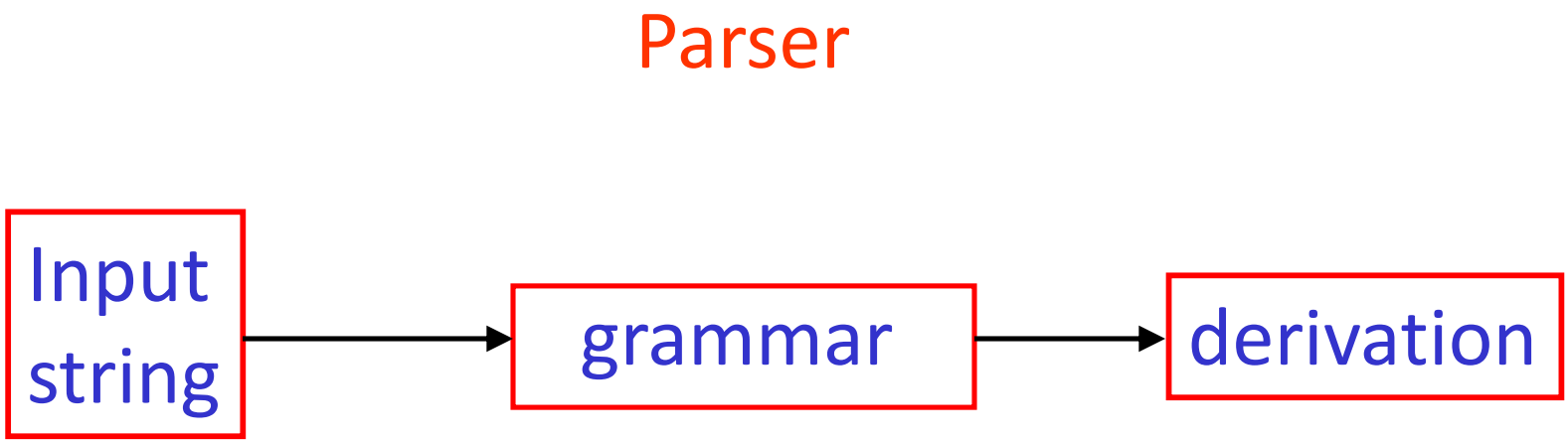
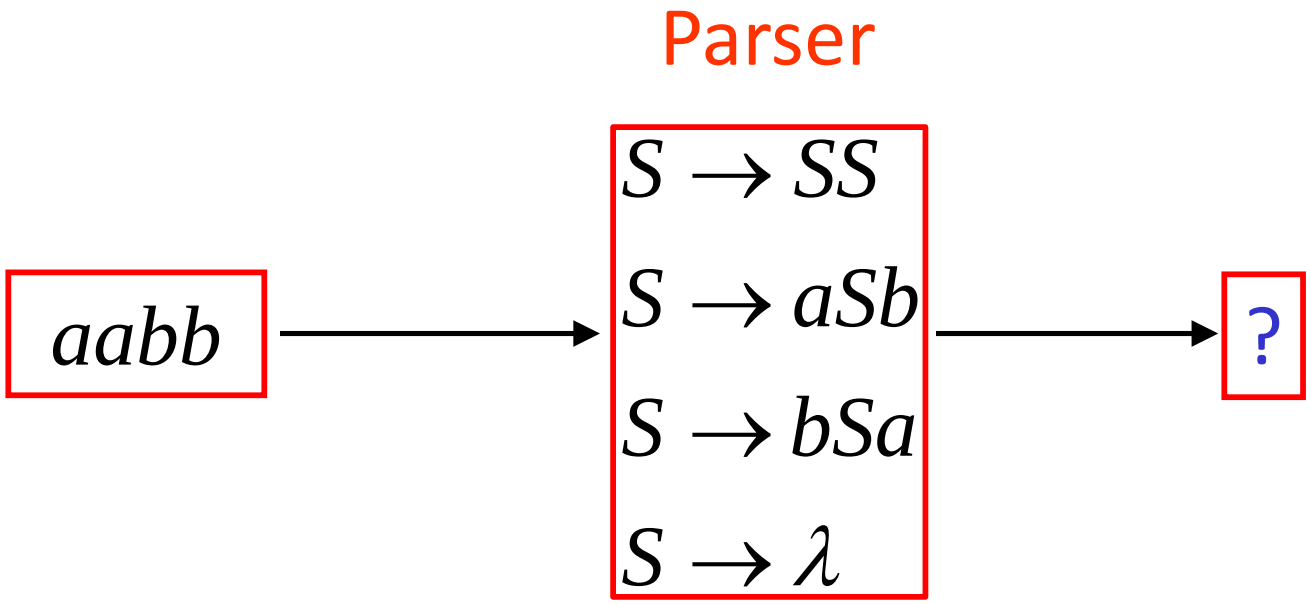
Parsing

Parser



Parser

- **Example**



Exhaustive Search

$S \to SS \mid aSb \mid bSa \mid \lambda$ Find derivation of $aabb$

- Phase 1:

$$S \Rightarrow SS$$

$$S \Rightarrow aSb$$

$$S \Rightarrow bSa$$

$$S \Rightarrow \lambda$$

- All possible derivations of length 1

Exhaustive Search

$S \to SS \mid aSb \mid bSa \mid \lambda$ Find derivation of $aabb$

- Phase 1:

$$S \Rightarrow SS$$

$$S \Rightarrow aSb$$

$$S \Rightarrow bSa$$

$$S \Rightarrow \lambda$$

- All possible derivations of length 1

Exhaustive Search

$$S \to SS \mid aSb \mid bSa \mid \lambda$$

- Phase 2:

$S \Rightarrow SS$

$S \Rightarrow aSb$

$S \Rightarrow SS \Rightarrow SSS$

$S \Rightarrow SS \Rightarrow aSbS$

$S \Rightarrow SS \Rightarrow bSaS$

$S \Rightarrow SS \Rightarrow S$

$S \Rightarrow aSb \Rightarrow aSSb$

$S \Rightarrow aSb \Rightarrow aaSbb$

$S \Rightarrow aSb \Rightarrow abSab$

$S \Rightarrow aSb \Rightarrow ab$

$aabb$

Exhaustive Search

$$S \rightarrow SS \mid aSb \mid bSa \mid \lambda$$

- Phase 2:

$S \Rightarrow SS$

$S \Rightarrow aSb$

$S \Rightarrow SS \Rightarrow SSS$

$S \Rightarrow SS \Rightarrow aSbS$

$S \Rightarrow SS \Rightarrow bSaS$

$S \Rightarrow SS \Rightarrow S$

$S \Rightarrow aSb \Rightarrow aSSb$

$S \Rightarrow aSb \Rightarrow aaSbb$

$S \Rightarrow aSb \Rightarrow abSab$

$S \Rightarrow aSb \Rightarrow ab$

$aabb$

Exhaustive Search

$$S \rightarrow SS \mid aSb \mid bSa \mid \lambda$$

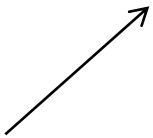
$S \Rightarrow SS \Rightarrow SSS$

$S \Rightarrow SS \Rightarrow aSbS$

$S \Rightarrow SS \Rightarrow S$

- Phase 3:

$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aabb$

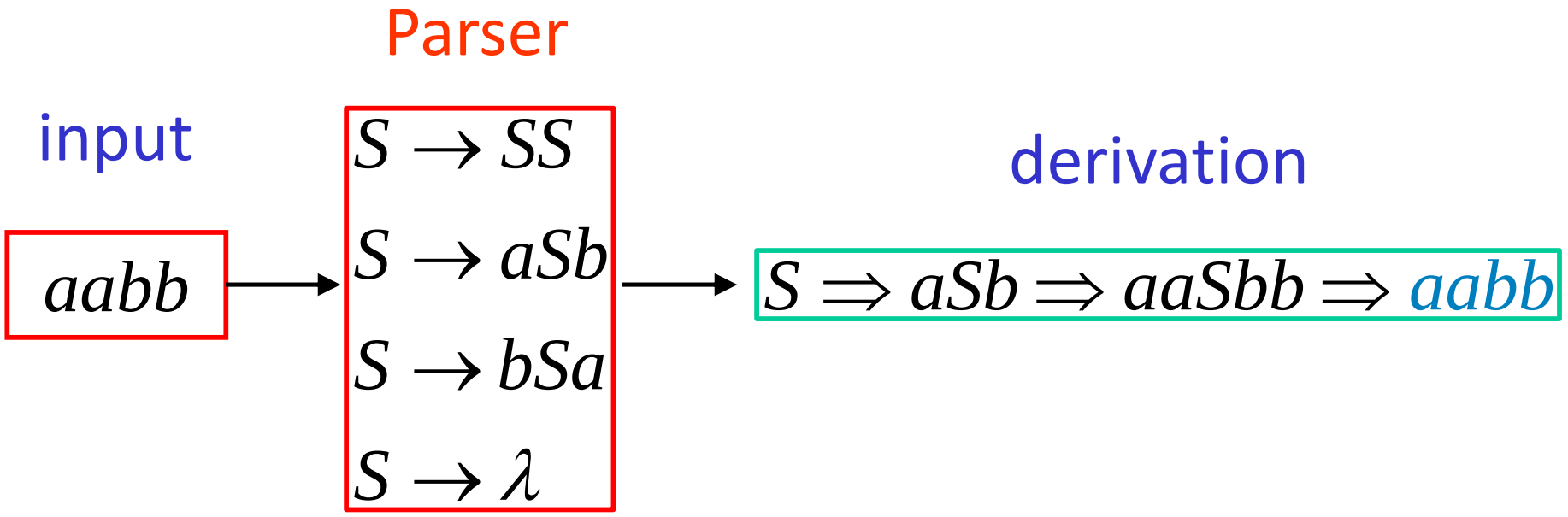


$S \Rightarrow aSb \Rightarrow aSSb$

$S \Rightarrow aSb \Rightarrow aaSbb$

Exhaustive Search

$$S \to SS \mid aSb \mid bSa \mid \lambda$$



Time Complexity of Exhaustive Search

- Suppose there are no productions of the form

$$A \to \lambda$$

$$A \to B$$

- Number of phases for string w : $2|w|$

Time Complexity of Exhaustive Search

- For grammar with k rules
- Time for phase 1: k

k possible derivations

Time Complexity of Exhaustive Search

- For grammar with k rules
- Time for phase 2: k^2

k^2 possible derivations

Time Complexity of Exhaustive Search

- For grammar with k rules
- Time for phase $2|w|$: $k^{2|w|}$

$k^{2|w|}$ possible derivations

Time Complexity of Exhaustive Search

- Total time needed for string w:

$$k + k^2 + \mathrm{L} + k^{2|w|}$$

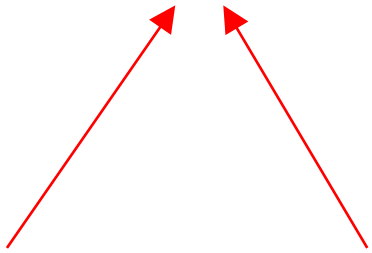
phase 1 phase 2 phase $2|w|$

Extremely bad!!!

Another Algorithm

- There exist faster algorithms for specialized grammars

s-grammar: $A \to ax$



symbol

string of variables

Pair (A, a) appears once

Another Algorithm

- s-grammar example:

$$S \rightarrow aS$$
$$S \rightarrow bSS$$
$$S \rightarrow c$$

- Each string has a unique derivation

$$S \Rightarrow aS \Rightarrow abSS \Rightarrow abcS \Rightarrow abcc$$

Another Algorithm

- For s-grammars: in the exhaustive search parsing there is only one choice in each phase
- Time for each phase: 1
- Total time for the parsing string w: $|w|$

Another Algorithm

- For general context-free grammars:

  There exists a parsing algorithm

  that parses a string w

  in time $|w|^3$.

Normal Forms for Context-free Grammars

* اَللّٰـهُمَّ إِنَّـا نَسْأَلُكَ لِسَانًا رَطِبًا بِذِكْرِكَ
وَقَلْبًـا مَفْعُمًـا بِشُكْـرِكَ وَبَدَنًا هَيِّنًا لَيِّنًا
بِطَاعَتِكَ اَللّٰهُمَّ إِنَّا نَسْأَلُكَ إِيْمَانًا كَامِلاً

وَنَسْأَلُكَ قَلْبًا خَاشِعًا وَنَسْأَلُكَ عِلْمًا نَافِعًا
وَنَسْأَلُكَ يَقِيْنًا صَادِقًا وَنَسْأَلُكَ دِيْنًا قَيِّمًا
وَنَسْأَلُكَ الْعَافِيَةَ مِنْ كُلِّ بَلِيَّةٍ وَنَسْأَلُكَ
تَمَامَ الْغِنَى عَنِ النَّاسِ وَهَبْ لَنَا حَقِيْقَةَ
الْإِيْمَـانِ بِكَ حَتَّى لاَ نَخَافَ وَلاَ نَرْجُوْ
غَيْرَكَ وَلاَ نَعْبُدَ شَيْئًا سِوَاكَ وَاجْعَلْ يَدَكَ
مَبْسُـوْطَةً عَلَيْنَا وَعَلى أَهْلِيْنَا وَأَوْلاَدِنَا
وَمَنْ مَعَنَـا بِرَحْمَتِـكَ وَلاَ تَكِـلْنَـا إِلَى
أَنْفُسِنَا طَرْفَةَ عَيْنٍ وَلاَ أَقَلَّ مِنْ ذٰلِكَ يَا
نِعْمَ الْمُجِيْبُ.

Context-free Grammars

- A context-free grammar:

$$G = (V, T, S, P)$$

- V: a set of variables (nonterminals)
- T: a set of terminals
- S: the start symbol (the axiom)
- P: a set of production rules of the form

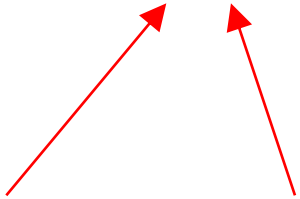
$$A \to x$$

- x is a string of variables and terminals

Chomsky Normal Form

- All productions have form:

$A \to BC$ and $A \to a$



variable variable terminal

Examples

$S \to AS$

$S \to a$

$A \to SA$

$A \to b$

Chomsky Normal Form

$S \to AS$

$S \to AAS$

$A \to SA$

$A \to aa$

Not Chomsky Normal Form

Conversion to Chomsky Normal Form

- **Example:**

$$S \rightarrow ABa$$

$$A \rightarrow aab$$

$$B \rightarrow Ac$$

Not Chomsky Normal Form

Conversion to Chomsky Normal Form

- Introduce variables for terminals: T_a, T_b, T_c

$$S \to ABa$$
$$A \to aab$$
$$B \to Ac$$

$$S \to ABT_a$$
$$A \to T_a T_a T_b$$
$$B \to AT_c$$
$$T_a \to a$$
$$T_b \to b$$
$$T_c \to c$$

Conversion to Chomsky Normal Form

- Introduce intermediate variable: V_1

$$S \to ABT_a$$
$$A \to T_a T_a T_b$$
$$B \to AT_c$$
$$T_a \to a$$
$$T_b \to b$$
$$T_c \to c$$

$$S \to AV_1$$
$$V_1 \to BT_a$$
$$A \to T_a T_a T_b$$
$$B \to AT_c$$
$$T_a \to a$$
$$T_b \to b$$
$$T_c \to c$$

Conversion to Chomsky Normal Form

- Introduce intermediate variable: V_2

$$S \to AV_1$$
$$V_1 \to BT_a$$
$$A \to T_aT_aT_b$$
$$B \to AT_c$$
$$T_a \to a$$
$$T_b \to b$$
$$T_c \to c$$

$$S \to AV_1$$
$$V_1 \to BT_a$$
$$A \to T_aV_2$$
$$V_2 \to T_aT_b$$
$$B \to AT_c$$
$$T_a \to a$$
$$T_b \to b$$
$$T_c \to c$$

Conversion to Chomsky Normal Form

- Final grammar in Chomsky Normal Form:

Initial grammar

$$S \to ABa$$
$$A \to aab$$
$$B \to Ac$$

$$S \to AV_1$$
$$V_1 \to BT_a$$
$$A \to T_aV_2$$
$$V_2 \to T_aT_b$$
$$B \to AT_c$$
$$T_a \to a$$
$$T_b \to b$$
$$T_c \to c$$

Conversion to Chomsky Normal Form

- In general:
- From any context-free grammar not in Chomsky Normal Form
- We can obtain an equivalent grammar in Chomsky Normal Form

Conversion to Chomsky Normal Form

- The procedure:
- First remove:

  Nullable variables

  Unit productions

Conversion to Chomsky Normal Form

- For every terminal symbol a

  Add production $T_a \to a$

  In productions: replace a with T_a

  New variable: T_a

Conversion to Chomsky Normal Form

- Replace production $A \to C_1 C_2 \mathrm{L}\ C_n$

with

$$A \to C_1 V_1$$
$$V_1 \to C_2 V_2$$
$$\mathrm{L}$$
$$V_{n-2} \to C_{n-1} C_n$$

New intermediate variables: $V_1, V_2, \mathrm{K}, V_{n-2}$

Chomsky Normal Form

- **Theorem**

  For any context-free grammar
  there is an equivalent grammar
  in Chomsky Normal Form

Chomsky Normal Form

- Observations

  Chomsky normal forms are good for parsing and proving theorems

  It is very easy to find the Chomsky normal form of any context-free grammar

An Application of Chomsky Normal Forms

The CYK Membership Algorithm

- Input:

  Grammar G in Chomsky Normal Form

  String w

- Output:

  Find if $w \in L(G)$

Cocke–Younger–Kasami

the **Cocke–Younger–Kasami algorithm** (alternatively called **CYK**, or **CKY**) is a parsing algorithm for context-free grammars published by Itiroo Sakai in 1961.[1] The algorithm is named after some of its rediscoverers: John Cocke, Daniel Younger, Tadao Kasami, and Jacob T. Schwartz. It employs bottom-up parsing and dynamic programming.

The Algorithm

- Input example:

  Grammar G

$$S \to AB$$
$$A \to BB$$
$$A \to a$$
$$B \to AB$$
$$B \to b$$

  String w: $aabbb$

The Algorithm

$$S \to AB, A \to BB, A \to a, B \to AB, B \to b$$

a	a	b	b	b
aa	ab	bb	bb	
aab	abb	bbb		
$aabb$	$abbb$			
$aabbb$				

The Algorithm

$$S \to AB, A \to BB, A \to a, B \to AB, B \to b$$

a	a	b	b	b
A	A	B	B	B
aa	ab	bb	bb	
aab	abb	bbb		
$aabb$	$abbb$			
$aabbb$				

The Algorithm

$S \to AB, A \to BB, A \to a, B \to AB, B \to b$

a	a	b	b	b
A	A	B	B	B
aa	ab	bb	bb	
	S,B	A	A	
aab	abb	bbb		
$aabb$	$abbb$			
$aabbb$				

The Algorithm

$$S \to AB, A \to BB, A \to a, B \to AB, B \to b$$

a	a	b	b	b
A	A	B	B	B
aa	ab	bb	bb	
	S,B	A	A	
aab	abb	bbb		
S,B	A	S,B		
$aabb$	$abbb$			
$aabbb$				

The Algorithm

$$S \to AB, A \to BB, A \to a, B \to AB, B \to b$$

a	a	b	b	b
A	A	B	B	B
aa	ab	bb	bb	
	S,B	A	A	
aab	abb	bbb		
S,B	A	S,B		
$aabb$	$abbb$			
A	S,B			
$aabbb$				

The Algorithm

$$S \to AB, A \to BB, A \to a, B \to AB, B \to b$$

a	a	b	b	b
A	A	B	B	B
aa	ab	bb	bb	
	S,B	A	A	
aab	abb	bbb		
S,B	A	S,B		
$aabb$	$abbb$			
A	S,B			
$aabbb$				
S,B				

The Algorithm

- Therefore:

$$aabbb \in L(G)$$

- Time Complexity:

$$|w|^3$$

- Observation:

  The CYK algorithm can be

  easily converted to a parser

Exercises

1. Eliminate all λ-productions from

$$S \to AaB|aaB$$
$$A \to \lambda$$
$$B \to bbA|\lambda$$

2. Remove all unit-productions, all useless productions, and all λ-productions from the grammar

$$S \to aA|aBB$$
$$A \to aaA|\lambda$$
$$B \to bB|bbC$$
$$C \to B$$

Exercises

3. Transform the grammar into Chomsky Normal Form:

$$S \to abAB$$
$$A \to bAB|\lambda$$
$$B \to BAa|A|\lambda$$

4. Convert the grammar into Chomsky Normal Form:

$$S \to aSb|ab$$

5. Convert the grammar into Chomsky Normal Form:

$$S \to aSA|A$$
$$A \to abA|b$$

Exercises

6. Transform the grammar into Chomsky Normal Form:

$$S \to AB|aB$$
$$A \to aab|\lambda$$
$$B \to bbA$$

7. Transform the grammars into CNF:

a) $S \to aSb|bSa|a|b$

b) $S \to aSb|ab$

c) $S \to ab|aS|aaS$

d) $S \to ABb|a,\ A \to aaA|B,\ B \to bAb$

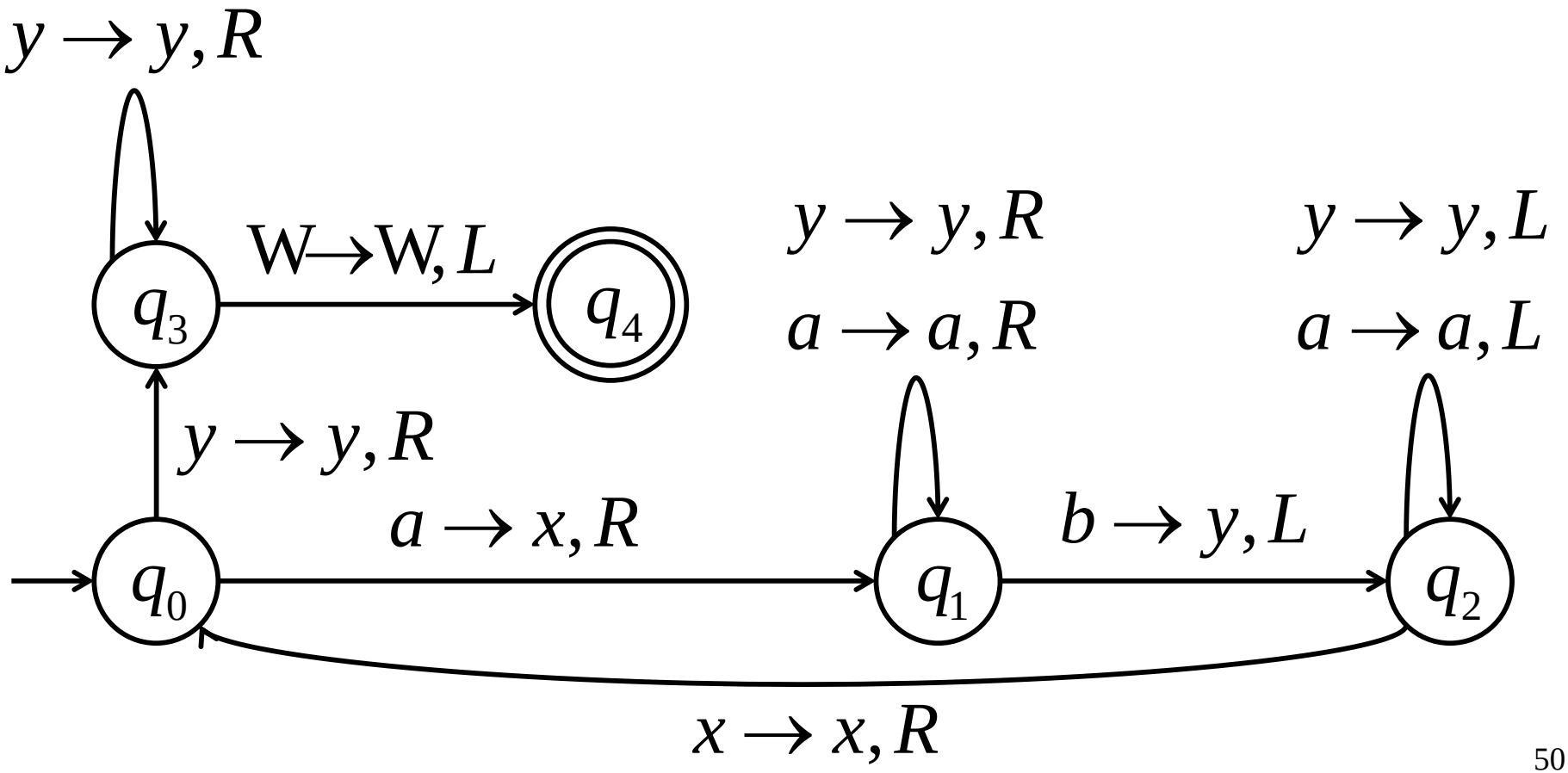
Exercises

8. Use the CYK algorithm to determine whether the strings $aabb$, $aabba$, and $abbbb$ are in the language generated by the grammar

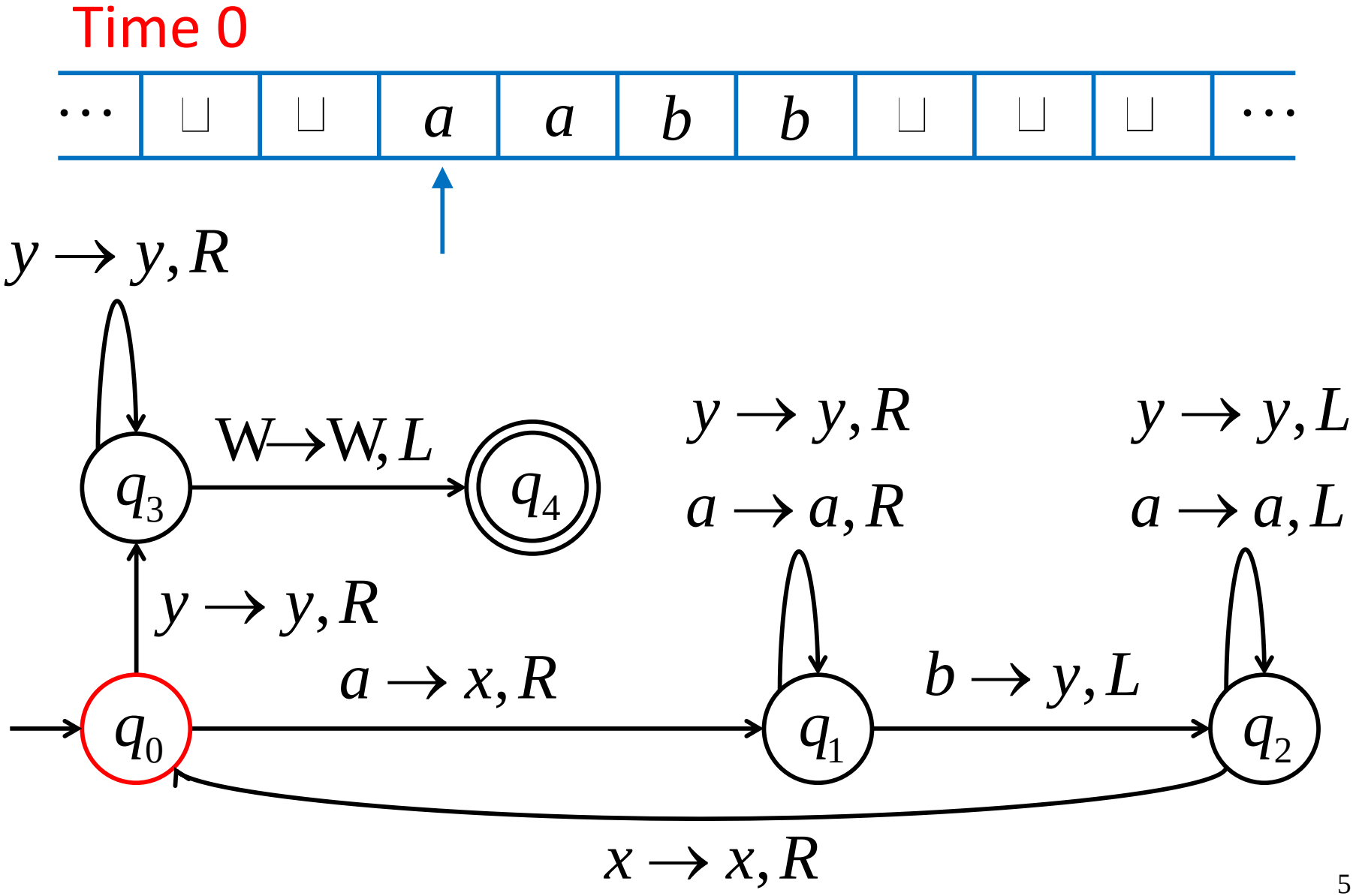
$$S \to AB$$
$$A \to BB|a$$
$$B \to AB|b$$

Another Example: Turing Machine

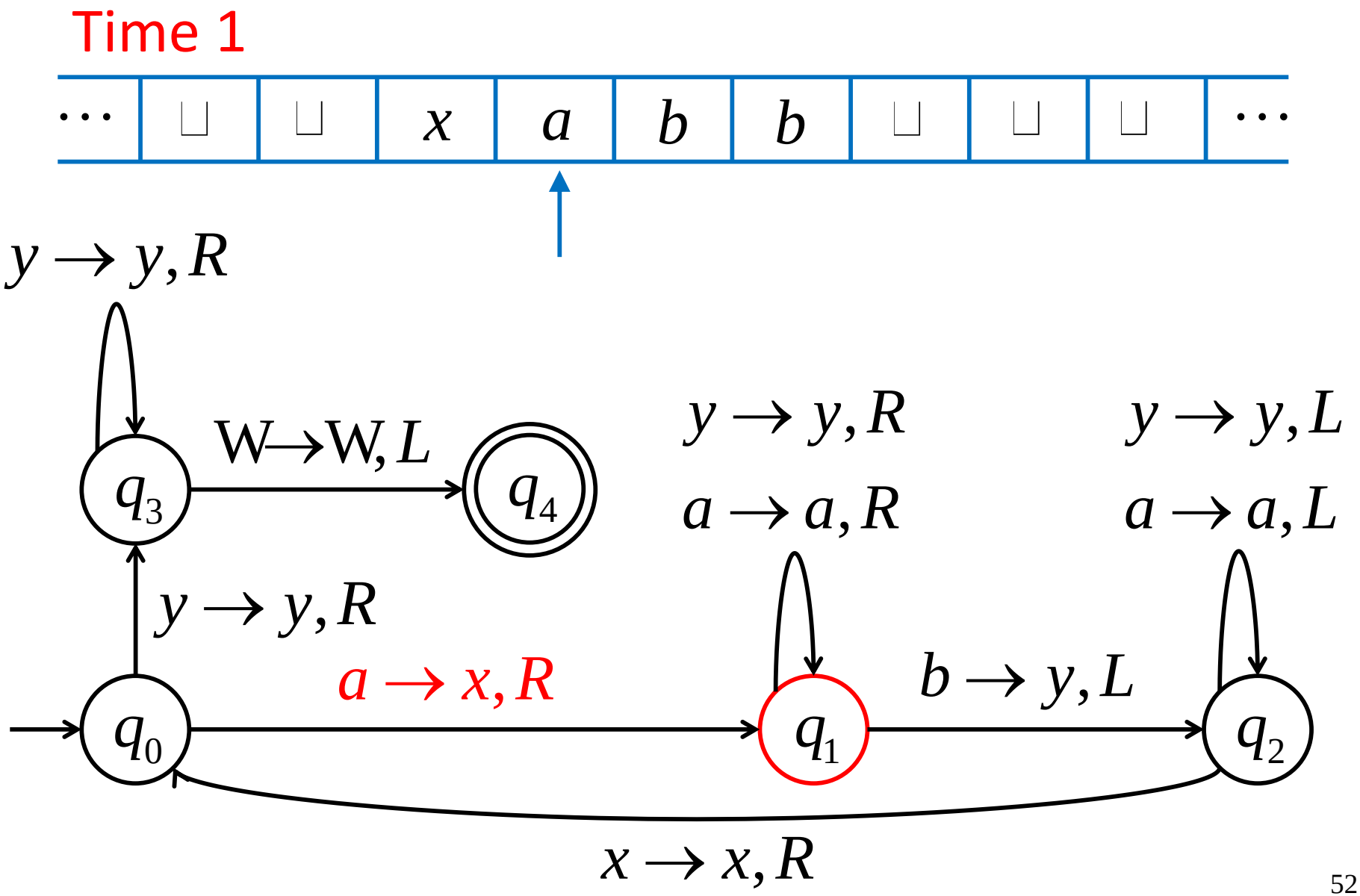
Turing machine for the language $\{a^n b^n\}$



Another Example: Turing Machine

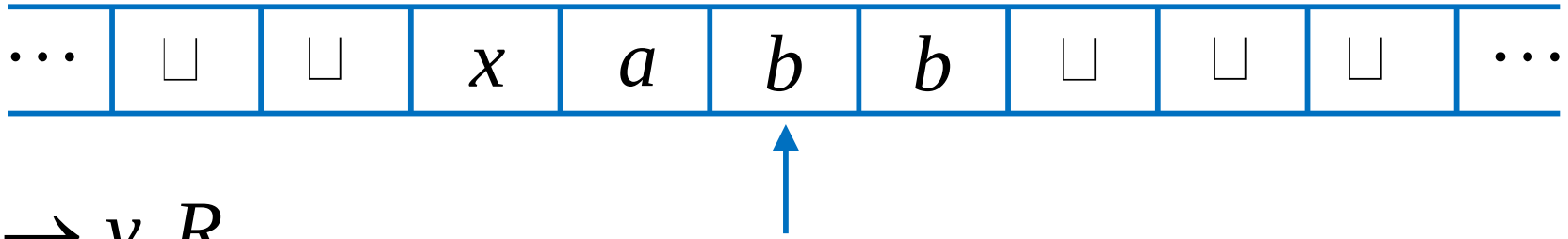


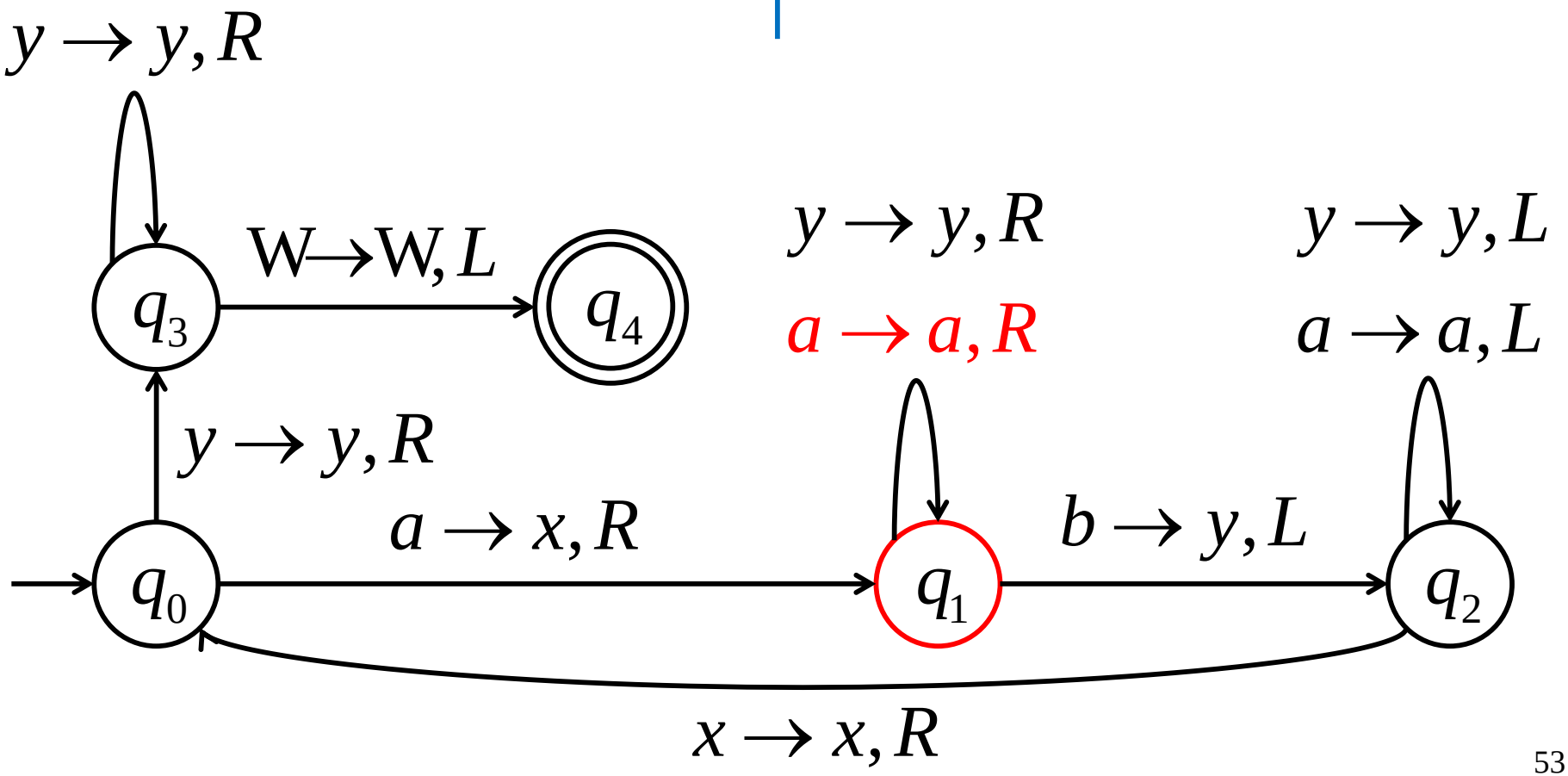
Another Example: Turing Machine



Another Example: Turing Machine

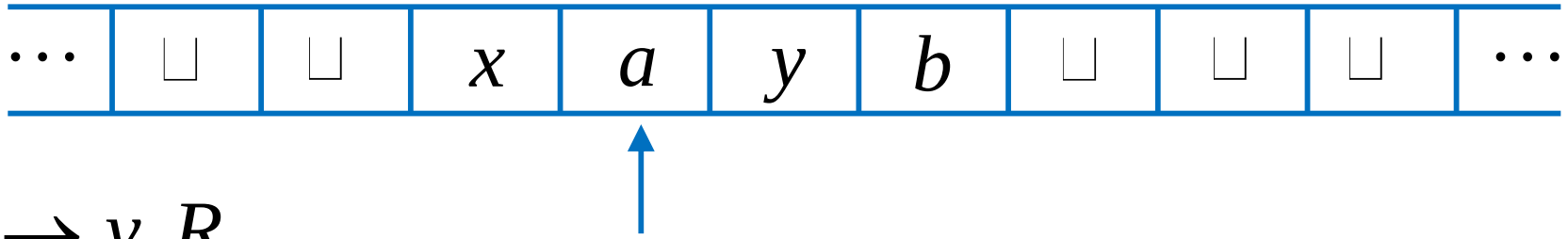
Time 2

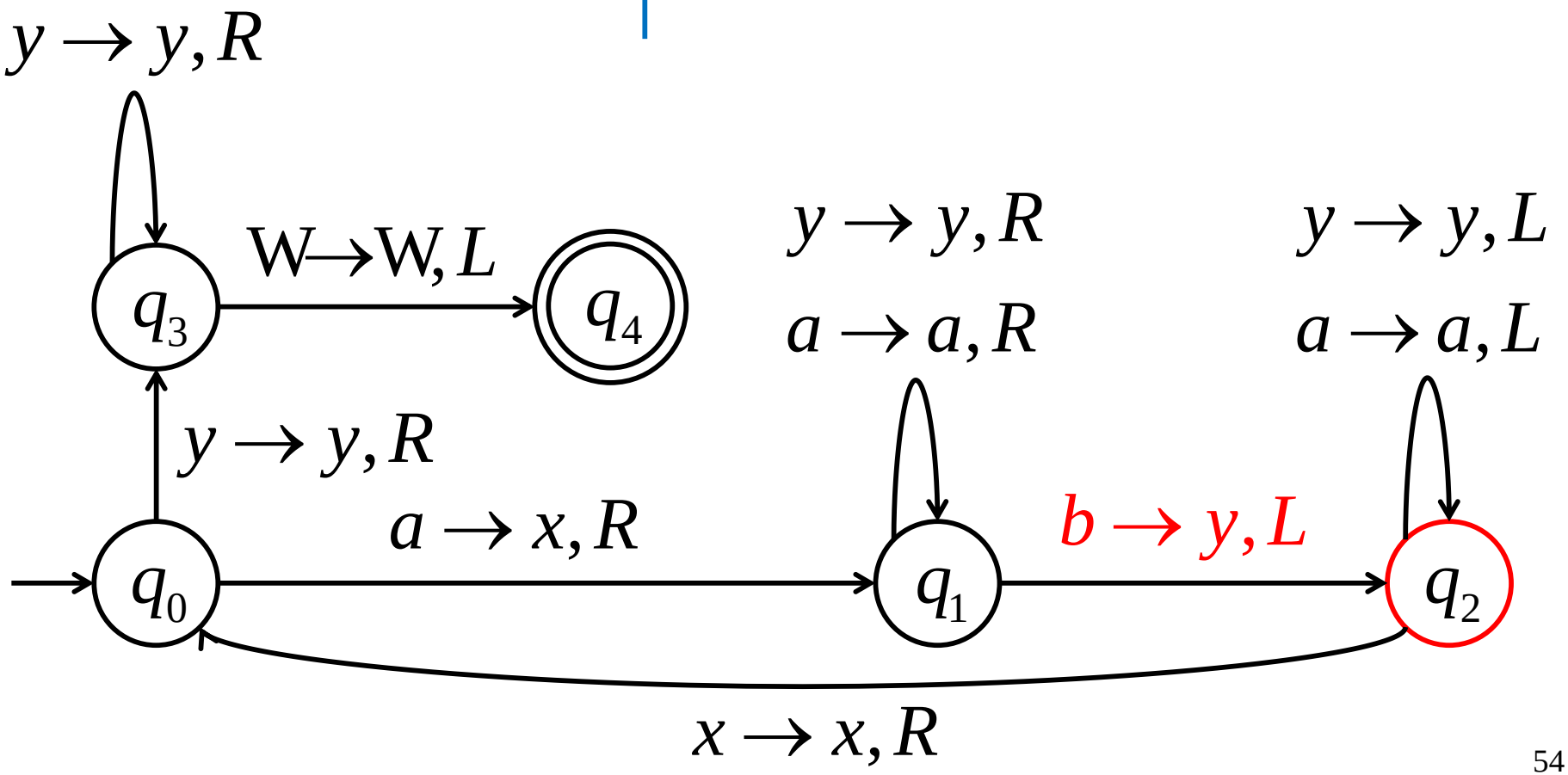




Another Example: Turing Machine

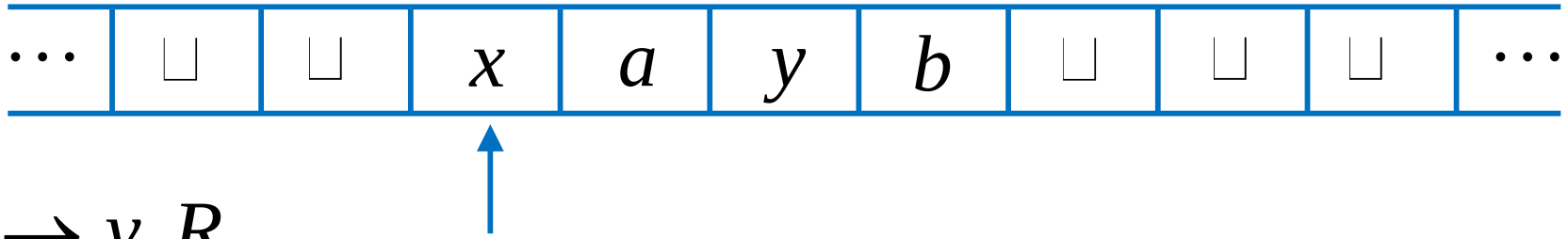
Time 3

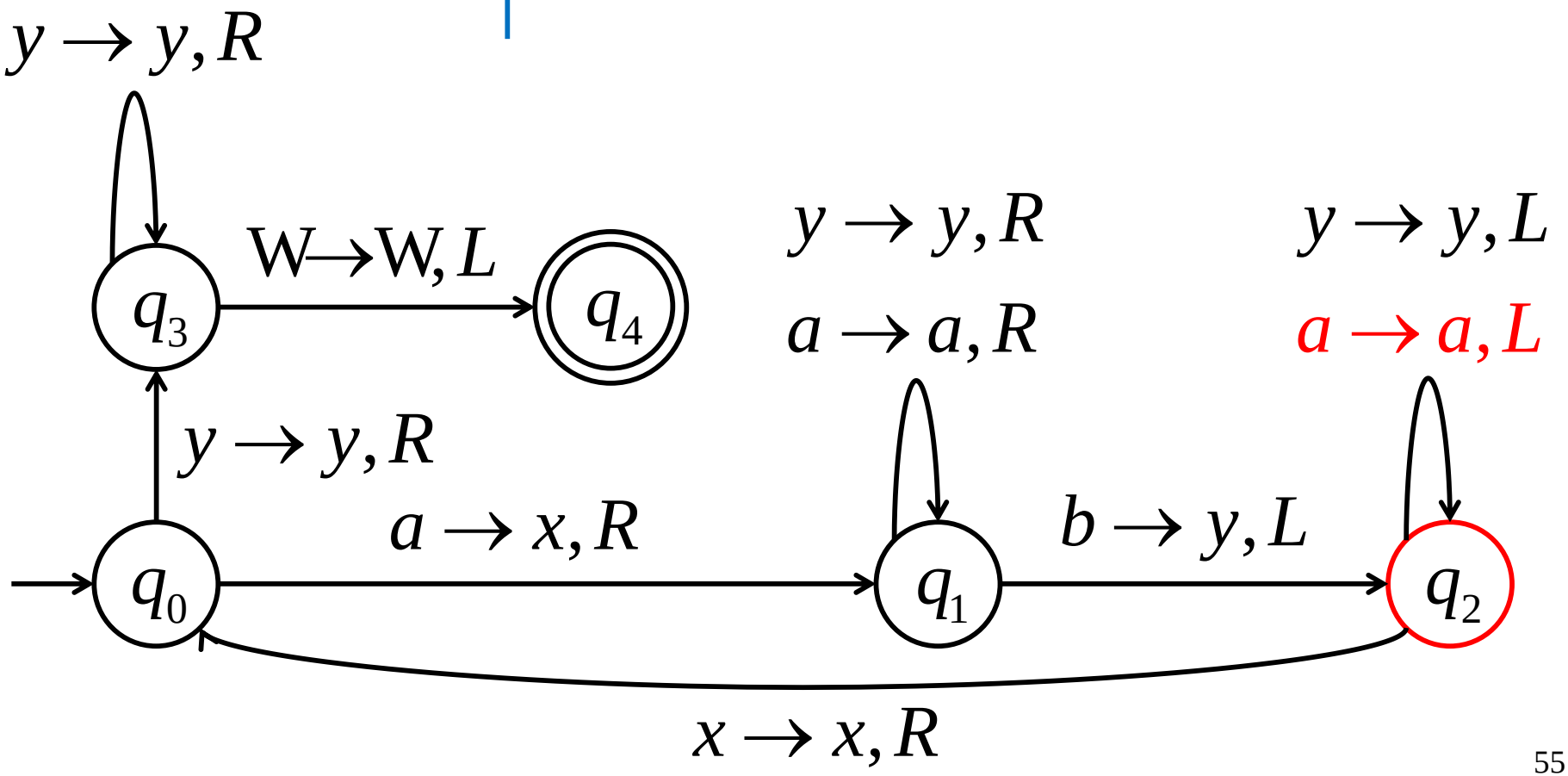




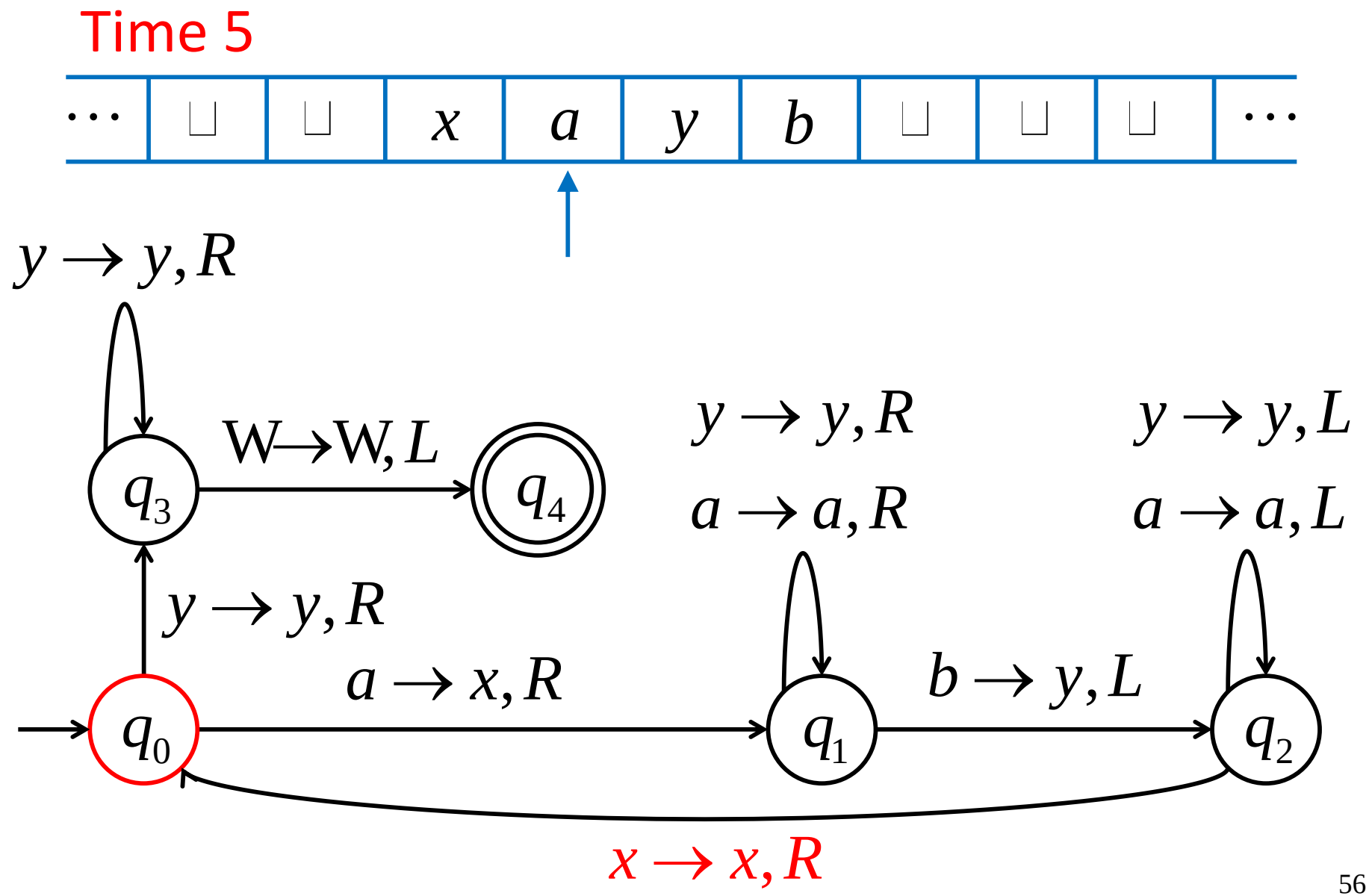
Another Example: Turing Machine

Time 4



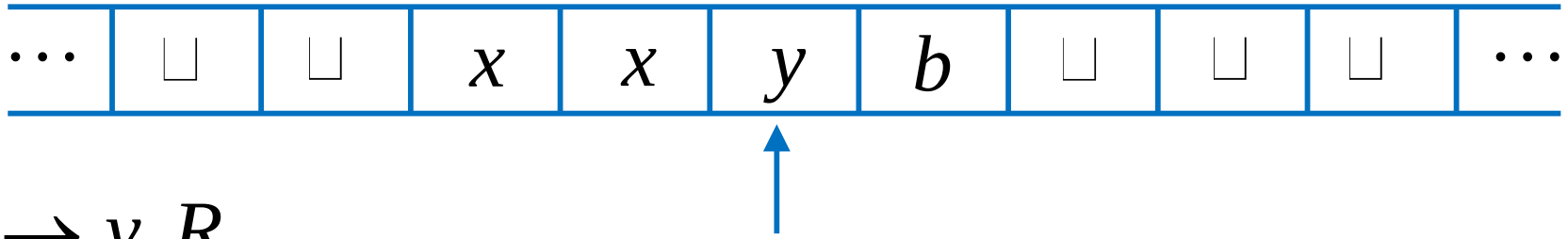


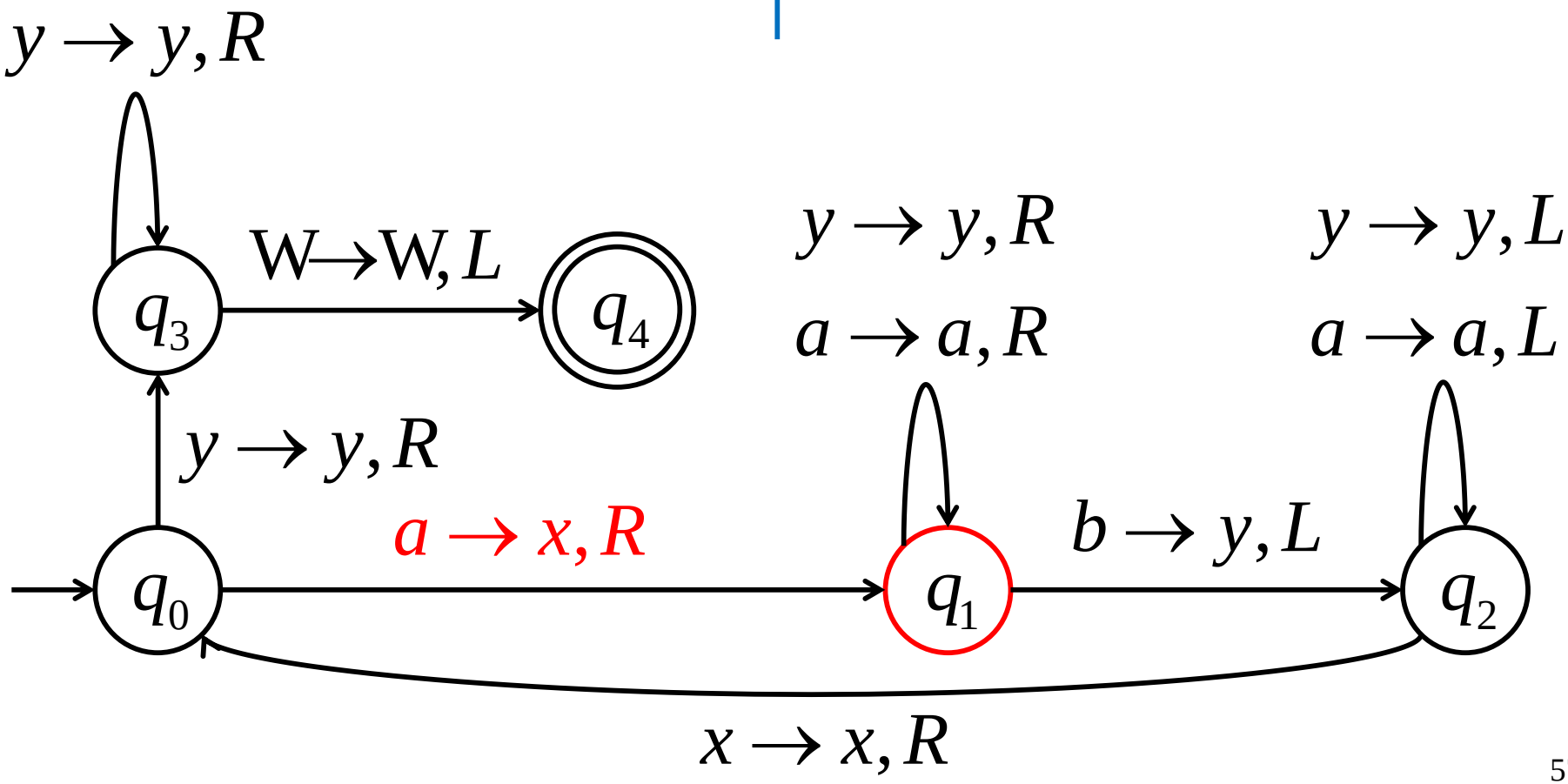
Another Example: Turing Machine



Another Example: Turing Machine

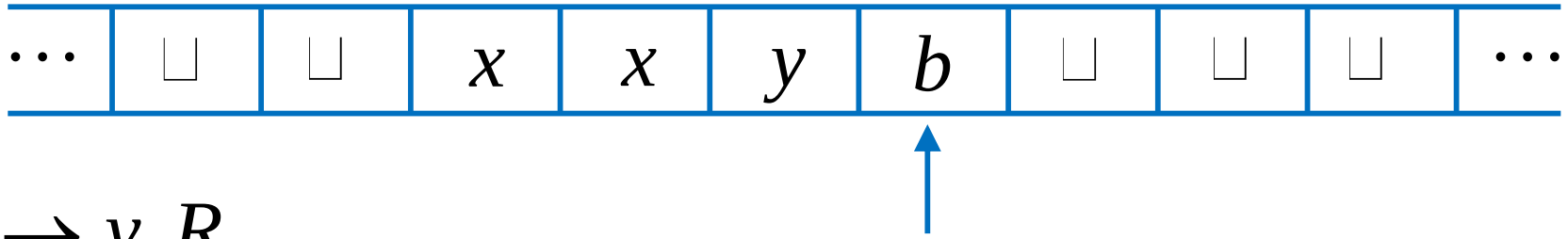
Time 6

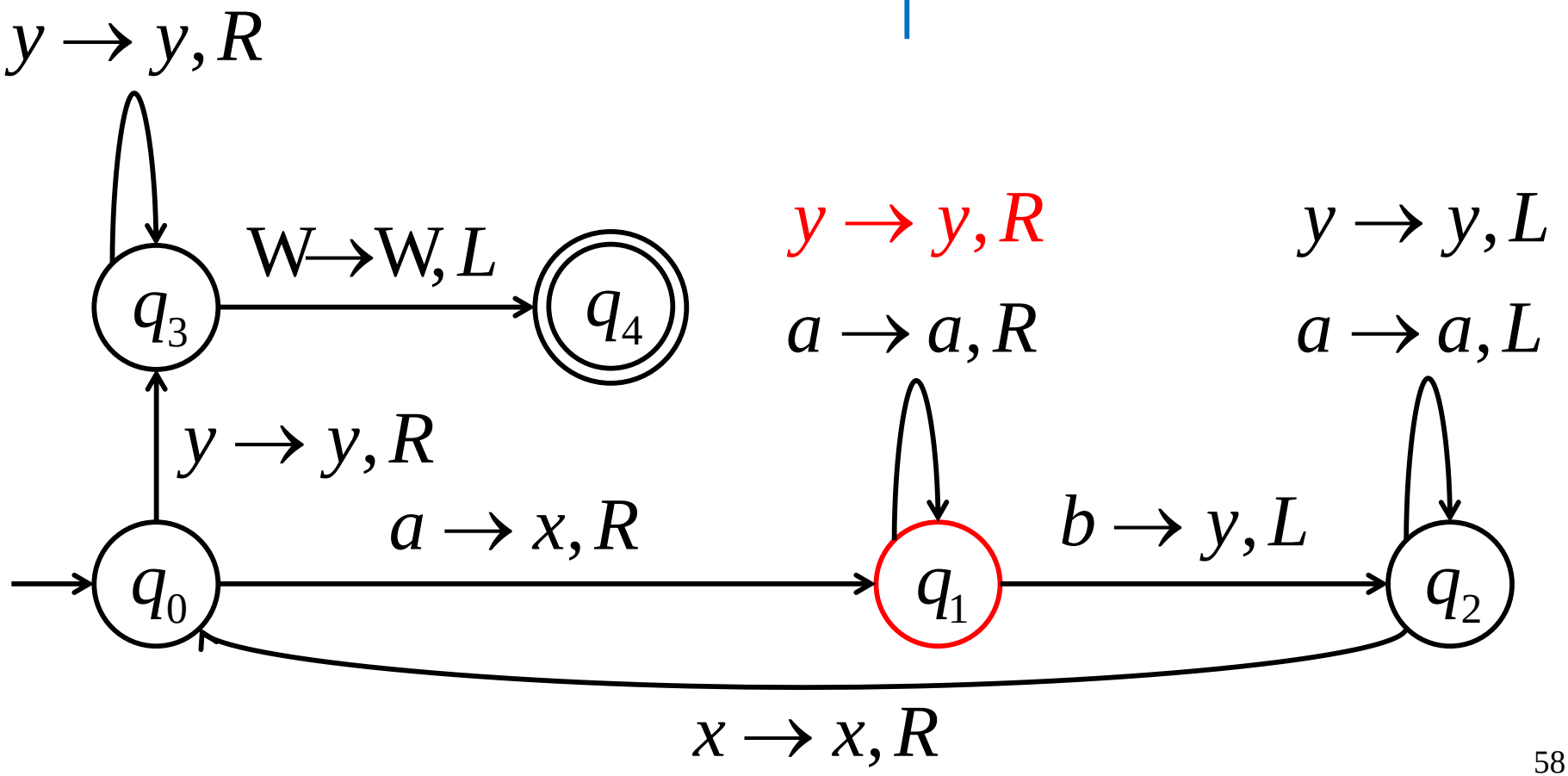




Another Example: Turing Machine

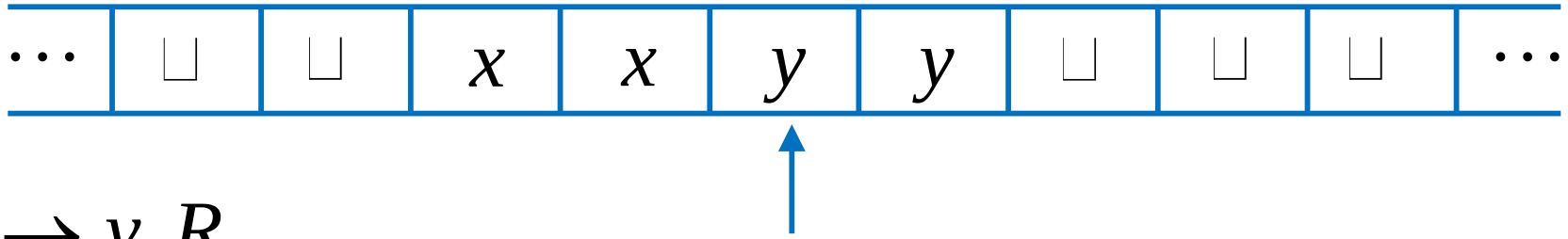
Time 7

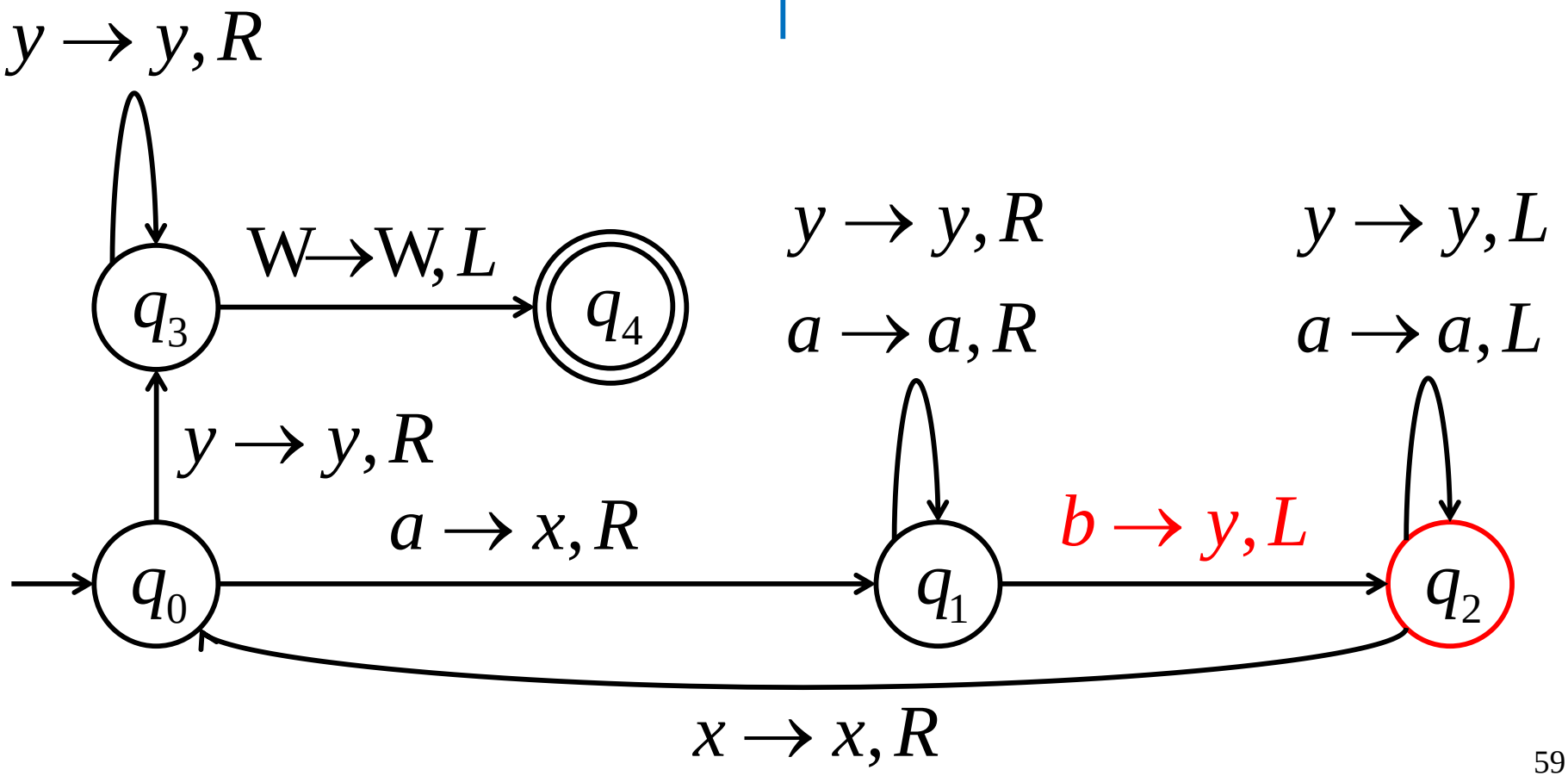




Another Example: Turing Machine

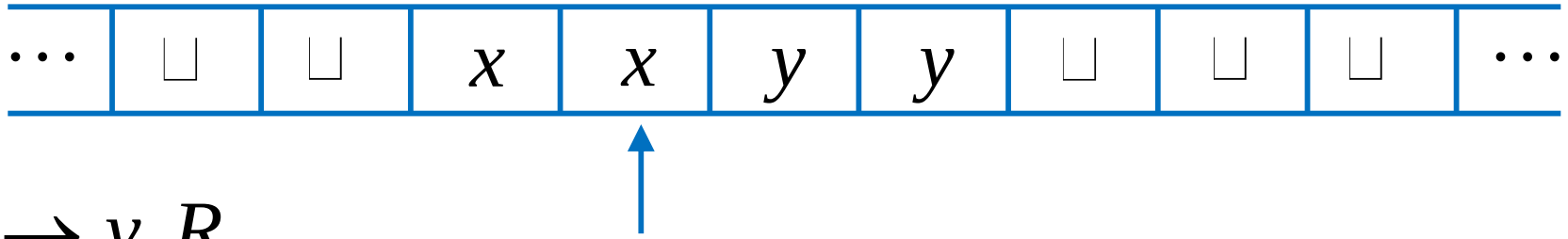
Time 8

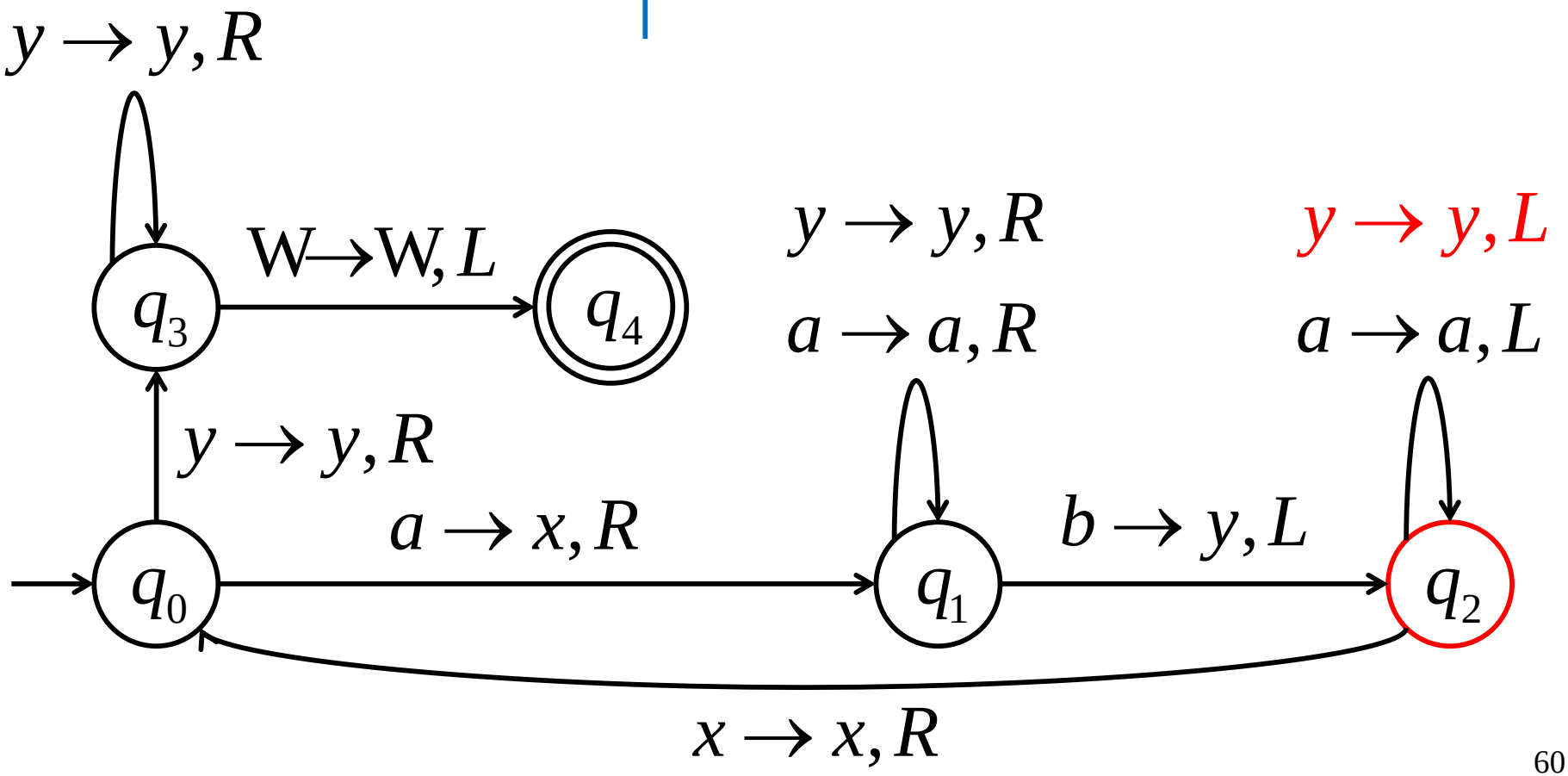




Another Example: Turing Machine

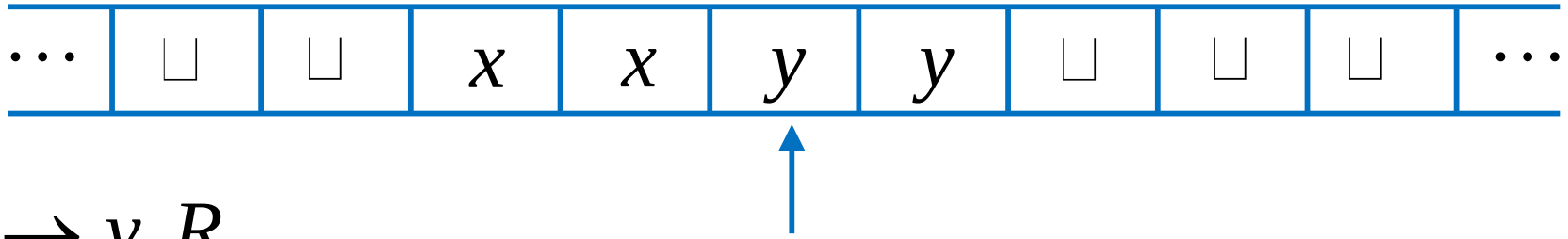
Time 9

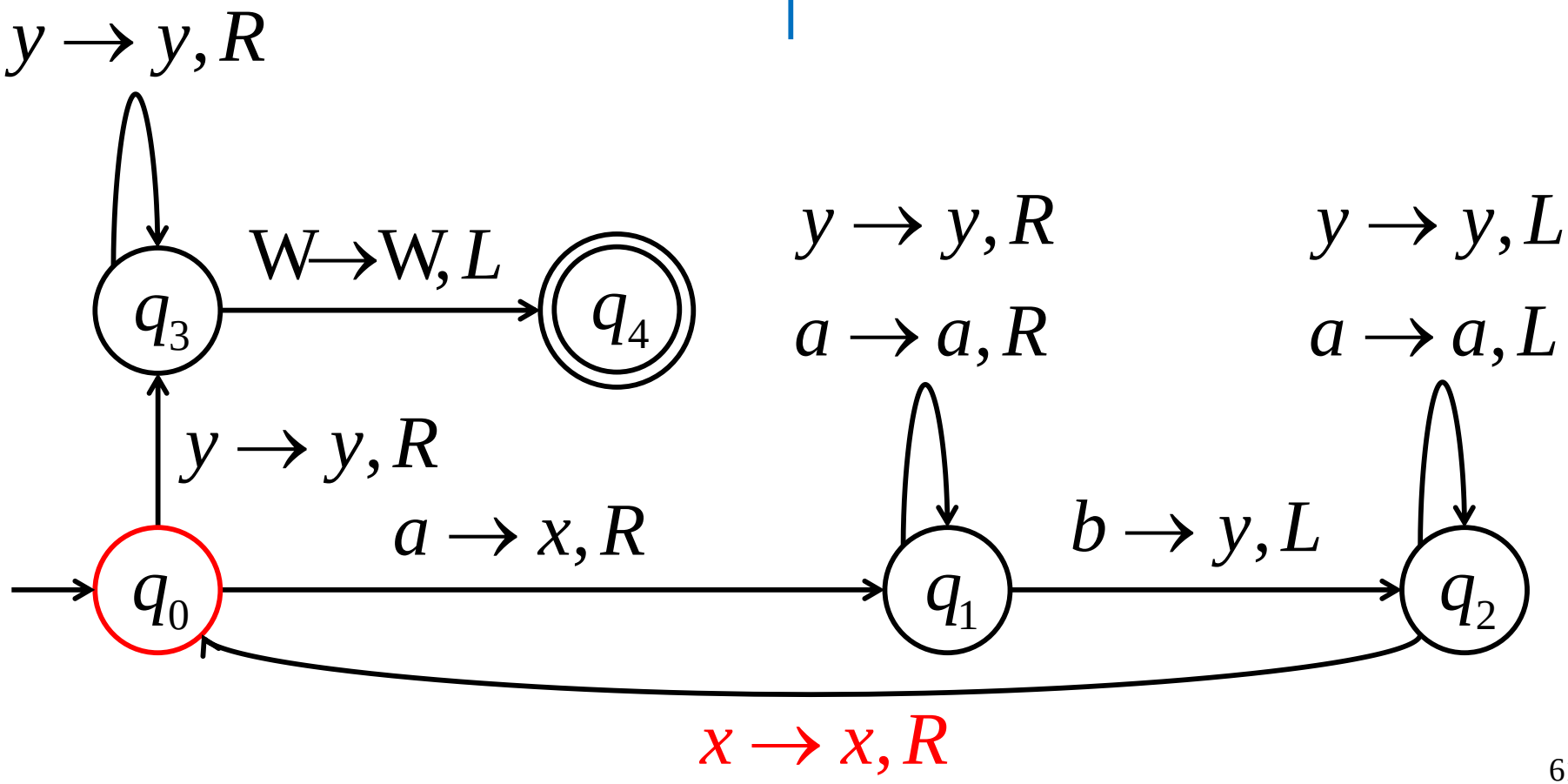




Another Example: Turing Machine

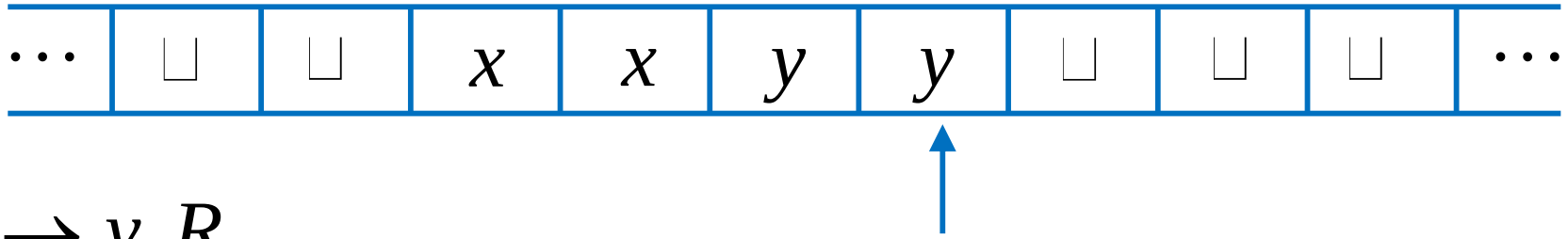
Time 10

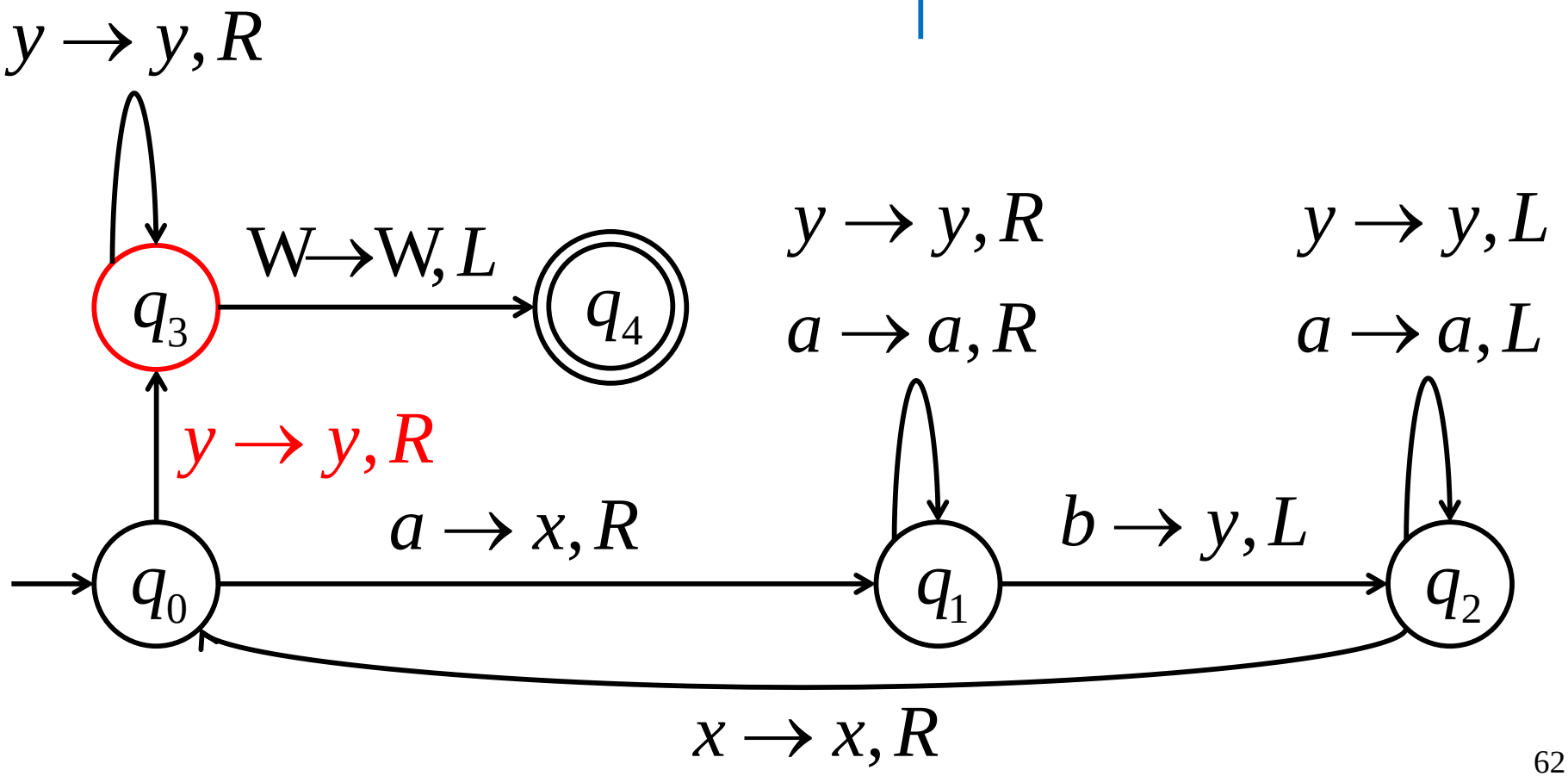




Another Example: Turing Machine

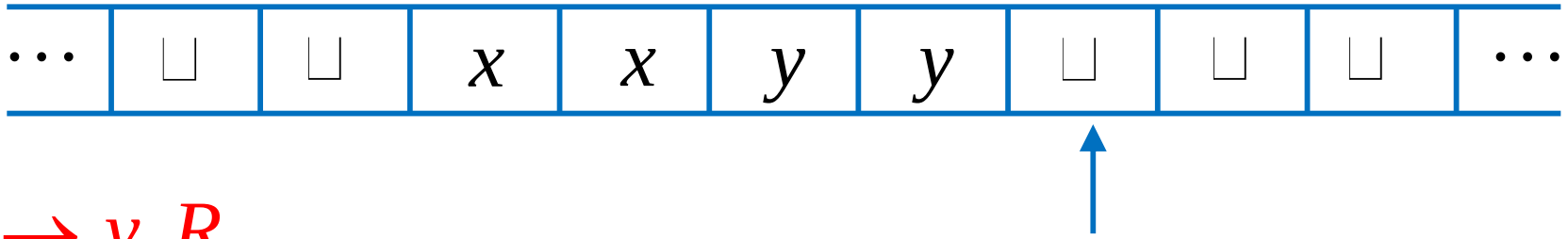
Time 11

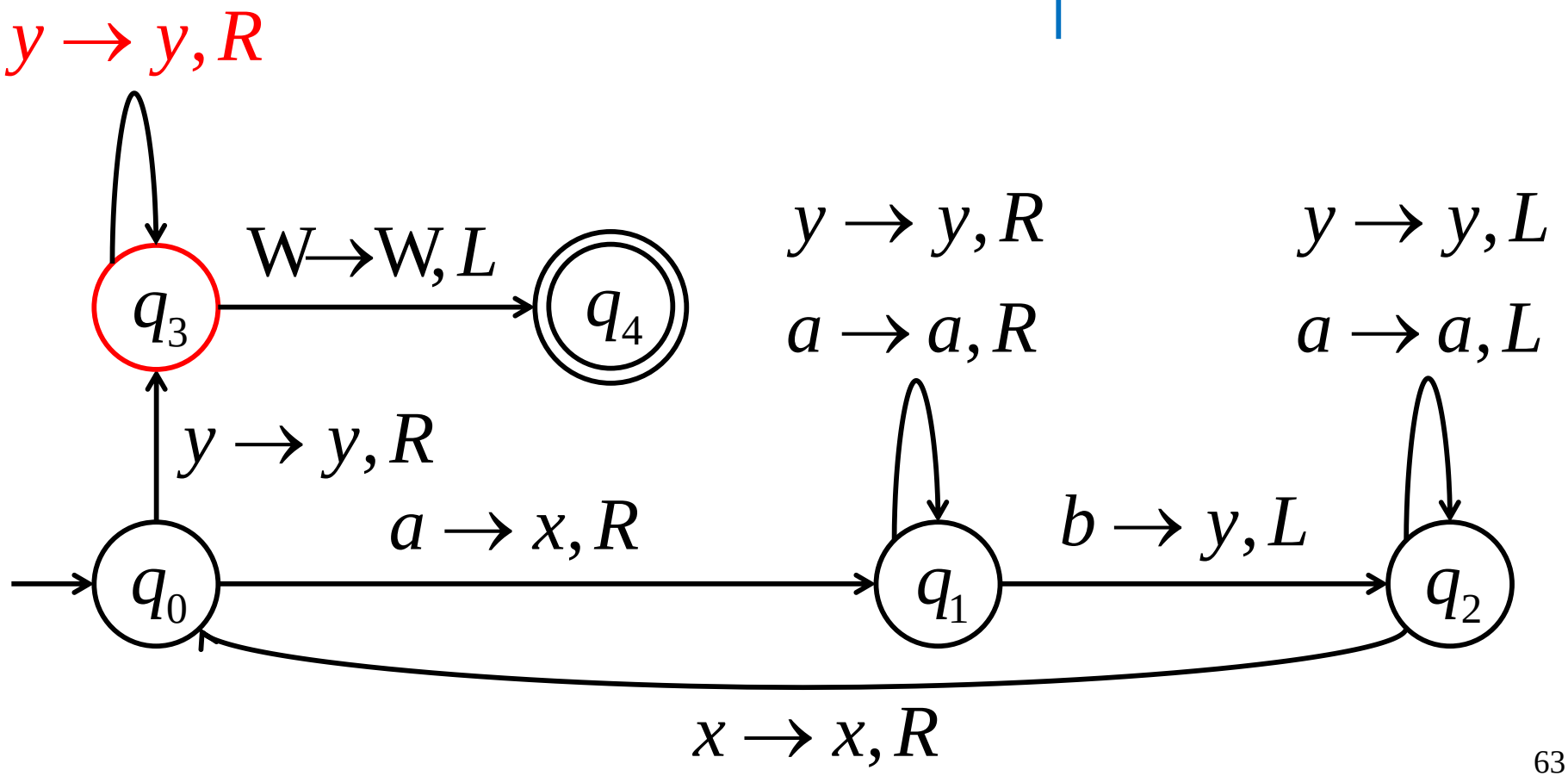




Another Example: Turing Machine

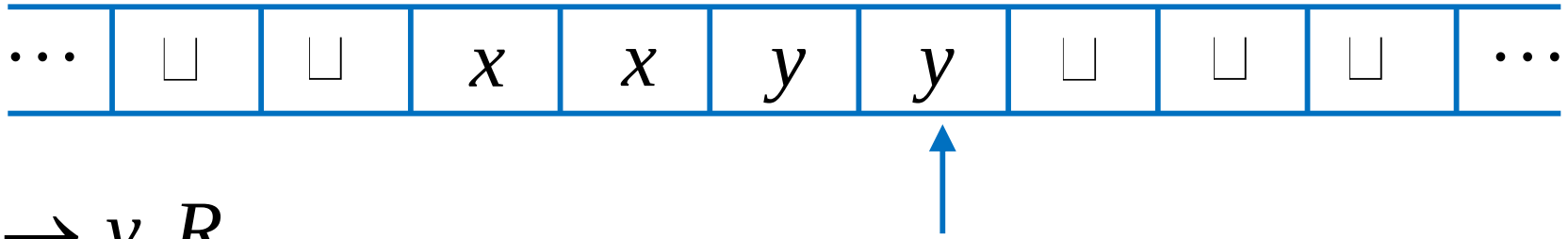
Time 12

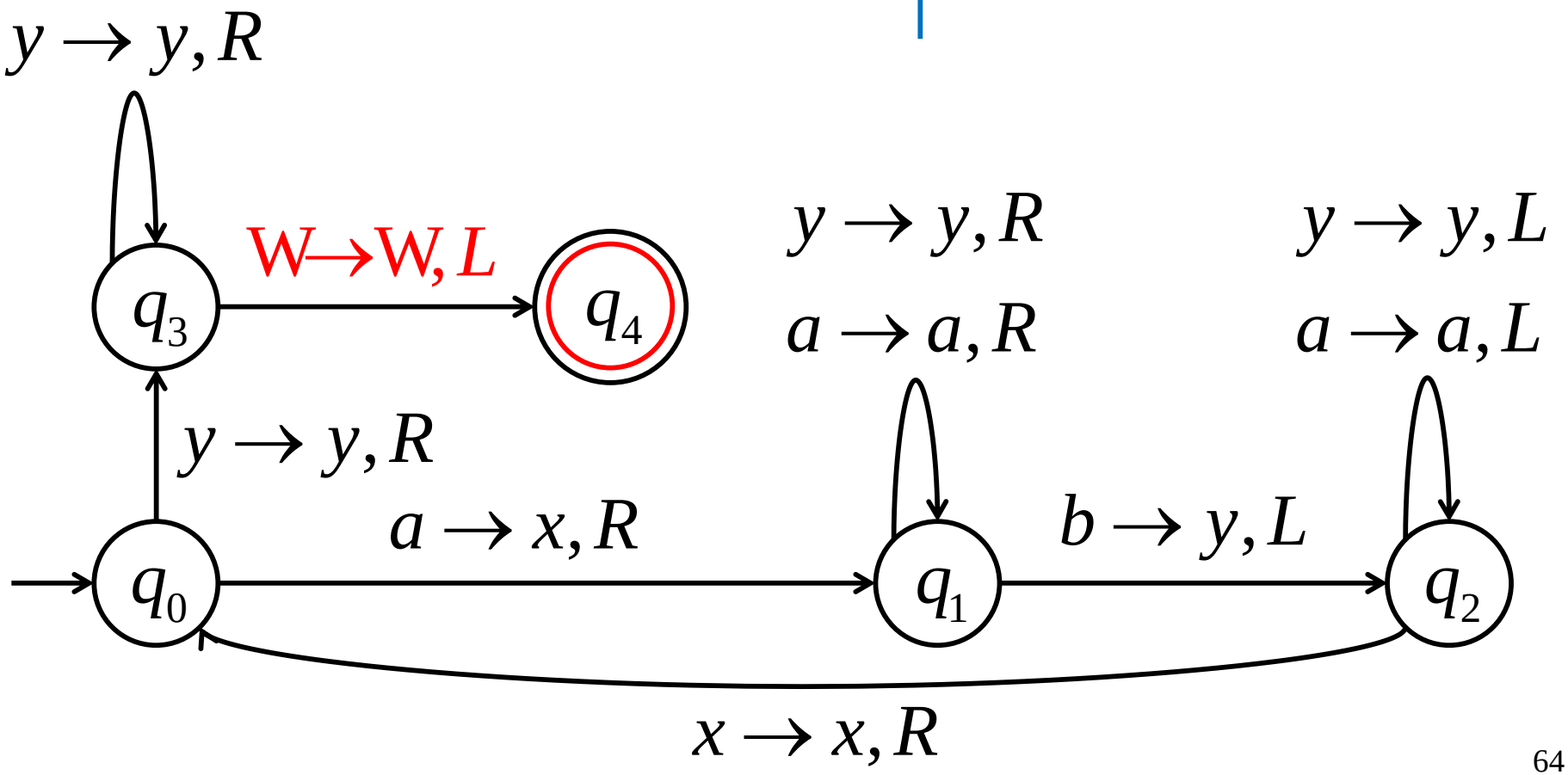




Another Example: Turing Machine

Time 13





Another Example: Turing Machine

Time 14