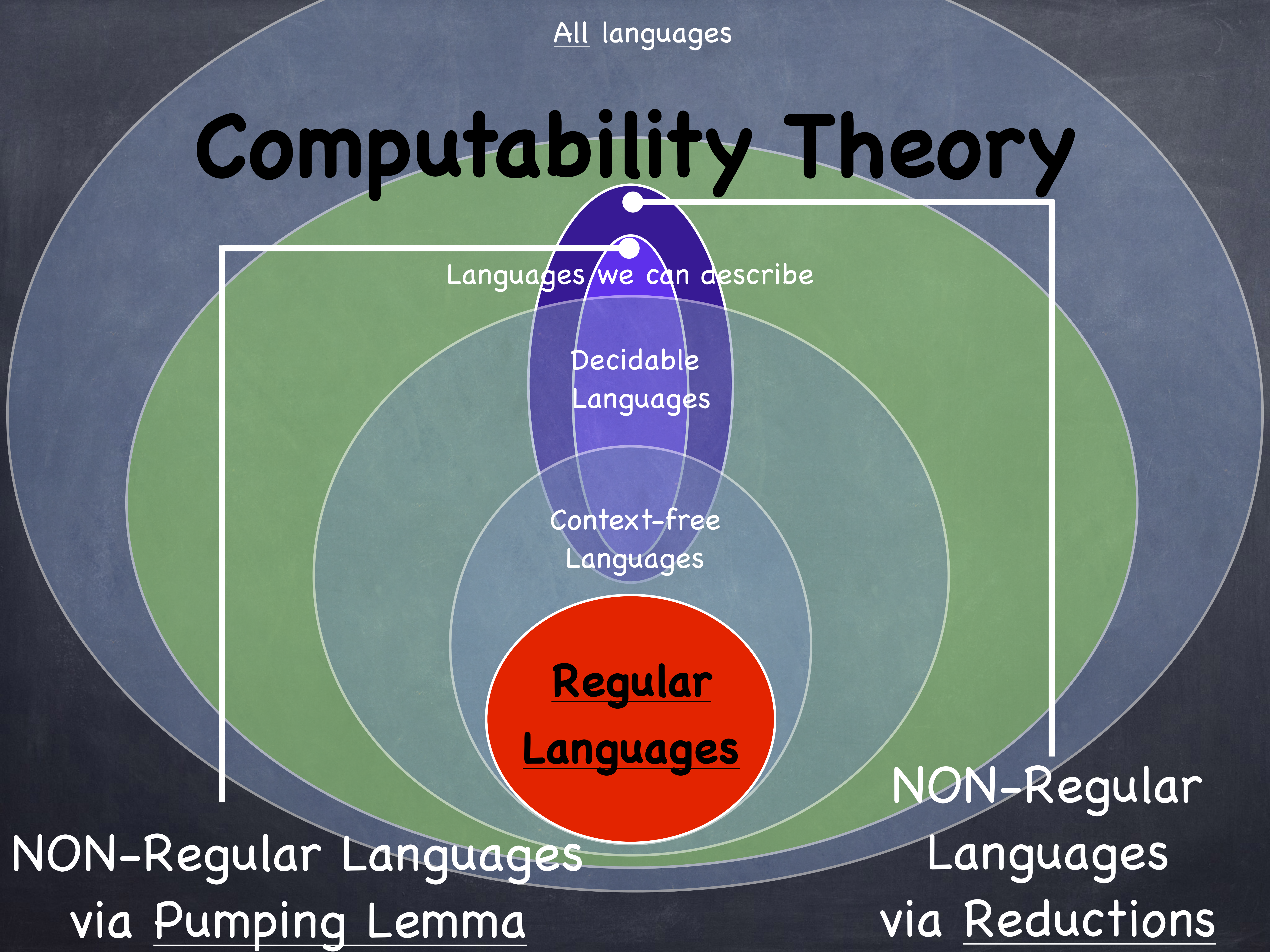
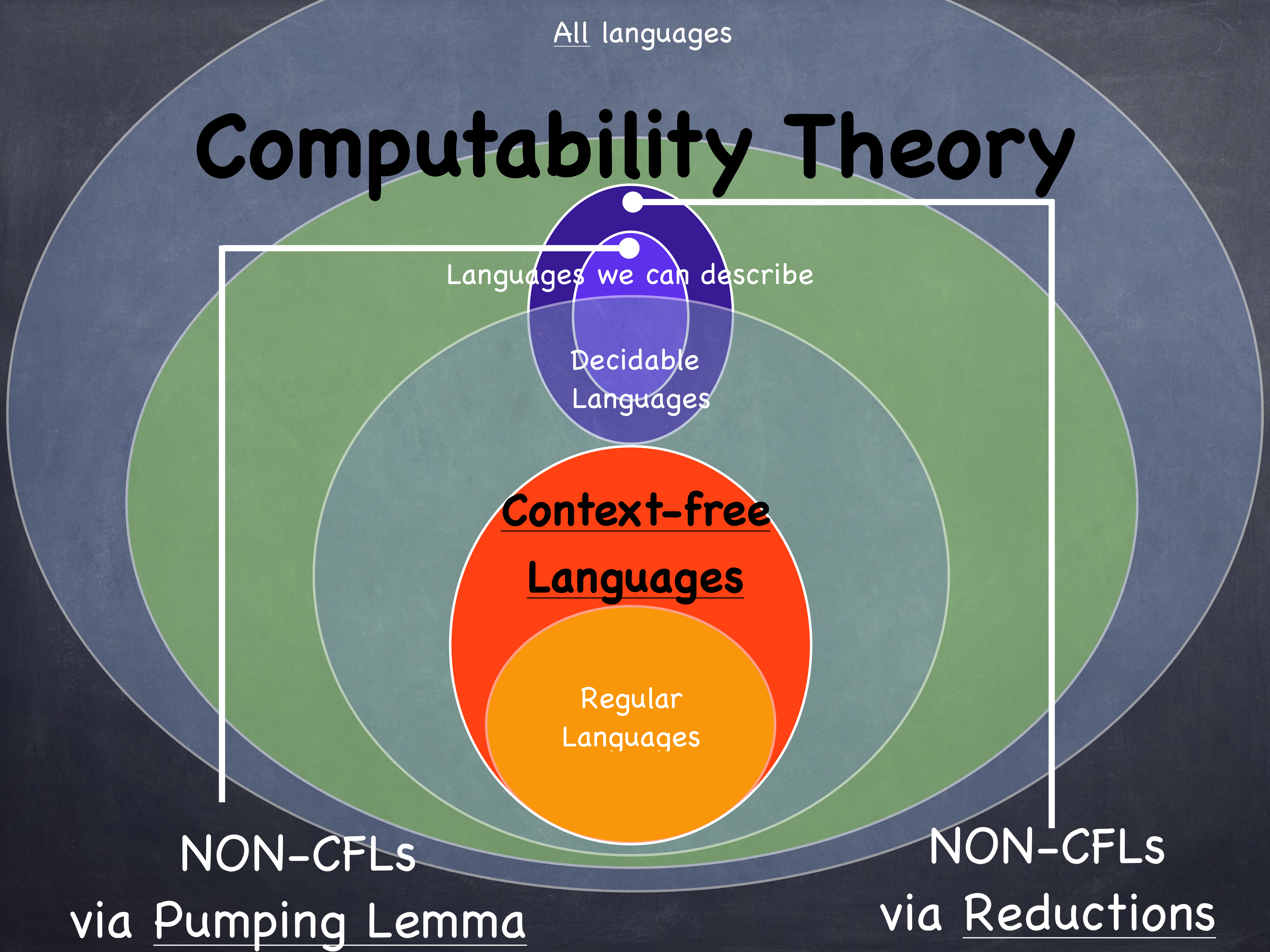
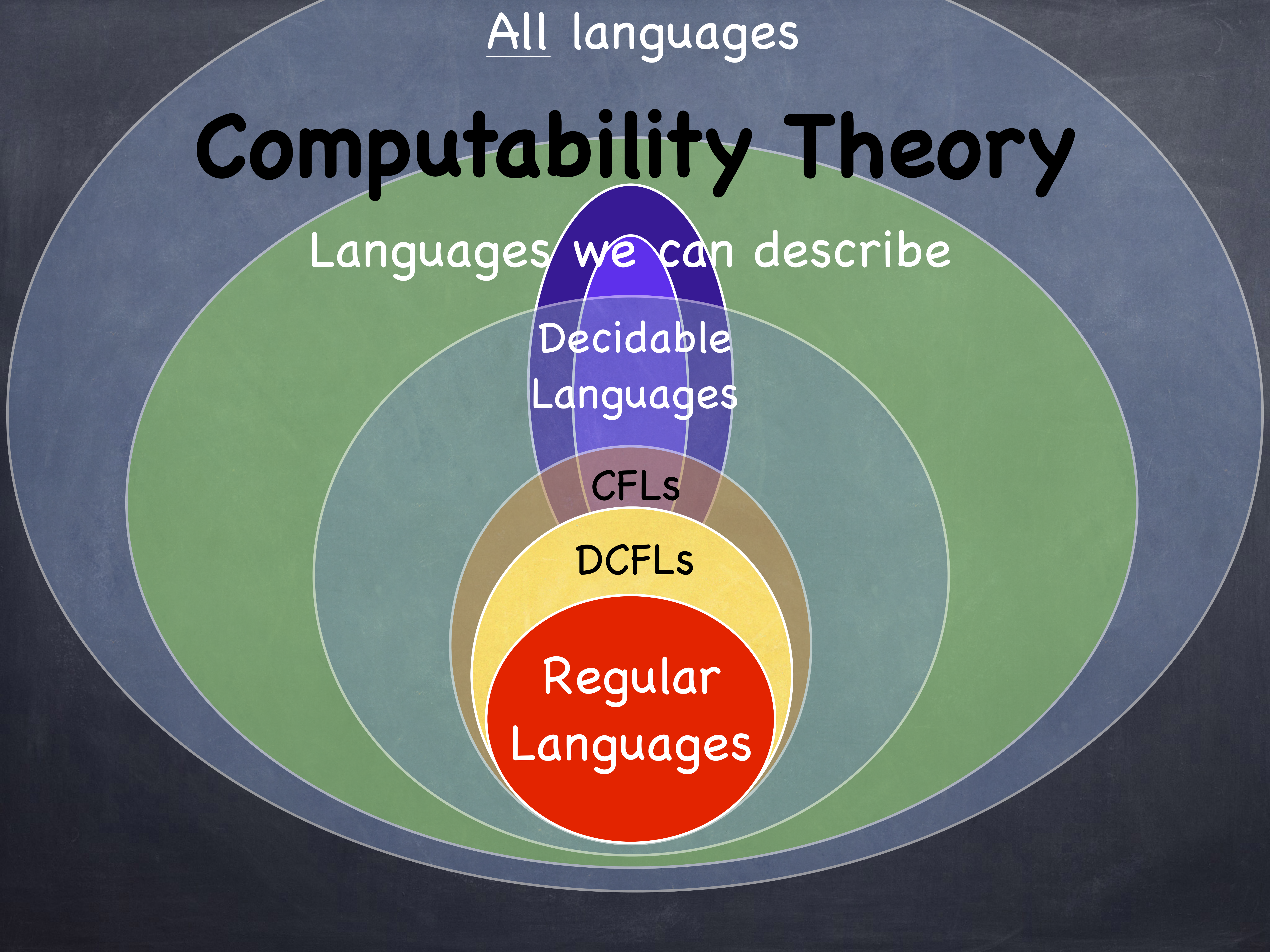
COMP-330
Theory of Computation

Fall 2019 -- Prof. Claude Crépeau

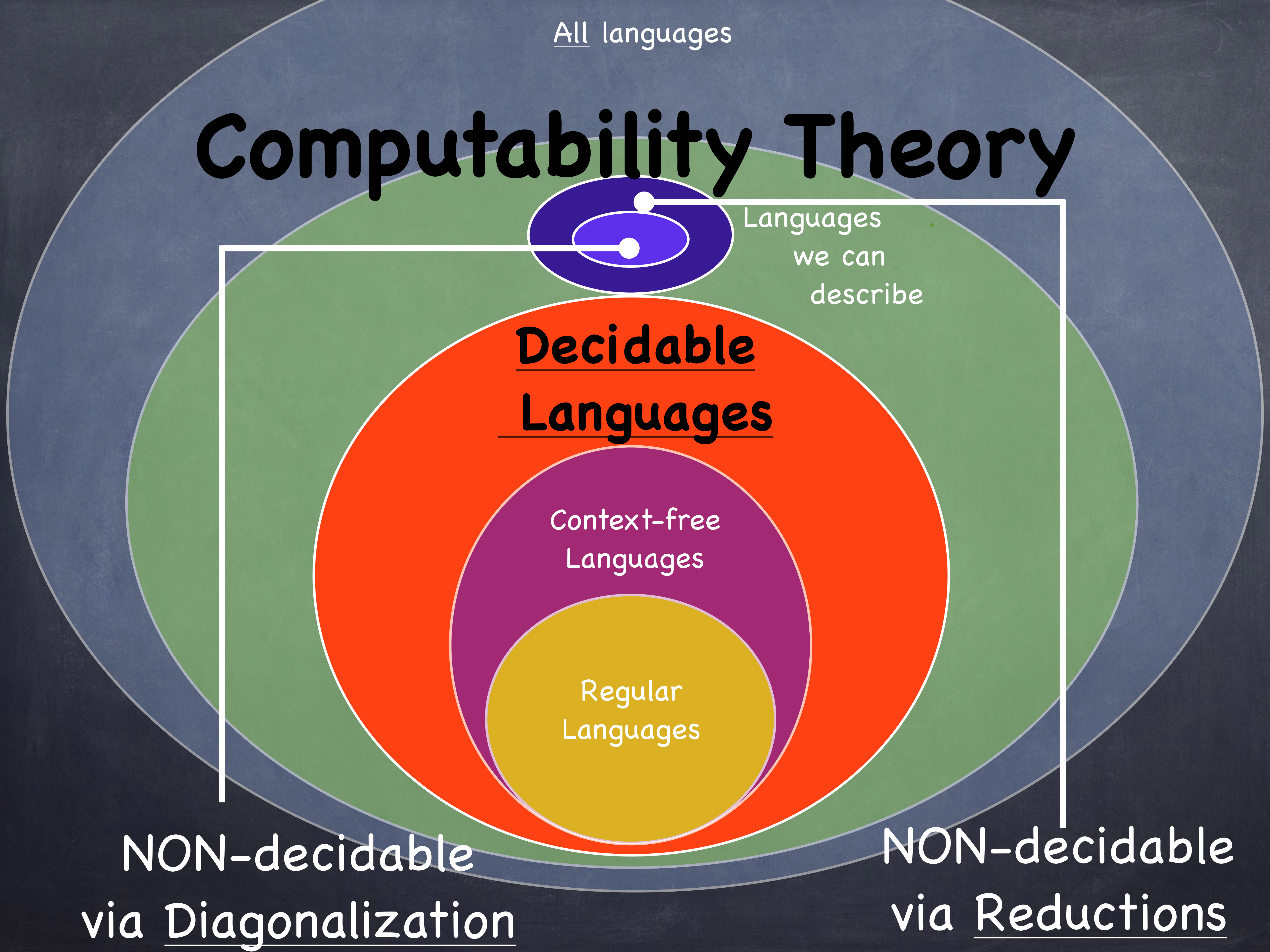
Lec. 16-17 :
Turing Machines &
Church-Turing Thesis


All languages
Computability Theory
Languages we can describe
Decidable Languages
Context-free Languages
Regular Languages
NON-Regular Languages via Pumping Lemma
NON-Regular Languages via Reductions


All languages
Computability Theory
Languages we can describe
Decidable Languages
Context-free Languages
Regular Languages
NON-CFLs via Pumping Lemma
NON-CFLs via Reductions

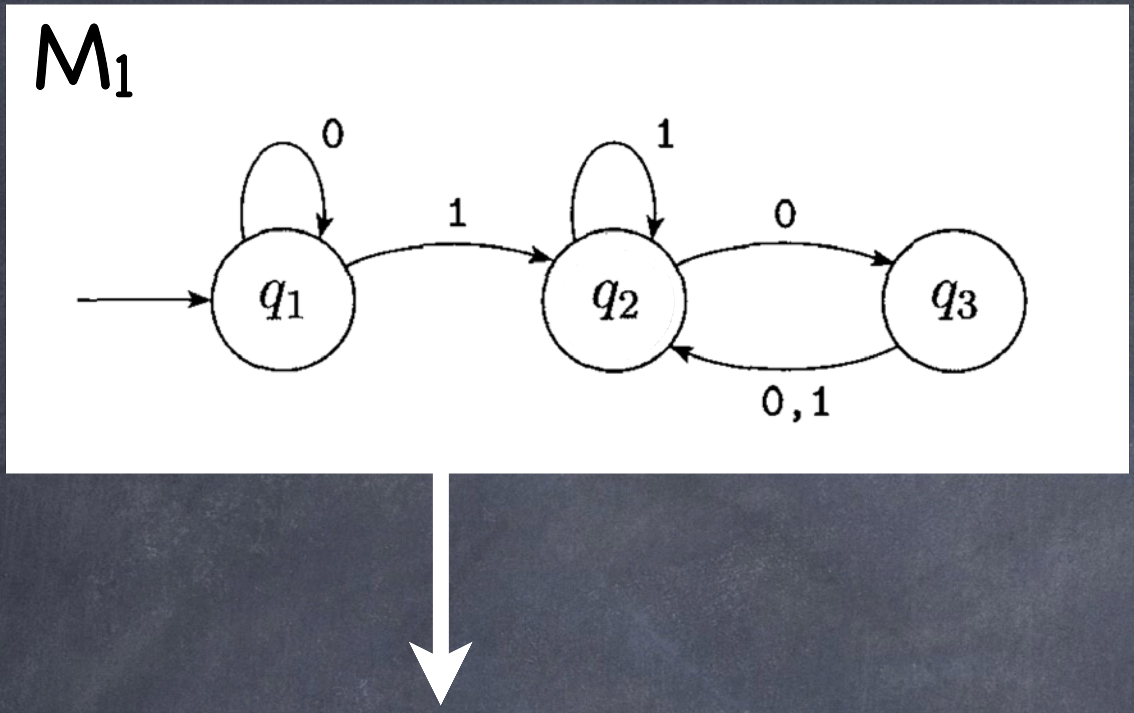

All languages
Computability Theory
Languages we can describe
Decidable
Languages
CFLs
DCFLs
Regular
Languages

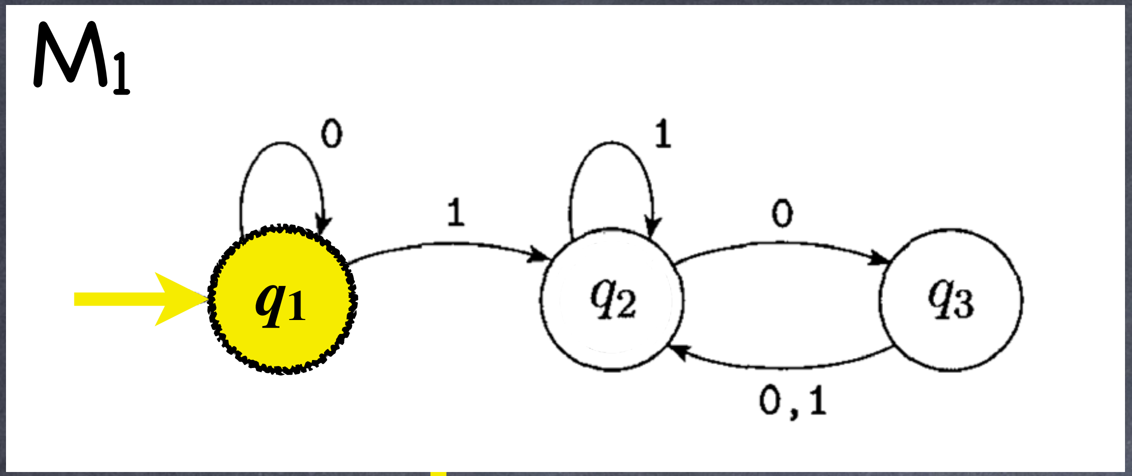
Computability Theory



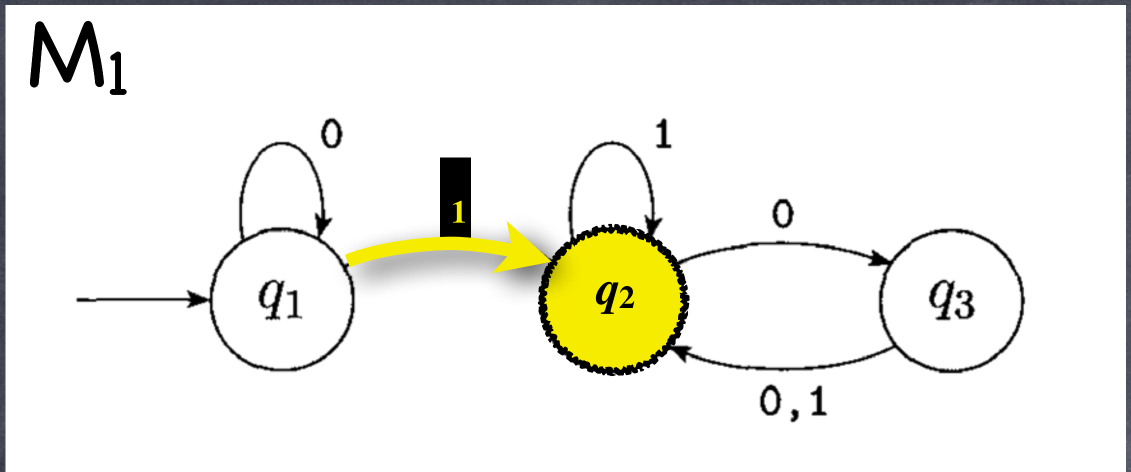
Turing MACHINES

Alan Turing

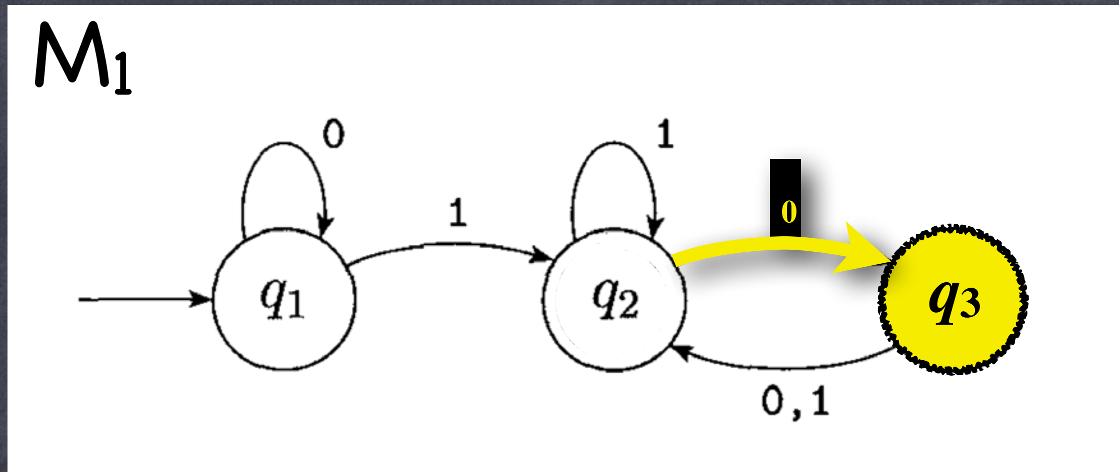

M1
0
1
1
0
q1
q2
q3
0,1



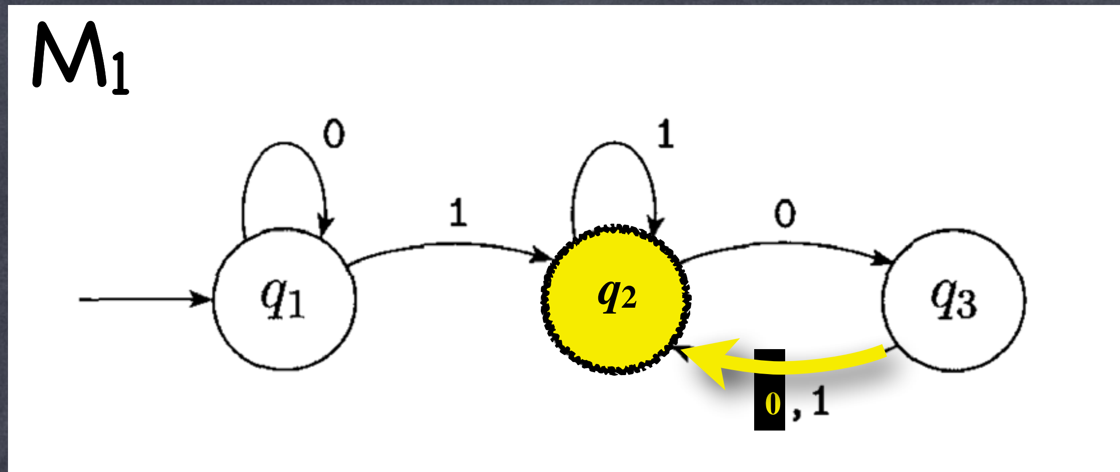
10010101



10010101



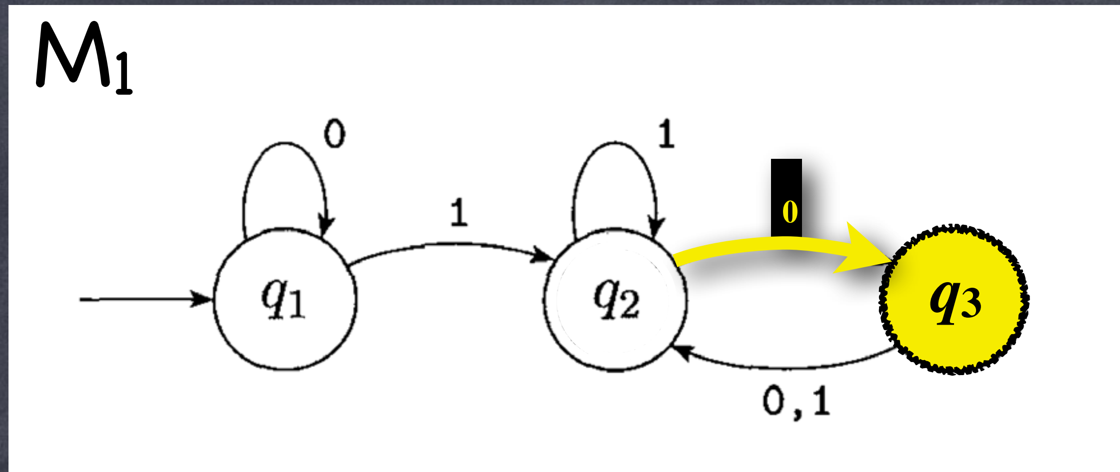
10010101



10110101

read and write!

moves Right and Left!



10110101

Turing Machines

The following list summarizes the differences between finite automata and Turing machines.

1. A Turing machine can both write on the tape and read from it.
2. The read–write head can move both to the left and to the right.
3. The tape is infinite.
4. The special states for rejecting and accepting take effect immediately.

TM Example

$M_1 =$ "On input string w:

1. Zig-zag across the tape to corresponding positions on either side of the # symbol to check whether these positions contain the same symbol. If they do not, or if no # is found, *reject*. Cross off symbols as they are checked to keep track of which symbols correspond.
2. When all symbols to the left of the # have been crossed off, check for any remaining symbols to the right of the #. If any symbols remain, *reject*; otherwise, *accept*."

The following figure contains several snapshots of M_1's tape while it is computing in stages 2 and 3 when started on input 011000#011000.

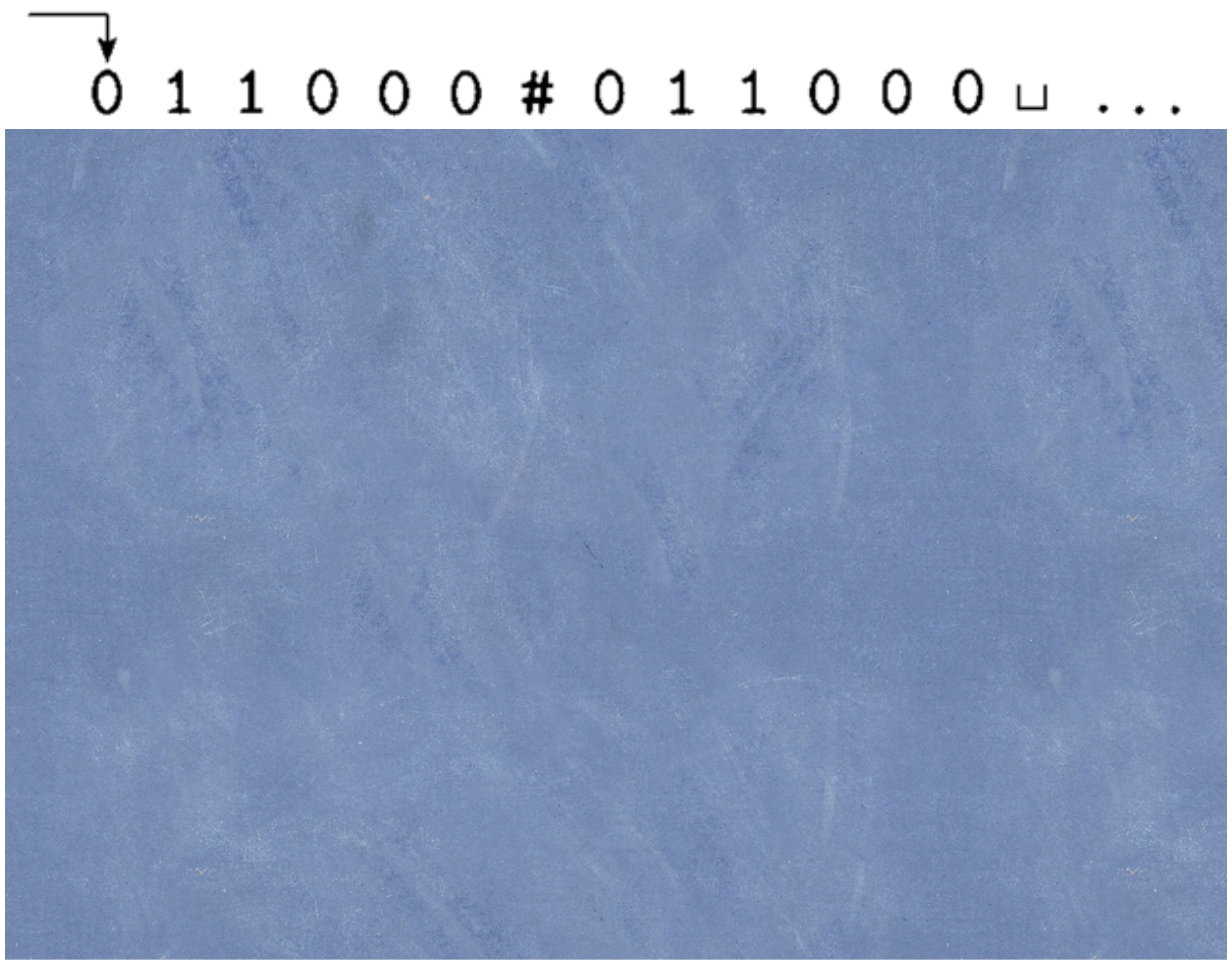


FIGURE 3.2
Snapshots of Turing machine M_1 computing on input 011000#011000

The following figure contains several snapshots of M_1's tape while it is computing in stages 2 and 3 when started on input 011000#011000.

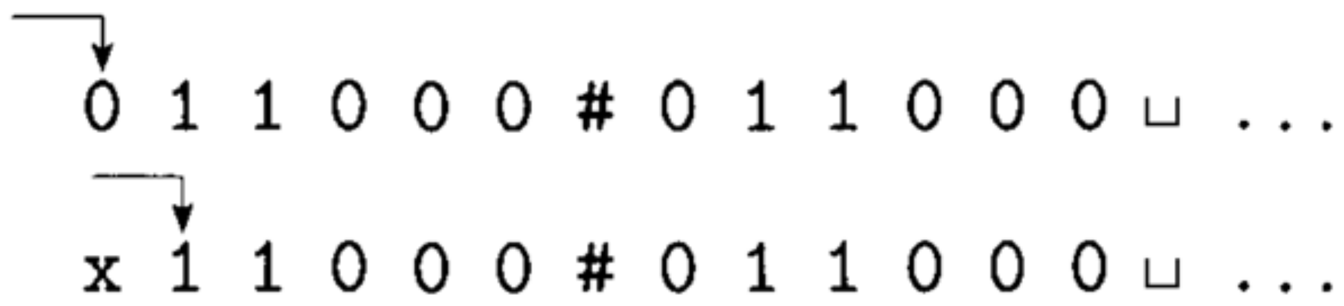


FIGURE 3.2
Snapshots of Turing machine M_1 computing on input 011000#011000

The following figure contains several snapshots of M_1's tape while it is computing in stages 2 and 3 when started on input 011000#011000.

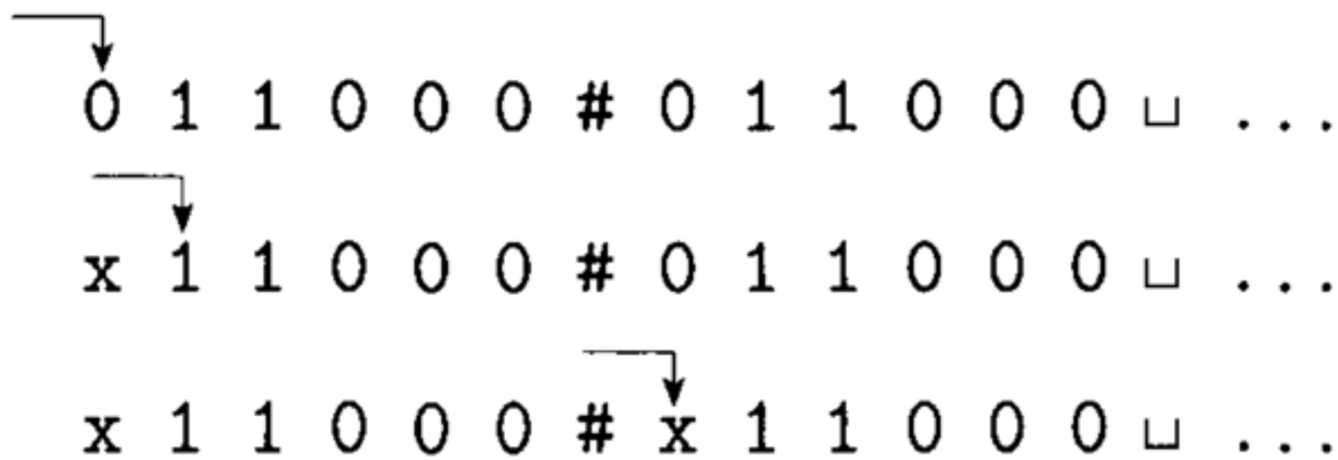


FIGURE 3.2

Snapshots of Tur #011000

The following figure contains several snapshots of M_1's tape while it is computing in stages 2 and 3 when started on input 011000#011000.

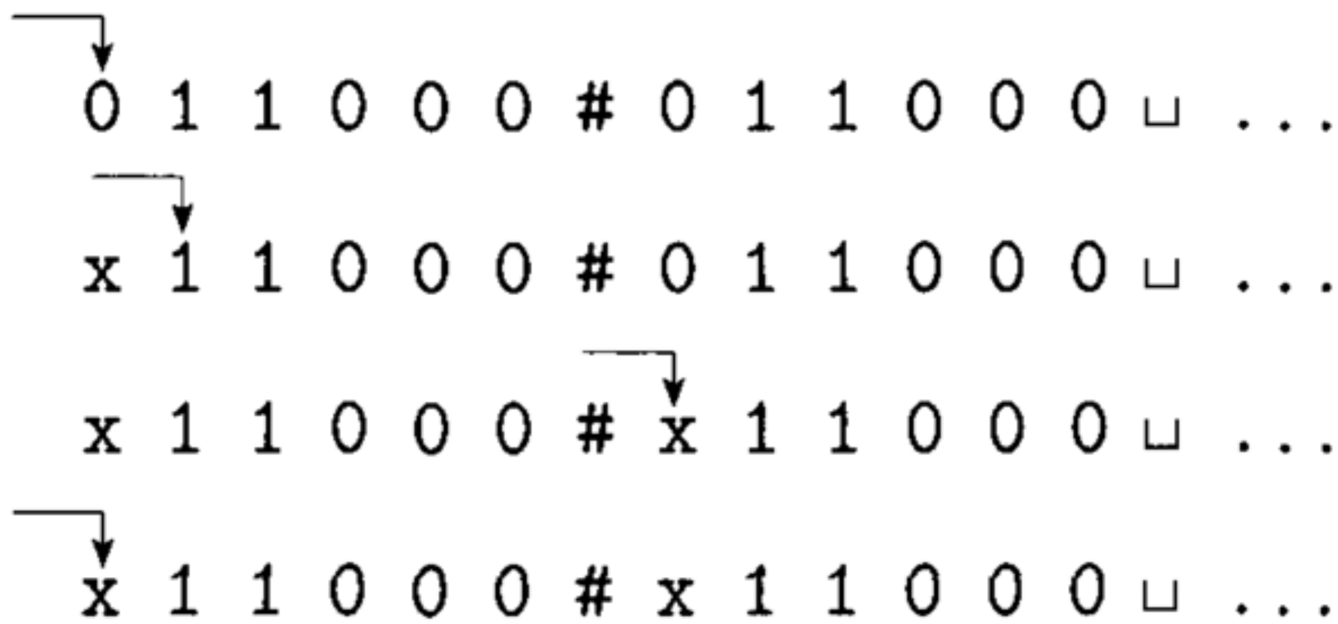


FIGURE 3.2

Snapshots of Tur ... #011000

The following figure contains several snapshots of M_1's tape while it is computing in stages 2 and 3 when started on input 011000#011000.

```
0 1 1 0 0 0 # 0 1 1 0 0 0 ␣ ...
x 1 1 0 0 0 # 0 1 1 0 0 0 ␣ ...
x 1 1 0 0 0 # x 1 1 0 0 0 ␣ ...
x 1 1 0 0 0 # x 1 1 0 0 0 ␣ ...
x x 1 0 0 0 # x 1 1 0 0 0 ␣ ...
```

FIGURE 3.2

Snapshots of Tur #011000

The following figure contains several snapshots of M_1's tape while it is computing in stages 2 and 3 when started on input 011000#011000.

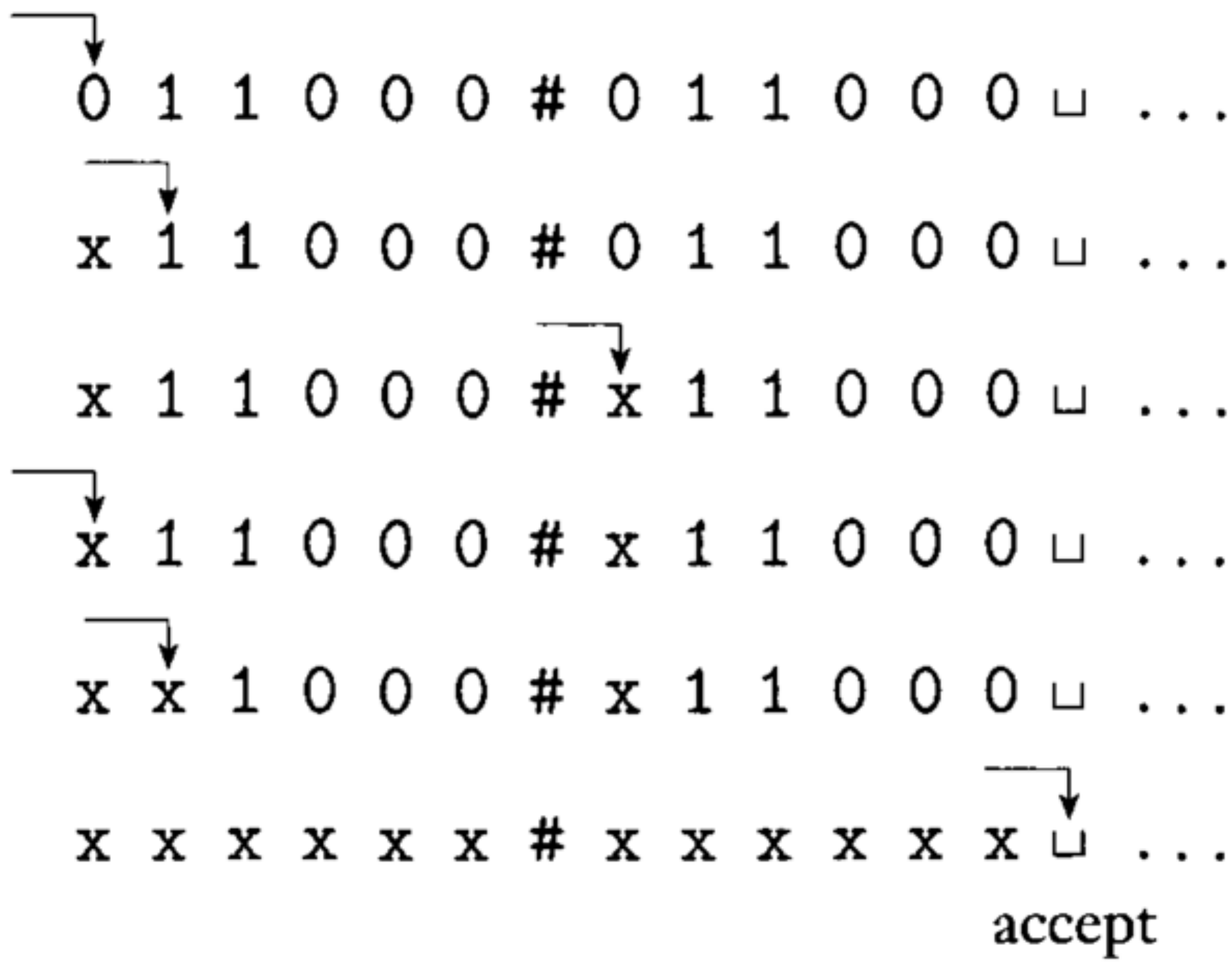


FIGURE 3.2

Snapshots of Tur #011000

Definition of TM

- States
- Input Alphabet
- Tape Alphabet
- Transition function
- Start state
- Accept state
- Reject state

Definition of TM

- States
- Input Alphabet
- Tape Alphabet
- Transition function
- Start state
- Accept state
- Reject state

Definition of TM

- States q_1 q_2 q_3
- Input Alphabet a,b,c
- Tape Alphabet
- Transition function
- Start state
- Accept state
- Reject state

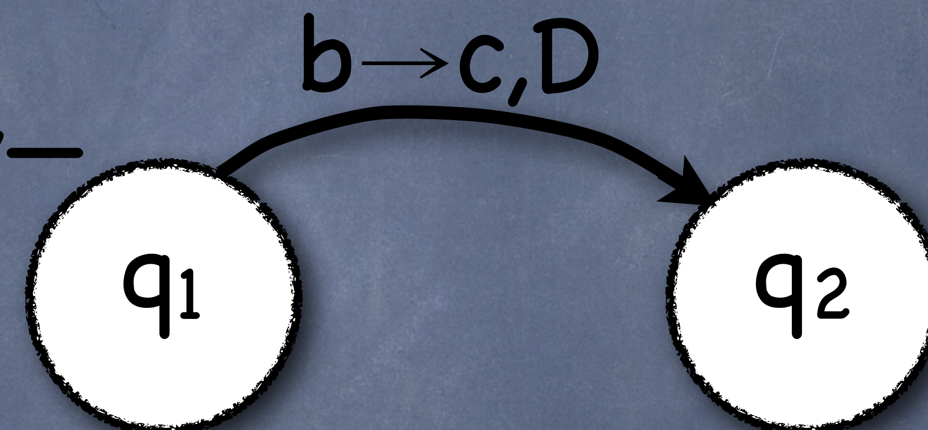
Definition of TM

- States q_1 q_2 q_3
- Input Alphabet a,b,c
- Tape Alphabet a,b,c,A,B,C,_
- Transition function
- Start state
- Accept state
- Reject state

Definition of TM

- States

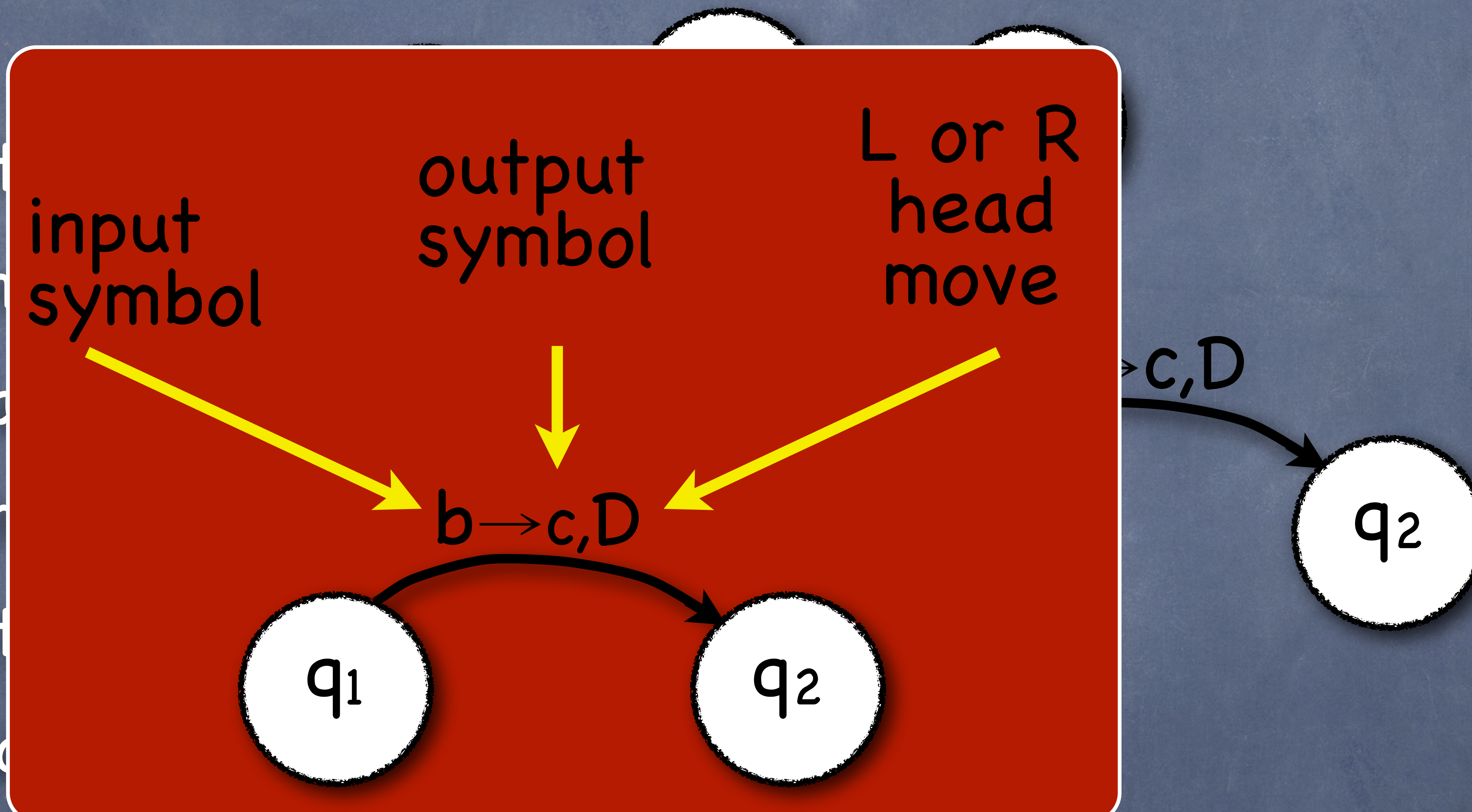
- Input Alphabet a,b,c
- Tape Alphabet a,b,c,A,B,C,_
- Transition function



- Start state
- Accept state
- Reject state

Definition of TM

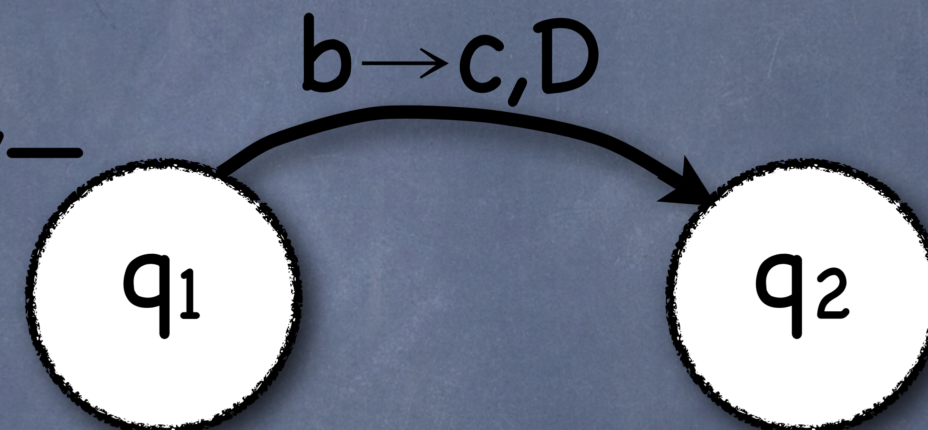
- St
- In
- Ta
- Tr
- St
- Ac
- Reject state



Definition of TM

- States

- Input Alphabet a,b,c
- Tape Alphabet a,b,c,A,B,C,_
- Transition function

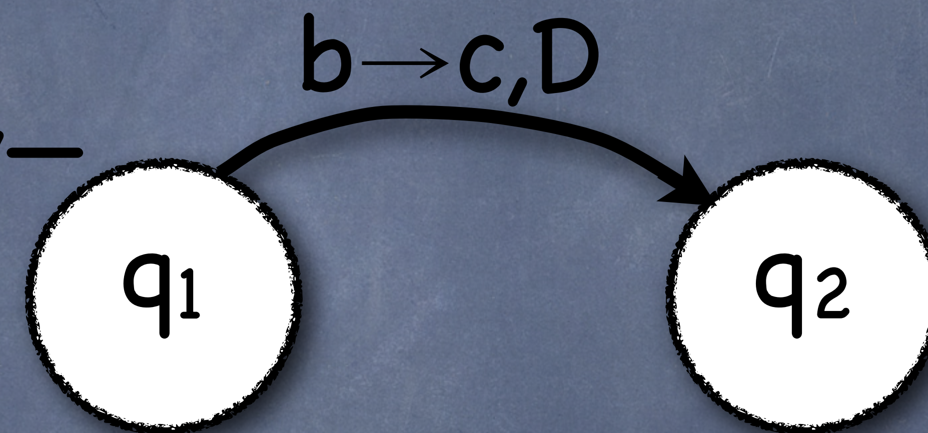


- Start state
- Accept state
- Reject state

Definition of TM

- States

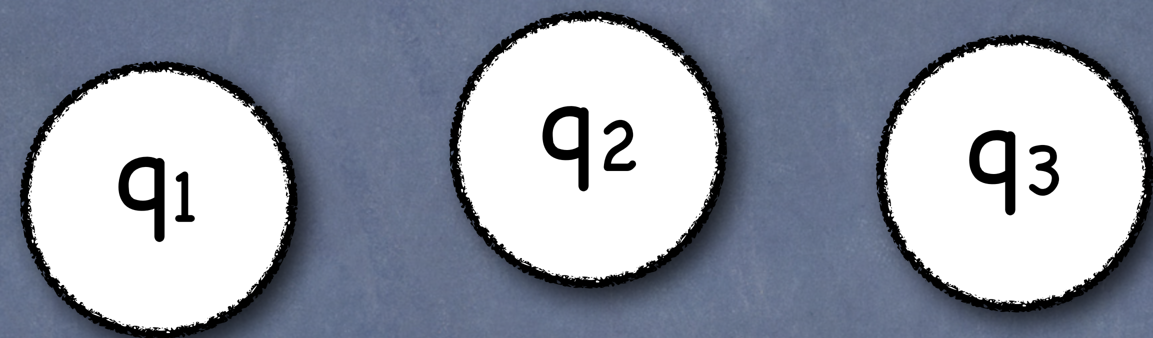
- Input Alphabet a,b,c
- Tape Alphabet a,b,c,A,B,C,_
- Transition function

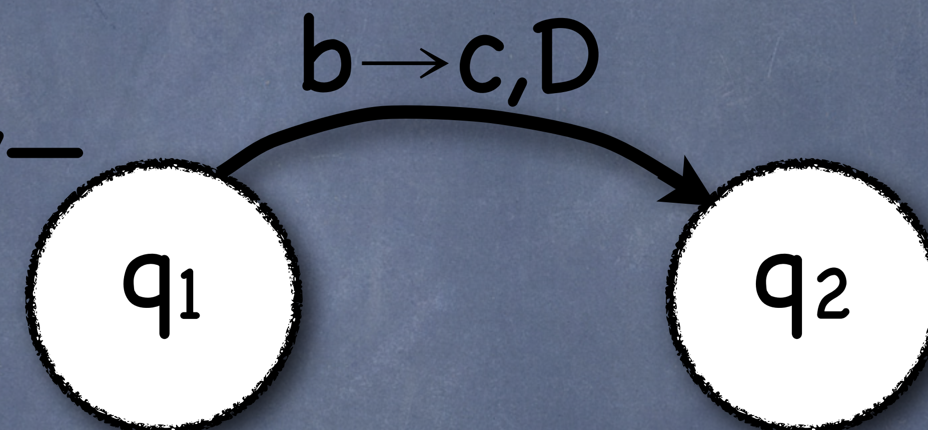


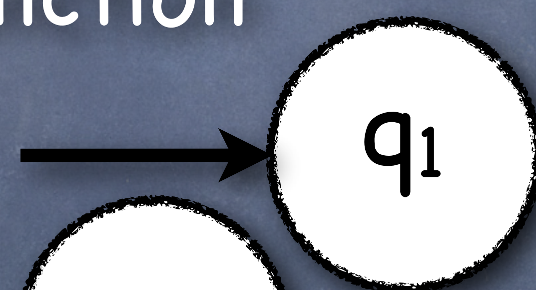
- Start state

- Accept state
- Reject state

Definition of TM

- States


- Input Alphabet a,b,c
- Tape Alphabet a,b,c,A,B,C,_
- Transition function


- Start state


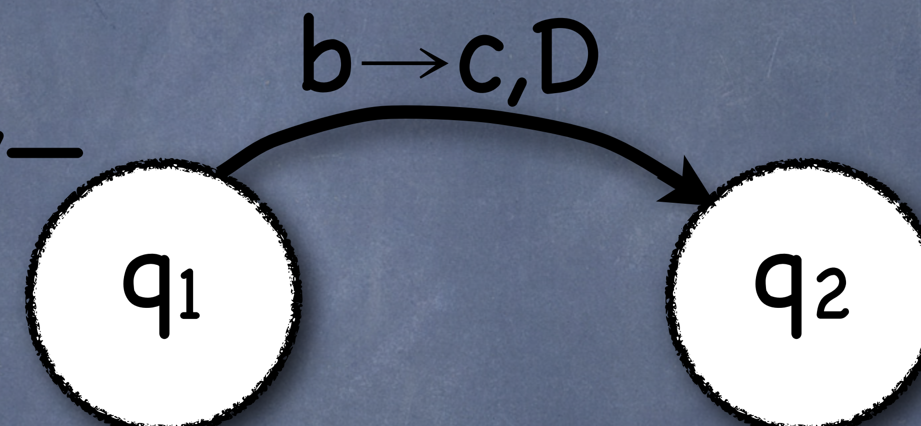
- Accept state


- Reject state

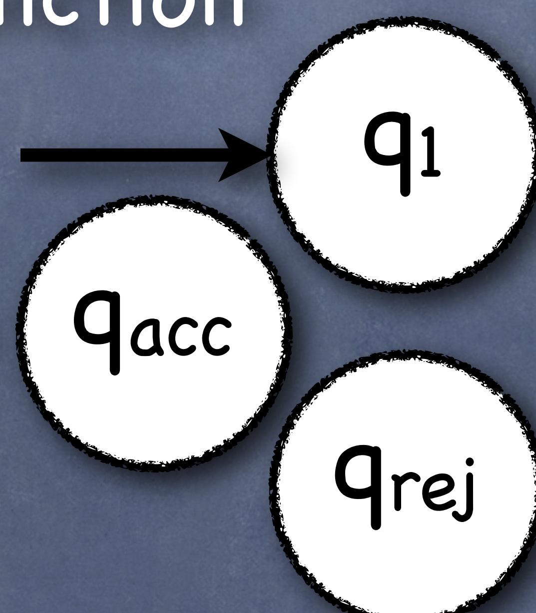
Definition of TM

- States

- Input Alphabet a,b,c
- Tape Alphabet a,b,c,A,B,C,_
- Transition function



- Start state
- Accept state
- Reject state



TM definition

DEFINITION 3.3

A ***Turing machine*** is a 7-tuple, $(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$, where Q, Σ, Γ are all finite sets and

1. Q is the set of states,
2. Σ is the input alphabet not containing the ***blank symbol*** $\sqcup$,
3. Γ is the tape alphabet, where $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma$,
4. $\delta\colon Q \times \Gamma \longrightarrow Q \times \Gamma \times \{\text{L}, \text{R}\}$ is the transition function,
5. $q_0 \in Q$ is the start state,
6. $q_{\text{accept}} \in Q$ is the accept state, and
7. $q_{\text{reject}} \in Q$ is the reject state, where $q_{\text{reject}} \neq q_{\text{accept}}$.

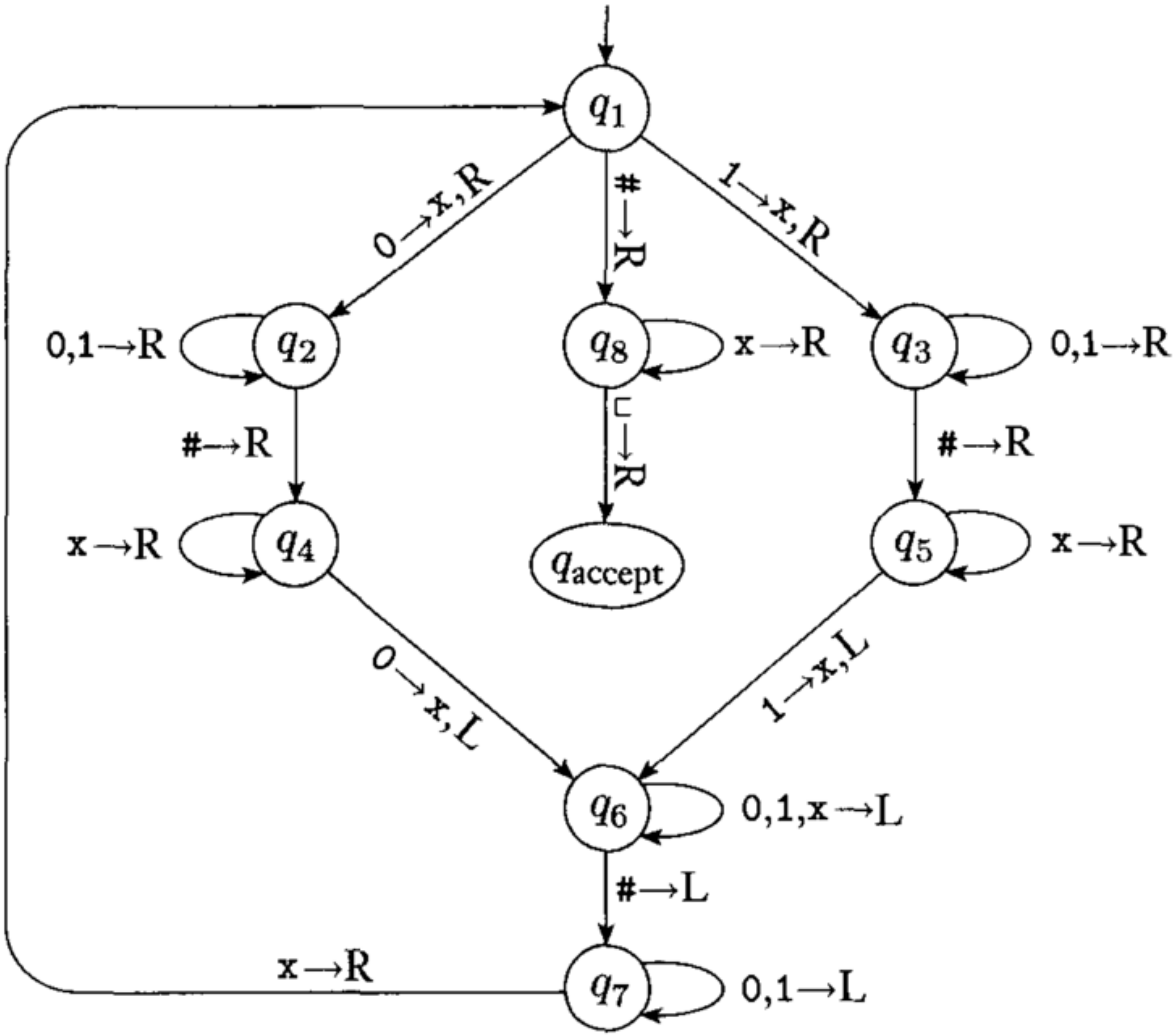


FIGURE 3.10
State diagram for Turing machine M_1

TM Configuration

As a Turing machine computes, changes occur in the current state, the current tape contents, and the current head location. A setting of these three items is called a ***configuration*** of the Turing machine. Configurations often are represented in a special way. For a state q and two strings u and v over the tape alphabet Γ we write $u\,q\,v$ for the configuration where the current state is q, the current tape contents is uv, and the current head location is the first symbol of v. The tape contains only blanks following the last symbol of v. For example, $1011q_7 01111$ represents the configuration when the tape is 101101111, the current state is q_7, and the head is currently on the second 0. The following figure depicts a Turing machine with that configuration.

TM Computation

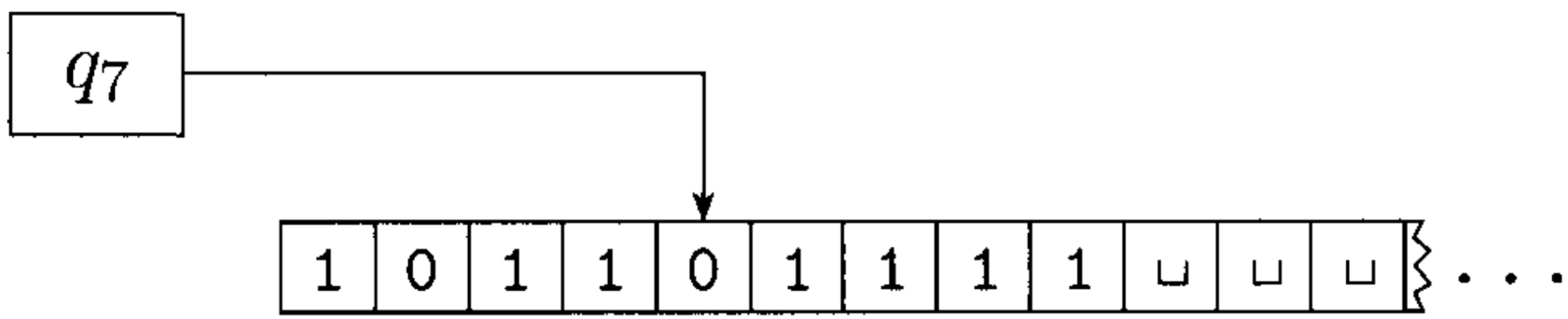


FIGURE 3.4
A Turing machine with configuration $1011q_701111$

TM definition

- For all $a,b,c \in \Gamma$, $u,v \in \Gamma^*$, $q_i,q_j \in Q$
- Config. $u a \mathbf{q_i} b v$ yields
  config. $u \mathbf{q_j} a c v$ if $\delta(q_i,b) = q_j,c,L$
- Config. $u a \mathbf{q_i} b v$ yields
  config. $u a c \mathbf{q_j} v$ if $\delta(q_i,b) = q_j,c,R$
- Special cases:
  Config. $\mathbf{q_i} b v$ yields $\mathbf{q_j} c v$ if $\delta(q_i,b) = q_j,c,L$
  Config. $\mathbf{q_i} b v$ yields $c \mathbf{q_j} v$ if $\delta(q_i,b) = q_j,c,R$

TM definition

- Fo
- Co
  co
- Co
  co
- Sp
  Co ... $_j$,c,L
  Config: q_i bv yields c q_j v if $\delta(q_i,b) = q_j,c,R$

ua q_i bv

yields (L)

u q_j acv

TM definition

- For all $a,b,c \in \Gamma$, $u,v \in \Gamma^*$, $q_i,q_j \in Q$
- Config. $ua\,q_i\,bv$ yields config. $u\,q_j\,acv$ if $\delta(q_i,b) = q_j,c,L$
- Config. $ua\,q_i\,bv$ yields config. $uac\,q_j\,v$ if $\delta(q_i,b) = q_j,c,R$
- Special cases:
  Config. $q_i\,bv$ yields $q_j\,cv$ if $\delta(q_i,b) = q_j,c,L$
  Config. $q_i\,bv$ yields $c\,q_j\,v$ if $\delta(q_i,b) = q_j,c,R$

TM definition

- Fo
- Co
  co
- Co
  co
- Sp
  Co ... $_j$,c,L
  Config: q_i bv yields c q_j v if $\delta(q_i,b) = q_j$,c,R

ua q_i bv

yields (R)

uac q_j v

TM definition

- For all $a,b,c \in \Gamma$, $u,v \in \Gamma^*$, $q_i,q_j \in Q$
- Config. $u a \mathbf{q_i} b v$ yields
  config. $u \mathbf{q_j} a c v$ if $\delta(q_i,b) = q_j,c,L$
- Config. $u a \mathbf{q_i} b v$ yields
  config. $u a c \mathbf{q_j} v$ if $\delta(q_i,b) = q_j,c,R$
- Special cases:
  Config. $\mathbf{q_i} b v$ yields $\mathbf{q_j} c v$ if $\delta(q_i,b) = q_j,c,L$
  Config. $\mathbf{q_i} b v$ yields $c \mathbf{q_j} v$ if $\delta(q_i,b) = q_j,c,R$

TM definition

- Fo
- Co
  co
- Co
  co
- Sp
  Co ... q_j,c,L
  Config. q_i bv yields c q_j v if $\delta(q_i,b) = q_j$,c,R

q_i bv

yields (L)

q_j cv

TM definition

- For all $a,b,c \in \Gamma$, $u,v \in \Gamma^*$, $q_i,q_j \in Q$
- Config. $u a \mathbf{q_i} b v$ yields
  config. $u \mathbf{q_j} a c v$ if $\delta(q_i,b) = q_j,c,L$
- Config. $u a \mathbf{q_i} b v$ yields
  config. $u a c \mathbf{q_j} v$ if $\delta(q_i,b) = q_j,c,R$
- Special cases:
  Config. $\mathbf{q_i} b v$ yields $\mathbf{q_j} c v$ if $\delta(q_i,b) = q_j,c,L$
  Config. $\mathbf{q_i} b v$ yields $c \mathbf{q_j} v$ if $\delta(q_i,b) = q_j,c,R$

TM definition

- Fo
- Co
  co
- Co
  co
- Sp
  Co ... $_j$,c,L
  Config: q_i bv yields c q_j v if $\delta(q_i,b) = q_j$,c,R

q_i bv

yields (R)

c q_j v

TM definition

- For all $a,b,c \in \Gamma$, $u,v \in \Gamma^*$, $q_i,q_j \in Q$
- Config. $u a \mathbf{q_i} b v$ yields
  config. $u \mathbf{q_j} a c v$ if $\delta(q_i,b) = q_j,c,L$
- Config. $u a \mathbf{q_i} b v$ yields
  config. $u a c \mathbf{q_j} v$ if $\delta(q_i,b) = q_j,c,R$
- Special cases:
  Config. $\mathbf{q_i} b v$ yields $\mathbf{q_j} c v$ if $\delta(q_i,b) = q_j,c,L$
  Config. $\mathbf{q_i} b v$ yields $c \mathbf{q_j} v$ if $\delta(q_i,b) = q_j,c,R$

TM Computation

- Start configuration: $q_0 w$ (w = input string)
- Accepting configuration: state = q_{accept}
- Rejecting configuration: state = q_{reject}

TM Computation

- Turing Machine M accepts input w if there exists configurations $C_0, C_1, ..., C_m$ such that
  - C_0 is a start configuration
  - C_i yields C_{i+1} for $0 \leq i < m$
  - C_m is an accepting configuration.
- The collection of strings that M accepts is the language of M or the language recognized by M, denoted $L(M)$.

TM Computation

DEFINITION 3.5

Call a language ***Turing-recognizable*** if some Turing machine recognizes it.[1]

TM Computation

DEFINITION 3.5

Call a language ***Turing-recognizable*** if some Turing machine recognizes it.[1]

- A TM decides a language if it recognizes it and halts (reaches an accepting or rejecting states) on all input strings.

TM Computation

DEFINITION 3.5

Call a language ***Turing-recognizable*** if some Turing machine recognizes it.[1]

- A TM decides a language if it recognizes it and halts (reaches an accepting or rejecting states) on all input strings.

DEFINITION 3.6

Call a language ***Turing-decidable*** or simply ***decidable*** if some Turing machine decides it.[2]

TM Computation

DEFINITION 3.5

Call a language ***Turing-recognizable*** if some Turing machine recognizes it.[1]

- A TM decides a language if it recognizes it and halts (reaches an accepting or rejecting states) on all input strings.

DEFINITION 3.6

Call a language ***Turing-decidable*** or simply ***decidable*** if some Turing machine decides it.[2]

[1]Often named **Recursively-Enumerable** in the literature.

[2]Often named **Recursive** in the literature.

TM Examples

EXAMPLE 3.7

Here we describe a Turing machine (TM) M_2 that decides $A = \{0^{2^n} \mid n \geq 0\}$, the language consisting of all strings of 0s whose length 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 tape contained more than a single 0 and the number of 0s was odd, *reject*.
4. Return the head to the left-hand end of the tape.
5. Go to stage 1."

TM Examples

Now we give the formal description of $M_2 = (Q, \Sigma, \Gamma, \delta, q_1, q_{\text{accept}}, q_{\text{reject}})$:

- $Q = \{q_1, q_2, q_3, q_4, q_5, q_{\text{accept}}, q_{\text{reject}}\}$,
- $\Sigma = \{0\}$, and
- $\Gamma = \{0, \text{x}, \sqcup\}$.
- We describe δ with a state diagram (see Figure 3.8).
- The start, accept, and reject states are q_1, q_{accept}, and q_{reject}.

TM Computation

$q_1 0000$

$\sqcup q_2 000$

$\sqcup \mathrm{x} q_3 00$

$\sqcup \mathrm{x} 0 q_4 0$

$\sqcup \mathrm{x} 0 \mathrm{x} q_3 \sqcup$

$\sqcup \mathrm{x} 0 q_5 \mathrm{x} \sqcup$

$\sqcup \mathrm{x} q_5 0 \mathrm{x} \sqcup$

$\sqcup q_5 \mathrm{x} 0 \mathrm{x} \sqcup$

$q_5 \sqcup \mathrm{x} 0 \mathrm{x} \sqcup$

$\sqcup q_2 \mathrm{x} 0 \mathrm{x} \sqcup$

$\sqcup \mathrm{x} q_2 0 \mathrm{x} \sqcup$

$\sqcup \mathrm{xx} q_3 \mathrm{x} \sqcup$

$\sqcup \mathrm{xxx} q_3 \sqcup$

$\sqcup \mathrm{xx} q_5 \mathrm{x} \sqcup$

$\sqcup \mathrm{x} q_5 \mathrm{xx} \sqcup$

$\sqcup q_5 \mathrm{xxx} \sqcup$

$q_5 \sqcup \mathrm{xxx} \sqcup$

$\sqcup q_2 \mathrm{xxx} \sqcup$

$\sqcup \mathrm{x} q_2 \mathrm{xx} \sqcup$

$\sqcup \mathrm{xx} q_2 \mathrm{x} \sqcup$

$\sqcup \mathrm{xxx} q_2 \sqcup$

$\sqcup \mathrm{xxx} \sqcup q_{\mathrm{accept}}$

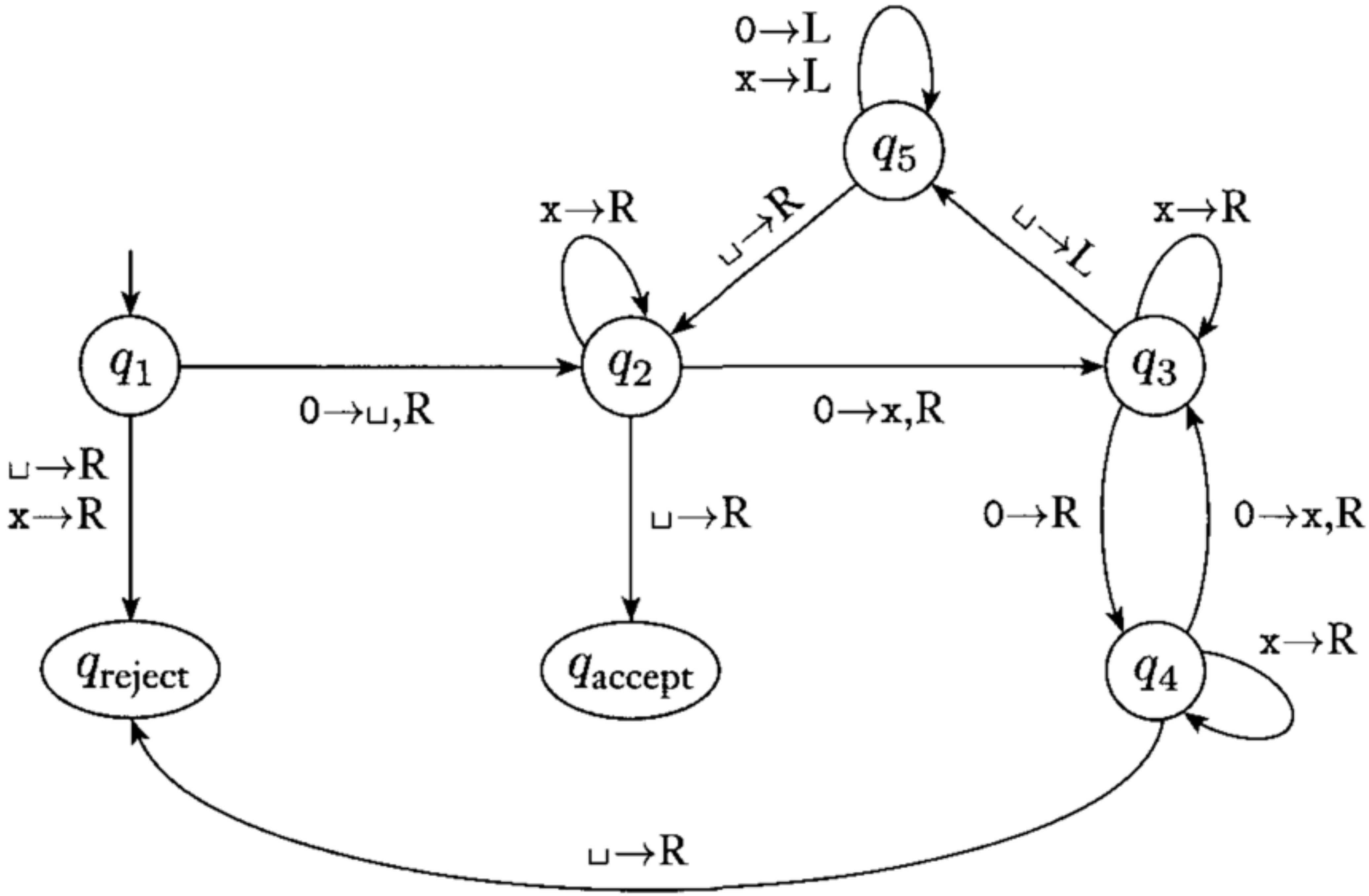


FIGURE 3.8
State diagram for Turing machine M_2

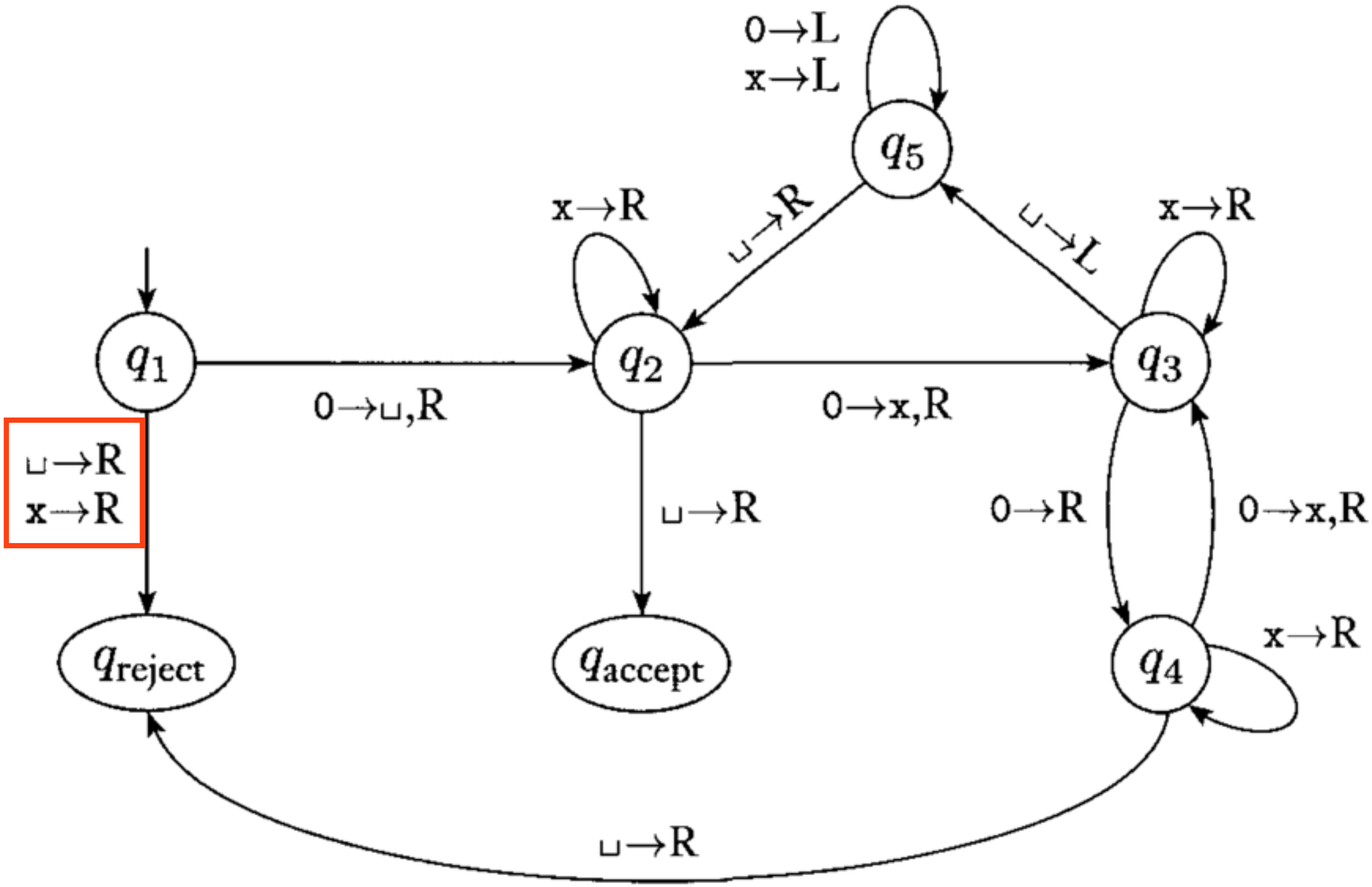


FIGURE 3.8
State diagram for Turing machine M_2

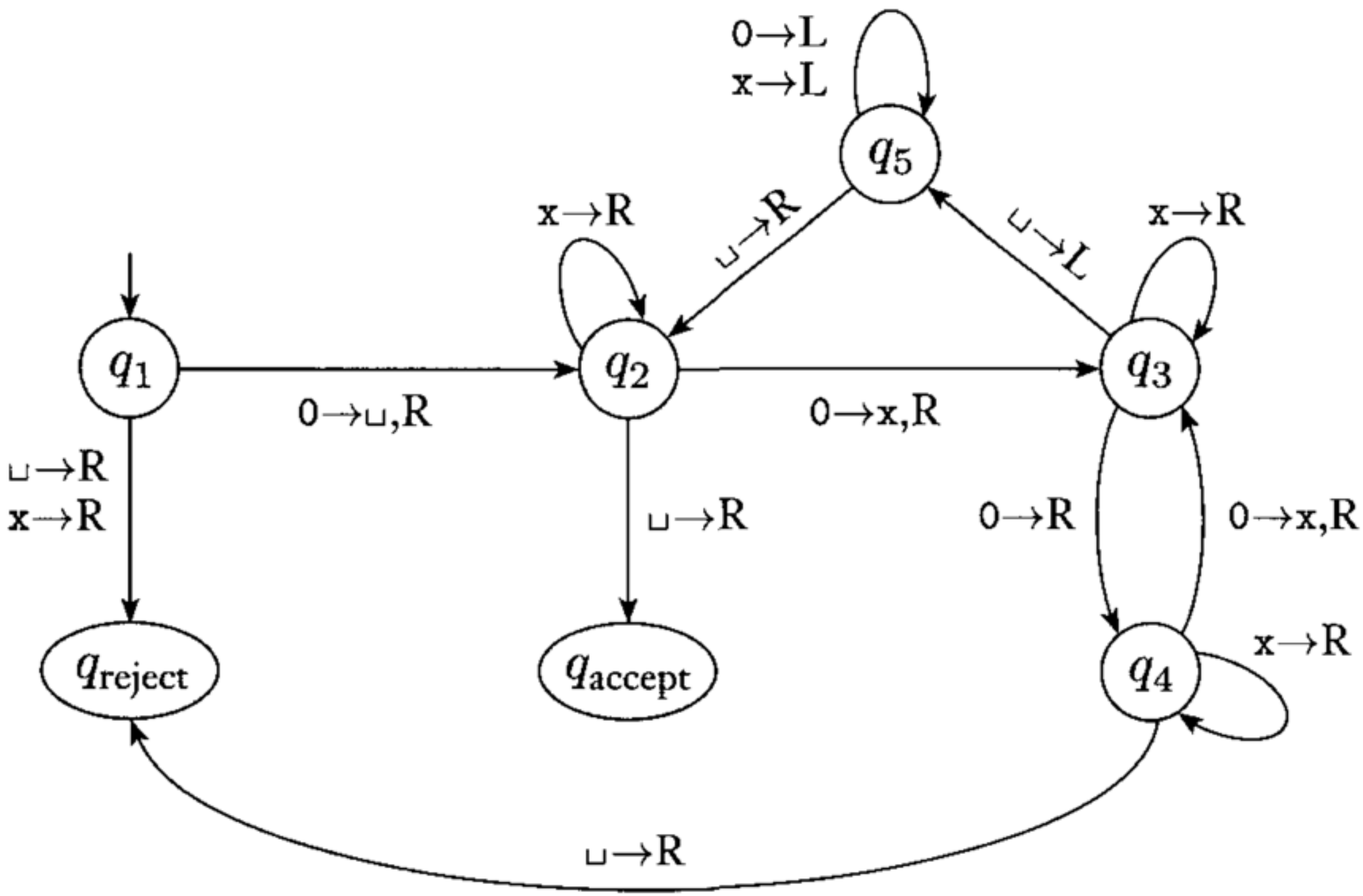


FIGURE 3.8
State diagram for Turing machine M_2

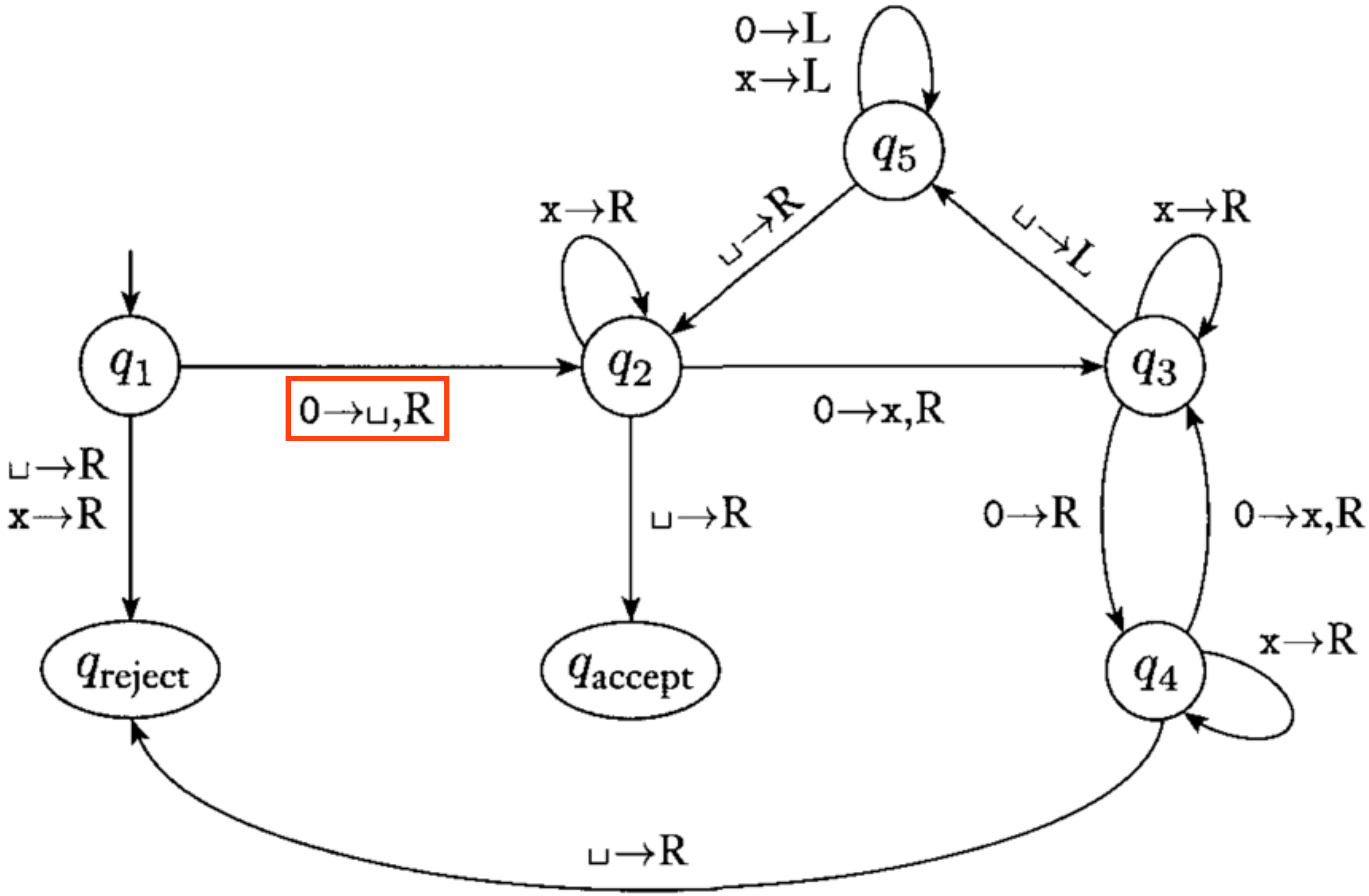


FIGURE 3.8
State diagram for Turing machine M_2

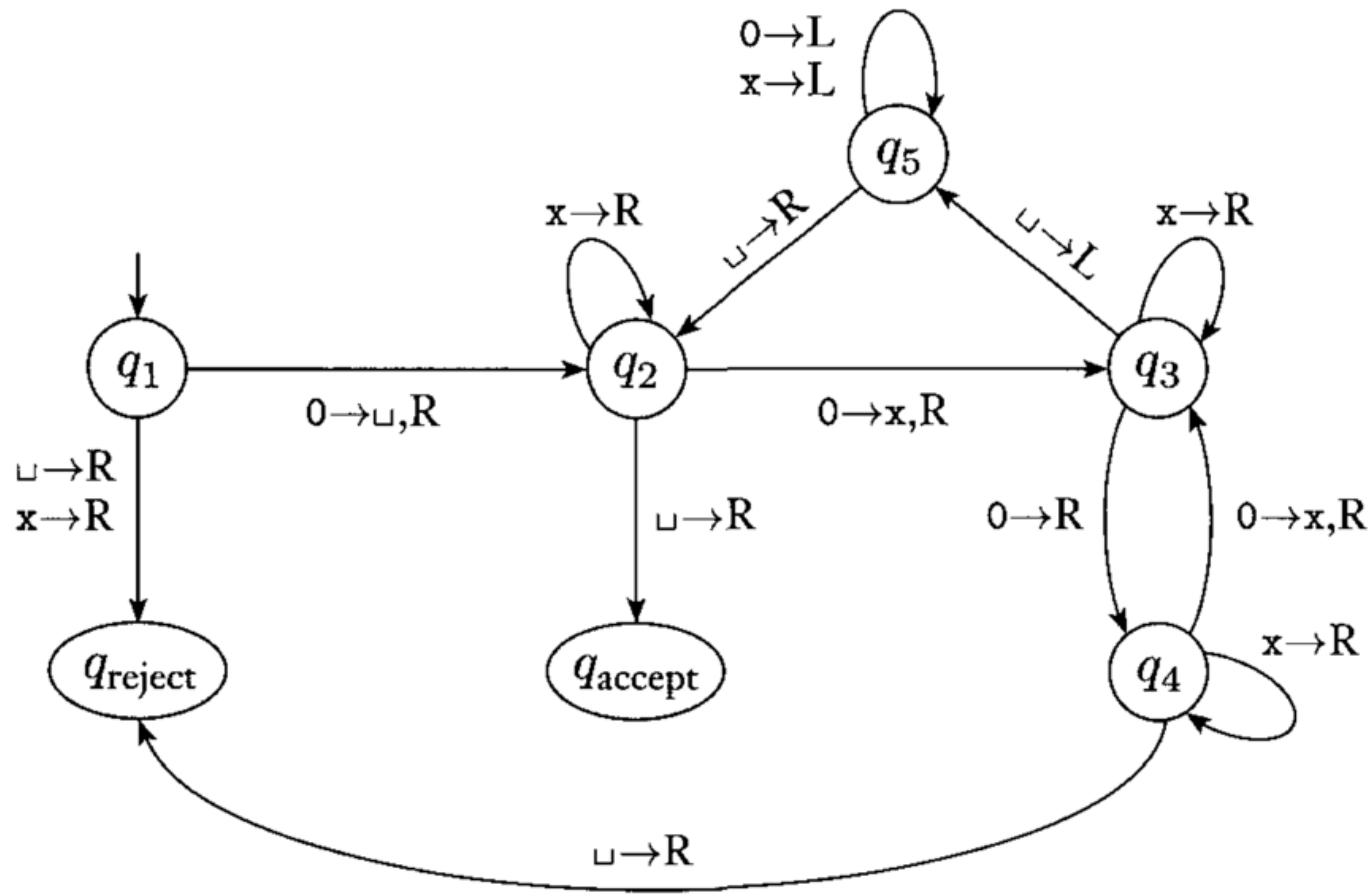


FIGURE 3.8
State diagram for Turing machine M_2

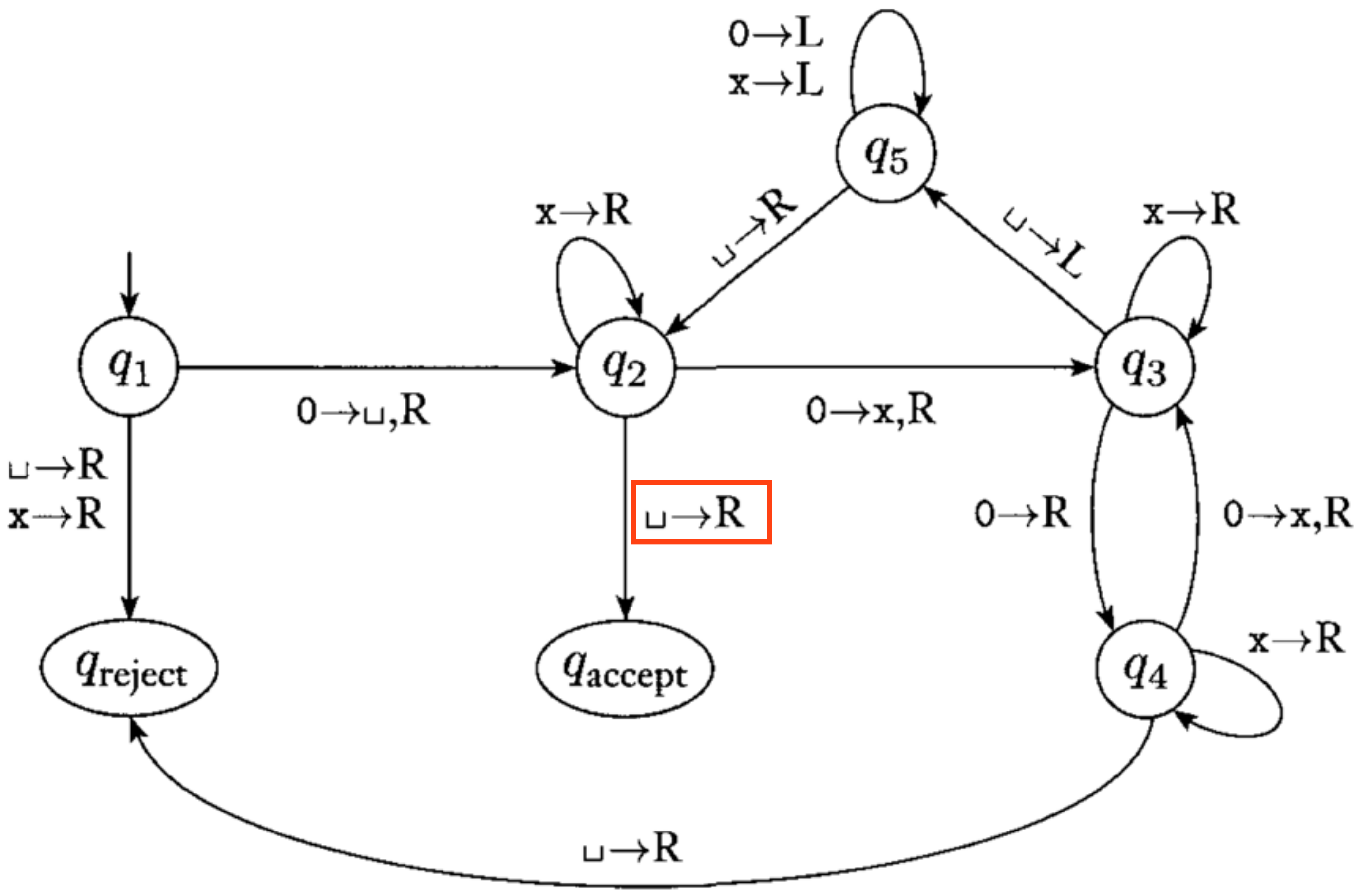


FIGURE 3.8

State diagram for Turing machine M_2

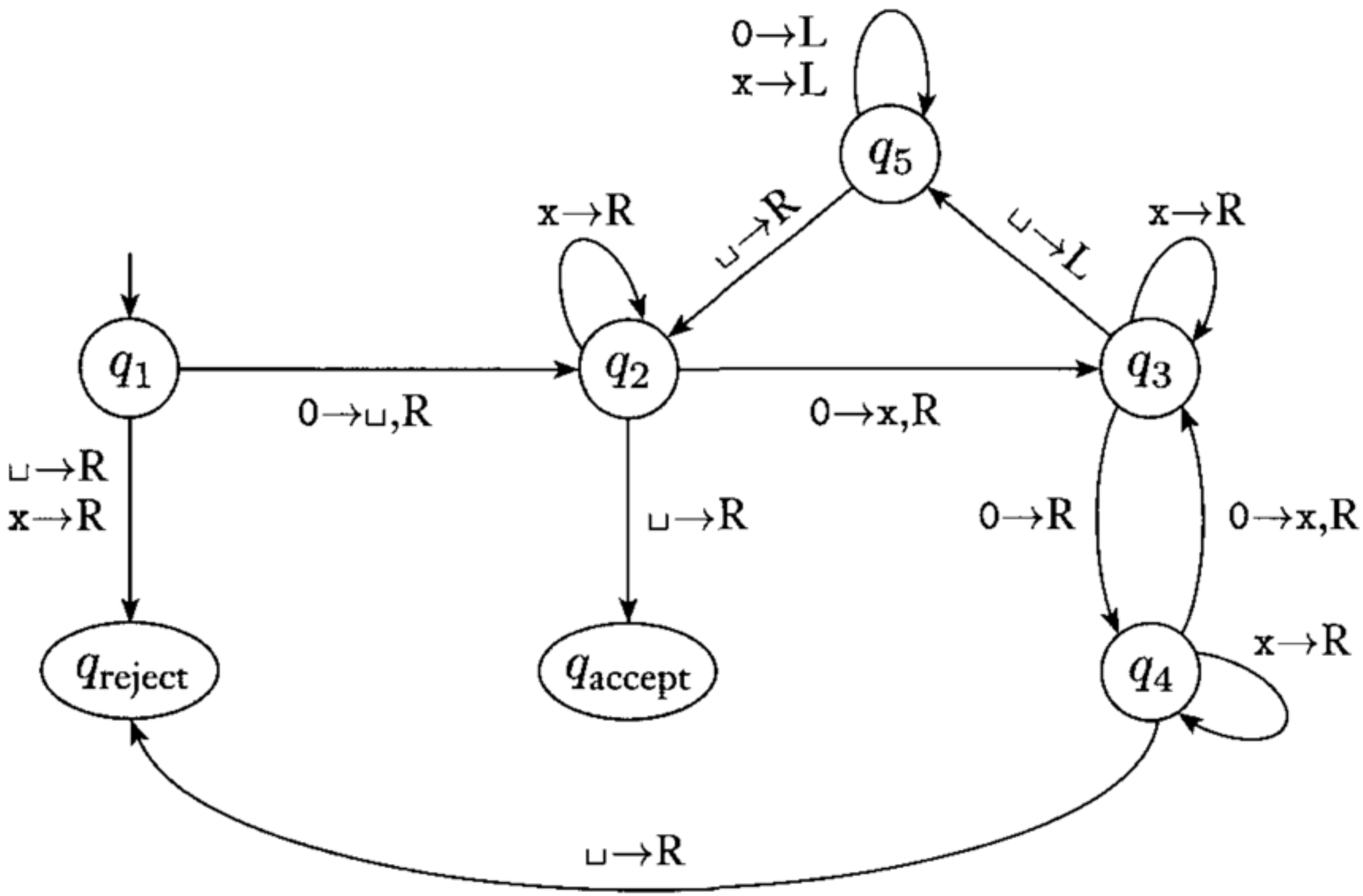


FIGURE 3.8
State diagram for Turing machine M_2

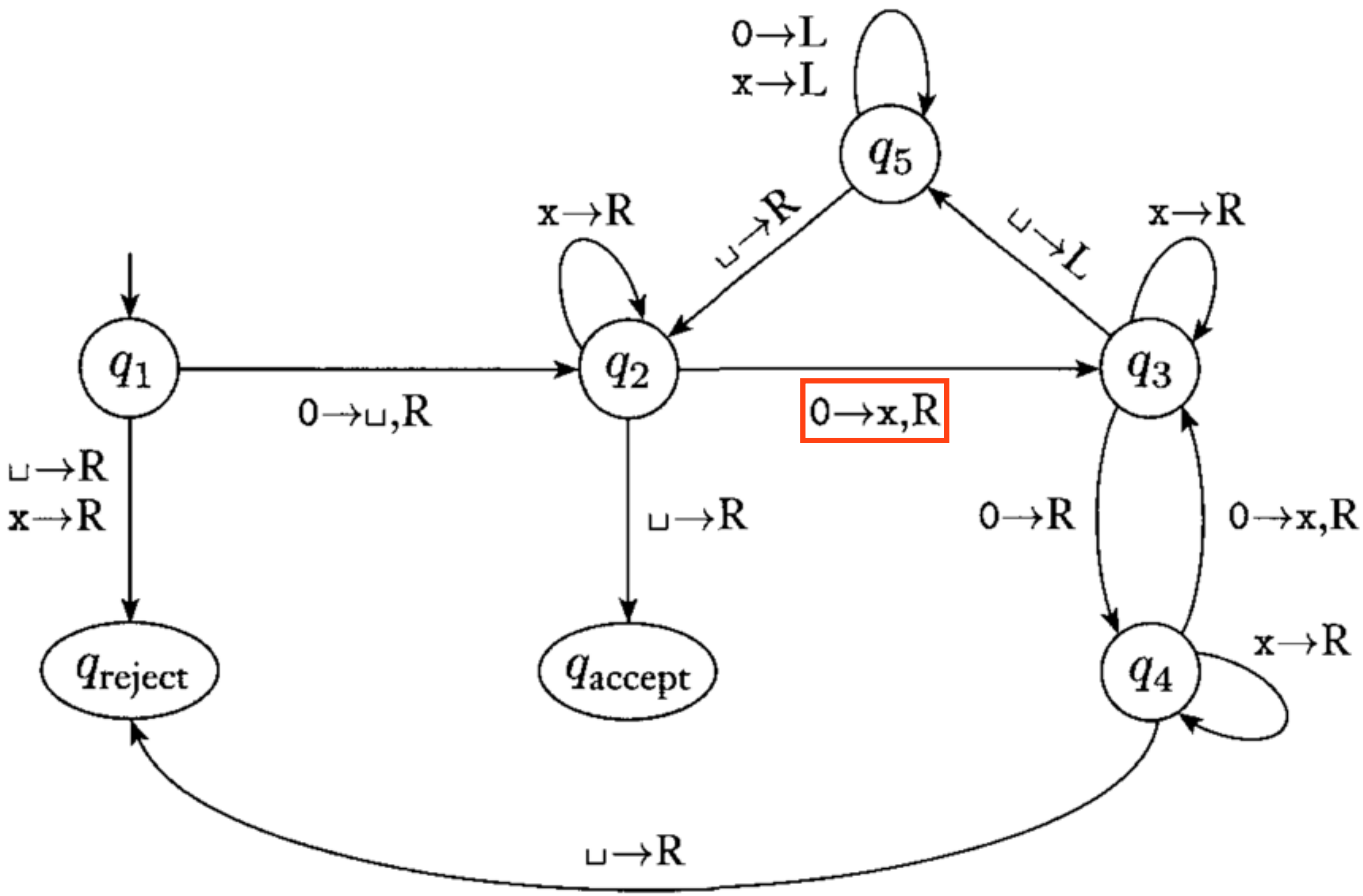


FIGURE 3.8
State diagram for Turing machine M_2

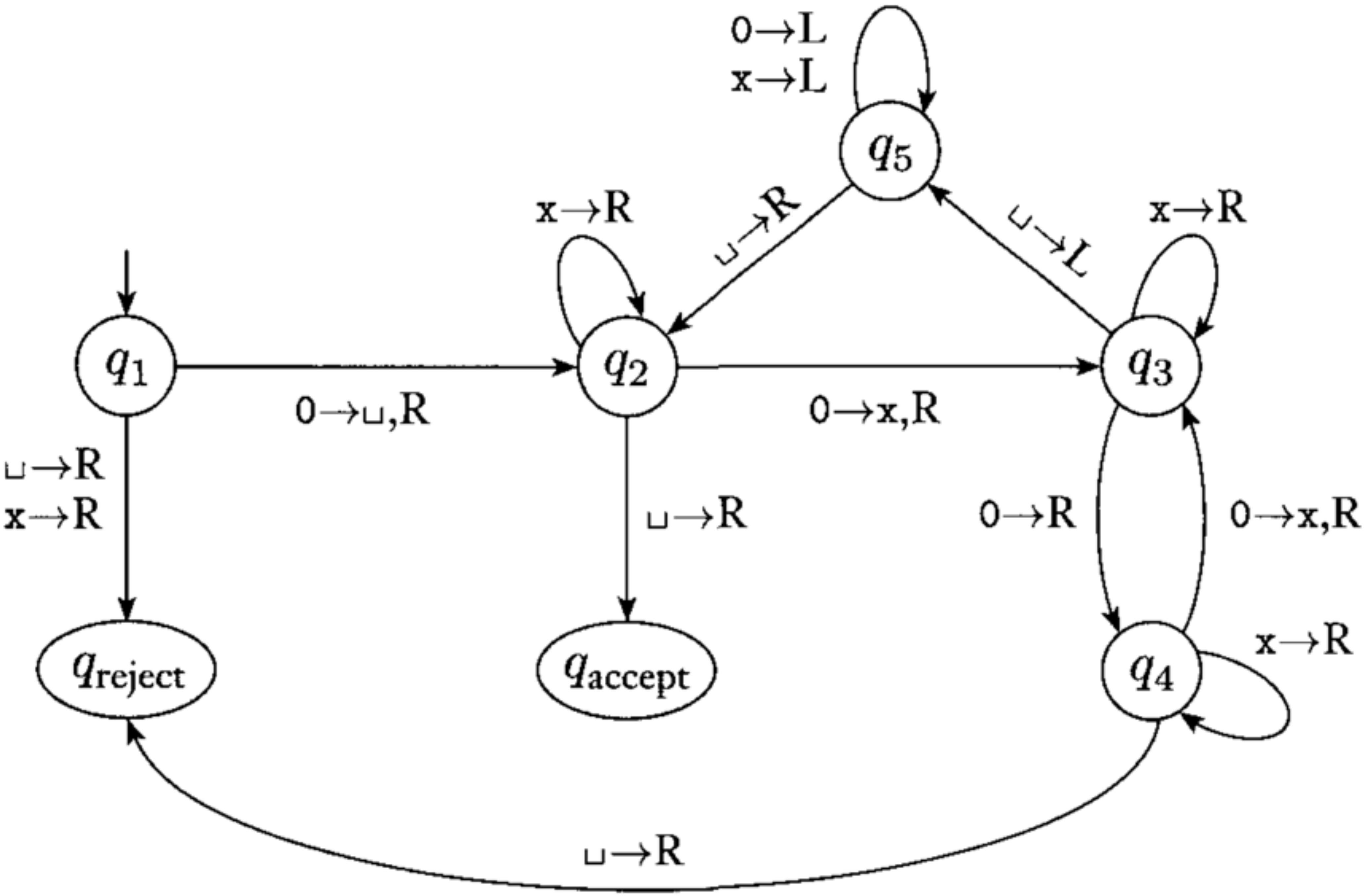


FIGURE 3.8
State diagram for Turing machine M_2

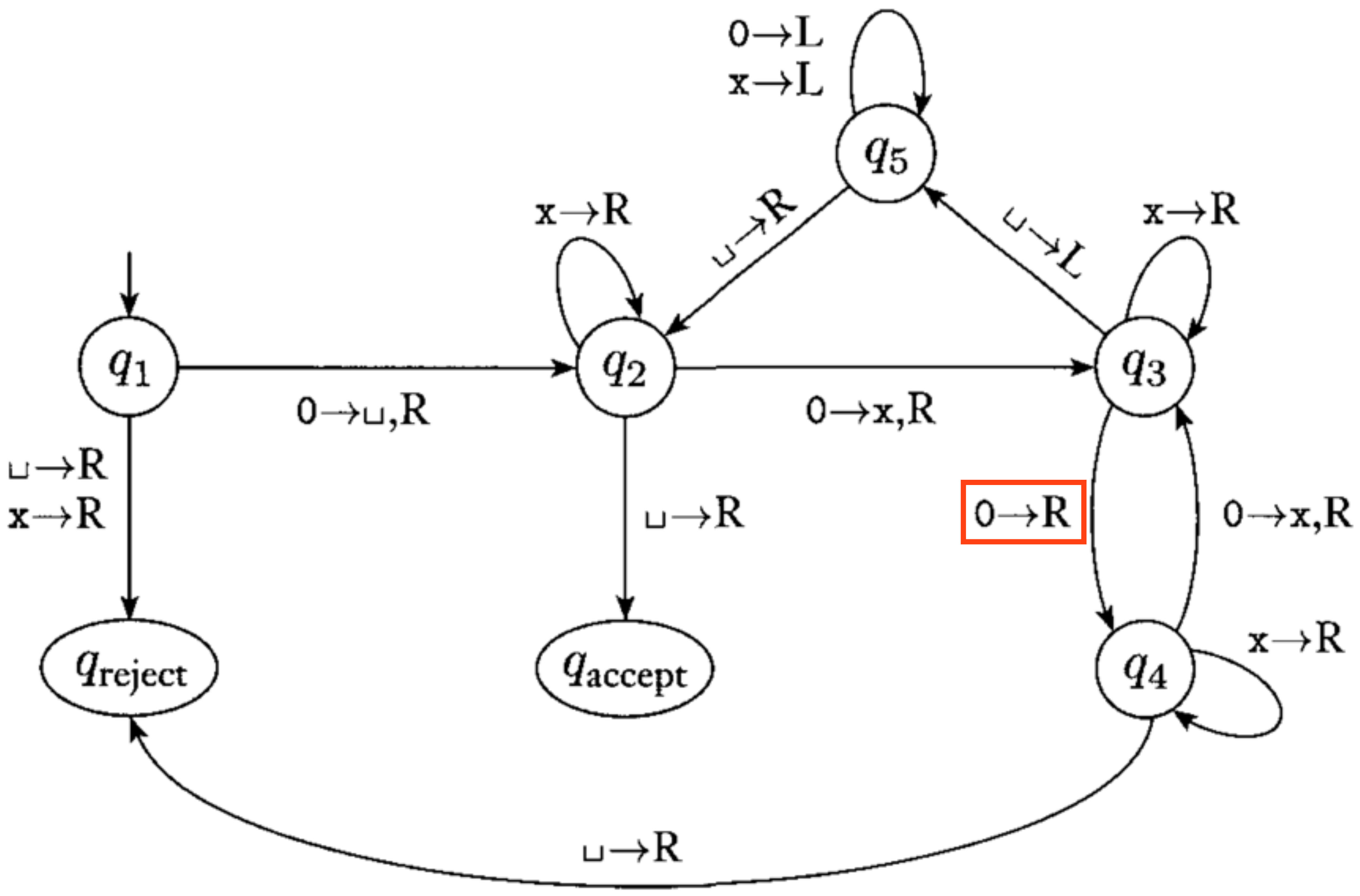


FIGURE 3.8

State diagram for Turing machine M_2

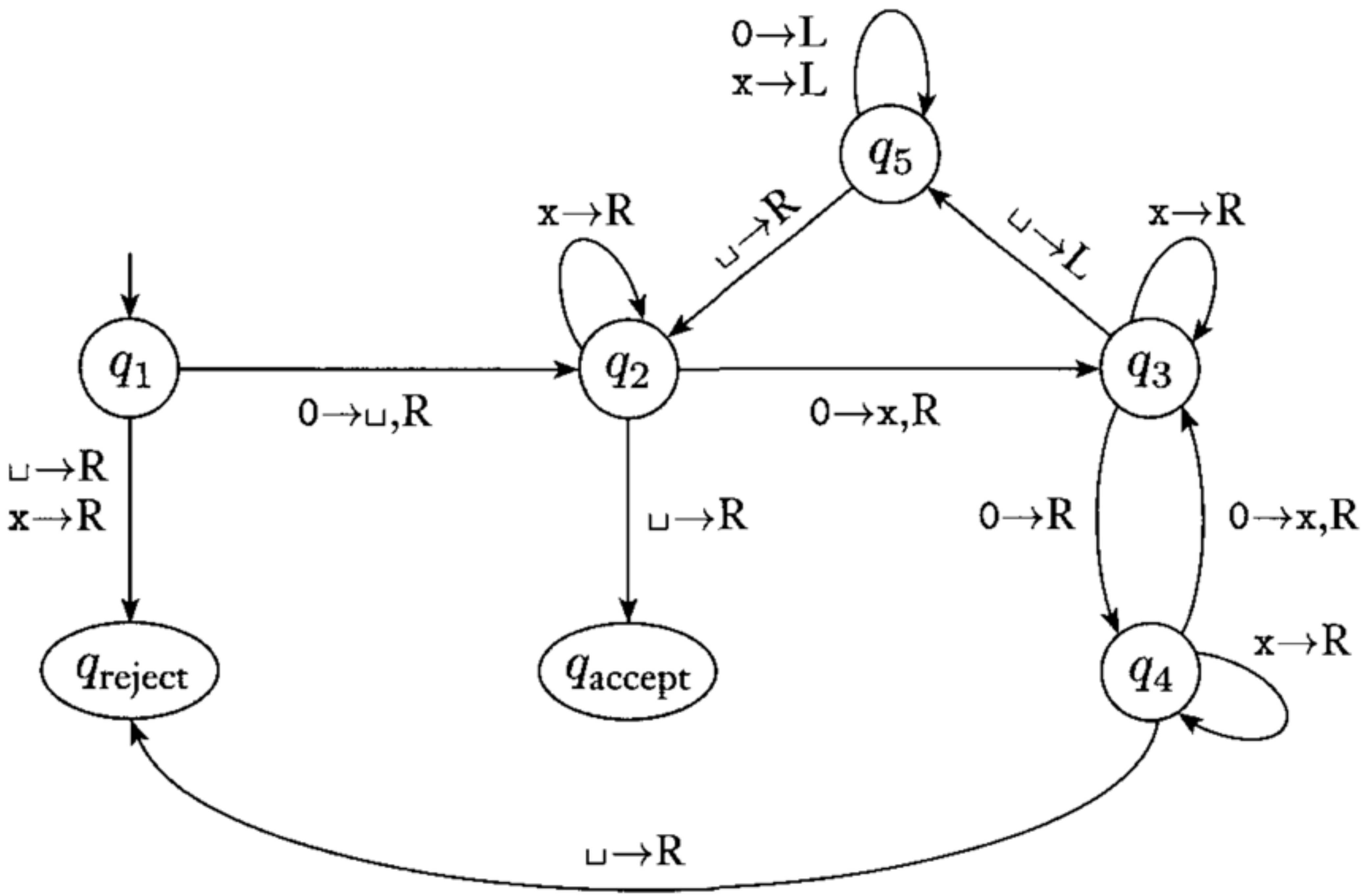


FIGURE 3.8
State diagram for Turing machine M_2

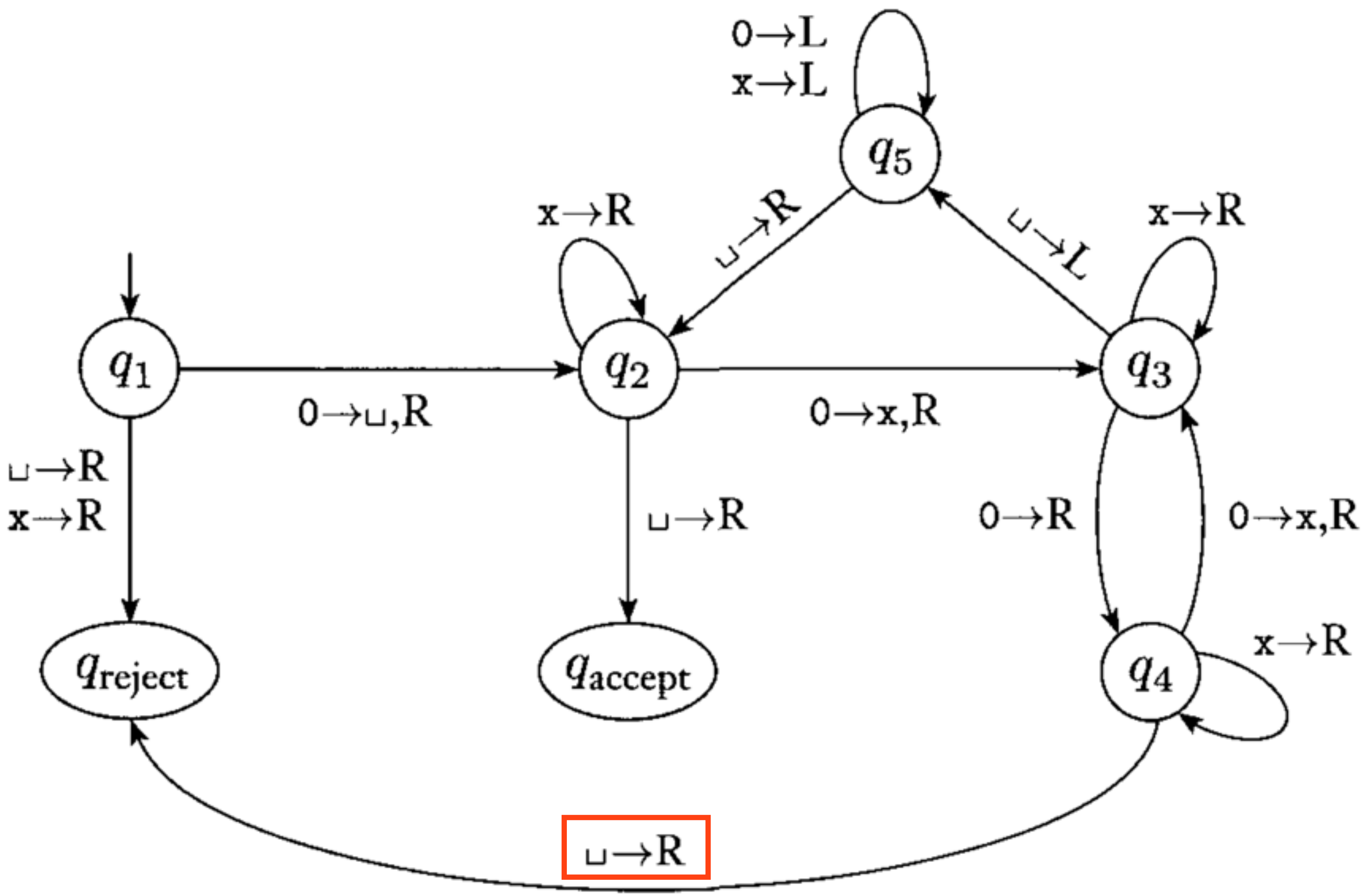


FIGURE 3.8
State diagram for Turing machine M_2

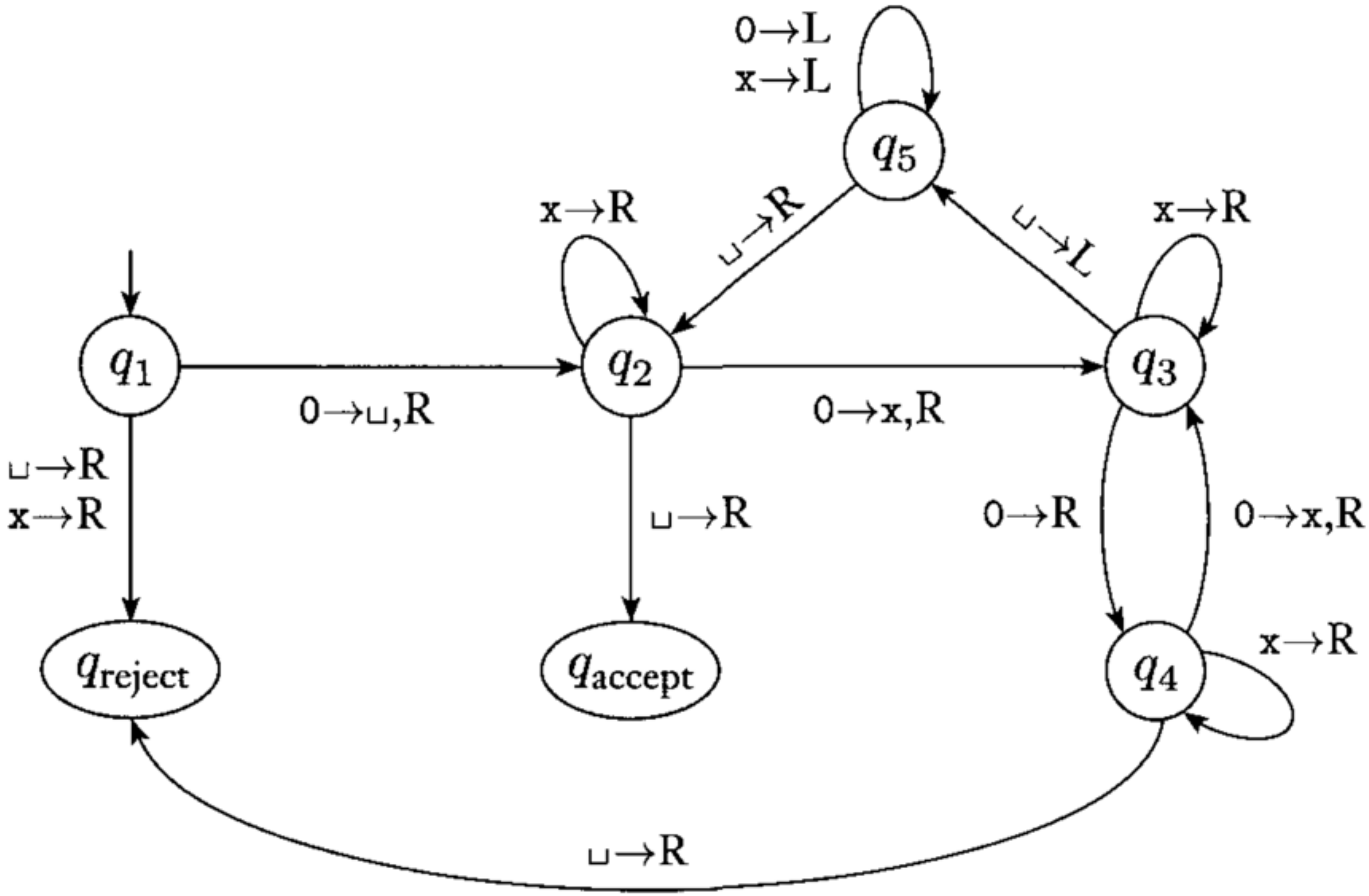


FIGURE 3.8

State diagram for Turing machine M_2

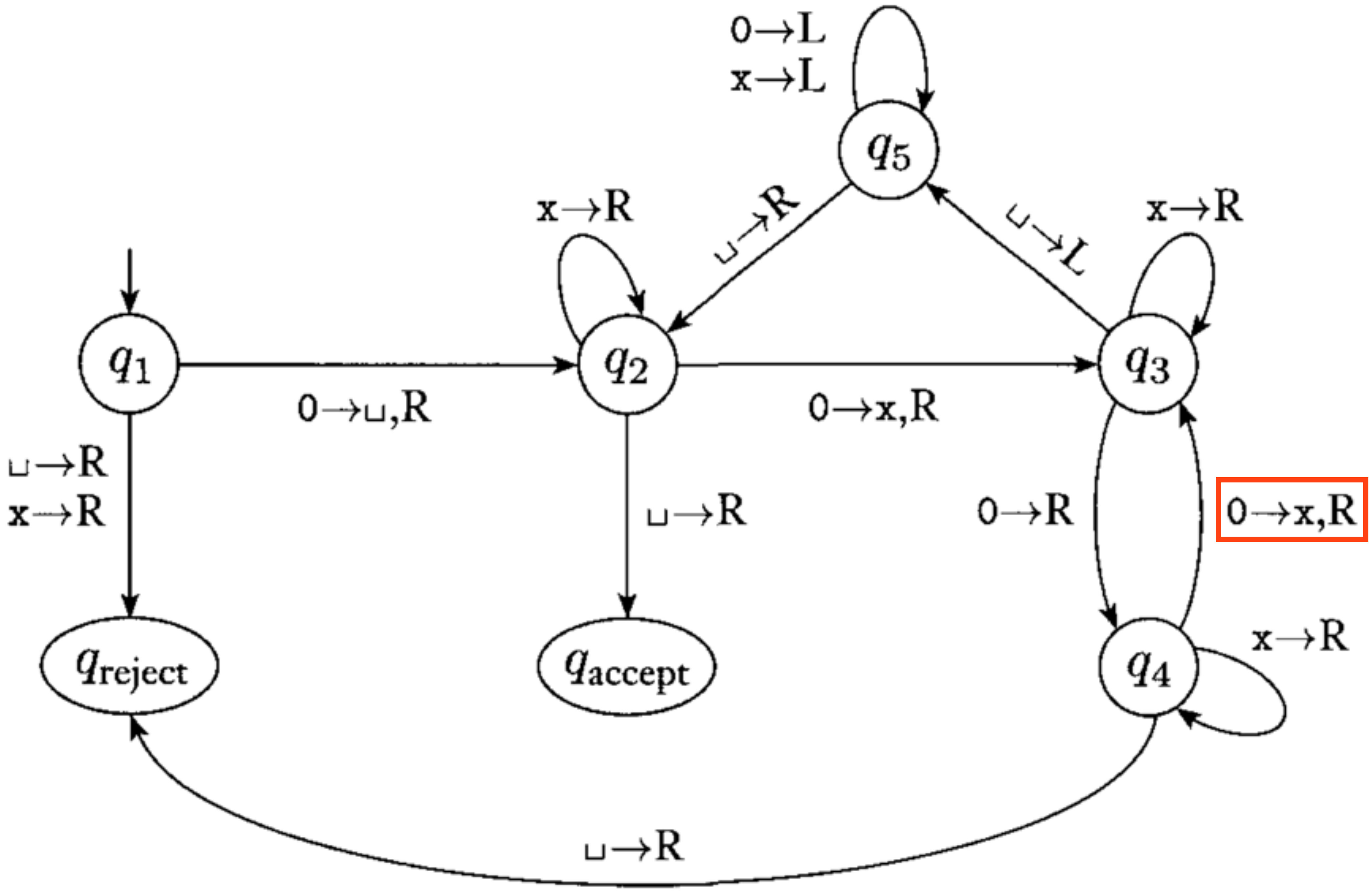


FIGURE 3.8
State diagram for Turing machine M_2

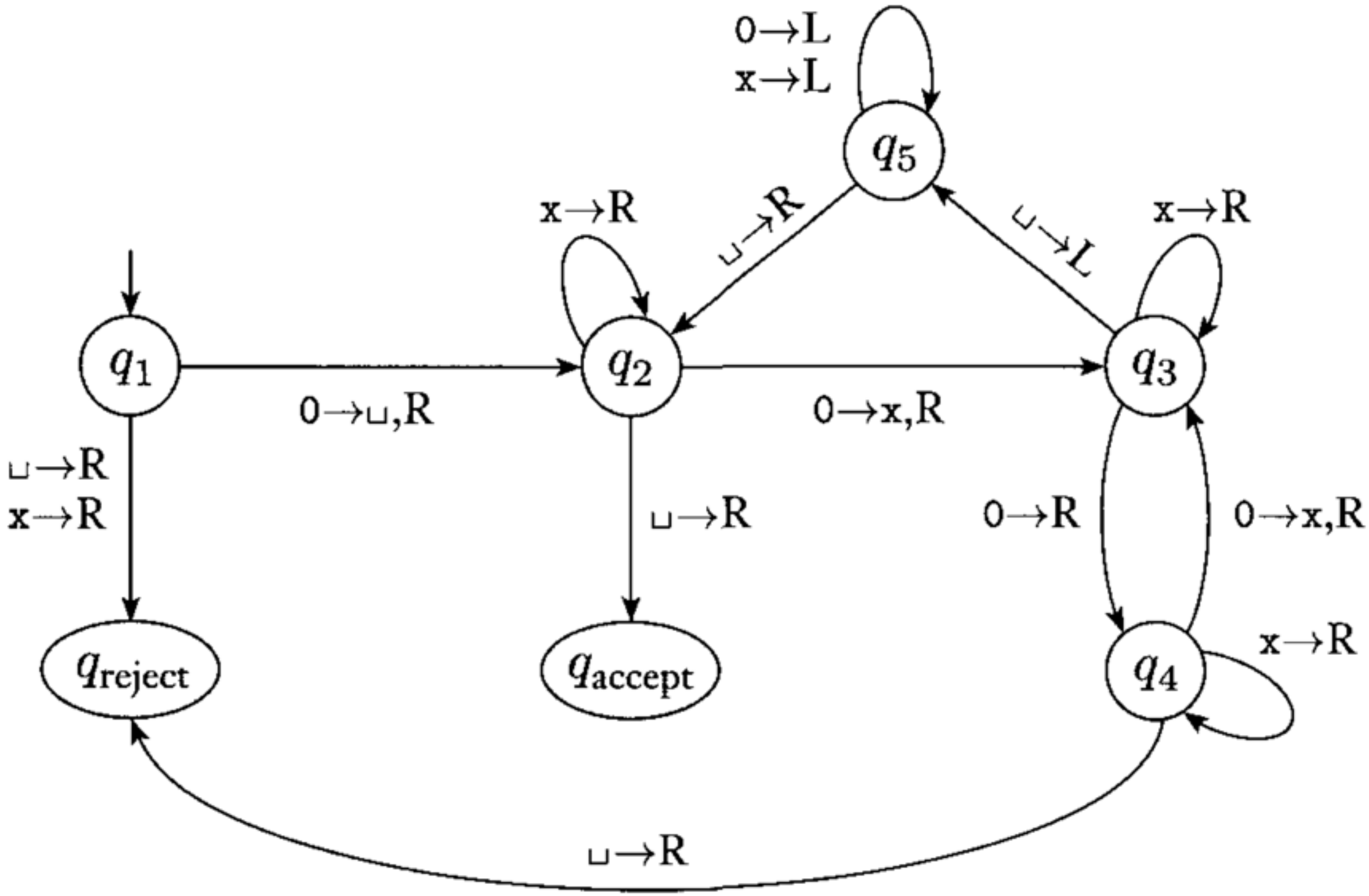


FIGURE 3.8
State diagram for Turing machine M_2

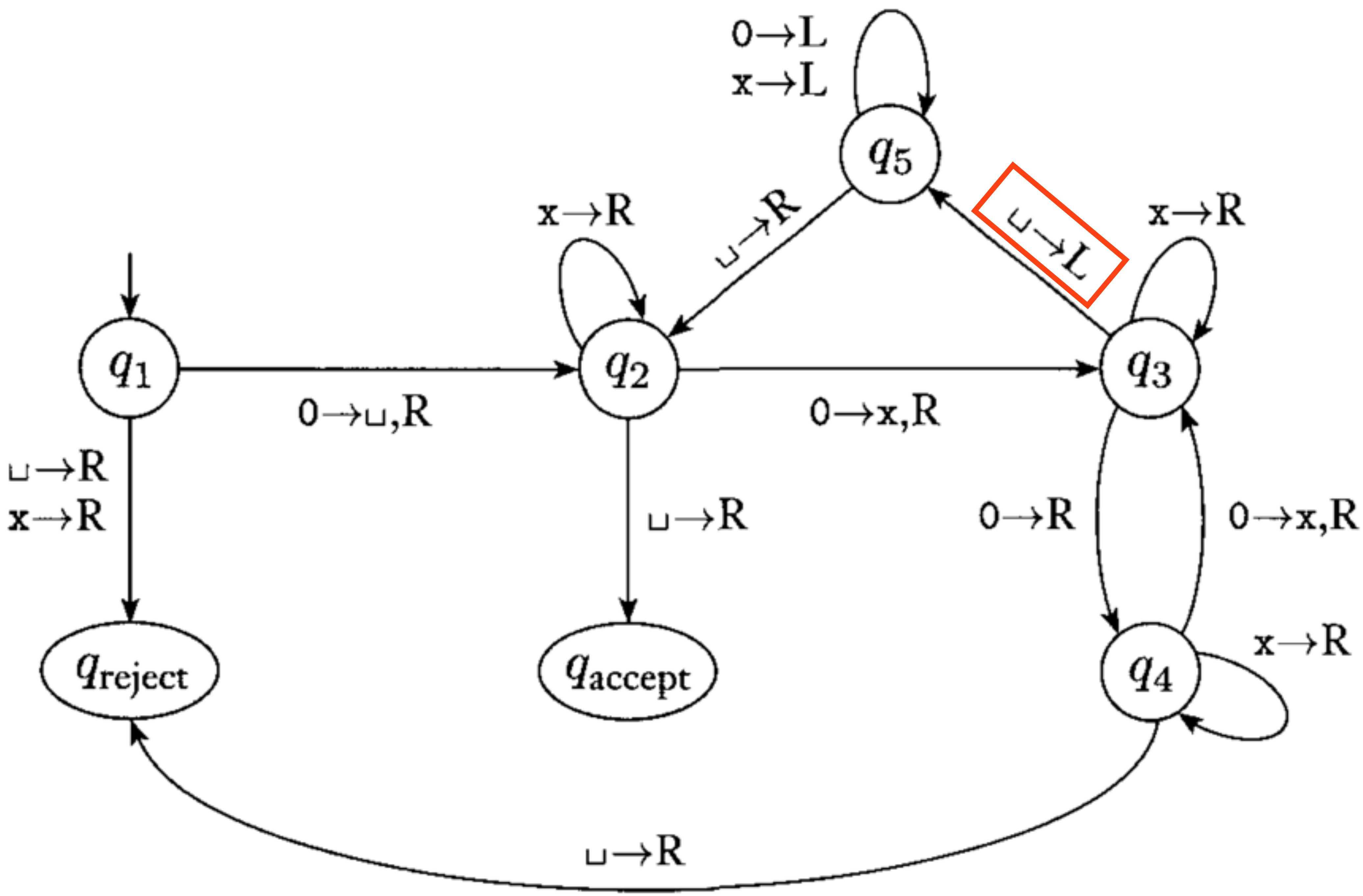


FIGURE 3.8
State diagram for Turing machine M_2

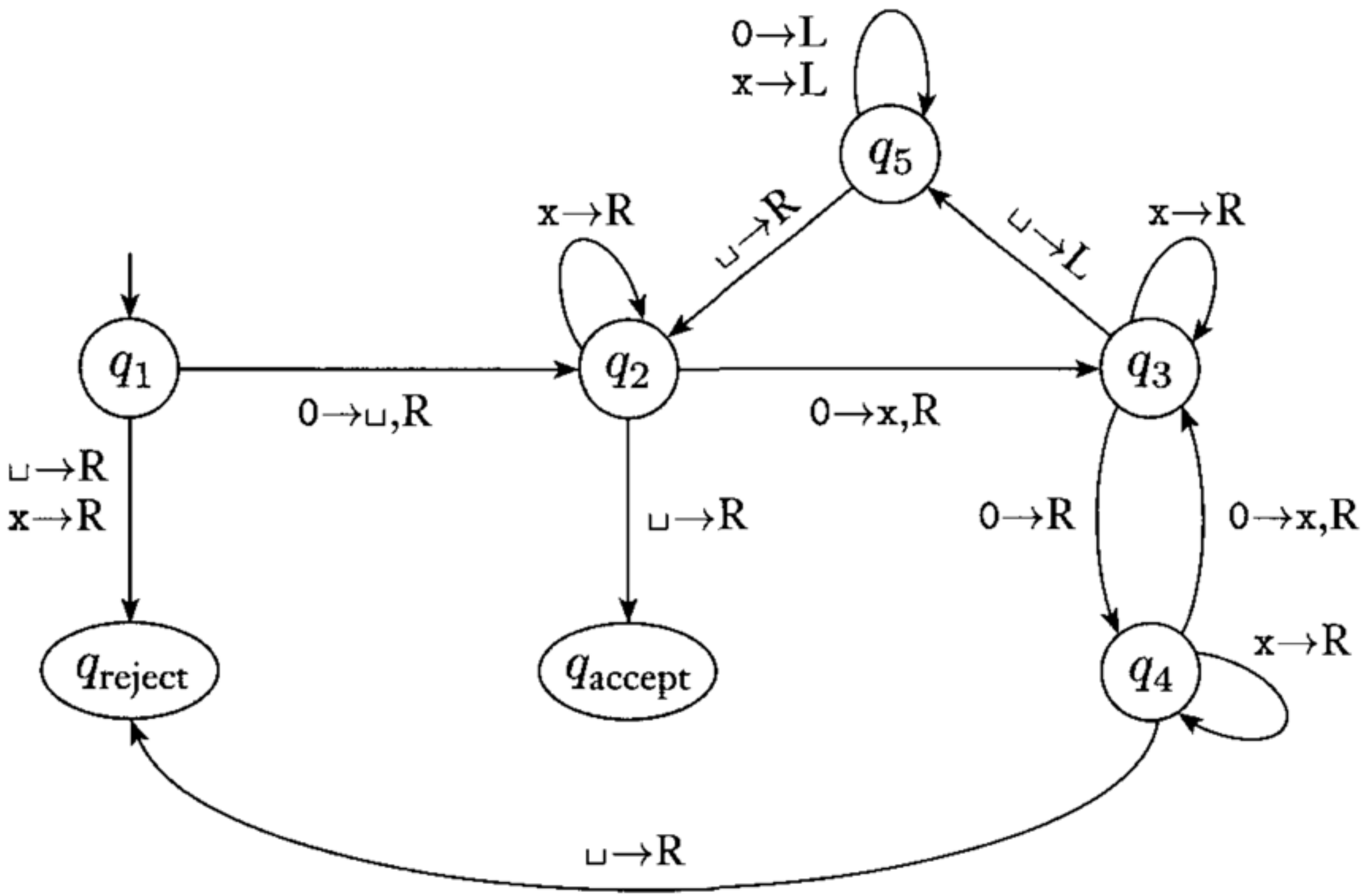


FIGURE 3.8
State diagram for Turing machine M_2

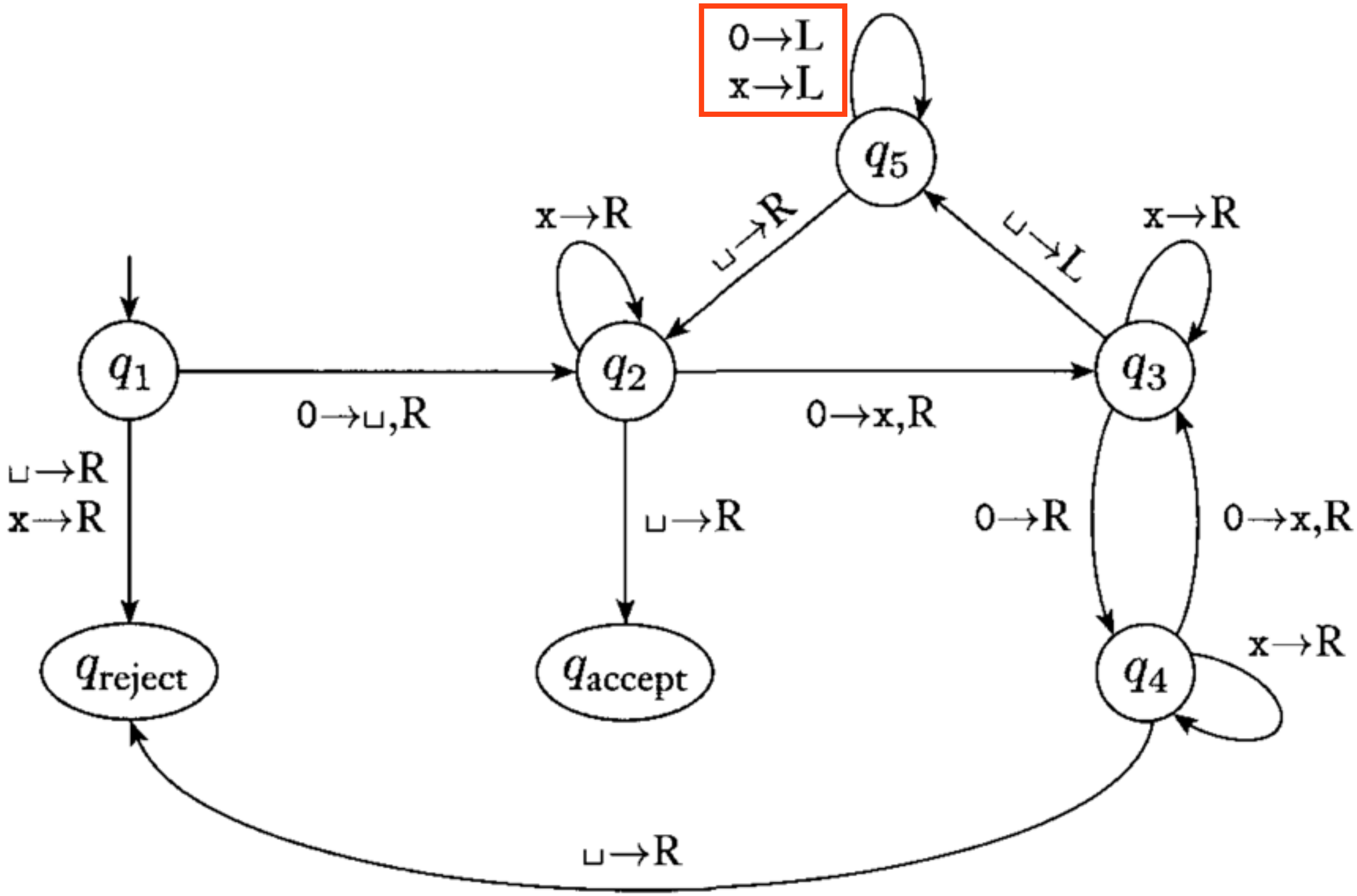


FIGURE 3.8
State diagram for Turing machine M_2

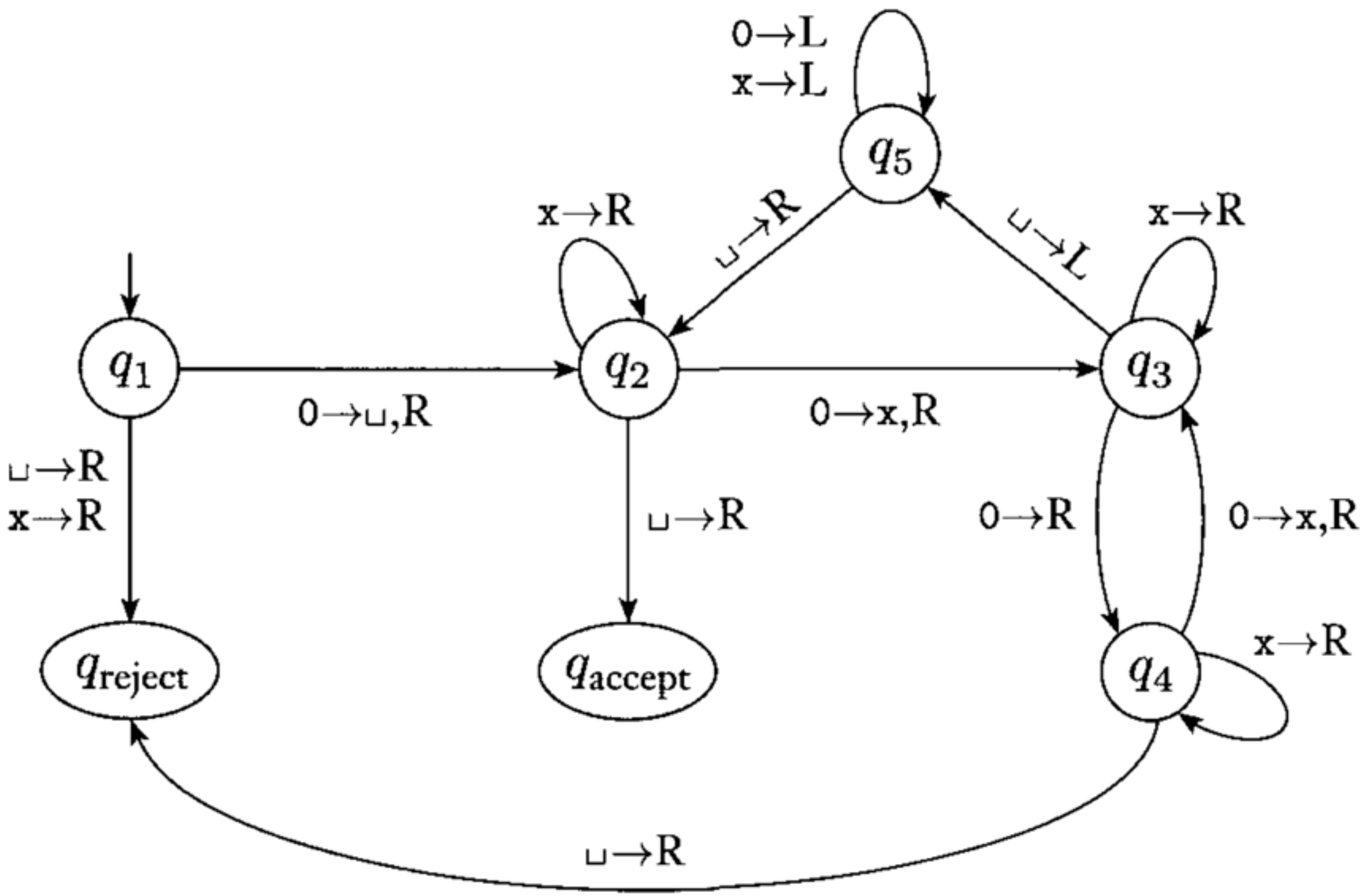


FIGURE 3.8
State diagram for Turing machine M_2

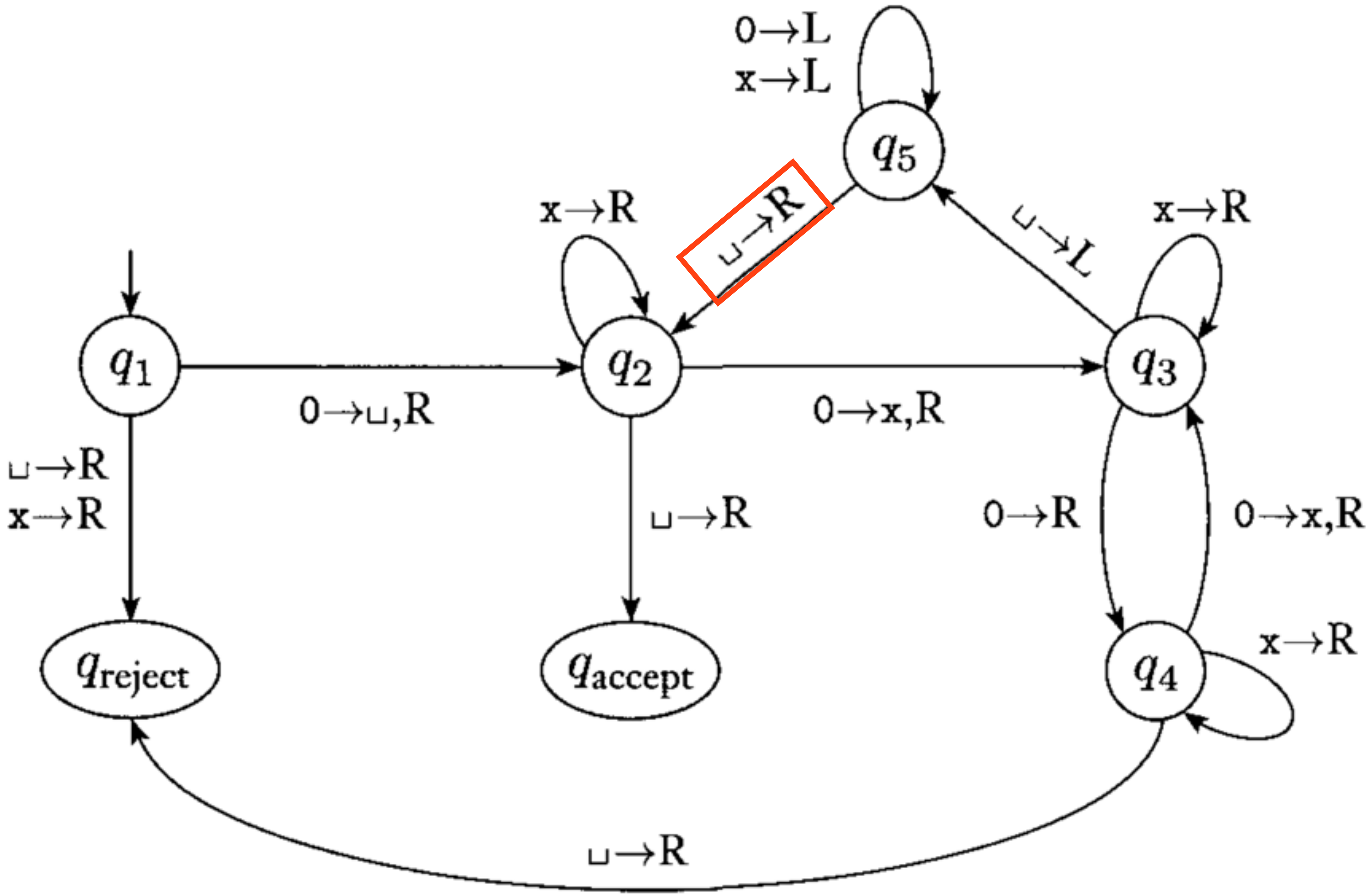


FIGURE 3.8
State diagram for Turing machine M_2

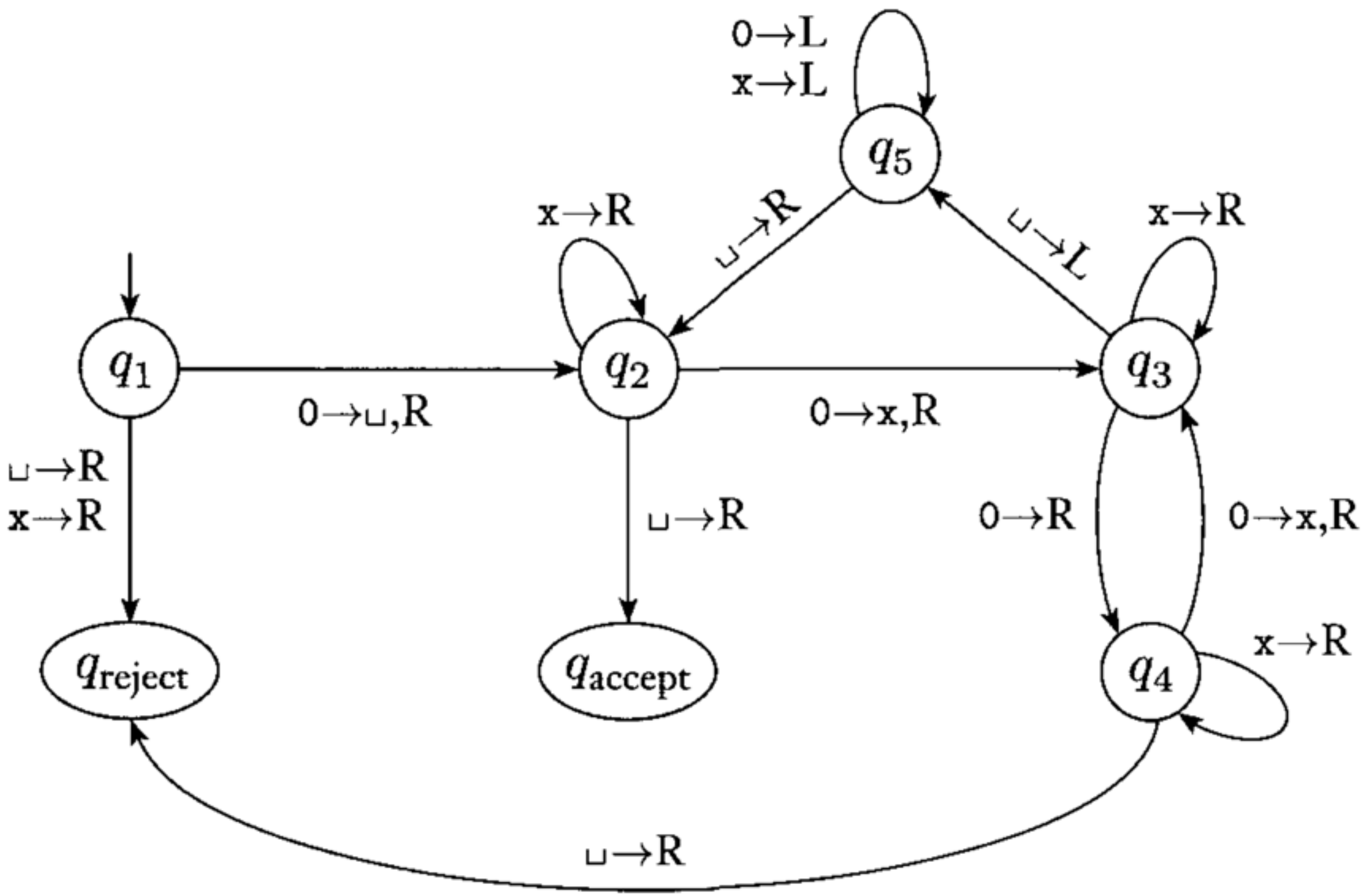


FIGURE 3.8
State diagram for Turing machine M_2

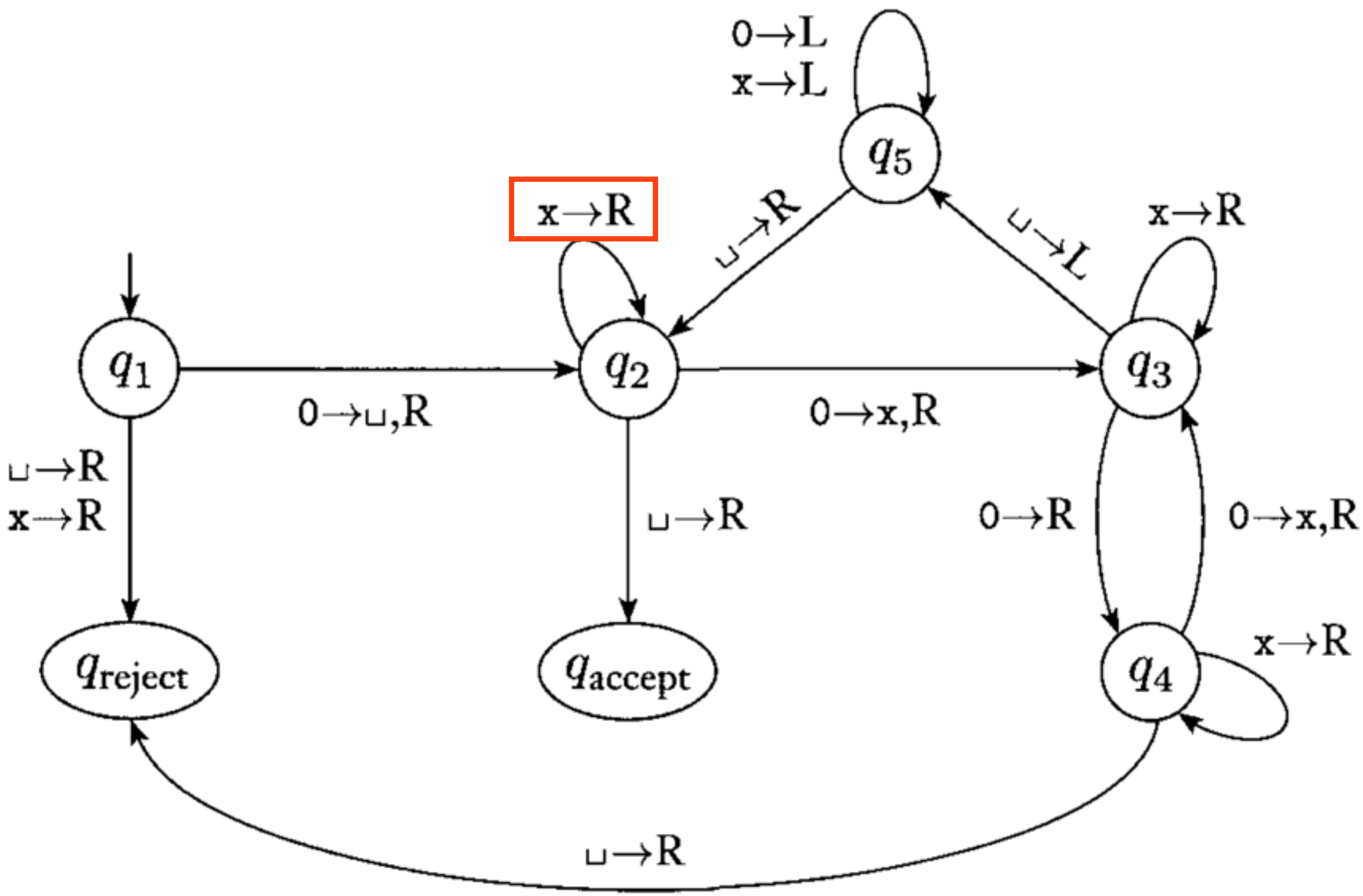


FIGURE 3.8
State diagram for Turing machine M_2

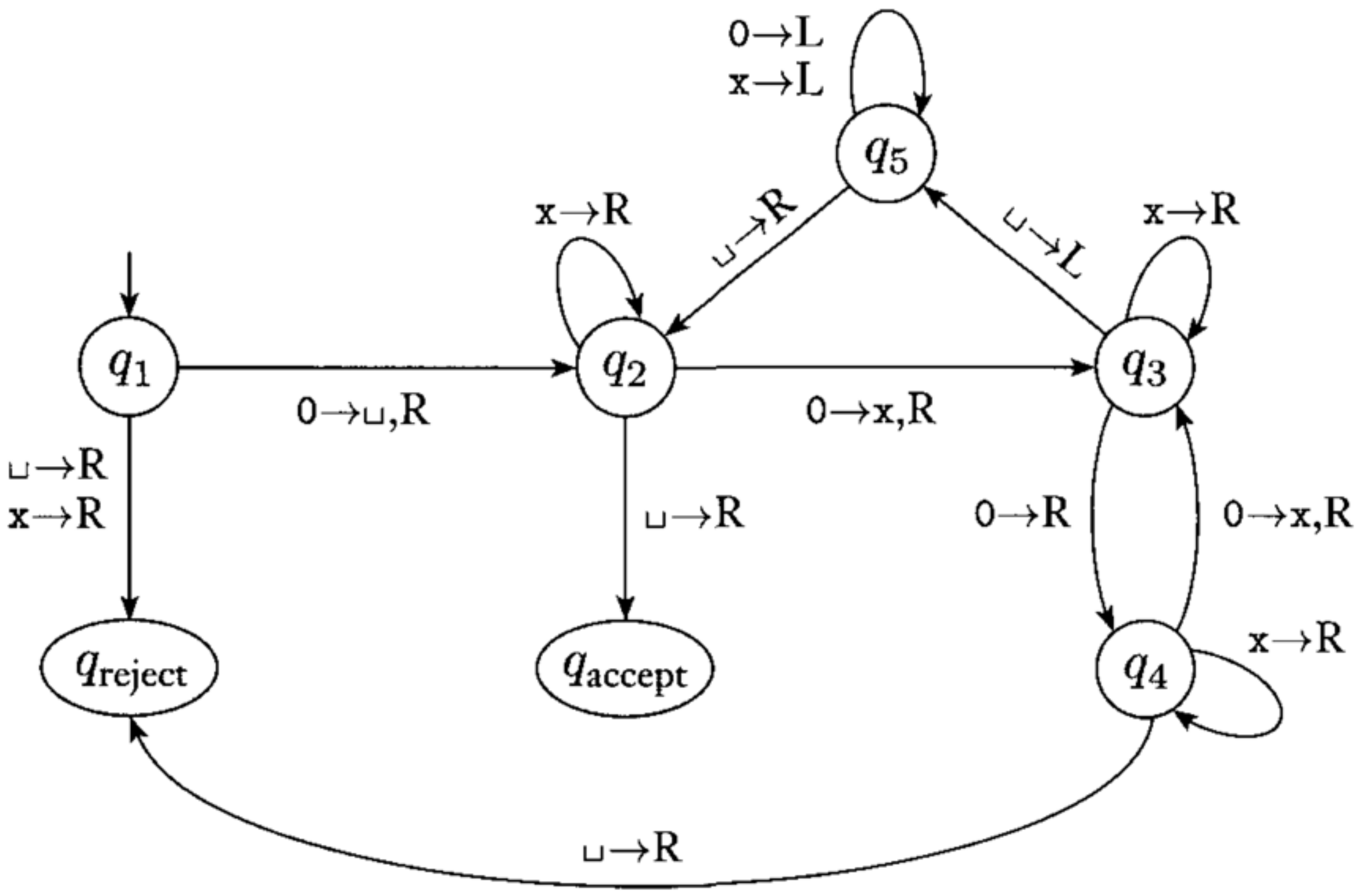


FIGURE 3.8
State diagram for Turing machine M_2

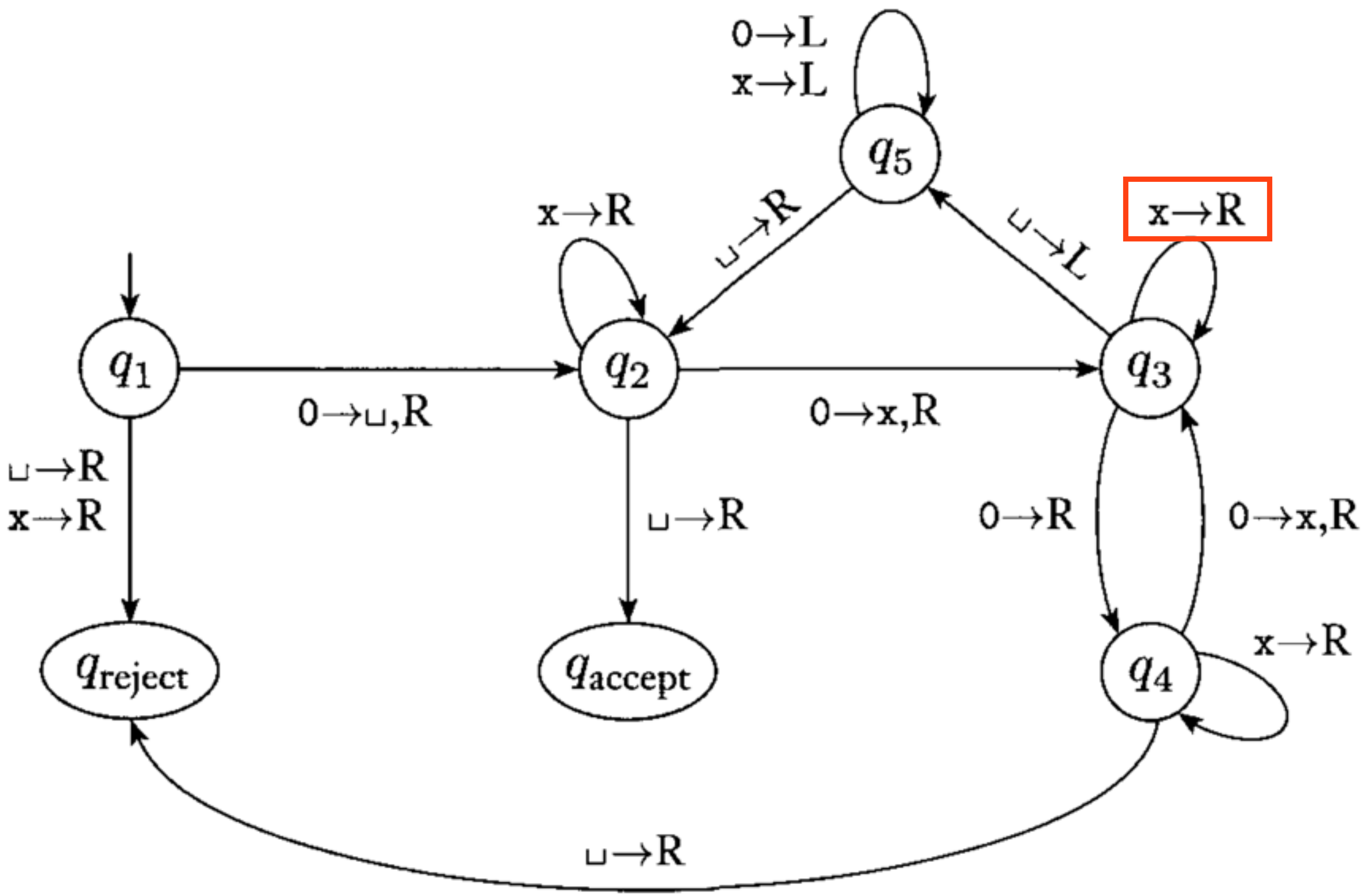


FIGURE 3.8

State diagram for Turing machine M_2

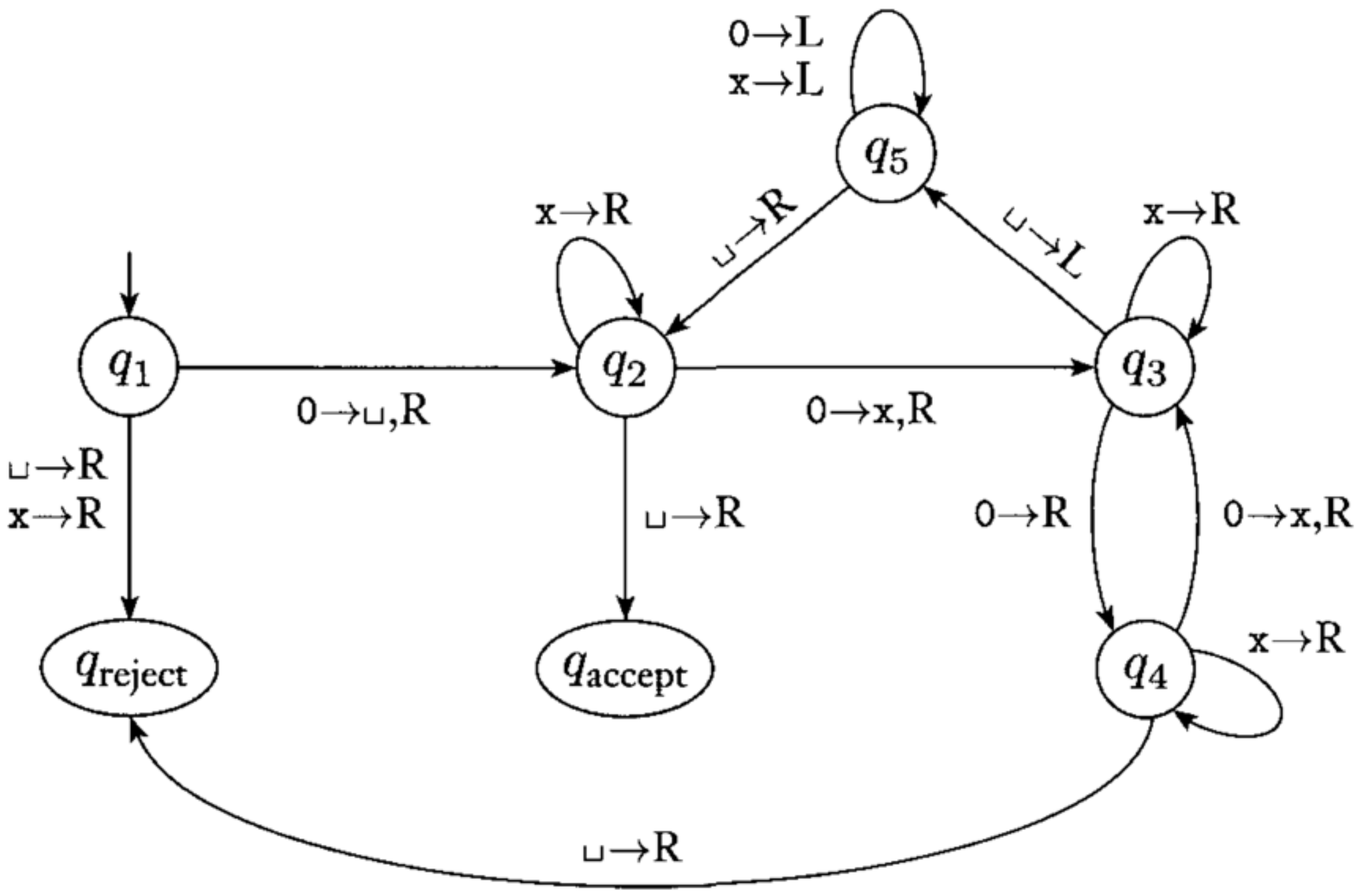


FIGURE 3.8
State diagram for Turing machine M_2

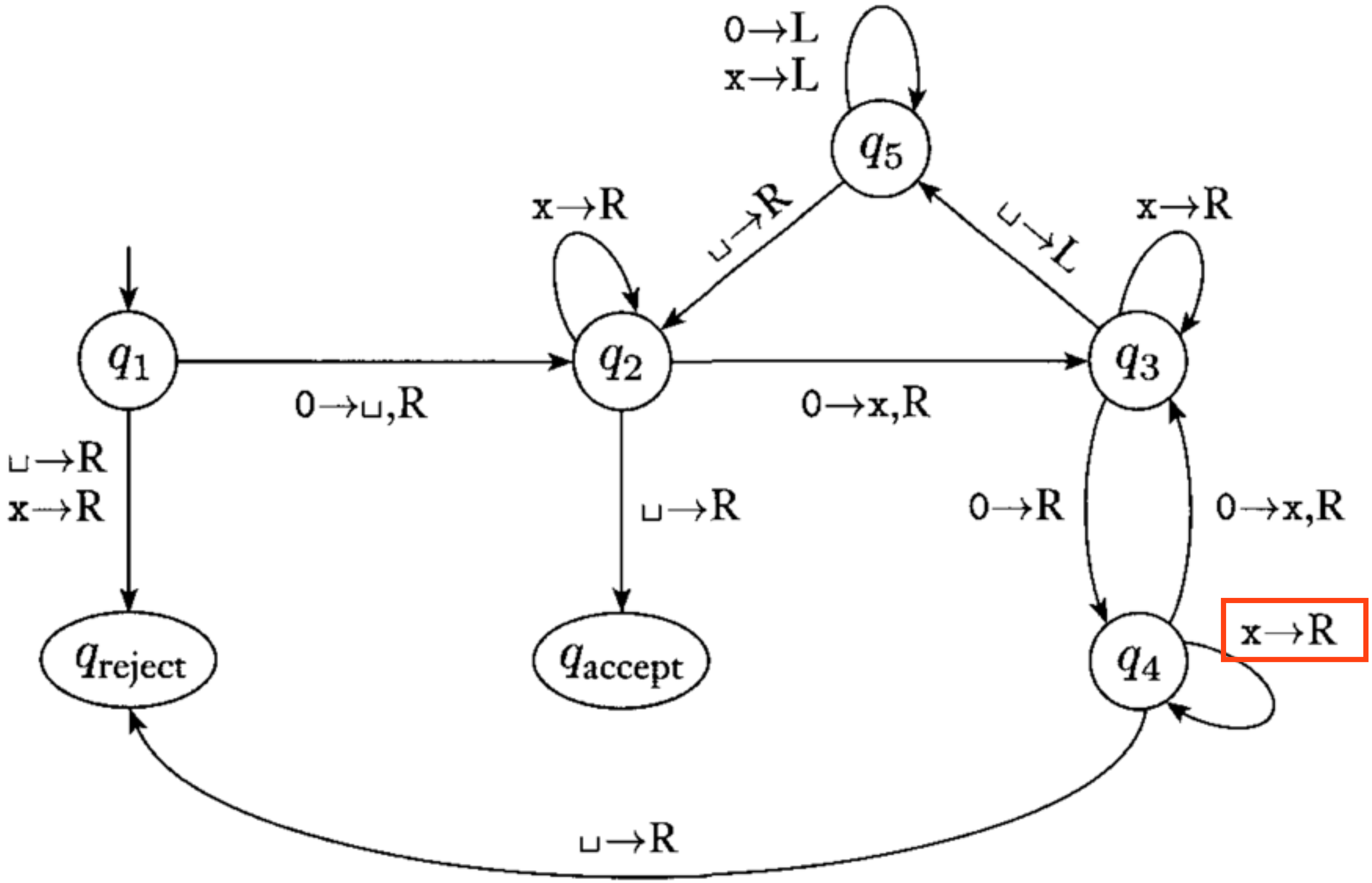


FIGURE 3.8

State diagram for Turing machine M_2

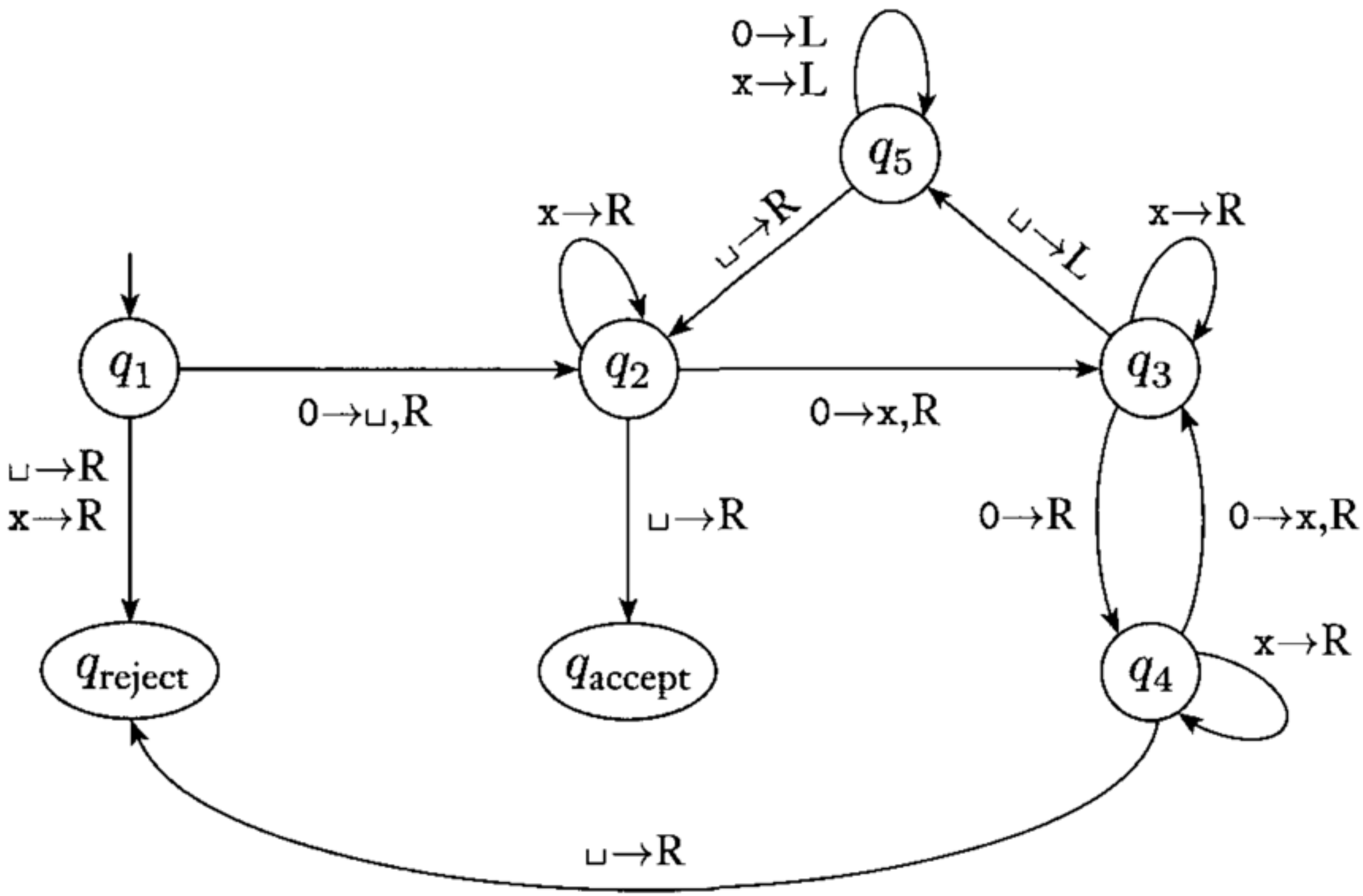


FIGURE 3.8

State diagram for Turing machine M_2

TM Examples

EXAMPLE 3.9

The following is a formal description of $M_1 = (Q, \Sigma, \Gamma, \delta, q_1, q_{accept}, q_{reject})$, the Turing machine that we informally described (page 139) for deciding the language $B = \{w\#w \mid w \in \{0,1\}^*\}$.

- $Q = \{q_1, \ldots, q_{14}, q_{accept}, q_{reject}\}$,
- $\Sigma = \{0,1,\#\}$, and $\Gamma = \{0,1,\#,x,\sqcup\}$.
- We describe δ with a state diagram (see the following figure).
- The start, accept, and reject states are q_1, q_{accept}, and q_{reject}.

M_1 = "On input string w:

1. Zig-zag across the tape to corresponding positions on either side of the # symbol to check whether these positions contain the same symbol. If they do not, or if no # is found, *reject*. Cross off symbols as they are checked to keep track of which symbols correspond.
2. When all symbols to the left of the # have been crossed off, check for any remaining symbols to the right of the #. If any symbols remain, *reject*; otherwise, *accept*."

EXAMPLE 3.9

The following is a formal description of $M_1 = (Q, \Sigma, \Gamma, \delta, q_1, q_{accept}, q_{reject})$, the Turing machine that we informally described (page 139) for deciding the language $B = \{w\#w \mid w \in \{0,1\}^*\}$.

- $Q = \{q_1, \ldots, q_{14}, q_{accept}, q_{reject}\}$,
- $\Sigma = \{0,1,\#\}$, and $\Gamma = \{0,1,\#,x,\sqcup\}$.
- We describe δ with a state diagram (see the following figure).
- The start, accept, and reject states are q_1, q_{accept}, and q_{reject}.

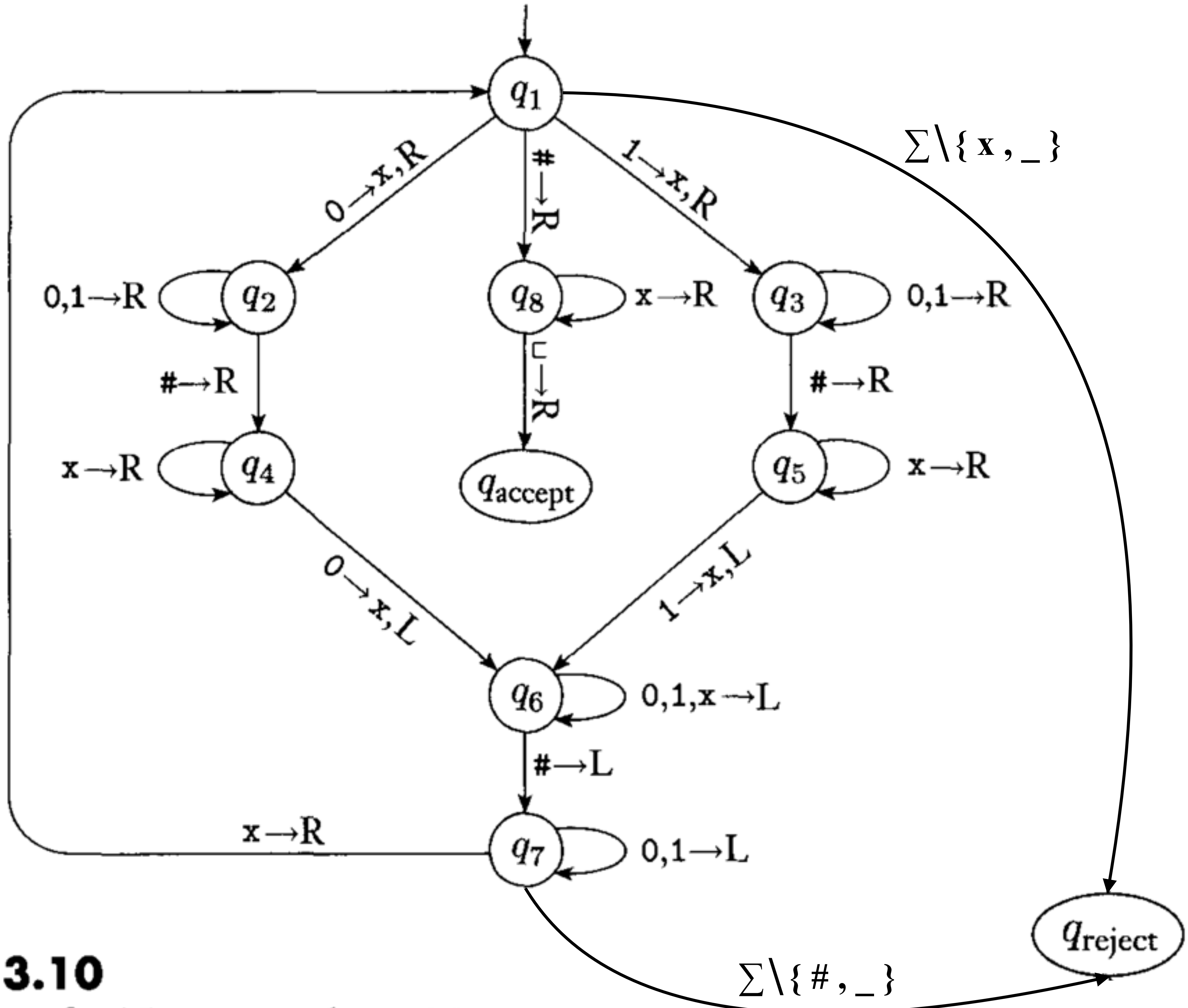


FIGURE 3.10

State diagram for Turing machine M_1

TM Examples

EXAMPLE 3.11

Here, a TM M_3 is doing some elementary arithmetic. It decides the language $C = \{a^i b^j c^k \mid i \times j = k \text{ and } i, j, k \geq 1\}$.

M_3 = "On input string w:

1. Scan the input from left to right to determine whether it is a member of $a^+b^+c^+$ and *reject* if it isn't.
2. Return the head to the left-hand end of the tape.
3. Cross off an a and scan to the right until a b occurs. Shuttle between the b's and the c's, crossing off one of each until all b's are gone. If all c's have been crossed off and some b's remain, *reject*.
4. Restore the crossed off b's and repeat stage 3 if there is another a to cross off. If all a's have been crossed off, determine whether all c's also have been crossed off. If yes, *accept*; otherwise, *reject*."

TM Examples

EXAMPLE 3.12

Here, a TM M_4 is solving what is called the *element distinctness problem*. It is given a list of strings over {0,1} separated by #s and its job is to accept if all the strings are different. The language is

$$E = \{\#x_1\#x_2\# \cdots \#x_l \mid \text{each } x_i \in \{0,1\}^* \text{ and } x_i \neq x_j \text{ for each } i \neq j\}.$$

Machine M_4 works by comparing x_1 with x_2 through x_l, then by comparing x_2 with x_3 through x_l, and so on. An informal description of the TM M_4 deciding this language follows.

TM Examples

$M_4 =$ "On input w:

1. Place a mark on top of the leftmost tape symbol. If that symbol was a blank, *accept*. If that symbol was a #, continue with the next stage. Otherwise, *reject*.
2. Scan right to the next # and place a second mark on top of it. If no # is encountered before a blank symbol, only x_1 was present, so *accept*.
3. By zig-zagging, compare the two strings to the right of the marked #s. If they are equal, *reject*.
4. Move the rightmost of the two marks to the next # symbol to the right. If no # symbol is encountered before a blank symbol, move the leftmost mark to the next # to its right and the rightmost mark to the # after that. This time, if no # is available for the rightmost mark, all the strings have been compared, so *accept*.
5. Go to Stage 3."

More Turing MACHINES

- Multitape Turing Machines
- Non-Deterministic Turing Machines
- Enumerator Turing Machines
- Everything else…

Multitape TM

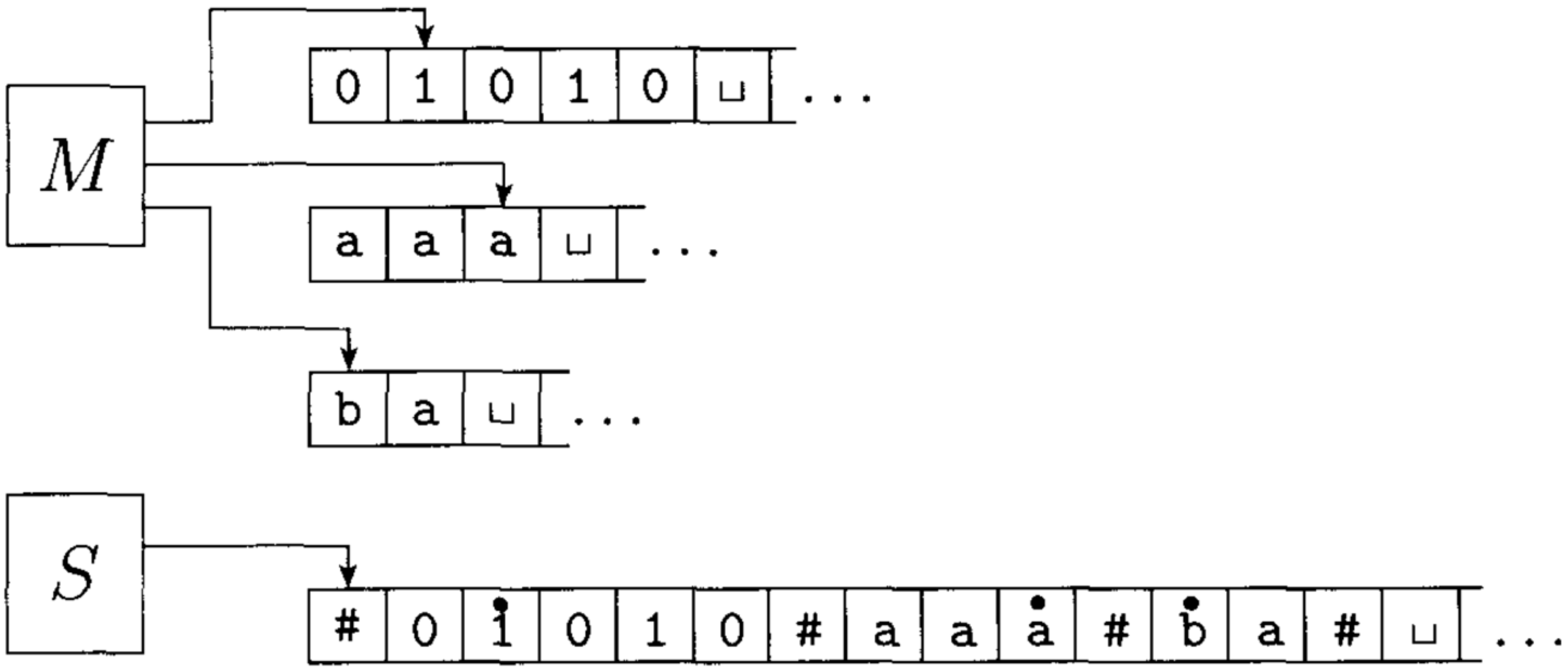


FIGURE 3.14
Representing three tapes with one

Multitape TM

$$\delta\colon Q \times \Gamma^k \longrightarrow Q \times \Gamma^k \times \{\mathrm{L}, \mathrm{R}, \mathrm{S}\}^k,$$

where k is the number of tapes. The expression

$$\delta(q_i, a_1, \ldots, a_k) = (q_j, b_1, \ldots, b_k, \mathrm{L}, \mathrm{R}, \ldots, \mathrm{L})$$

Multitape TM

$$\delta \colon Q \times \Gamma^k \longrightarrow Q \times \Gamma^k \times \{\mathrm{L}, \mathrm{R}, \mathrm{S}\}^k,$$

where k is the number of tapes. The expression

$$\delta(q_i, a_1, \ldots, a_k) = (q_j, b_1, \ldots, b_k, \mathrm{L}, \mathrm{R}, \ldots, \mathrm{L})$$

THEOREM 3.13

Every multitape Turing machine has an equivalent single-tape Turing machine.

Multitape TM

$S =$ "On input $w = w_1 \cdots w_n$:

1. First S puts its tape into the format that represents all k tapes of M. The formatted tape contains

$$\#\dot{w}_1 w_2 \cdots w_n \#\dot{\sqcup}\#\dot{\sqcup}\# \cdots \#$$

2. To simulate a single move, S scans its tape from the first #, which marks the left-hand end, to the $(k+1)$st #, which marks the right-hand end, in order to determine the symbols under the virtual heads. Then S makes a second pass to update the tapes according to the way that M's transition function dictates.
3. If at any point S moves one of the virtual heads to the right onto a #, this action signifies that M has moved the corresponding head onto the previously unread blank portion of that tape. So S writes a blank symbol on this tape cell and shifts the tape contents, from this cell until the rightmost #, one unit to the right. Then it continues the simulation as before."

Multitape TM

COROLLARY 3.15

A language is Turing-recognizable if and only if some multitape Turing machine recognizes it.

PROOF A Turing-recognizable language is recognized by an ordinary (single-tape) Turing machine, which is a special case of a multitape Turing machine. That proves one direction of this corollary. The other direction follows from Theorem 3.13.

Non-deterministic TM

The transition function for a nondeterministic Turing machine has the form

$$\delta\colon Q \times \Gamma \longrightarrow \mathcal{P}(Q \times \Gamma \times \{\mathrm{L}, \mathrm{R}\}).$$

THEOREM 3.16

Every nondeterministic Turing machine has an equivalent deterministic Turing machine.

Non-deterministic TM

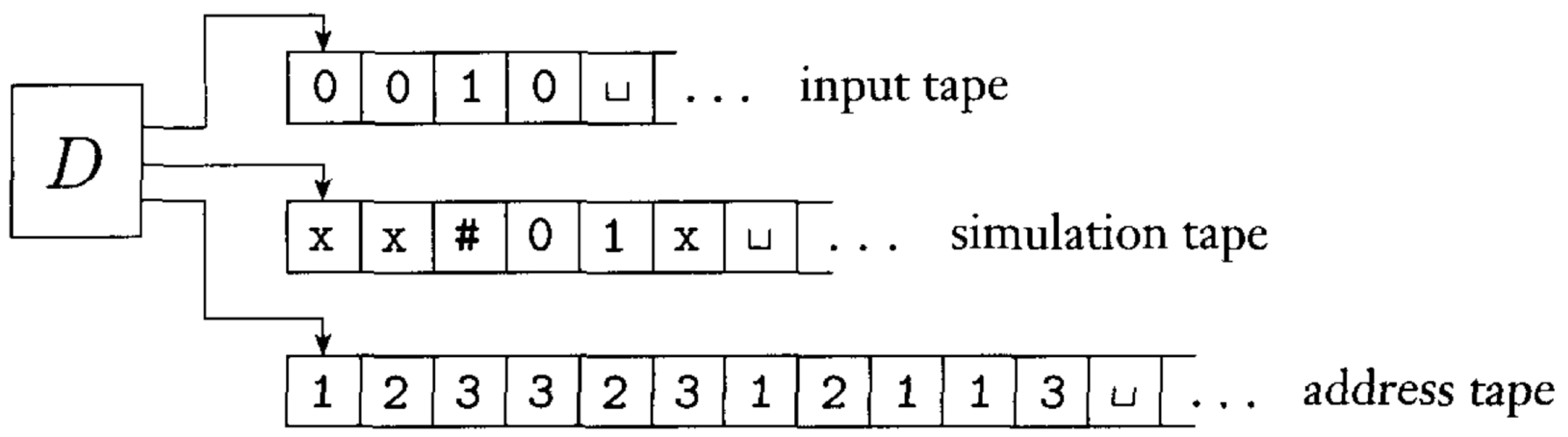


FIGURE 3.17
Deterministic TM D simulating nondeterministic TM N

Non-deterministic TM

1. Initially tape 1 contains the input w, and tapes 2 and 3 are empty.
2. Copy tape 1 to tape 2.
3. Use tape 2 to simulate N with input w on one branch of its nondeterministic computation. Before each step of N consult the next symbol on tape 3 to determine which choice to make among those allowed by N's transition function. If no more symbols remain on tape 3 or if this nondeterministic choice is invalid, abort this branch by going to stage 4. Also go to stage 4 if a rejecting configuration is encountered. If an accepting configuration is encountered, *accept* the input.
4. Replace the string on tape 3 with the lexicographically next string. Simulate the next branch of N's computation by going to stage 2.

Non-deterministic TM

1. Initi
2. Cop
3. Use … etermin-
istic … on tape 3
to d … ransition
func … rminisitic
choi … o stage 4
if a … uration is
enc
4. Rep … g. Simu-
late

Non-deterministic TM

1. Initially tape 1 contains the input w, and tapes 2 and 3 are empty.
2. Copy tape 1 to tape 2.
3. Use tape 2 to simulate N with input w on one branch of its nondeterministic computation. Before each step of N consult the next symbol on tape 3 to determine which choice to make among those allowed by N's transition function. If no more symbols remain on tape 3 or if this nondeterministic choice is invalid, abort this branch by going to stage 4. Also go to stage 4 if a rejecting configuration is encountered. If an accepting configuration is encountered, *accept* the input.
4. Replace the string on tape 3 with the lexicographically next string. Simulate the next branch of N's computation by going to stage 2.

Enumerator TM

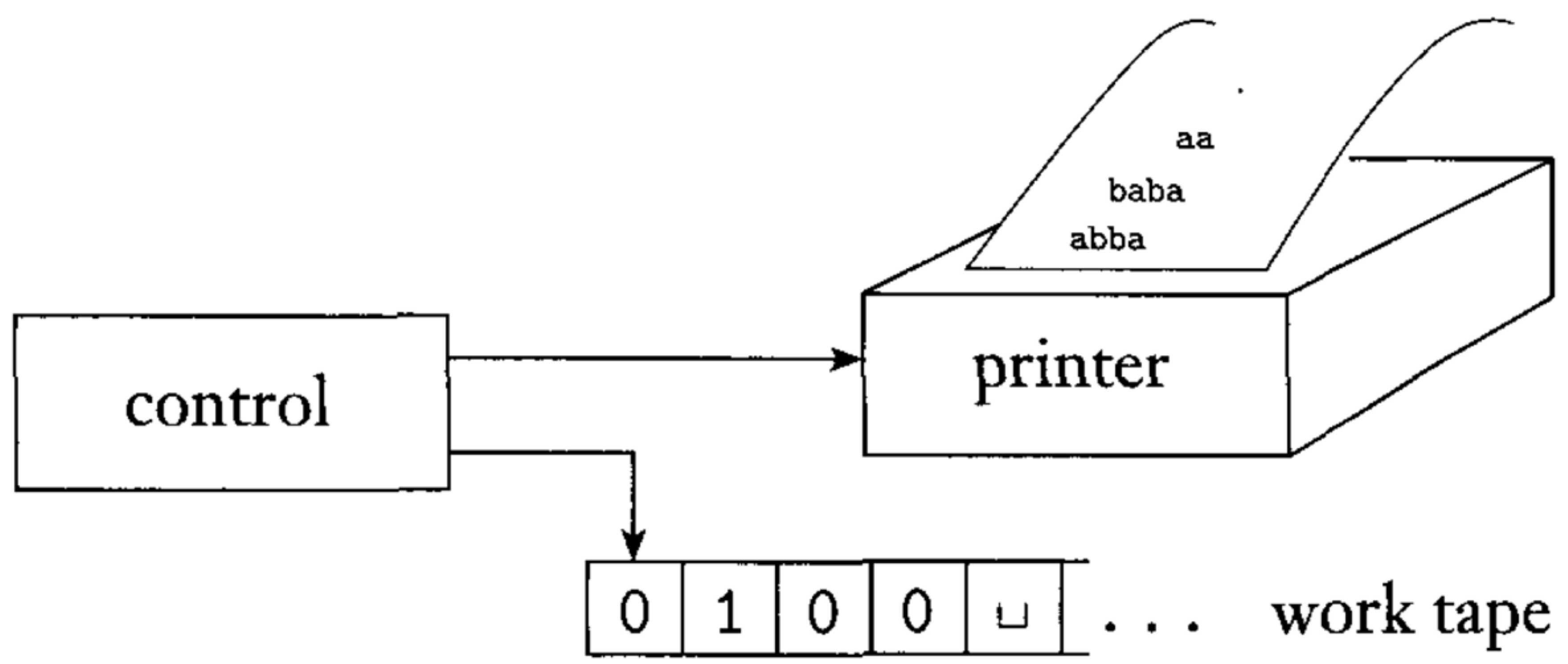


FIGURE 3.20
Schematic of an enumerator

Enumerator TM

THEOREM 3.21

A language is Turing-recognizable if and only if some enumerator enumerates it.

Enumerator TM

PROOF First we show that if we have an enumerator E that enumerates a language A, a TM M recognizes A. The TM M works in the following way.

M = "On input w:

1. Run E. Every time that E outputs a string, compare it with w.
2. If w ever appears in the output of E, *accept*."

Clearly, M accepts those strings that appear on E's list.

Enumerator TM

Now we do the other direction. If TM M recognizes a language A, we can construct the following enumerator E for A. Say that $s_1, s_2, s_3, \ldots$ is a list of all possible strings in Σ^*.

E = "Ignore the input.

1. Repeat the following for $i = 1, 2, 3, \ldots$
2. Run M for i steps on each input, $s_1, s_2, \ldots, s_i$.
3. If any computations accept, print out the corresponding s_j."

If M accepts a particular string s, eventually it will appear on the list generated by E. In fact, it will appear on the list infinitely many times because M runs from the beginning on each string for each repetition of step 1. This procedure gives the effect of running M in parallel on all possible input strings.

Enumerator TM

Now w … 4, we can
construct … list of all
possible st …

E = "Ign …

1. …
2. …
3. … s_j."

If M acce … generated
by E. In … e M runs
from the b … rocedure
gives the e …

Enumerator TM

Now we do the other direction. If TM M recognizes a language A, we can construct the following enumerator E for A. Say that $s_1, s_2, s_3, \ldots$ is a list of all possible strings in Σ^*.

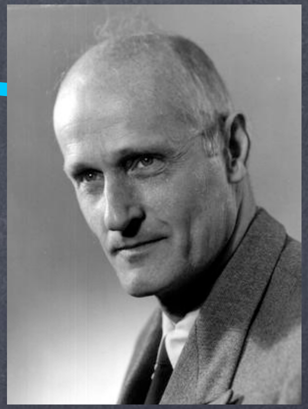
E = "Ignore the input.

1. Repeat the following for $i = 1, 2, 3, \ldots$
2. Run M for i steps on each input, $s_1, s_2, \ldots, s_i$.
3. If any computations accept, print out the corresponding s_j."

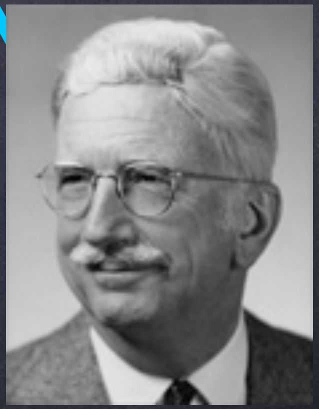
If M accepts a particular string s, eventually it will appear on the list generated by E. In fact, it will appear on the list infinitely many times because M runs from the beginning on each string for each repetition of step 1. This procedure gives the effect of running M in parallel on all possible input strings.

Everything Else

Alonzo Church



Stephen Kleene

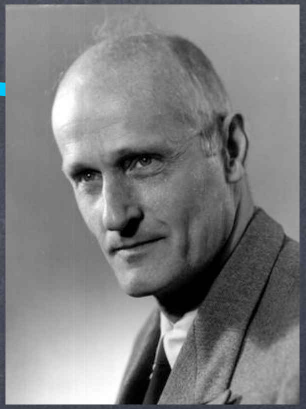


J. Barkley Rosser

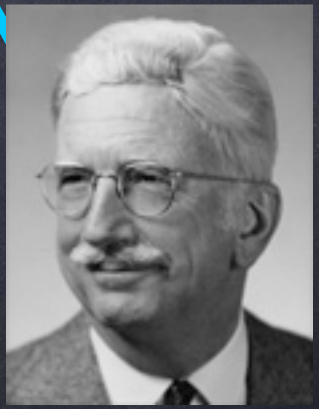
Everything Else

- Lambda-calculus

Alonzo Church



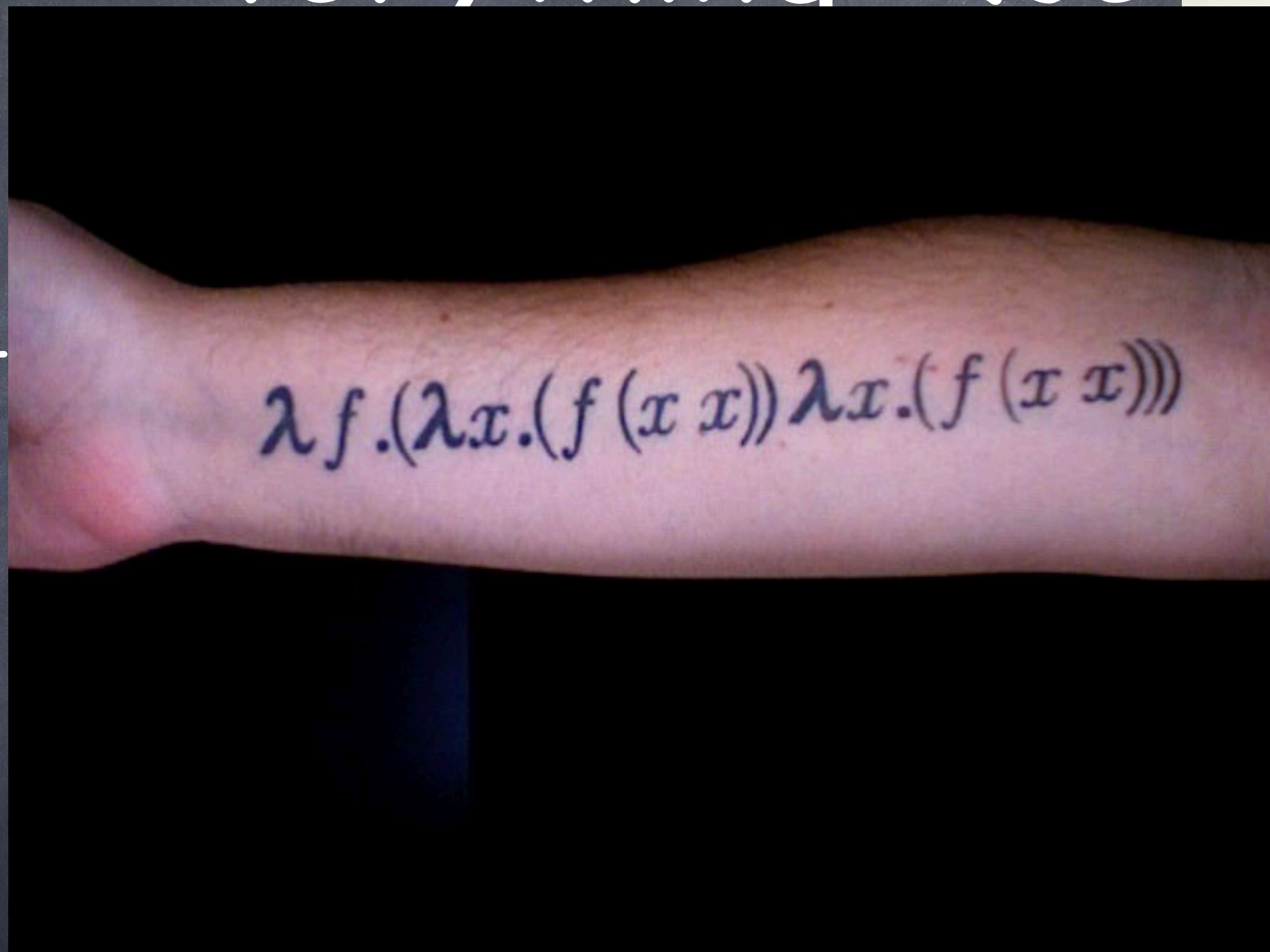
Stephen Kleene



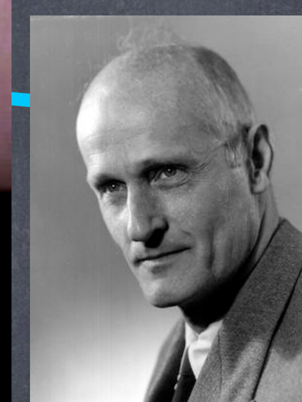
J. Barkley Rosser

Everything Else

- L




Alonzo Church

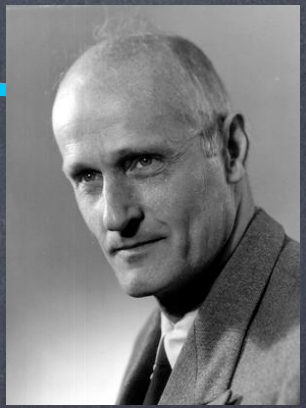

Stephen Kleene


J. Barkley Rosser

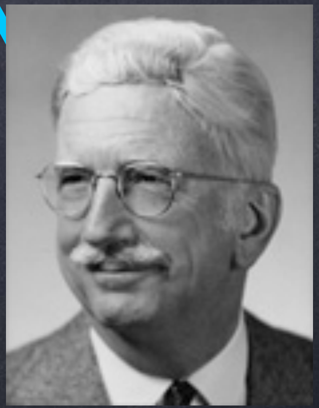
Everything Else

- Lambda-calculus

Alonzo Church



Stephen Kleene

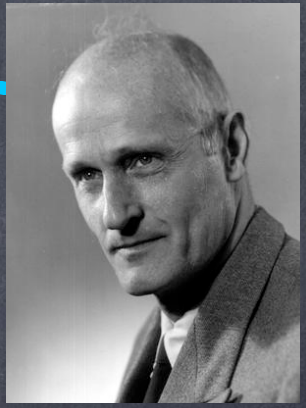


J. Barkley Rosser

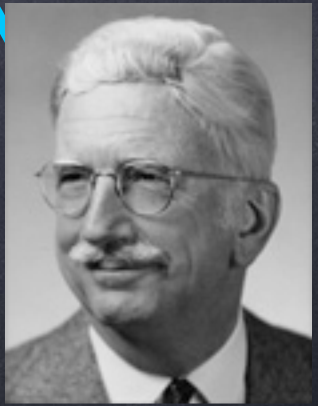
Everything Else

- Lambda-calculus
- Recursive Functions

Alonzo Church



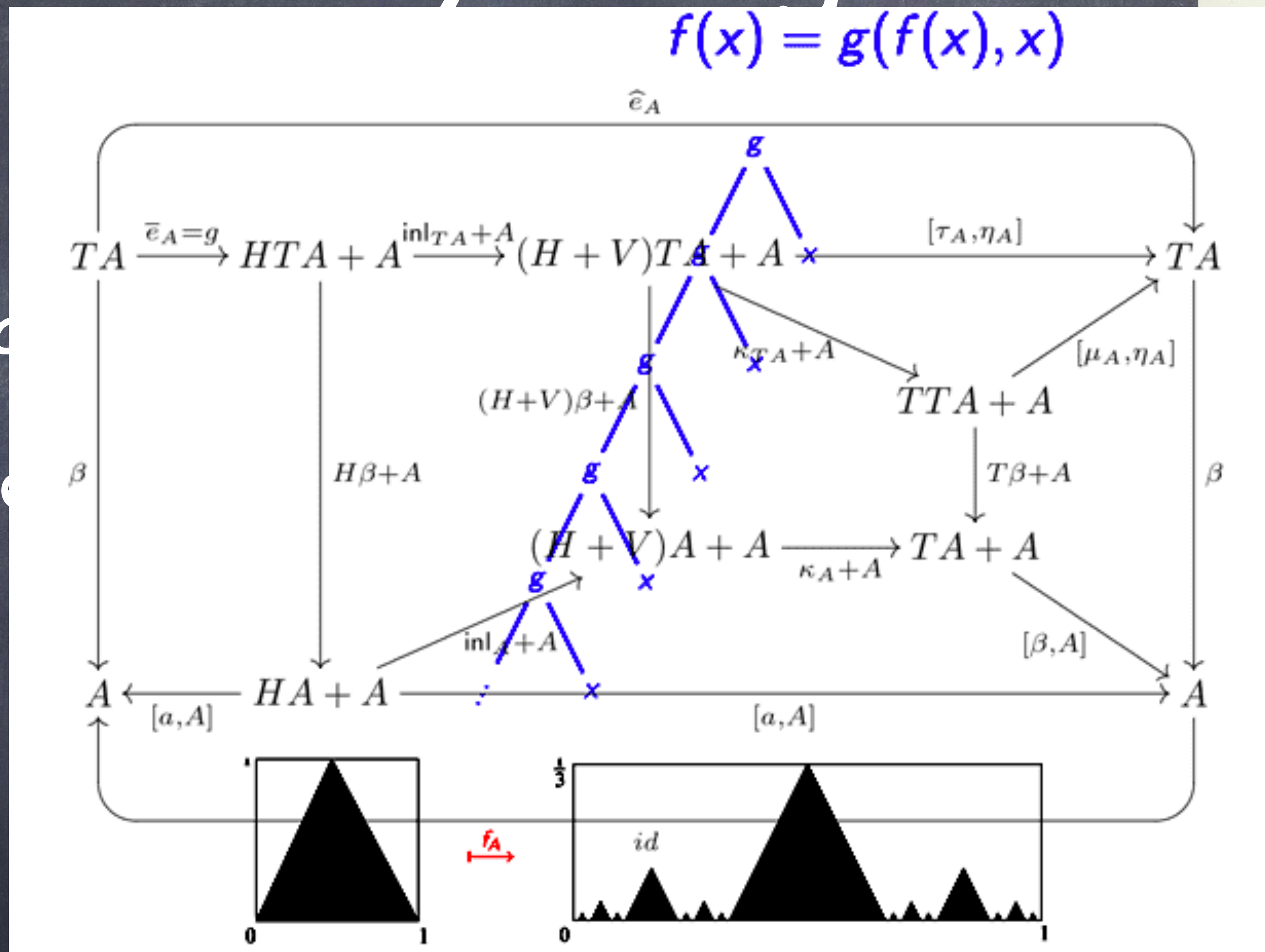
Stephen Kleene



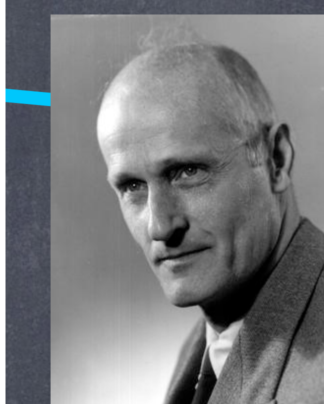
J. Barkley Rosser

Everything Else

- Lo
- Re




Alonzo Church

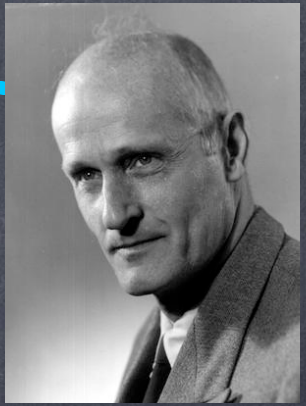

Stephen Kleene


J. Barkley Rosser

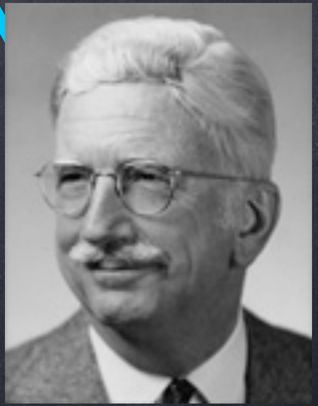
Everything Else

- Lambda-calculus
- Recursive Functions

Alonzo Church



Stephen Kleene

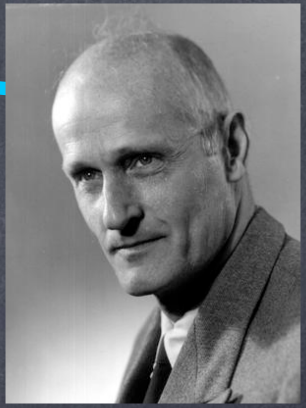


J. Barkley Rosser

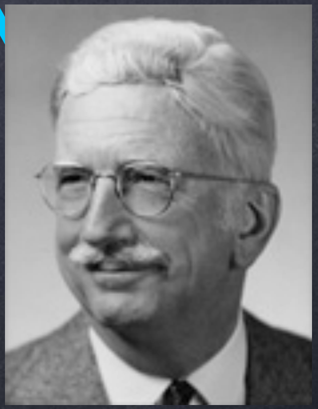
Everything Else

- Lambda-calculus
- Recursive Functions
- Programming languages:

Alonzo Church



Stephen Kleene

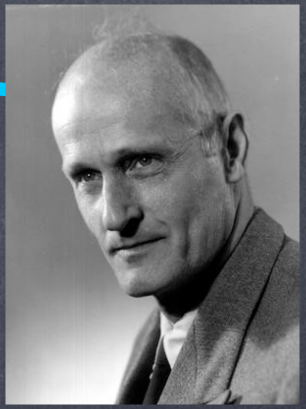


J. Barkley Rosser

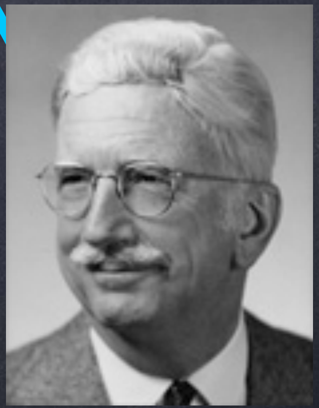
Everything Else

- Lambda-calculus
- Recursive Functions
- Programming languages:
  - FORTRAN, PASCAL, C, JAVA,...

Alonzo Church



Stephen Kleene

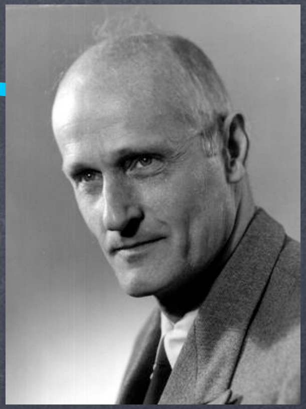


J. Barkley Rosser

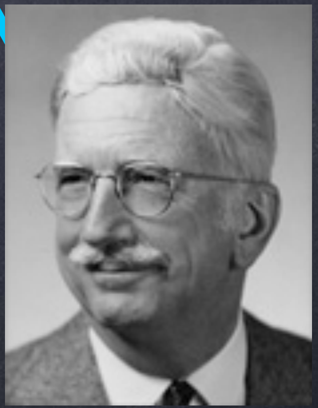
Everything Else

- Lambda-calculus
- Recursive Functions
- Programming languages:
  - FORTRAN, PASCAL, C, JAVA,...
  - LISP, SCHEME,...

Alonzo Church



Stephen Kleene



J. Barkley Rosser

Church-Turing Thesis

Alonzo Church

Alan Turing

Church-Turing Thesis

Intuitive notion of algorithms	equals	*Turing machine algorithms*

FIGURE 3.22
The Church–Turing Thesis

Paris, 1900


David Hilbert

Paris, 1900


David Hilbert

- Speaking on 8 August 1900, at the Paris 2nd International Congress of Mathematicians, at La Sorbonne, German mathematician David Hilbert presented ten problems in mathematics.


David Hilbert

Paris, 1900

- Speaking on 8 August 1900, at the Paris 2^{nd} International Congress of Mathematicians, at La Sorbonne, German mathematician David Hilbert presented ten problems in mathematics.
- The problems were all unsolved at the time, and several of them turned out to be very influential for 20^{th} century mathematics.

Hilbert's 10^{th} problem

- Let P be a polynomial in **several variables**:
  $$P(x,y,z)=24x^2y^3+17x+5y+25$$
- Is there a set of integers for x,y,z such that
  $$P(x,y,z)=0\ ?$$
- This problem is undecidable… but is Turing-Recognizable…
- Needed a formal model of computing to prove impossibility.

Yuri Matiyasevich

Single variable Poly

Let

$$D_1 = \{p \mid p \text{ is a polynomial over } x \text{ with an integral root}\}.$$

Here is a TM M_1 that recognizes D_1:

M_1 = "The input is a polynomial p over the variable x.

1. Evaluate p with x set successively to the values 0, 1, -1, 2, -2, 3, -3, ... If at any point the polynomial evaluates to 0, *accept*."

Single variable Poly

Let

$$D_1 = \{p | \, p \text{ is a polynomial over } x \text{ with an integral root}\}.$$

Here is a TM M_1 that recognizes D_1:

$M_1 =$ "The input is a polynomial p over the variable x.

1. Evaluate p with x set successively to the values 0, 1, −1, 2, −2, 3, −3, ... If at any point the polynomial evaluates to 0, *accept*."

3.21 Let $c_1x^n + c_2x^{n-1} + \cdots + c_nx + c_{n+1}$ be a polynomial with a root at $x = x_0$. Let $c_{\max}$ be the largest absolute value of a c_i. Show that

$$|x_0| < (n+1)\frac{c_{\max}}{|c_1|}.$$

COMP-330
Theory of Computation

Fall 2019 -- Prof. Claude Crépeau

Lec. 16-17 :
Turing Machines &
Church-Turing Thesis