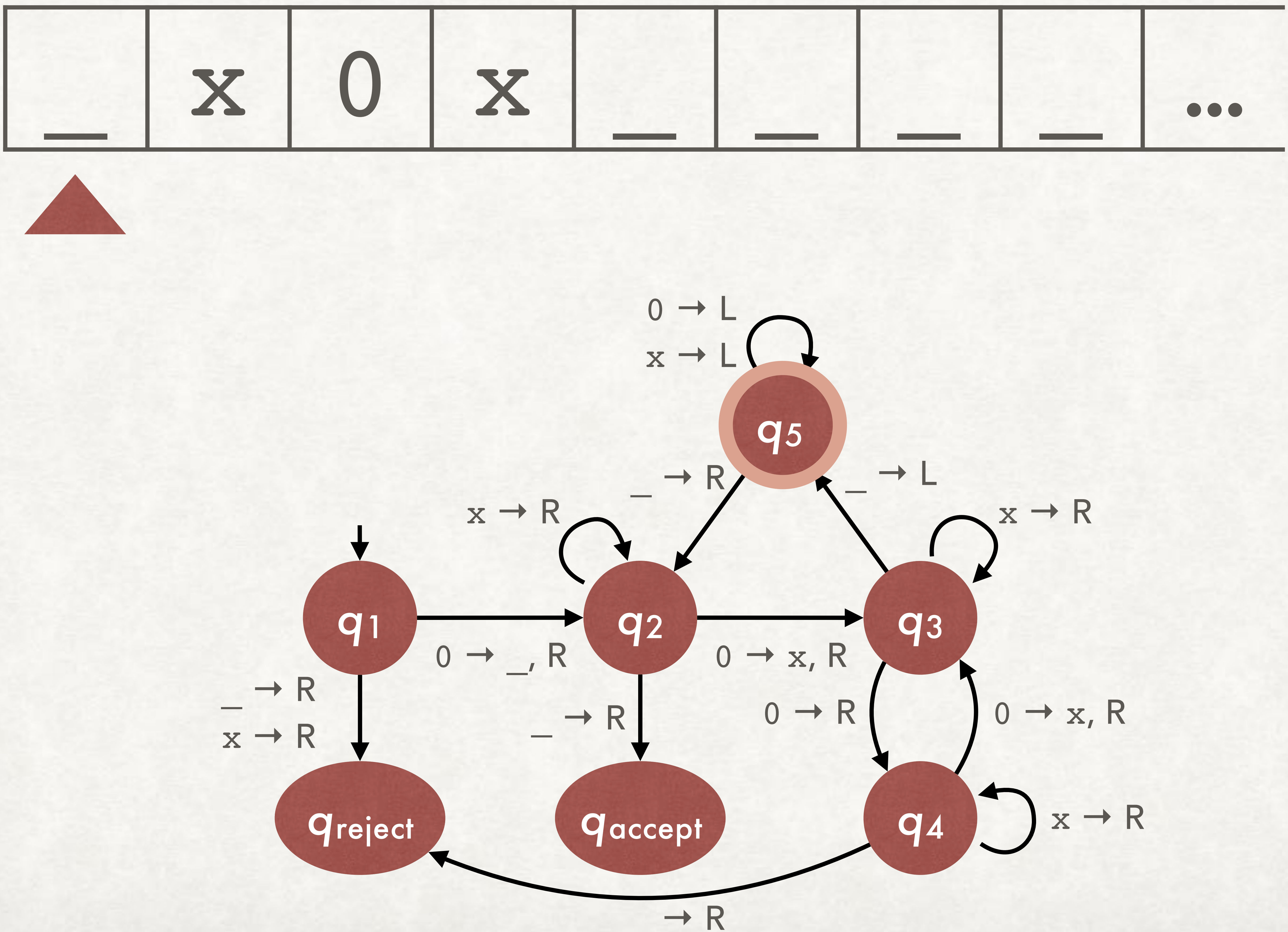

_
x
0
x
_
_
_
_
…
0 → L
x → L
q5
_ → R
_ → L
x → R
x → R
q1
q2
q3
0 → _, R
0 → x, R
_ → R
x → R
_ → R
0 → R
0 → x, R
qreject
qaccept
q4
x → R
_ → R

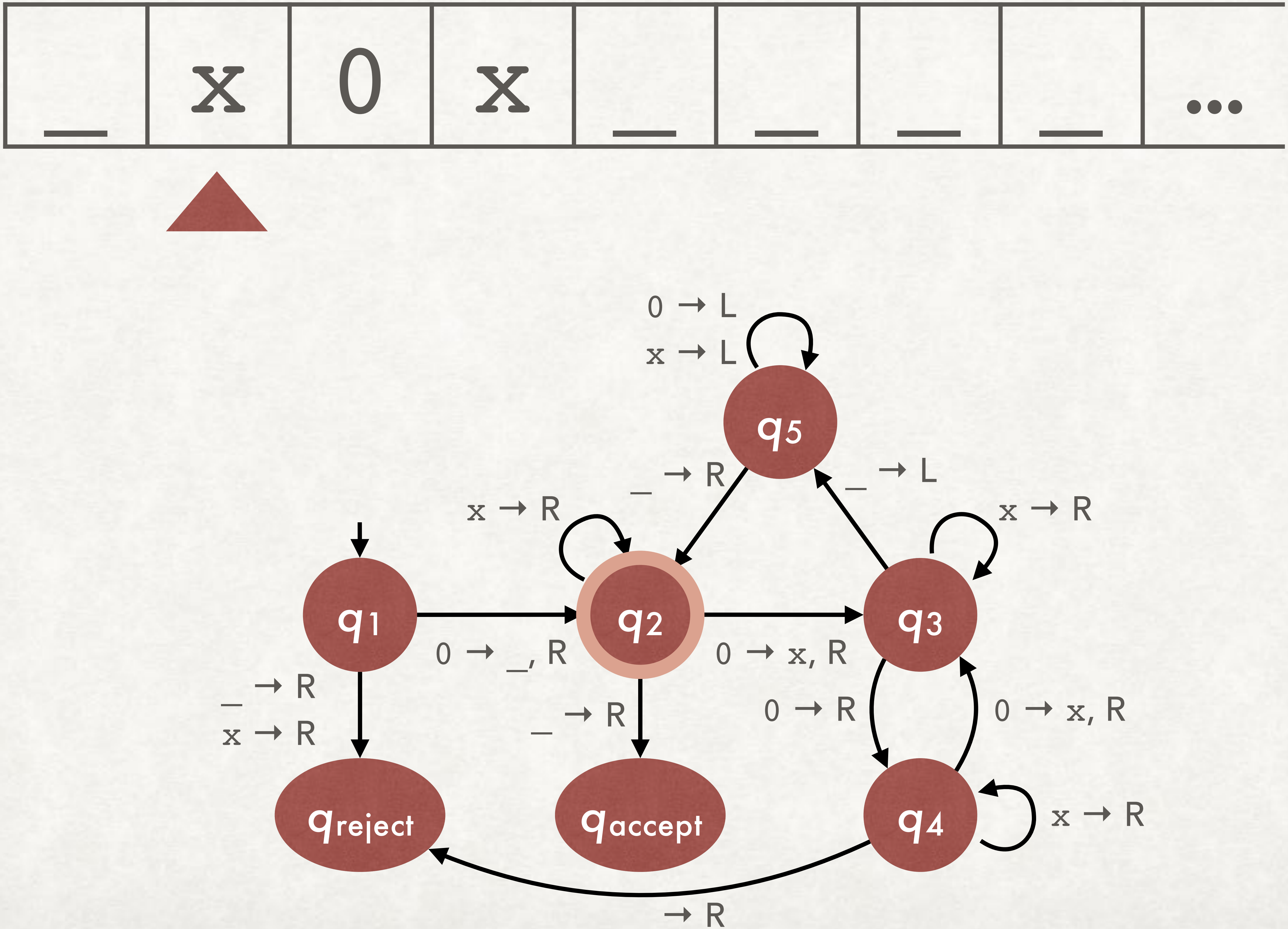


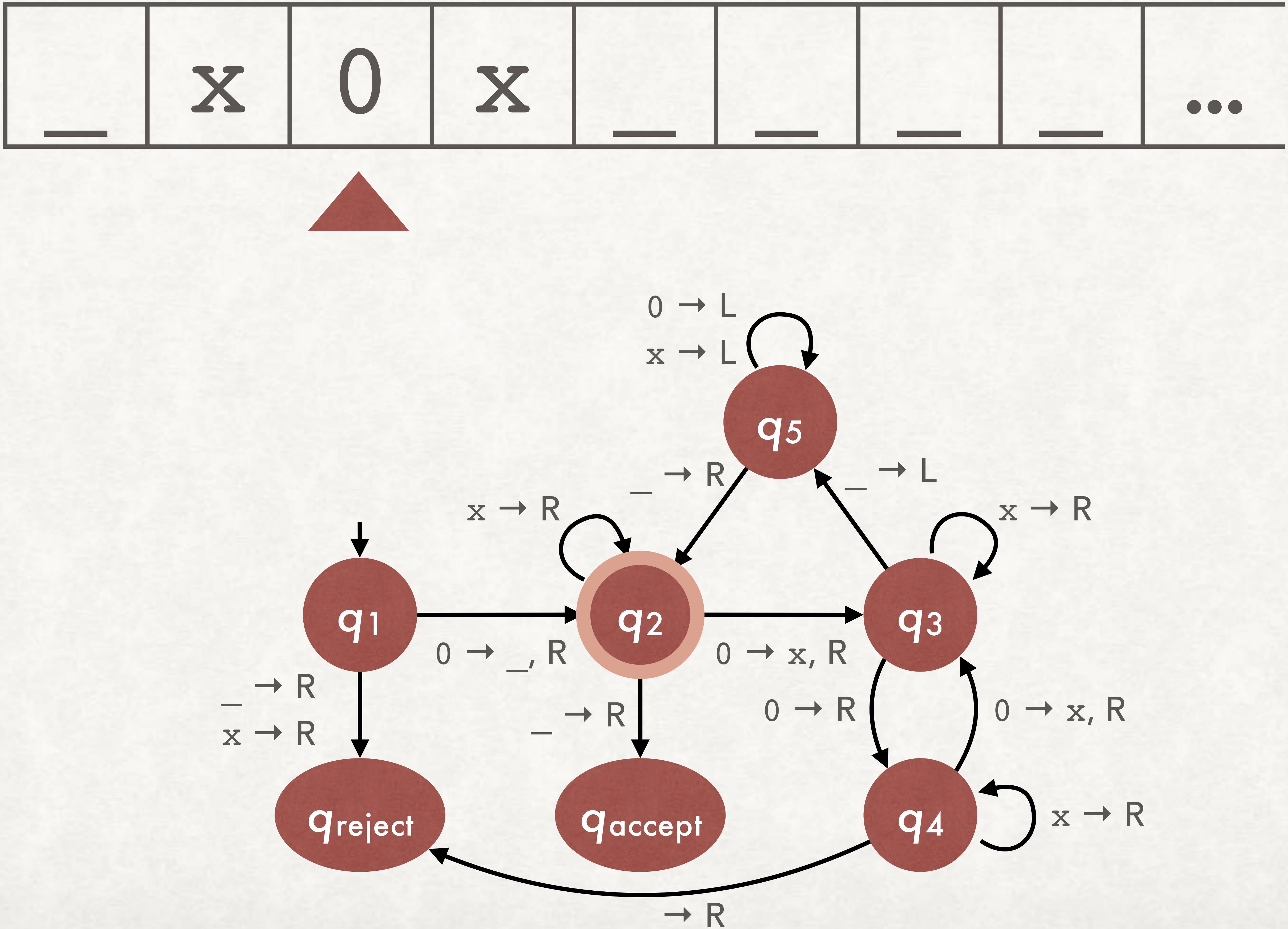
_
x
0
x
_
_
_
_
...
0 → L
x → L
q5
_ → R
_ → L
x → R
x → R
q1
q2
q3
0 → _, R
0 → x, R
_ → R
x → R
_ → R
0 → R
0 → x, R
qreject
qaccept
q4
x → R
_ → R


_
x
0
x
_
_
_
_
...

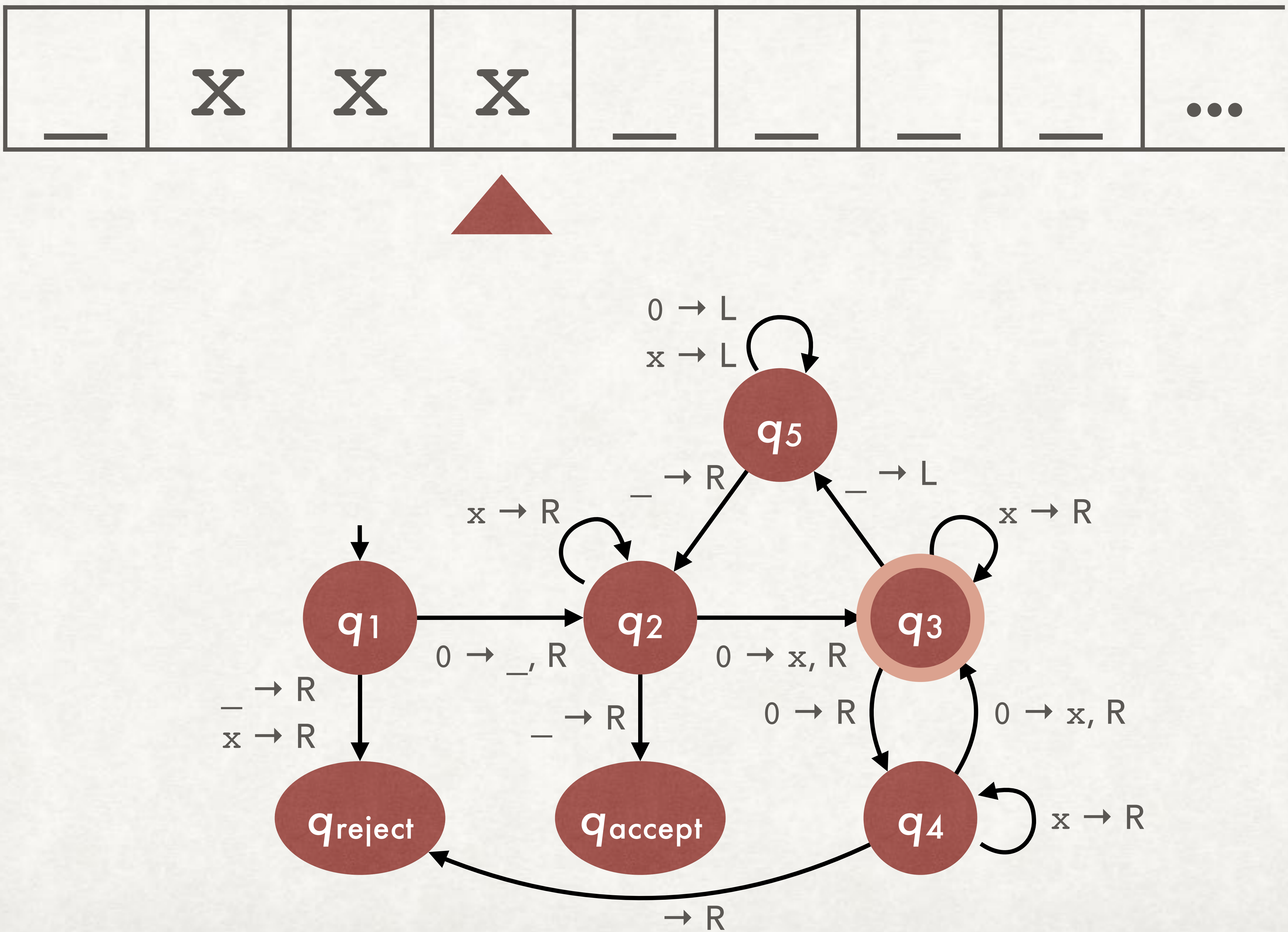

0 → L
x → L
q5
_ → R
_ → L
x → R
x → R
q1
q2
q3
0 → _, R
0 → x, R
_ → R
x → R
_ → R
0 → R
0 → x, R
qreject
qaccept
q4
x → R
_ → R

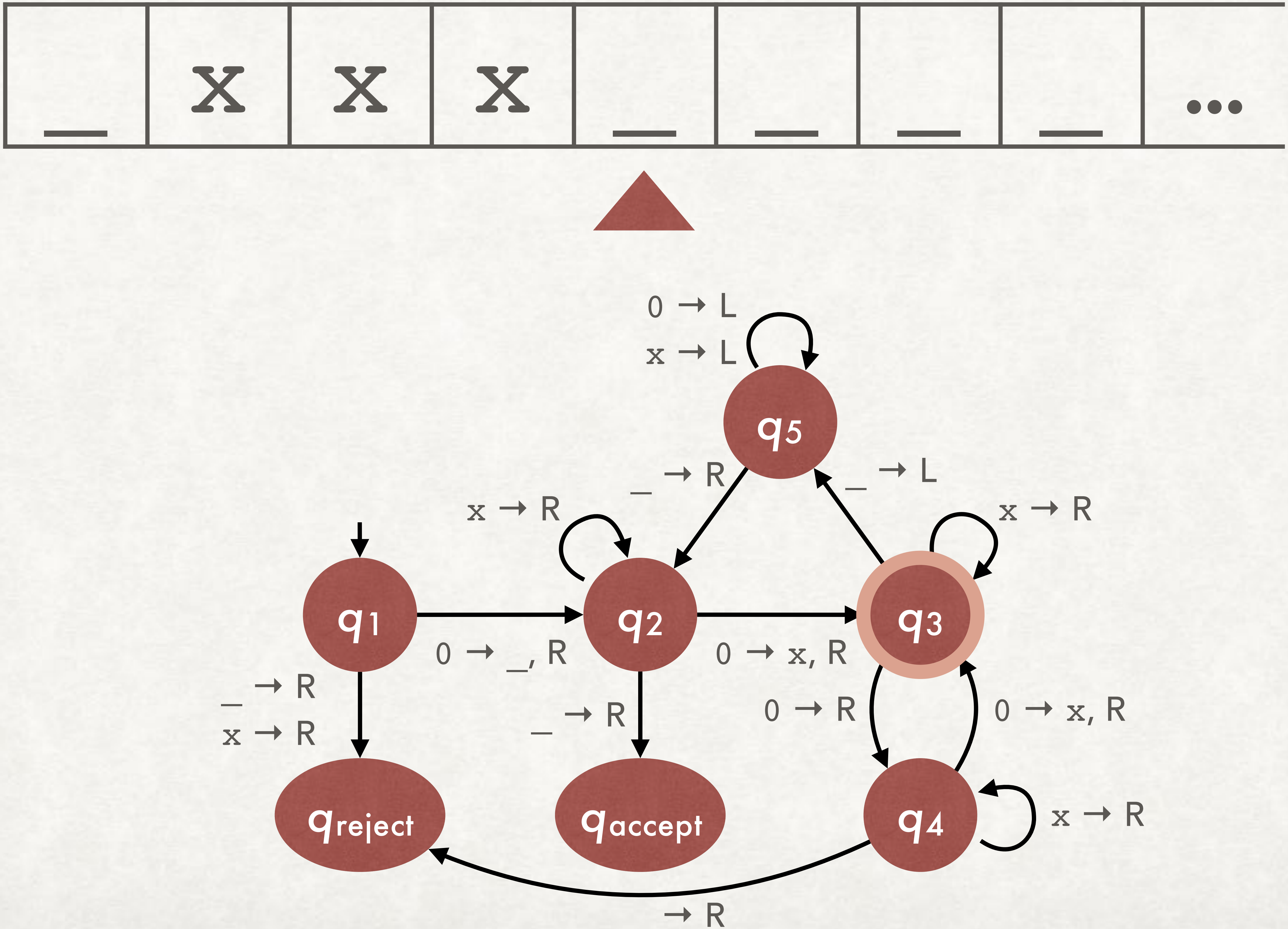

_
x
0
x
_
_
_
_
...
0 → L
x → L
q5
_ → R
_ → L
x → R
x → R
q1
q2
q3
0 → _, R
0 → x, R
_ → R
x → R
_ → R
0 → R
0 → x, R
qreject
qaccept
q4
x → R
_ → R


_
x
0
x
_
_
_
_
...
0 → L
x → L
q5
_ → R
_ → L
x → R
x → R
q1
q2
q3
0 → _, R
0 → x, R
_ → R
x → R
_ → R
0 → R
0 → x, R
qreject
qaccept
q4
x → R
_ → R

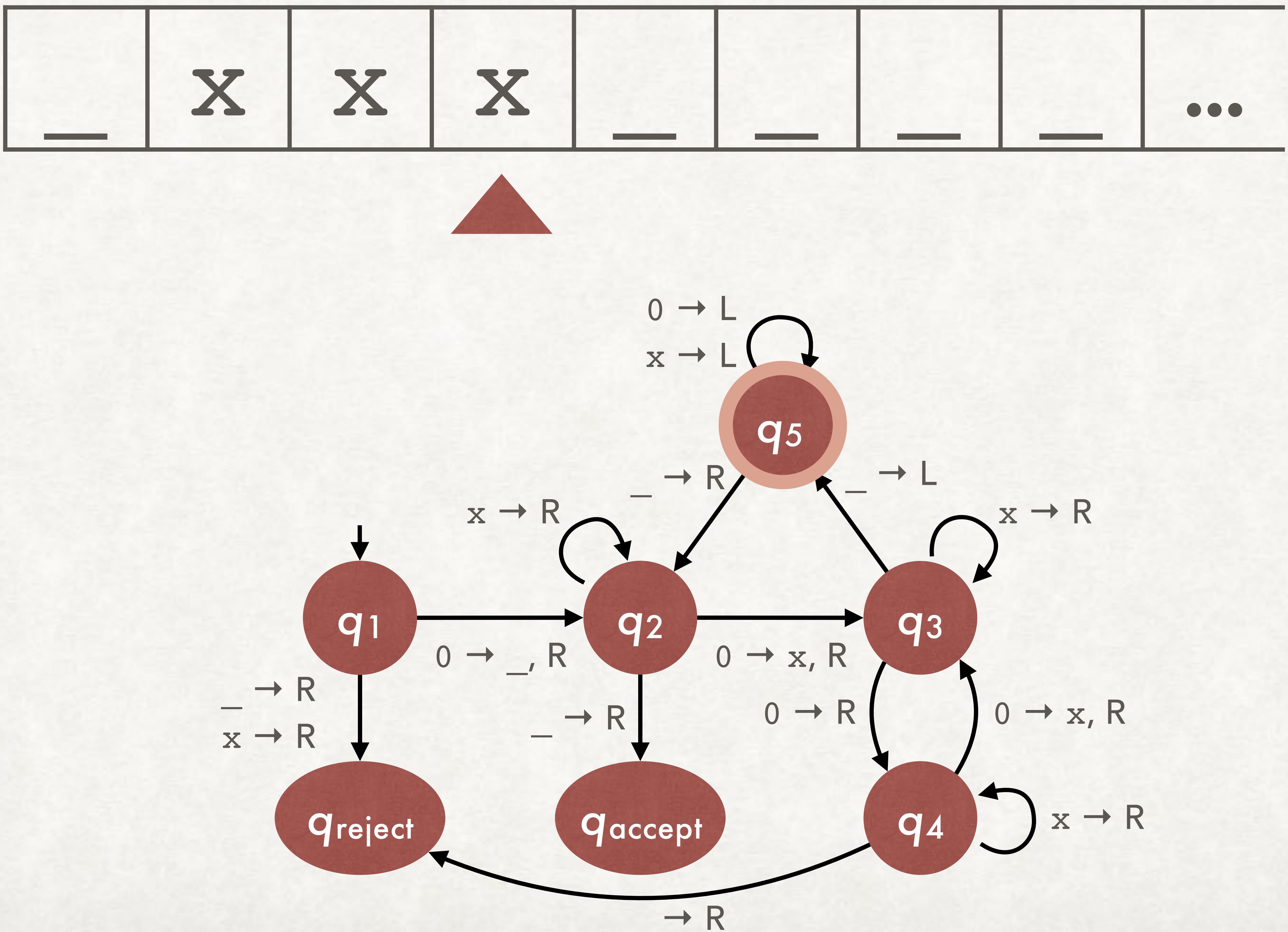


_
x
0
x
_
_
_
_
...
0 → L
x → L
q5
_ → R
_ → L
x → R
x → R
q1
q2
q3
0 → _, R
0 → x, R
_ → R
x → R
_ → R
0 → R
0 → x, R
qreject
qaccept
q4
x → R
_ → R

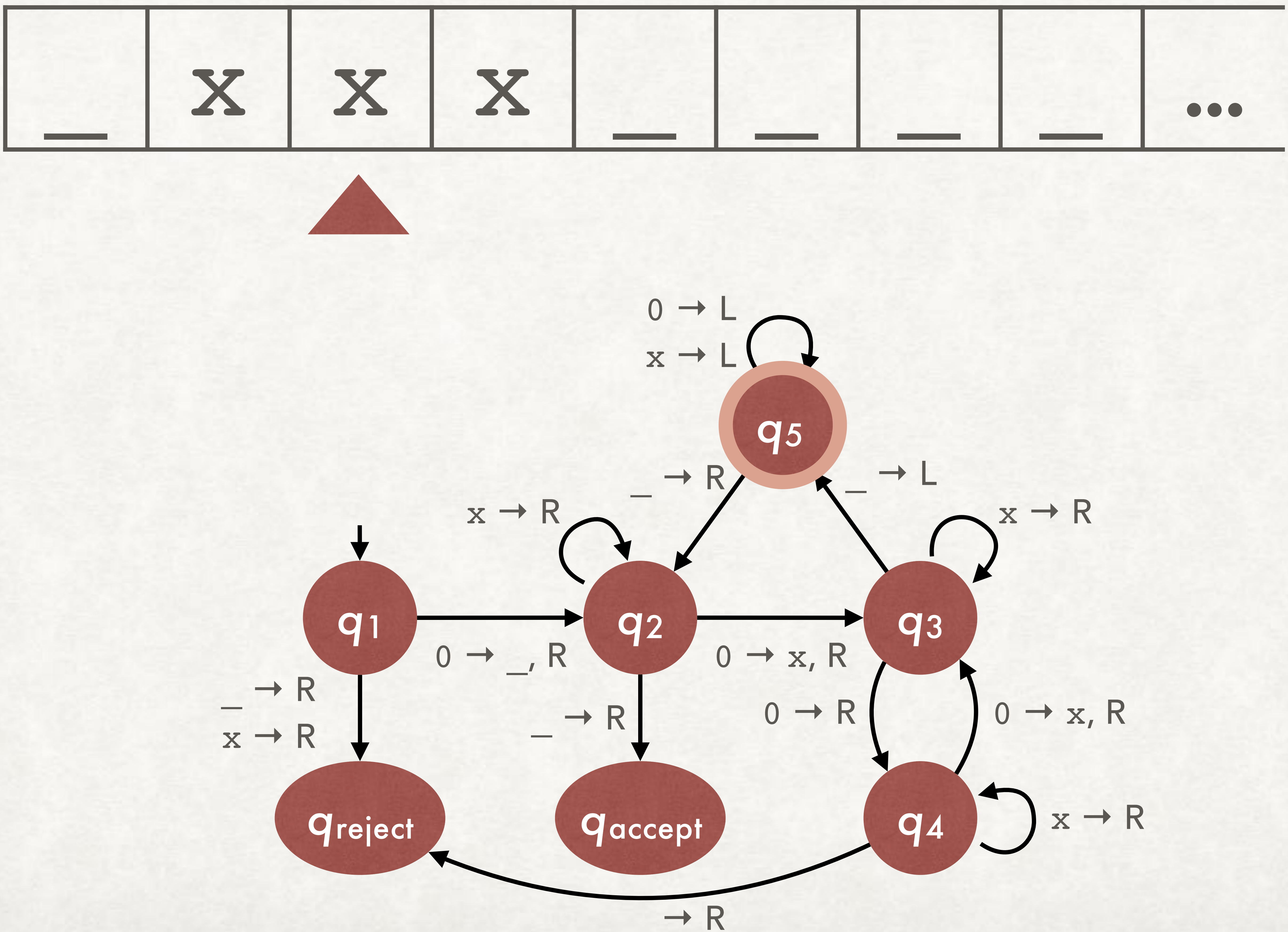

_
x
x
x
_
_
_
_
...
0 → L
x → L
q5
_ → R
_ → L
x → R
x → R
q1
q2
q3
0 → _, R
0 → x, R
_ → R
x → R
_ → R
0 → R
0 → x, R
qreject
qaccept
q4
x → R
_ → R



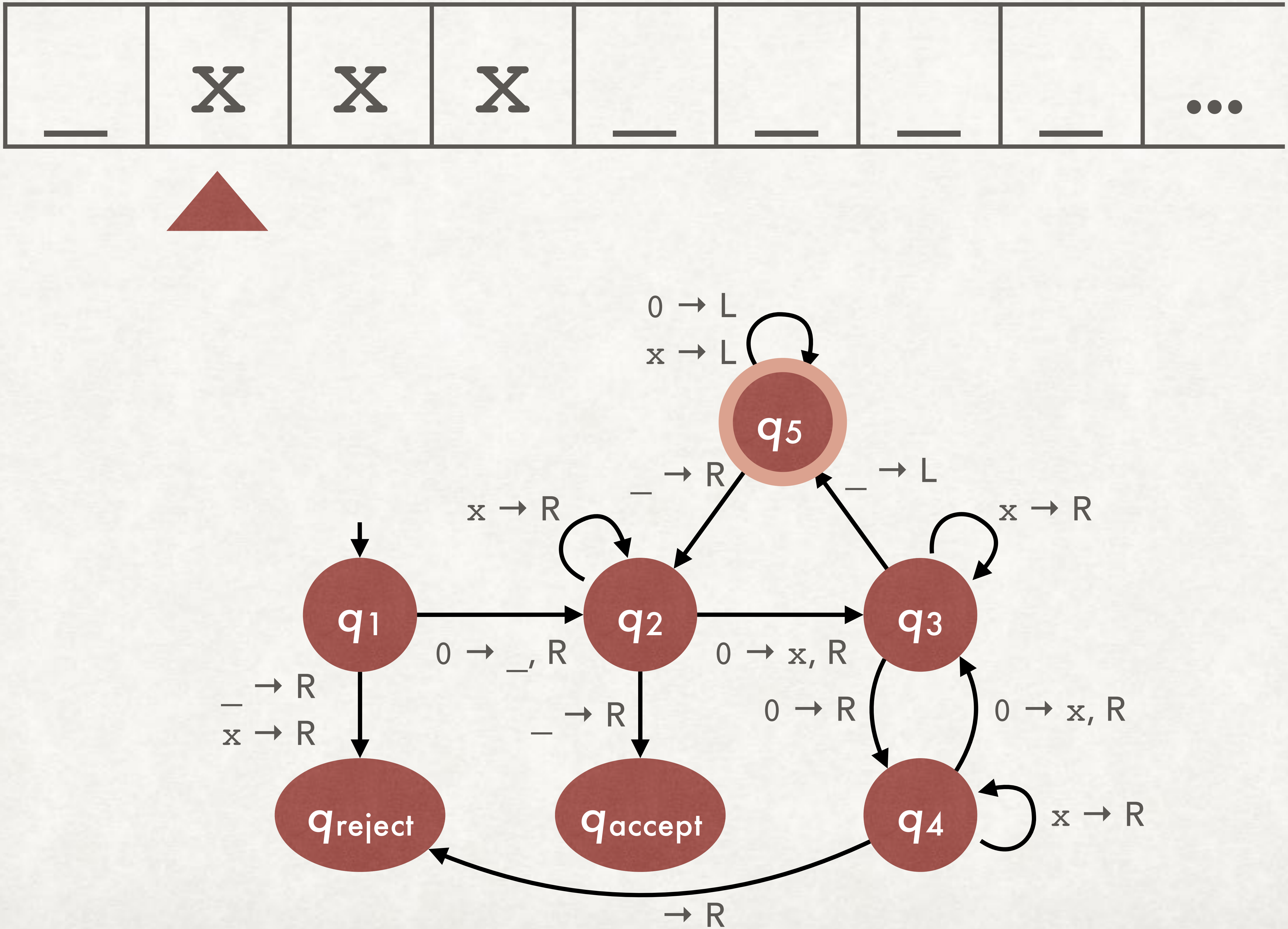
_
x
x
x
_
_
_
_
...
0 → L
x → L
q5
_ → R
_ → L
x → R
x → R
q1
q2
q3
0 → _, R
0 → x, R
_ → R
x → R
_ → R
0 → R
0 → x, R
qreject
qaccept
q4
x → R
_ → R



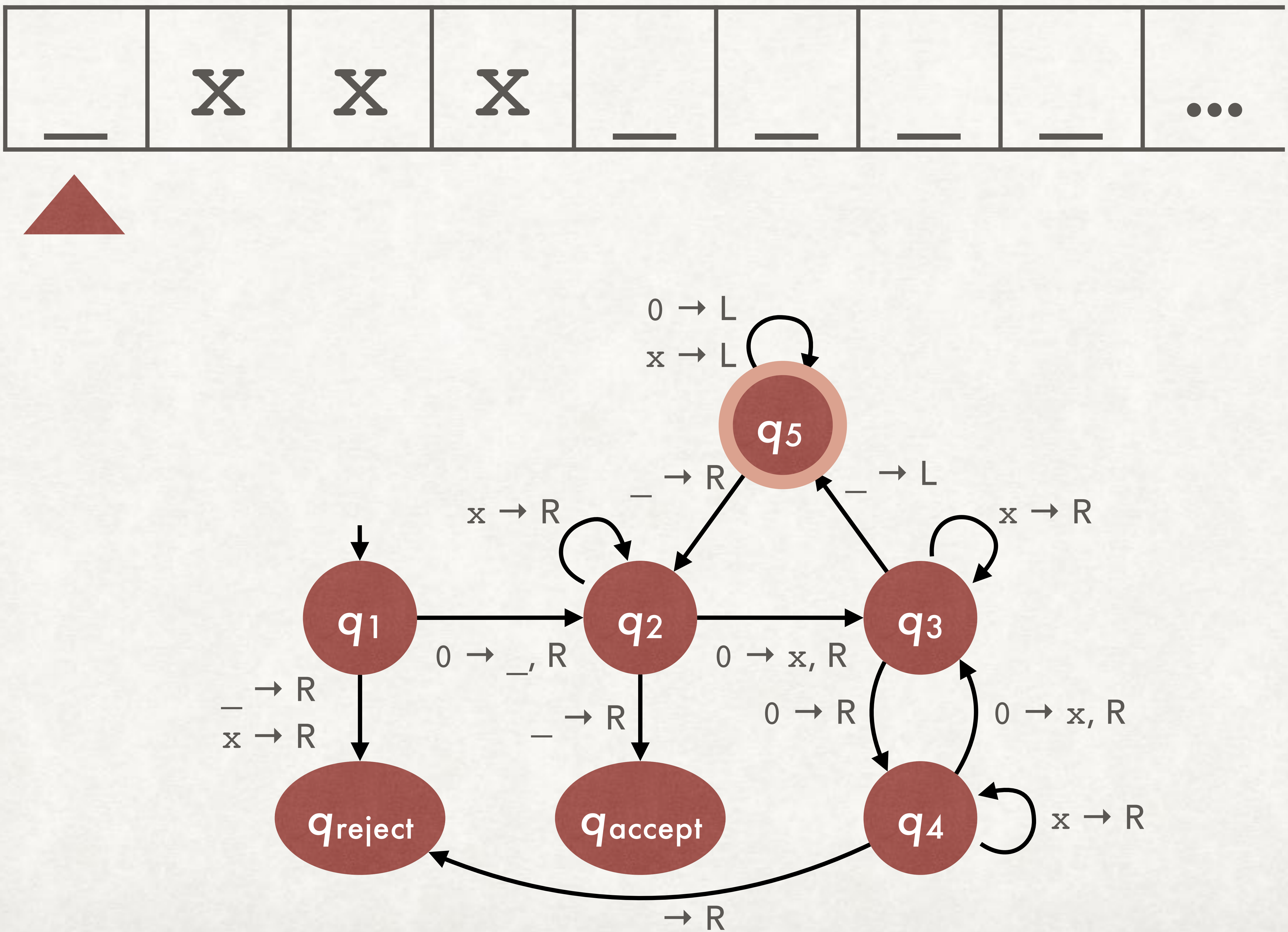
_
x
x
x
_
_
_
_
...
0 → L
x → L
q5
_ → R
_ → L
x → R
x → R
q1
q2
q3
0 → _, R
0 → x, R
_ → R
x → R
_ → R
0 → R
0 → x, R
qreject
qaccept
q4
x → R
_ → R



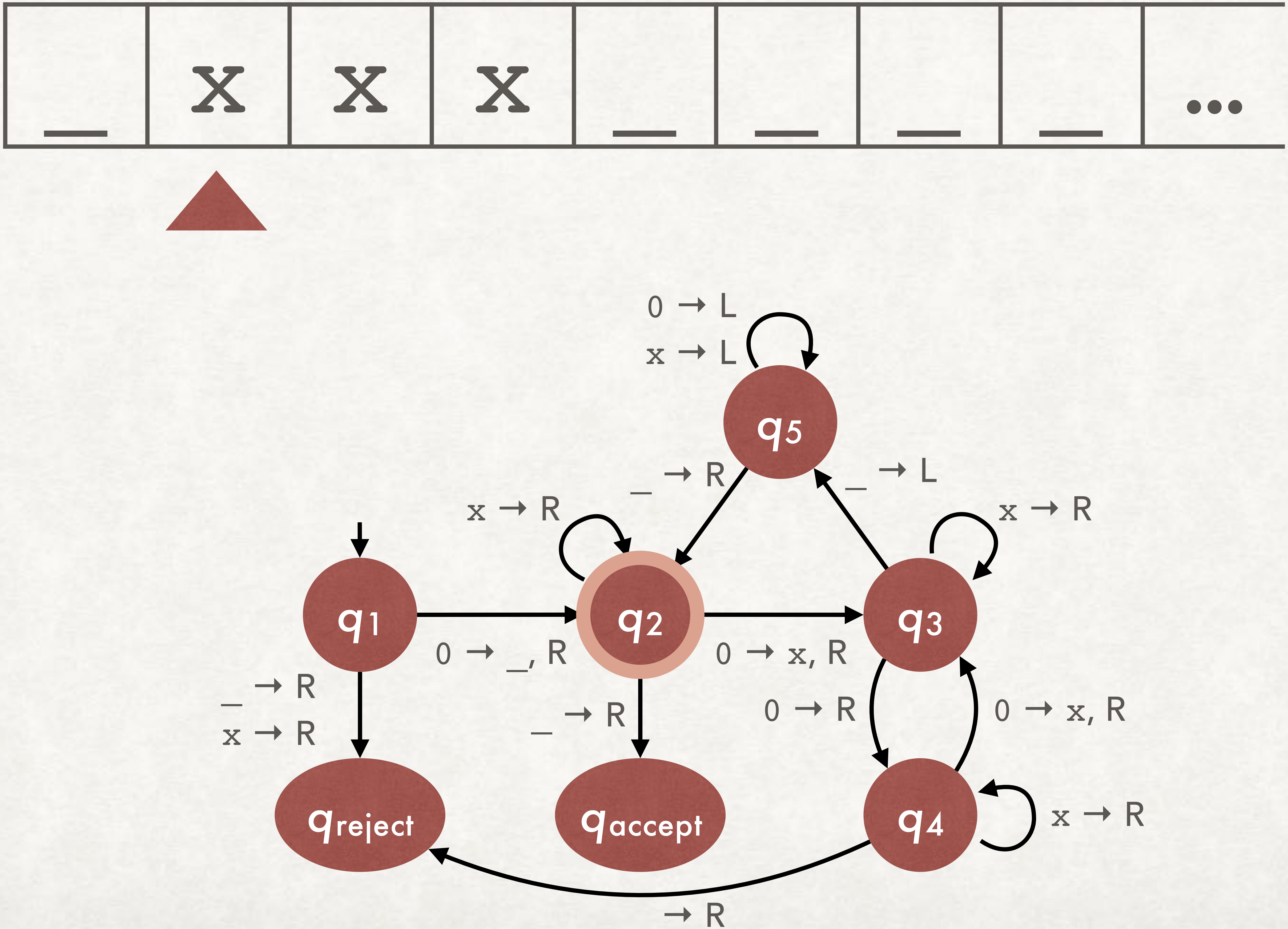
_
x
x
x
_
_
_
_
...
0 → L
x → L
q5
_ → R
_ → L
x → R
x → R
q1
q2
q3
0 → _, R
0 → x, R
_ → R
x → R
_ → R
0 → R
0 → x, R
qreject
qaccept
q4
x → R
_ → R



_
x
x
x
_
_
_
_
...
0 → L
x → L
q5
_ → R
_ → L
x → R
x → R
q1
q2
q3
0 → _, R
0 → x, R
_ → R
x → R
_ → R
0 → R
0 → x, R
qreject
qaccept
q4
x → R
_ → R

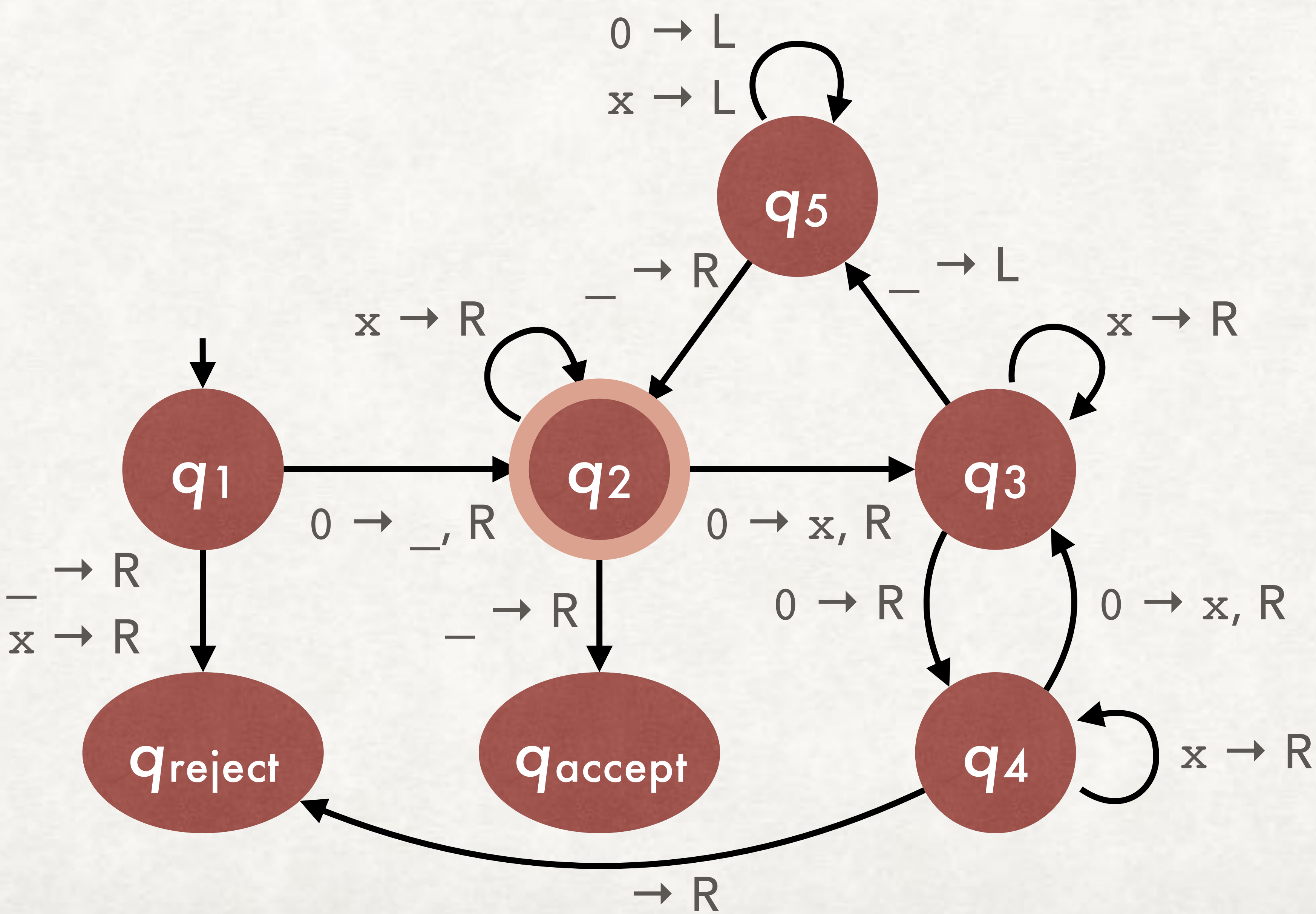


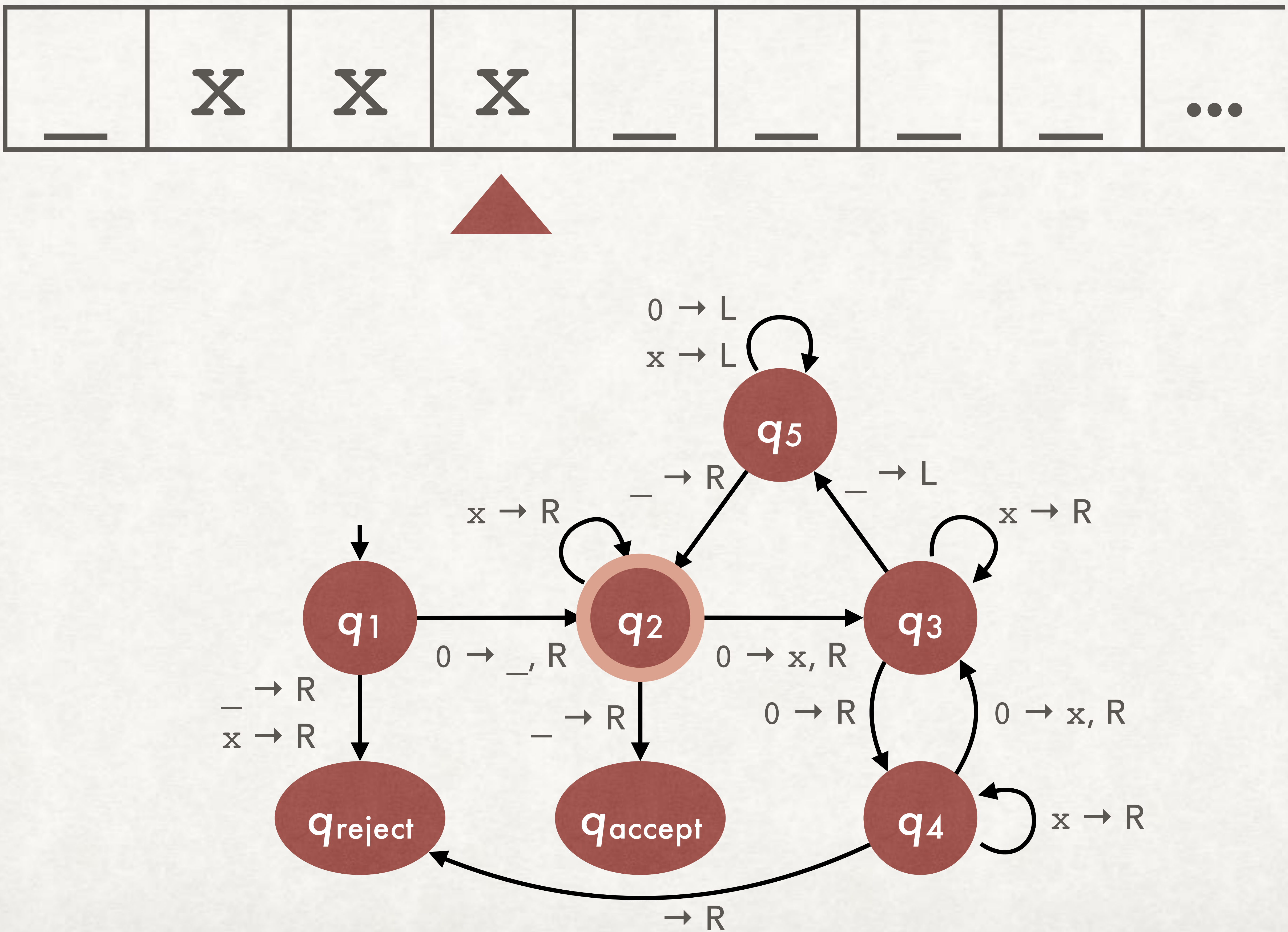
_
x
x
x
_
_
_
_
...
0 → L
x → L
q5
_ → R
_ → L
x → R
x → R
q1
q2
q3
0 → _, R
0 → x, R
_ → R
x → R
_ → R
0 → R
0 → x, R
qreject
qaccept
q4
x → R
_ → R



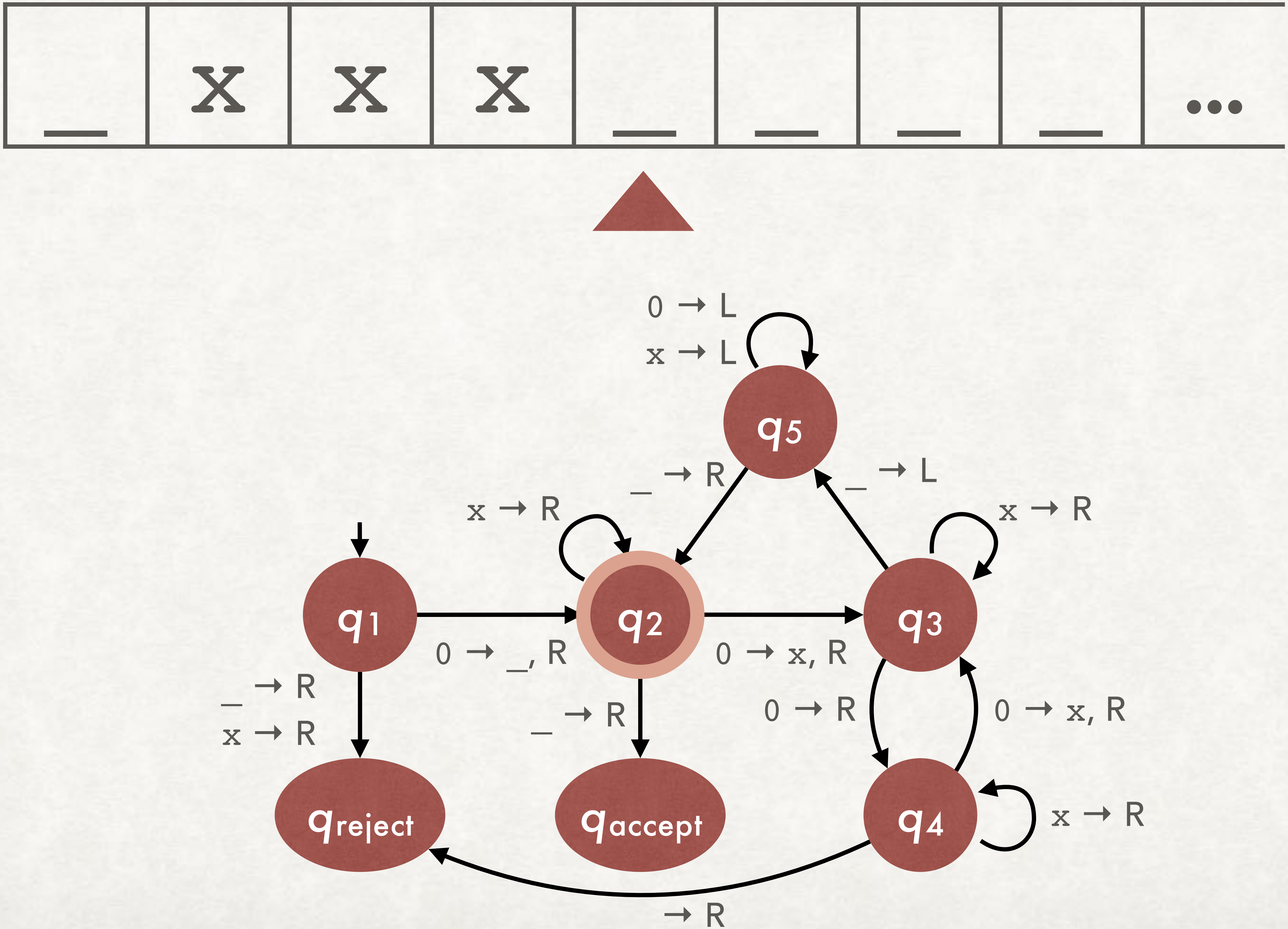
_
x
x
x
_
_
_
_
...
0 → L
x → L
q5
_ → R
_ → L
x → R
x → R
q1
q2
q3
0 → _, R
0 → x, R
_ → R
x → R
_ → R
0 → R
0 → x, R
qreject
qaccept
q4
x → R
_ → R

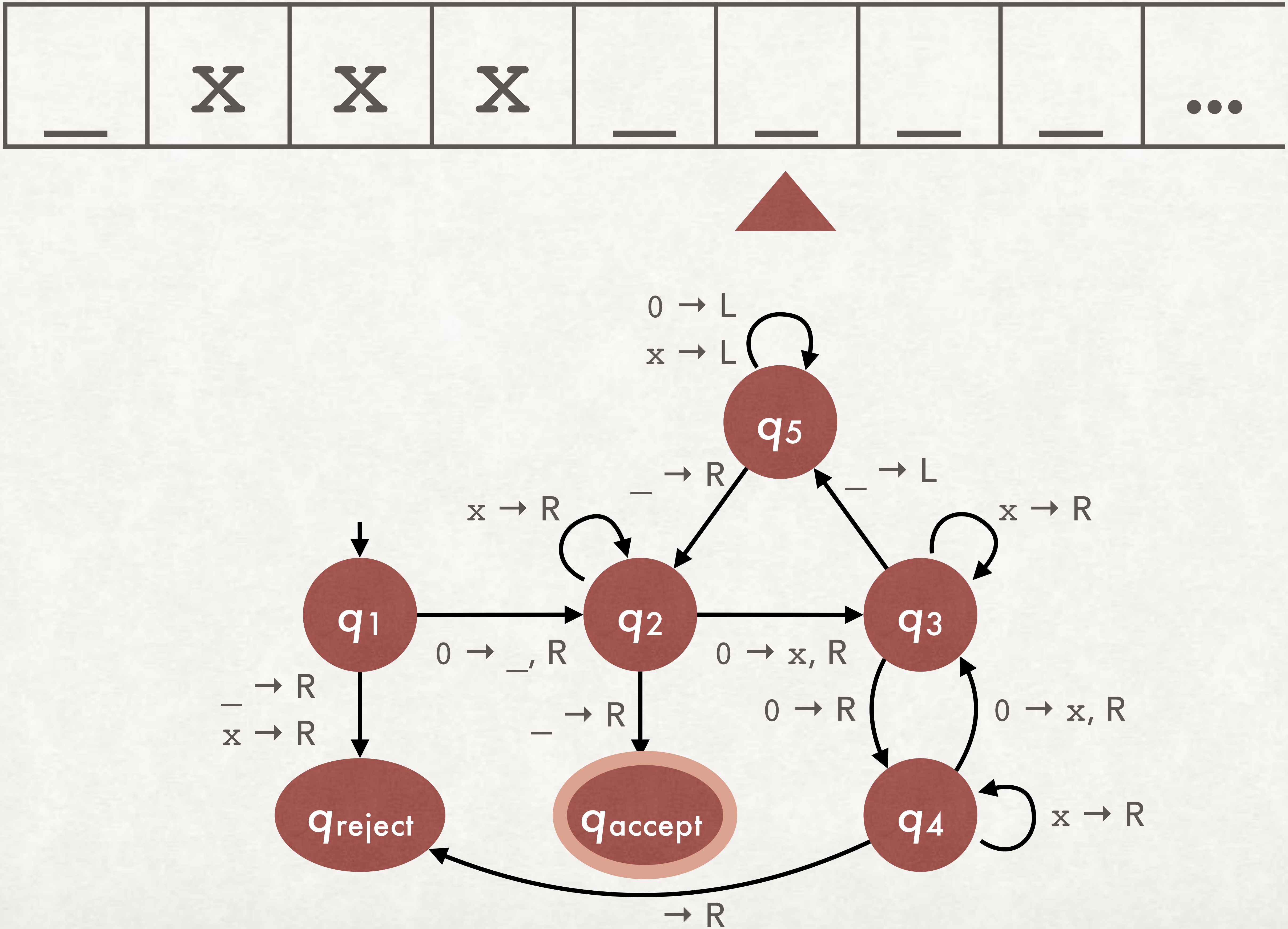

_
x
x
x
_
_
_
_
...


0 → L
x → L
q5
_ → R
_ → L
x → R
x → R
q1
q2
q3
0 → _, R
0 → x, R
_ → R
x → R
_ → R
0 → R
0 → x, R
qreject
qaccept
q4
x → R
_ → R



_
x
x
x
_
_
_
_
…
0 → L
x → L
q5
_ → R
_ → L
x → R
x → R
q1
q2
q3
0 → _, R
0 → x, R
_ → R
x → R
_ → R
0 → R
0 → x, R
qreject
qaccept
q4
x → R
_ → R


_
x
x
x
_
_
_
_
...
0 → L
x → L
q5
_ → R
_ → L
x → R
x → R
q1
q2
q3
0 → _, R
0 → x, R
_ → R
x → R
_ → R
0 → R
0 → x, R
qreject
qaccept
q4
x → R
_ → R



_
x
x
x
_
_
_
_
...
0 → L
x → L
q5
_ → R
_ → L
x → R
x → R
q1
q2
q3
0 → _, R
0 → x, R
_ → R
x → R
_ → R
0 → R
0 → x, R
qreject
qaccept
q4
x → R
_ → R

TURING MACHINES

YOUR TURN

Write a state diagram for a Turing machine recognizing the language

$\{a^{2n} \mid n \geq 0\}$.

TURING MACHINES

YOUR TURN

Write an implementation description, then a state diagram for a Turing machine recognizing the language

$\{ww^R \mid w \in \{0,1\}^*\}$.

THURSDAY, 2018/03/22
READING: SIPSER 3.2

CHURCH-TURING THESIS

IN MODERN LANGUAGE

Intuitive notion of algorithm

=

Turing machine algorithm

CHURCH-TURING THESIS

WHY SHOULD WE BELIEVE IT?

- Turing's original argument
- Convergence of several proposed models
  - Turing machines (1936)
  - Untyped lambda calculus (1936)
  - Partial recursive functions (1920, 1935, 1952)
  - Unrestricted (type 0) grammars (1956)

CHURCH-TURING THESIS

1+1=2 IN LAMBDA CALCULUS

$$(\lambda m.\lambda n.\lambda f.\lambda x.mf(nfx))\,\underline{(\lambda f.\lambda x.fx)}\,(\lambda f.\lambda x.fx)$$

$$(\lambda n.\lambda f.\lambda x.(\lambda f.\lambda x.fx)f(nfx))\,\underline{(\lambda f.\lambda x.fx)}$$

$$\lambda f.\lambda x.(\lambda f.\lambda x.fx)f((\lambda f.\lambda x.fx)fx)$$

$$\lambda f.\lambda x.(\lambda x.fx)\,\underline{((\lambda f.\lambda x.fx)fx)}$$

$$\lambda f.\lambda x.f((\lambda f.\lambda x.fx)fx)$$

$$\lambda f.\lambda x.f((\lambda x.fx)x)$$

$$\lambda f.\lambda x.f(fx)$$

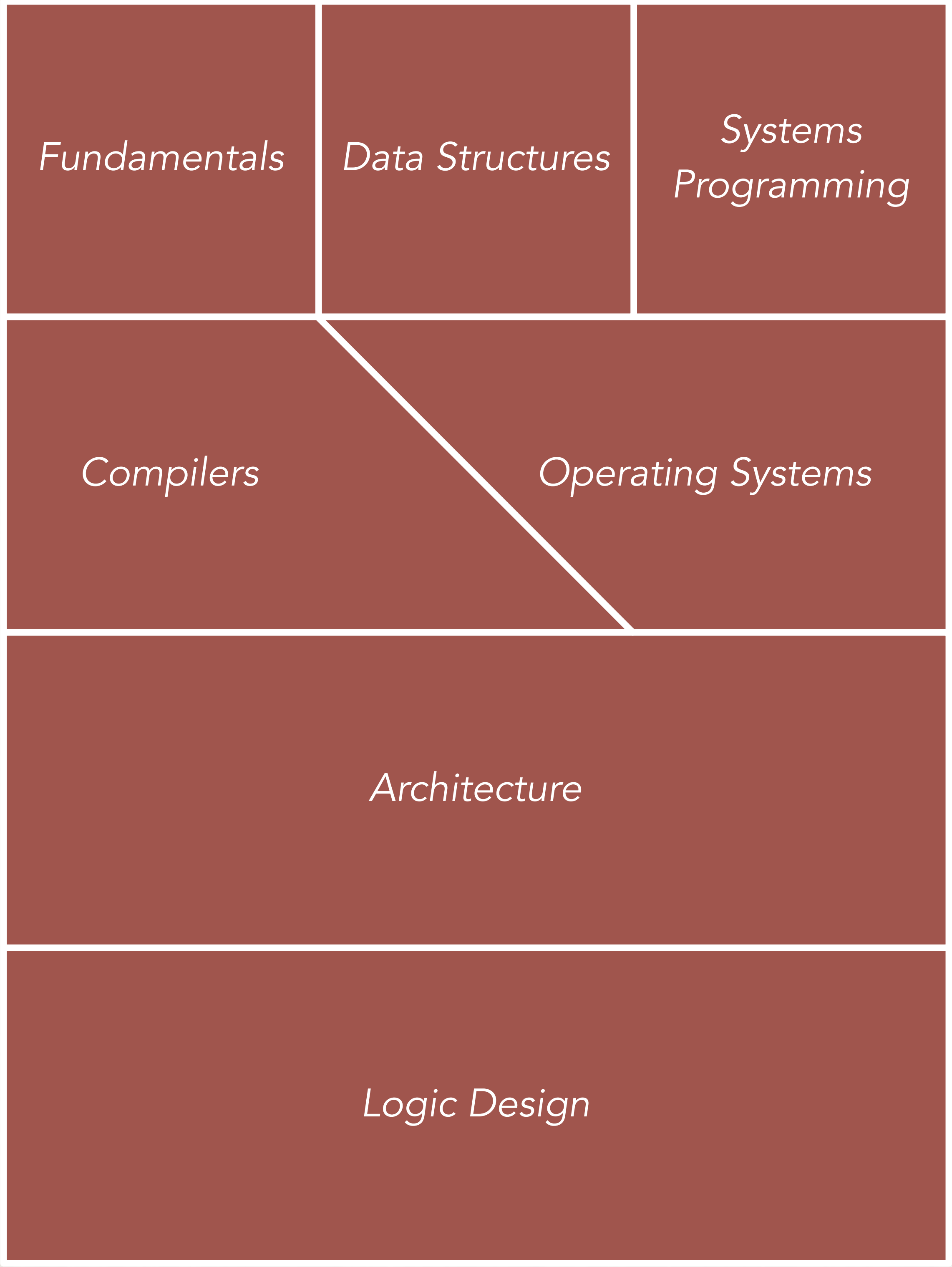
CHURCH-TURING THESIS

WHY SHOULD WE BELIEVE IT?

- Turing's original argument
- Convergence of several proposed models
  - Turing machines (1936)
  - Untyped lambda calculus (1936)
  - Partial recursive functions (1920, 1935, 1952)
  - Unrestricted (type 0) grammars (1956)
- Today: Explore extensions to Turing machines

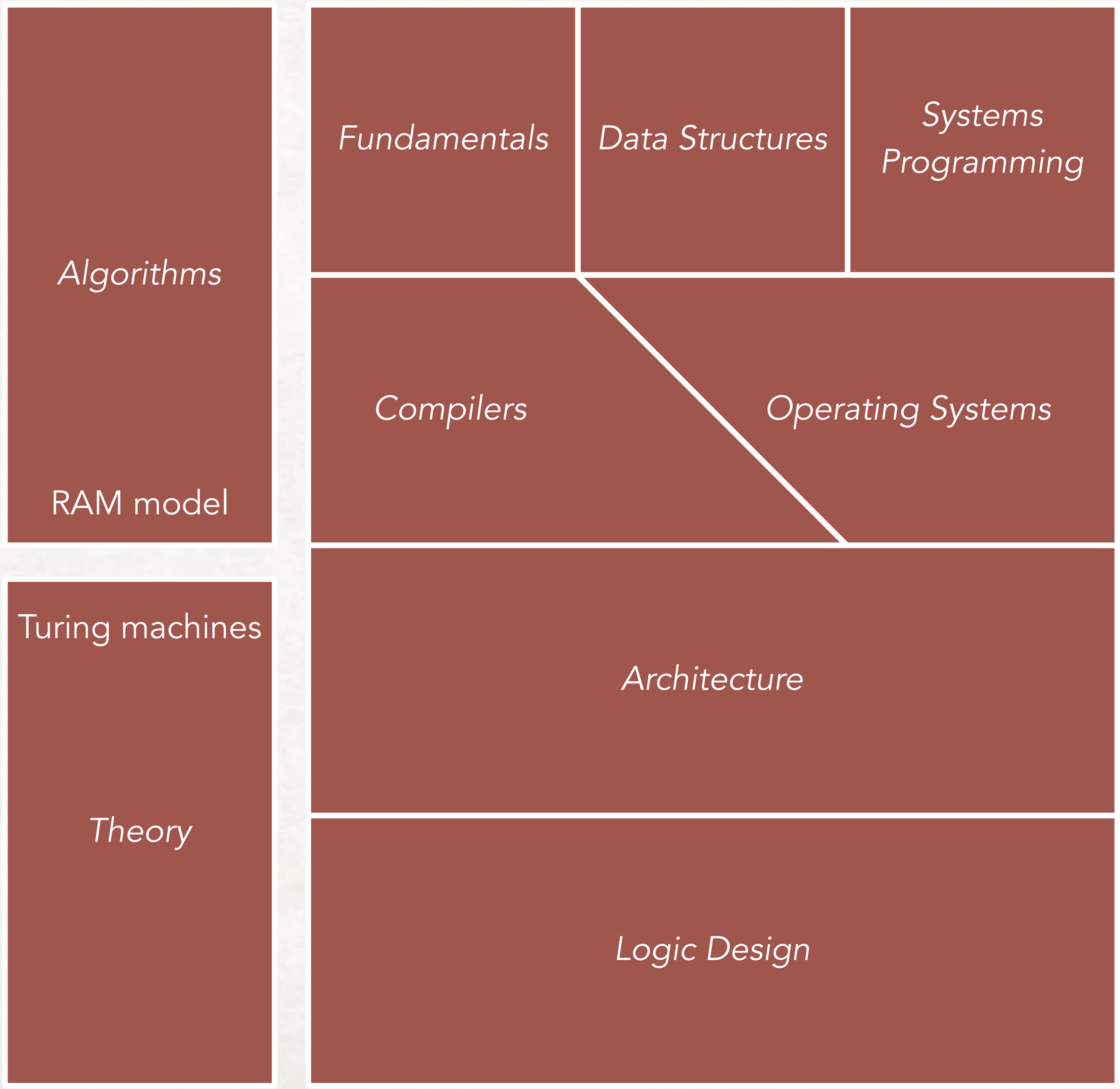
ALL OF UNDERGRADUATE COMPUTER SCIENCE

ACCORDING TO ME



ALL OF UNDERGRADUATE COMPUTER SCIENCE

ACCORDING TO ME



TURING MACHINES

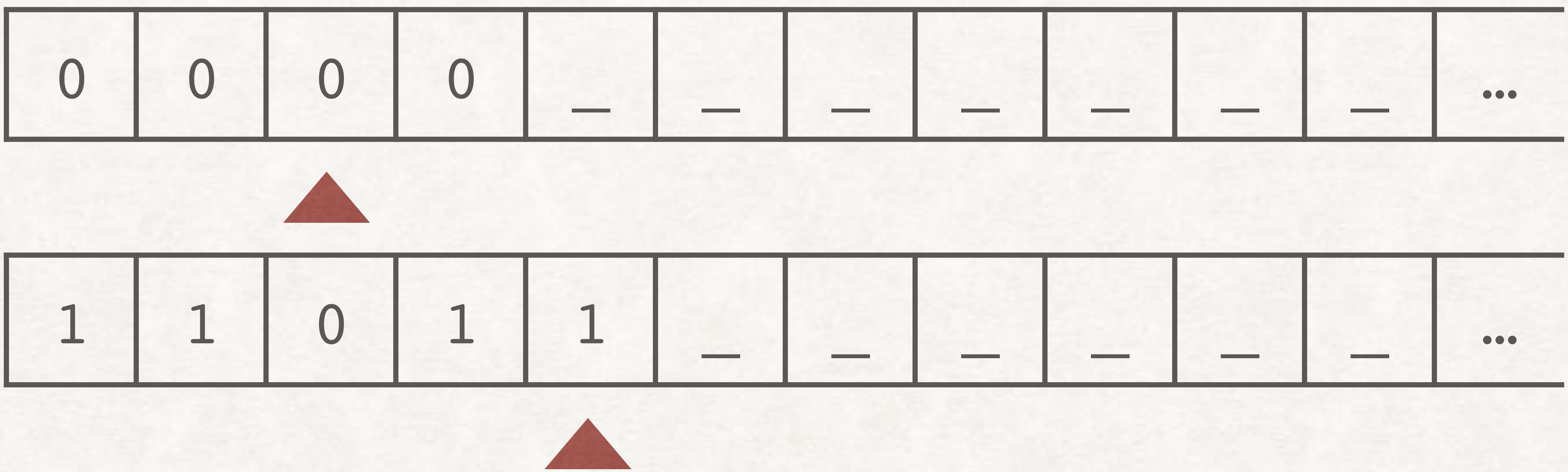
DISCUSS

What do computers (or computer languages) have that Turing Machines don't?

variables	output of strings, numbers, etc.
numbers, arithmetic	output, e.g.,graphics, sound, music
process one character at a time	input, e.g., mouse, keyboard
loops, if/then/else	network
functions	
data structures	
random access memory	
concurrency	
classes	

MULTITAPE TURING MACHINES

IDEA



- Fixed (usually small) number of **tapes**
- One **head** per tape, each moving independently
- Single global **state**

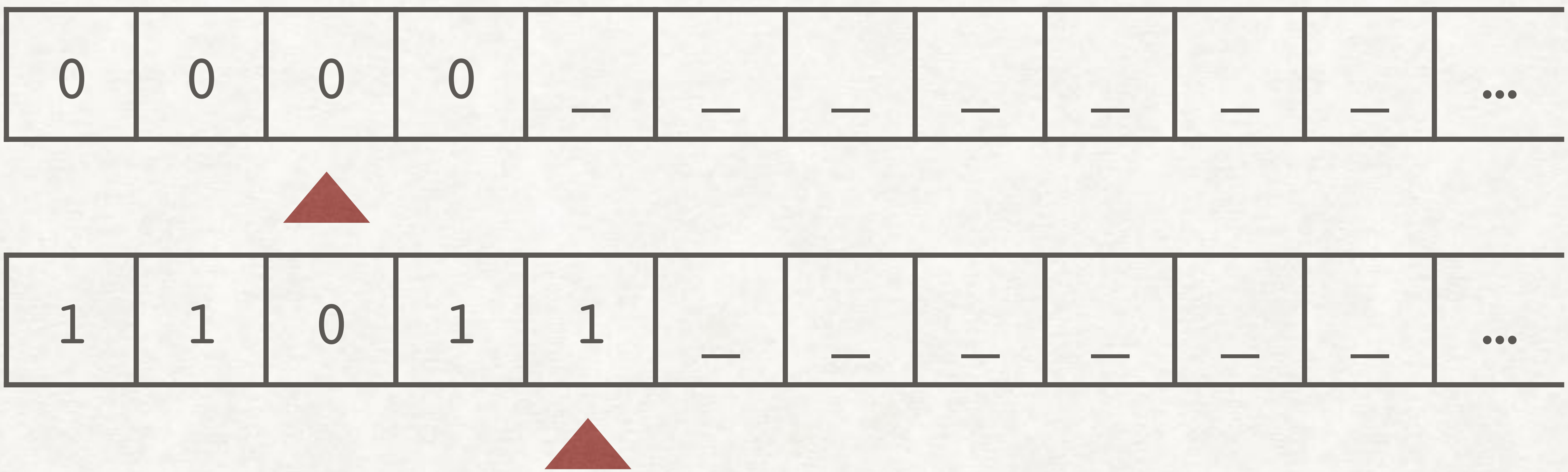
MULTITAPE TURING MACHINES

EQUIVALENCE WITH SINGLE-TAPE

How do you convert a multitape Turing machine into an equivalent single-tape Turing machine?

MULTITAPE TURING MACHINES

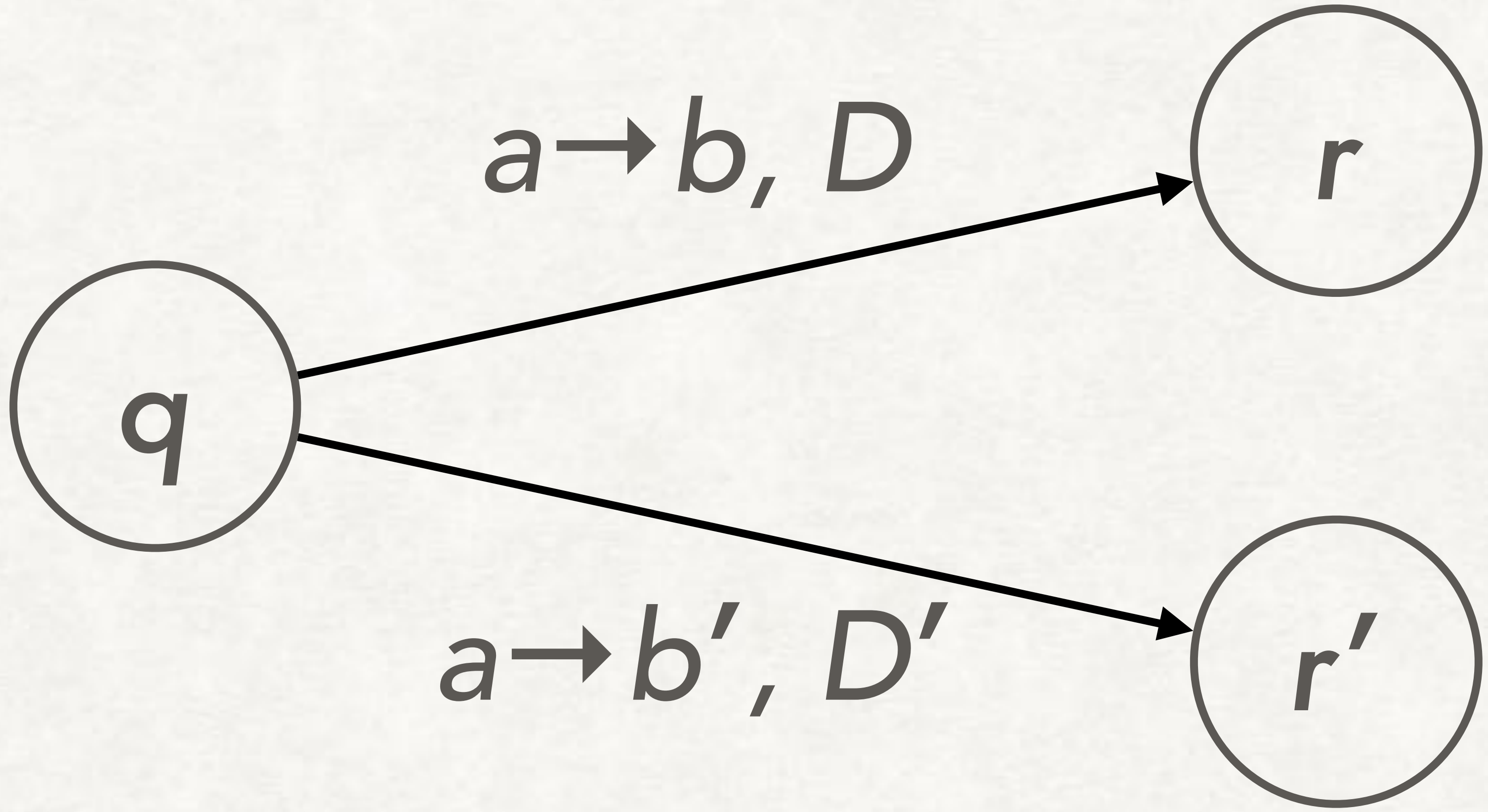
EQUIVALENCE WITH SINGLE-TAPE



becomes

NONDETERMINISTIC TURING MACHINES

IDEA



Machine will follow *both* transitions in two computation branches

NONDETERMINISTIC TURING MACHINES

IDEA

accept and halt	when *any* branch enters q_{accept}
reject and halt	when *all* branches enter q_{reject}
loop	otherwise

How do you convert a nondeterministic Turing machine into an equivalent deterministic Turing machine?

NONDETERMINISTIC TURING MACHINES

EQUIVALENCE WITH DETERMINISTIC MULTITAPE

0	0	1	0	_	_	_	_	_	_	_	…

x	x	#	0	1	x	_	_	_	_	_	…

1	2	3	3	2	3	1	2	1	1	3	…

Each string here selects a branch: choose #1, then #2, etc.

Enumerate all branches in BFS order: 1, 2, 3, …, 11, 12, 13, …, 21, 22, 23, …

NONDETERMINISTIC TURING MACHINES

EQUIVALENCE WITH DETERMINISTIC SINGLE-TAPE

q_1	0	0	1	0	#	1	q_2	0	1	0	#	…

- Tape holds a queue of simulated configurations
  - State on tape means head is on *next* square
- While front configuration is not accepting:
  - For each possible move in front configuration:
    - Push new configuration to back of queue
  - Pop front configuration

RANDOM ACCESS MACHINES

IDEA

1	0	0	123	5	0	-6	7	1	-88	1	…

$X[i] \leftarrow C$	write constant
$X[i] \leftarrow X[j] + X[k]$	add (also subtract) cells
$X[i] \leftarrow X[X[j]]$	copy from dereferenced cell
$X[X[i]] \leftarrow X[j]$	copy to dereferenced cell
TRA m if $X[j] > 0$	conditional branch

How do you convert a random access machine into an equivalent multitape Turing machine?

RANDOM ACCESS MACHINES

EQUIVALENCE WITH TURING MACHINES

1	0	0	123	5	0	-6	7	1	-88	1	…

becomes

$	*	0	#	1	*	1	1	#	1	1	…

- Cook-Reckhow targeted a multitape TM: additional tapes for an address register, a value register, and scratch space?
- I don't know how they represented negative numbers

RANDOM ACCESS MACHINES

EQUIVALENCE WITH TURING MACHINES

Demo:

C code → ELVM assembly → Turing machine

https://github.com/shinh/elvm

TURING MACHINES

WHAT'S NEXT?

- The most powerful Turing machine
- Is there a language that is undecidable?
- What other languages are undecidable?
- Is there really nothing beyond Turing machines?