Computational Learning Theory
Mathematics of String Data and Finite State Automata

Akihiro Yamamoto 山本 章博
http://www.iip.ist.i.kyoto-u.ac.jp/member/akihiro/
akihiro@i.kyoto-u.ac.jp

Machine Leaning from String Data

Alphabets and Stings

- Σ : a finite set of symbols and called an alphabet
- Σ^* : the set of all finite strings (sequences) consisting of the symbols in Σ.
  - An empty string is denoted by ε.
  - $\Sigma^+ = \Sigma^* - \{\varepsilon\}$
  - The size of a string *w*, denoted by $|w|$, is the total number of symbols occurring in *w*.

Examples

- $\Sigma = \{a, b\}$
  $\Sigma^* = \{\varepsilon, a, b, aa, ab, ba, bb, aaa, aab, \ldots\}$
  $|aaa| = |aab| = 3$, $|a| = |b| = 1$, $|\varepsilon| = 0$
- $\Sigma = \{A, T, C, G\}$
  $\Sigma^* = \{\varepsilon, A, T, C, G, AA, \ldots, AG, TA, \ldots, AAA, \ldots\}$

Question

- Assume that we have provided

  $C \subset \Sigma^*$: a finite set of positive examples, and

  $D \subset \Sigma^*$: a finite set of negative examples

  such that $C \cap D = \emptyset$.
- Develop a computer program to find a rule which accepts all positive examples and rejects all negative examples.

Examples

Example 1

$C_1 = \{$ab, aab, abaab, aaab, aaaabbbb,abab$\}$

$D_1 = \{$a, b, bbbb, abba, baaaaba, babb$\}$

- It could hold that *every string in C_1 starts with* a *and end with* b.

Example 2

$C_2 = \{$ba, bababa, babababa, bababababa$\}$

$D_2 = \{$a, b, bbbb, abb, baaaaba, babbb$\}$

- It might hold that *every string in C_2 is made of some repetition of* ba.

Examples

Example 3

$C_3 = \{$aaabbb, ab, aaaabbbb, aaaaabbbbb, aabb$\}$

$D_3 = \{$a, b, bbbb, abb, baaaaba, babbb$\}$

- *Every string in C_3 consists of two strings: The first string consists only of* a*'s, and the second consists of the same number of* b*'s.*

Example 4

$C_4 = \{$aa, abaaba, aaaaaaa, baaab, abab$\}$

$D_4 = \{$a, b, bbbb, abb, bbbbbbba, babbb$\}$

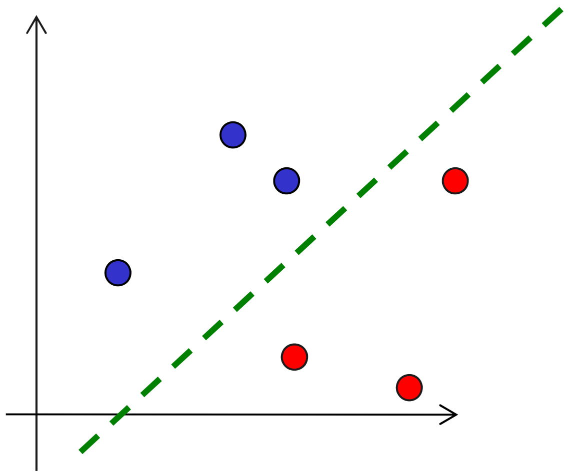
- *In every string in C_4 has more than two* a*'s.*

The First Problem

- What is the grammar and vocabulary with which we represent the rule to distinguish C and D?
  - In the linear classification case, the rule to be found is represented in the form of $(\boldsymbol{w}, \boldsymbol{x}) + c$ s.t.

$$\boldsymbol{x} \in C \Rightarrow (\boldsymbol{w}, \boldsymbol{x}) + c \geq 0$$

$$\boldsymbol{x} \in D \Rightarrow (\boldsymbol{w}, \boldsymbol{x}) + c \leq 0$$



- The region including C is represented with an inequation

$$(\boldsymbol{w}, \boldsymbol{x}) + c \geq 0$$

Solutions to the Problem

- We adopt some representation method with which we represent a subset of Σ^* which includes C.
  - Since the rule found by some learning mechanism is expected to be "general", the set should be sufficiently large.

    Rules should not overfit the examples.
  - A rule which represents a rule is sometimes called a predicate.

Examples

Example 1

C_1 = {ab, aab, abaab, aaab, aaaabbbb,abab}

D_1 = {a, b, bbbb, abba, baaaaba, babb}

- The rule which is output by a learning machine would represent a set

L_1 = {ab, aab, abb, aaab, aabb, abab, abbb,
aaaab, aaabb, ..., abaab, ..., abbbb,
..., aaaabbbb, ...}

Examples

Example 3

$C_2 = \{$aaabbb, ab, aaaabbbb, aaaaabbbbb, aabb$\}$

$D_2 = \{$a, b, bbbb, abb, baaaaba, babbb$\}$

- You may imagine that the rule which is output by a learning machine would represent a set

$L_2 = \{$ab, aabb, aaabbb, aaaabbbb, aaaaabbbbb, aaaaaabbbbbb,...$\}$

Formal Languages

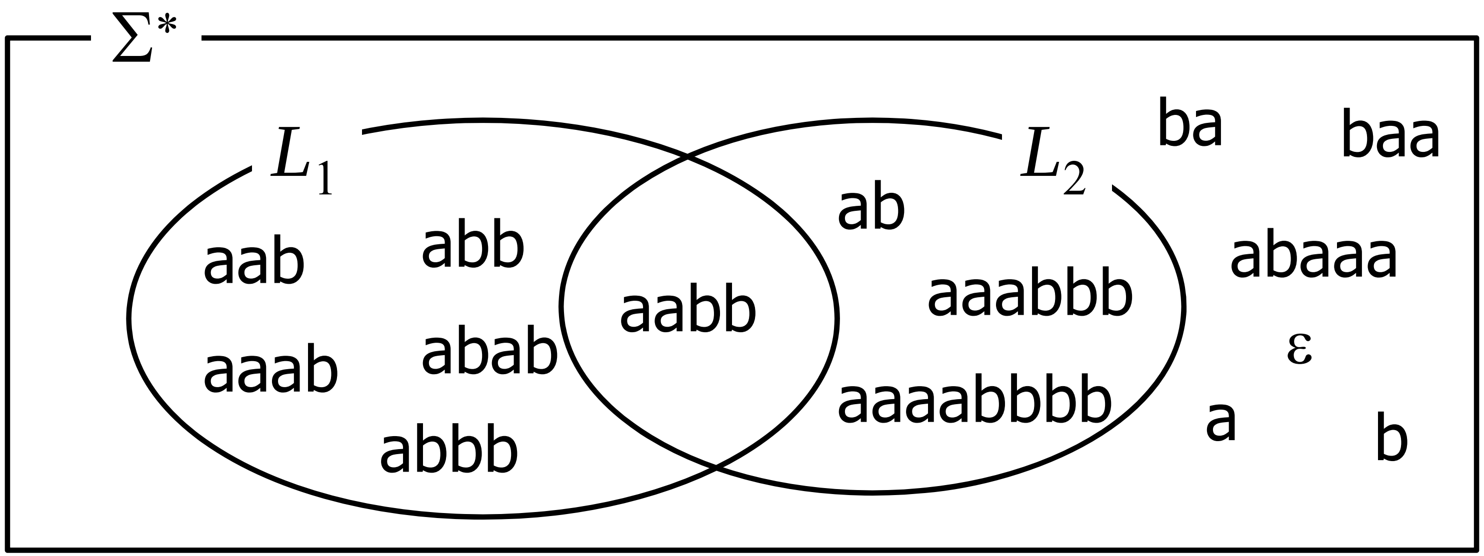
- Every subset of Σ^* is called a formal language.

Example

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

$L_1 = \{$aab, abb, aaab, aabb, abab, abbb,... $\}$

$L_2 = \{$ab, aabb, aaabbb, aaaabbbb,... $\}$



An Instance of the Abstract *Learning*

- A language L which contains C (yes) and excludes D (no) is to be learned
- The language L is represented by some predicate p_θ with some representation f .
  - Let $L(f) = \{x \in \Sigma^* \mid p_f(\mathbf{x})\}$ for a predicate p_f defined with f .

  Then the search space (version space) is

  $$\mathsf{L} = \{L(f) \mid f \in \mathsf{H}\}.$$

  where H is the set of representations f.
- The training examples are provided as the sets C and D.
- A learning algorithm is provided.

An Instance of the Abstract *Learning*

- As an instance of the formulation

  $$\mathrm{argmin}_{f \in \mathrm{H}} \left(\Sigma_{\boldsymbol{x} \in Data} \mathrm{Loss}(f, \boldsymbol{x}) + \lambda \mathrm{P}(f)\right)$$

learning languages can be formalized with letting

H : the set of all representation,

f : an representation,

Data : a finite set of pairs $\boldsymbol{x} = \langle w, s \rangle$ of a string with a sign such that $s = +$ if $w \in C$ and $s = -$ if $w \in D$,

$$\mathrm{Loss}(f, \boldsymbol{x}) = \begin{cases} 0 \ \text{ if } \quad \boldsymbol{x} = \langle w, + \rangle \text{ and } w \in L(f) \\ \qquad \text{or } \boldsymbol{x} = \langle w, - \rangle \text{ and } w \notin L(f), \\ 1, \text{ otherwise.} \end{cases}$$

Three Types of Representation

- We introduce some predicates which distinguishes strings in a formal language, and exclude strings outside of the formal language.

$$x \in C \Rightarrow x \in L(\theta)$$

$$x \in D \Rightarrow x \notin L(\theta)$$

- The predicates are categorized into three types:
  - Algorithms for recognizing formal languages
  - Monomials just like in algebra
  - Grammar for defining formal languages
- Analogy to the cases of numbers would be helpful to understand the first and the second.

Polynomials as Predicates

- Let us consider sets of natural numbers.
  - Natural numbers are 0, 1, 2,... and **N** denotes the set of all natural numbers.
- For example, let us consider the sets

$$F = \{\, x \in \mathbf{N} : x = 2y \text{ for some } y \in \mathbf{N} \,\}$$

$$P = \{\, x \in \mathbf{N} : x = 2y + 1 \text{ for some } y \in \mathbf{N} \,\}$$

- Then

$$F = \{2 \times 0, 2 \times 1, 2 \times 2, 2 \times 3, \ldots\} = \{0, 2, 4, 6, \ldots\}$$

$$P = \{2 \times 0+1, 2 \times 1+1, 2 \times 2+1, 2 \times 3+1, \ldots\} = \{1, 3, 5, 7, \ldots\}$$

Polynomials as Predicates

- Let us consider sets of natural numbers.
  - Natural numbers are 0, 1, 2,... and **N** denotes the set of all natural numbers.
- For example, let us consider the sets

$$E = \{ x \in \mathbf{N} : x \textit{ is an even number} \}$$

$$O = \{ x \in \mathbf{N} : x \textit{ is an odd number} \}$$

What is "*even*"? What is "*odd*"?

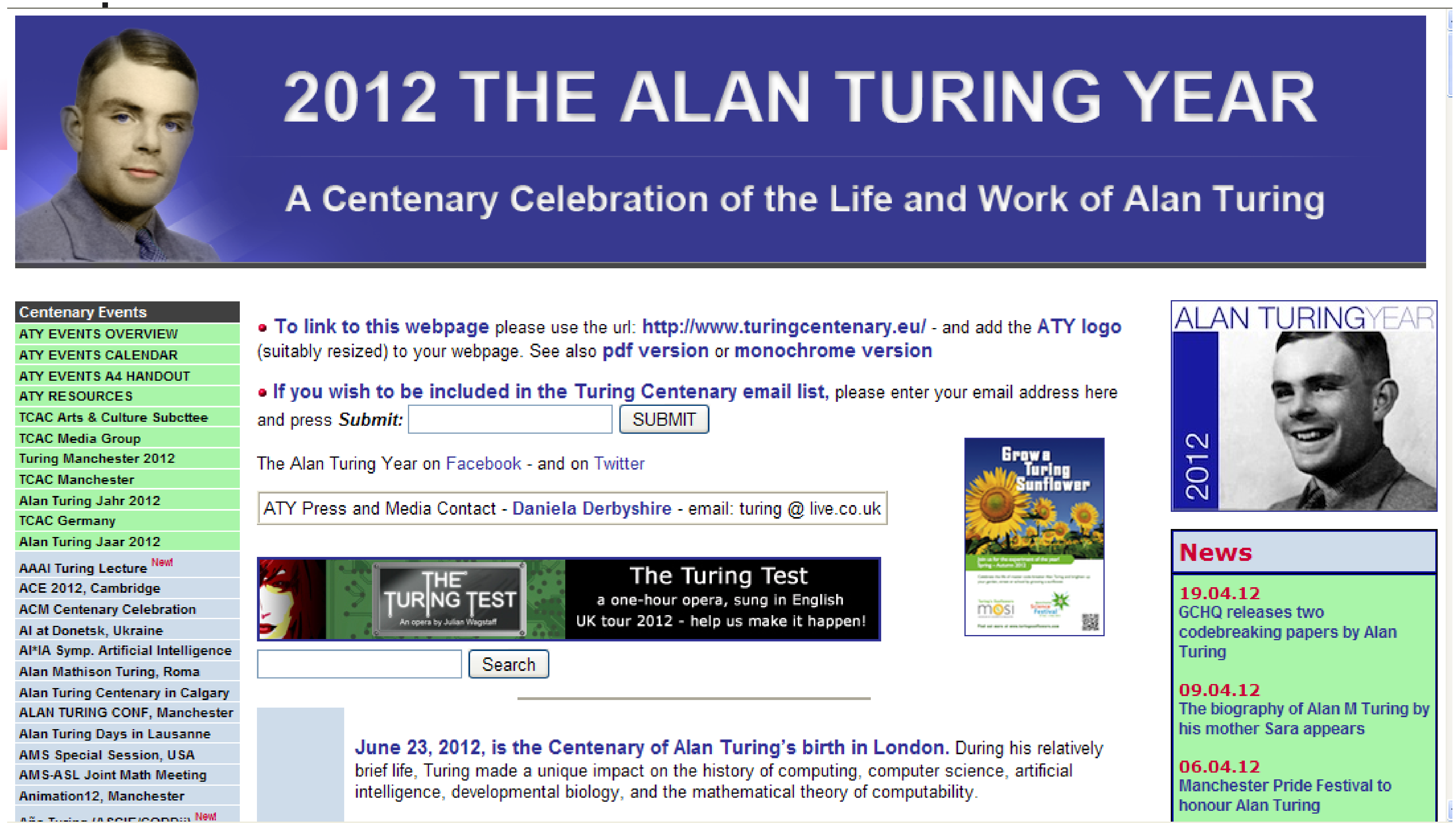
- To see whether a number x is an even number or odd, apply the following algorithm:

  Step 1 : Divide x with 2 and see the remainder y.

  Step 2 : If y is equal to 0, then answer " x is an even number"

  else answer " x is an even number"

Algorithms and Automata



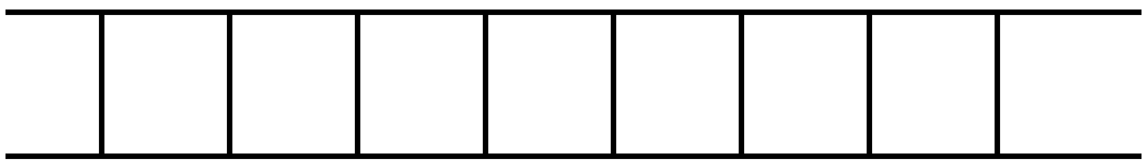
- A. Turing: On computable numbers, with an application to the Entscheidungsproblem, 1936.

Observation by Turing (1)

- Computing is normally done by writing certain symbols on paper. We may suppose this paper is divided into squares like a child's arithmetic book.

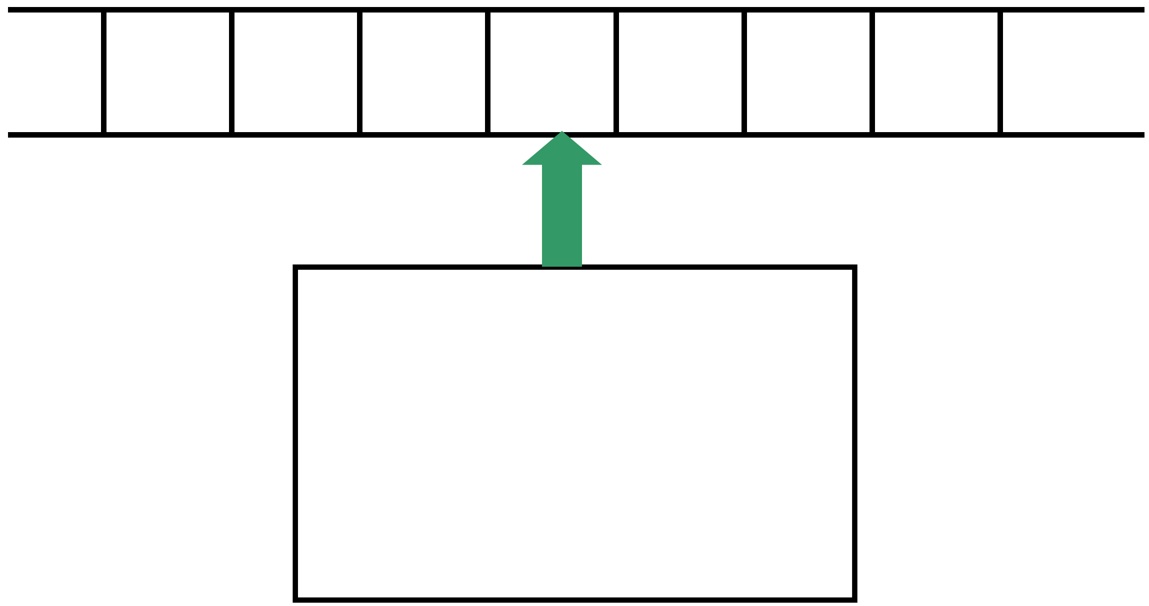
  ...

- I assume then that the computation is carried out on one-dimensional paper, i.e. on a tape divided into squares.



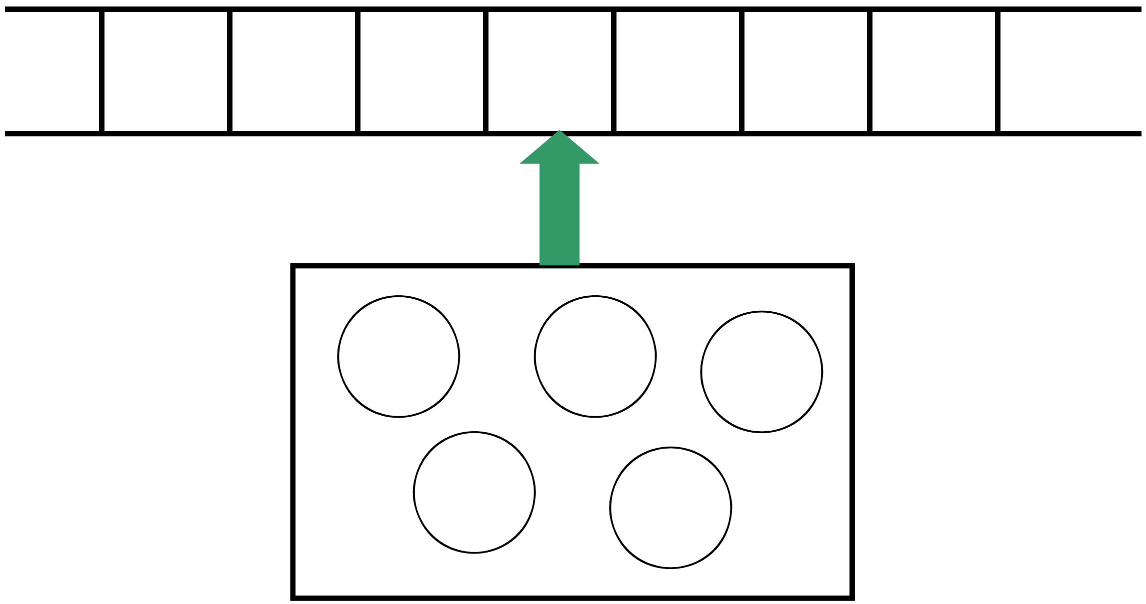
Observation by Turing(2)

- The behaviour of the computer at any moment is determined by the symbols which he is observing and his "state of mind" at that moment.



Observation by Turing(3)

- We will also suppose that the number of states of mind which need be taken into account is finite.



Observation by Turing(4)

(a) Changes of the symbol on one of the observed squares.

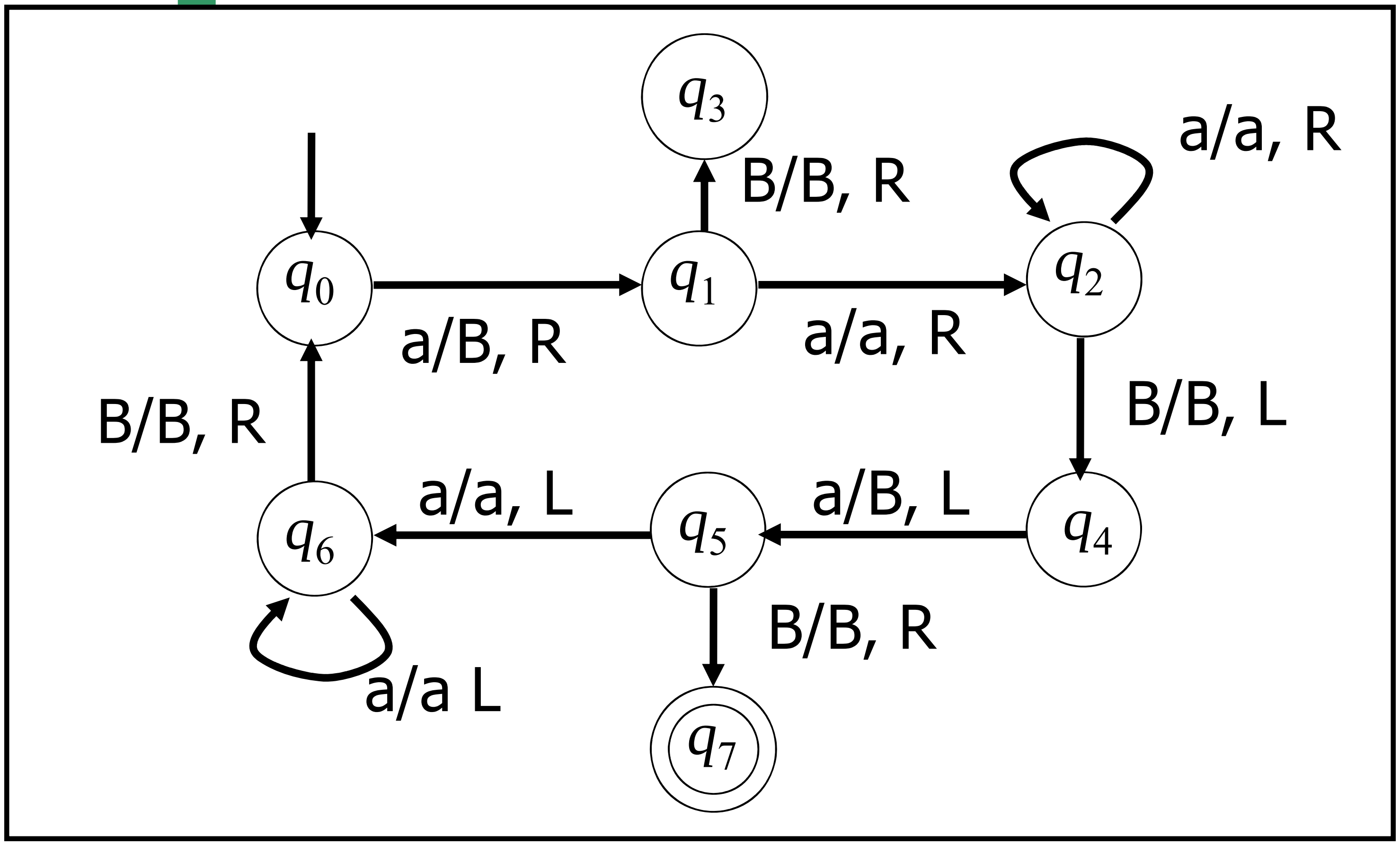
(b) Changes of one of the squares observed to another square within L squares of one of the previously observed squares.

A. A possible change (a) of symbol together with a possible change of state of mind.

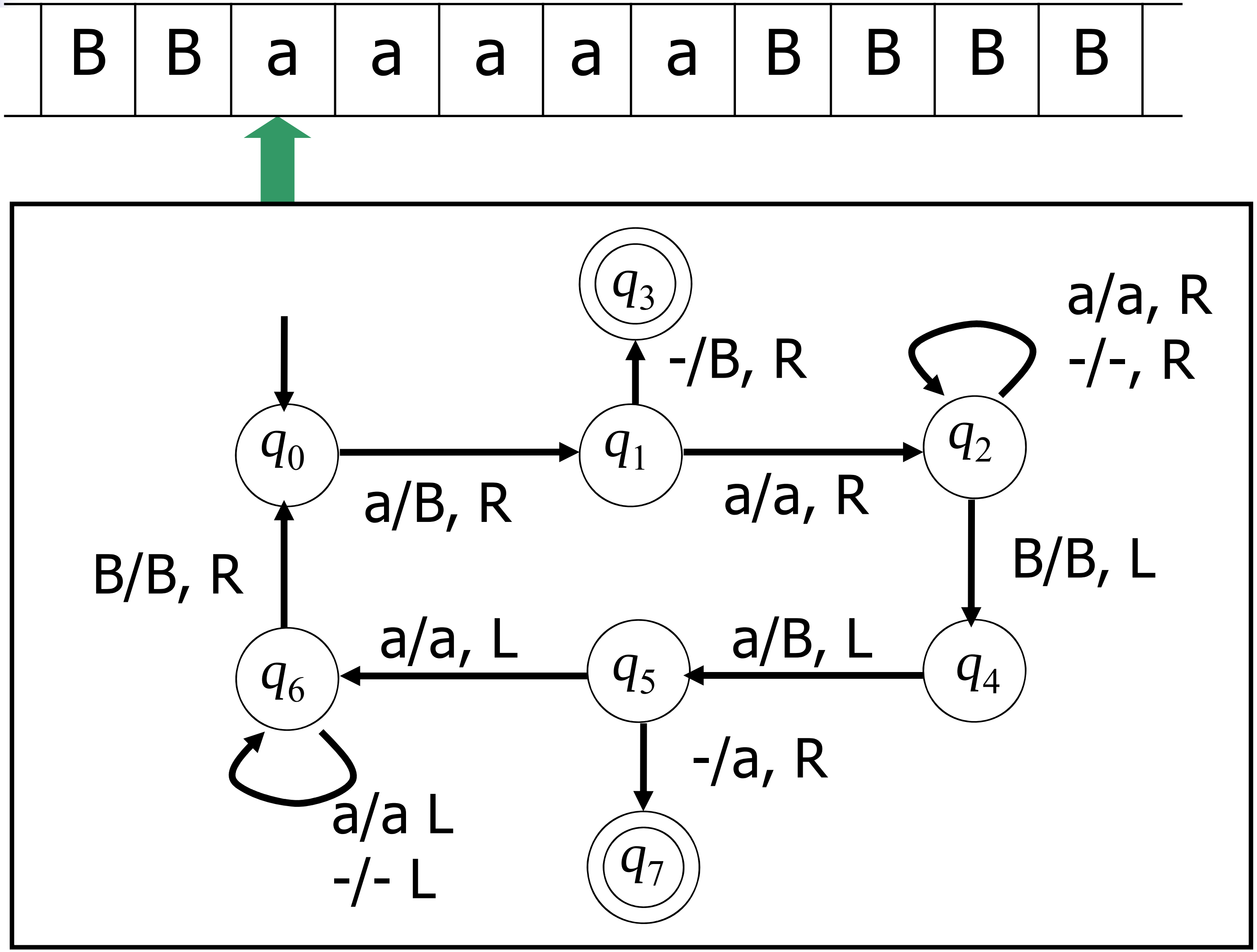
B. A possible change (b) of observed squares, together with a possible change of state of mind.

Dividing x by 2 (1-1)

B	a	a	a	a	a	a	B	B	B	B

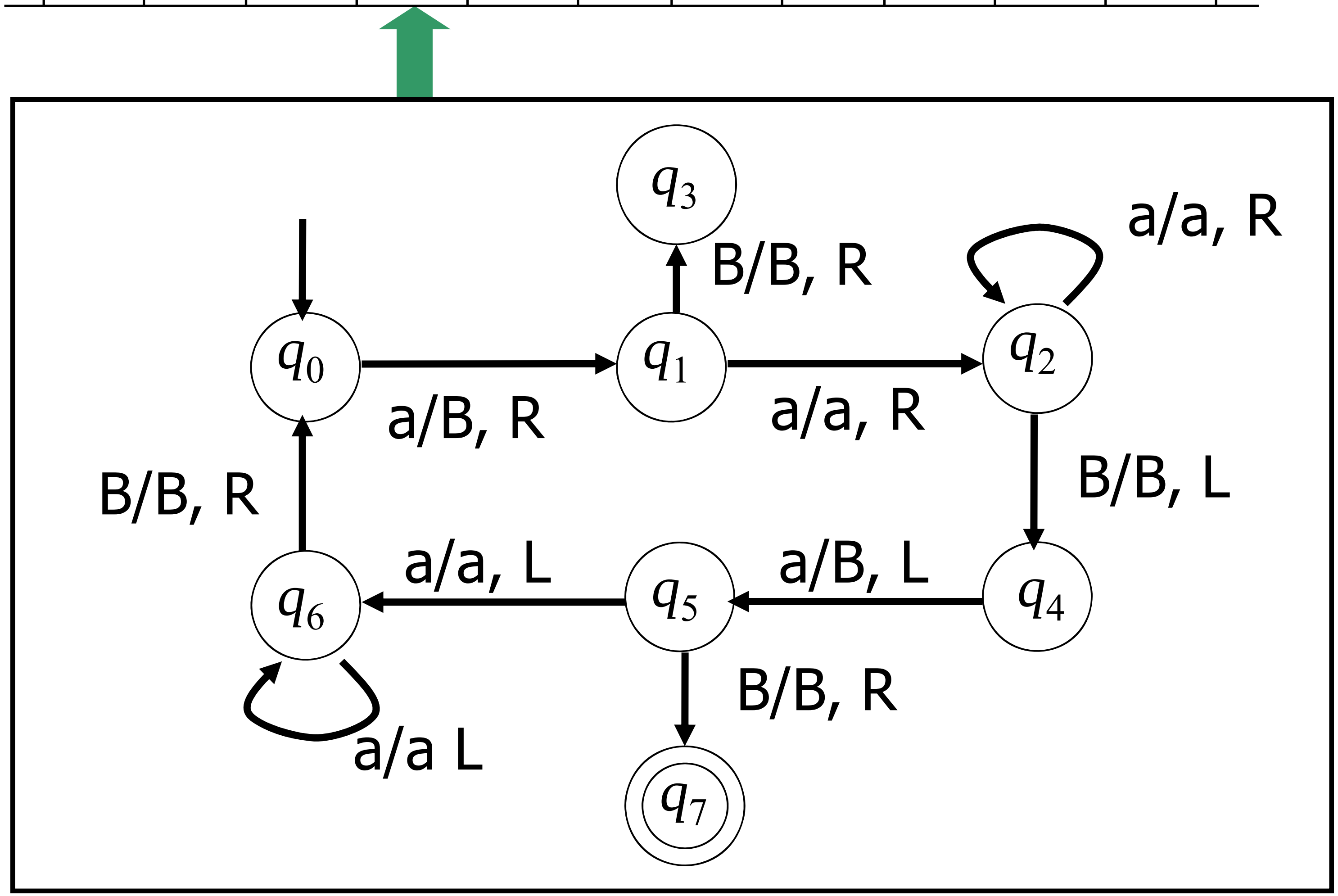


Dividing x by 2 (1-2)



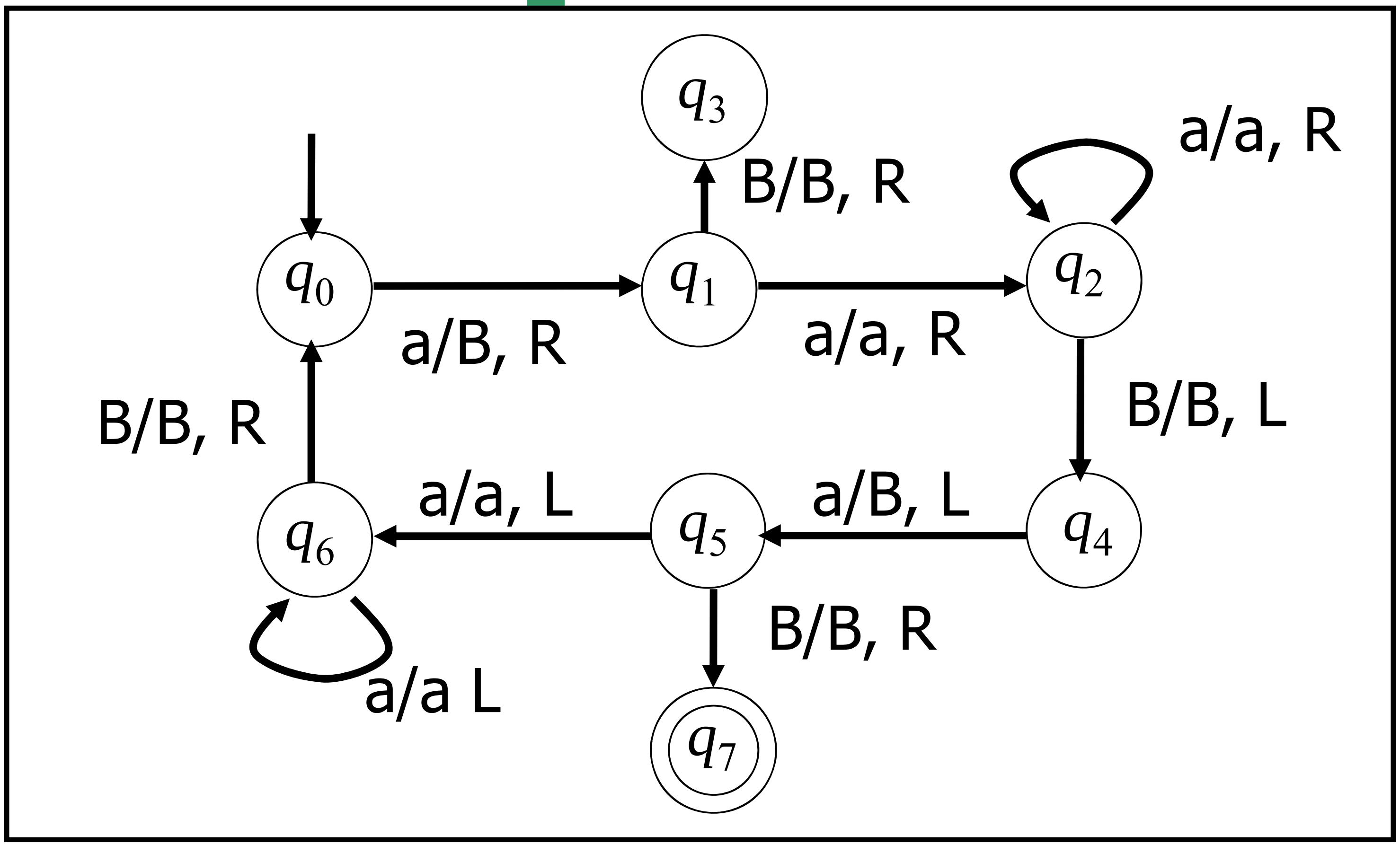
Dividing x by 2 (1-3)

B	B	a	a	a	a	a	B	B	B	B



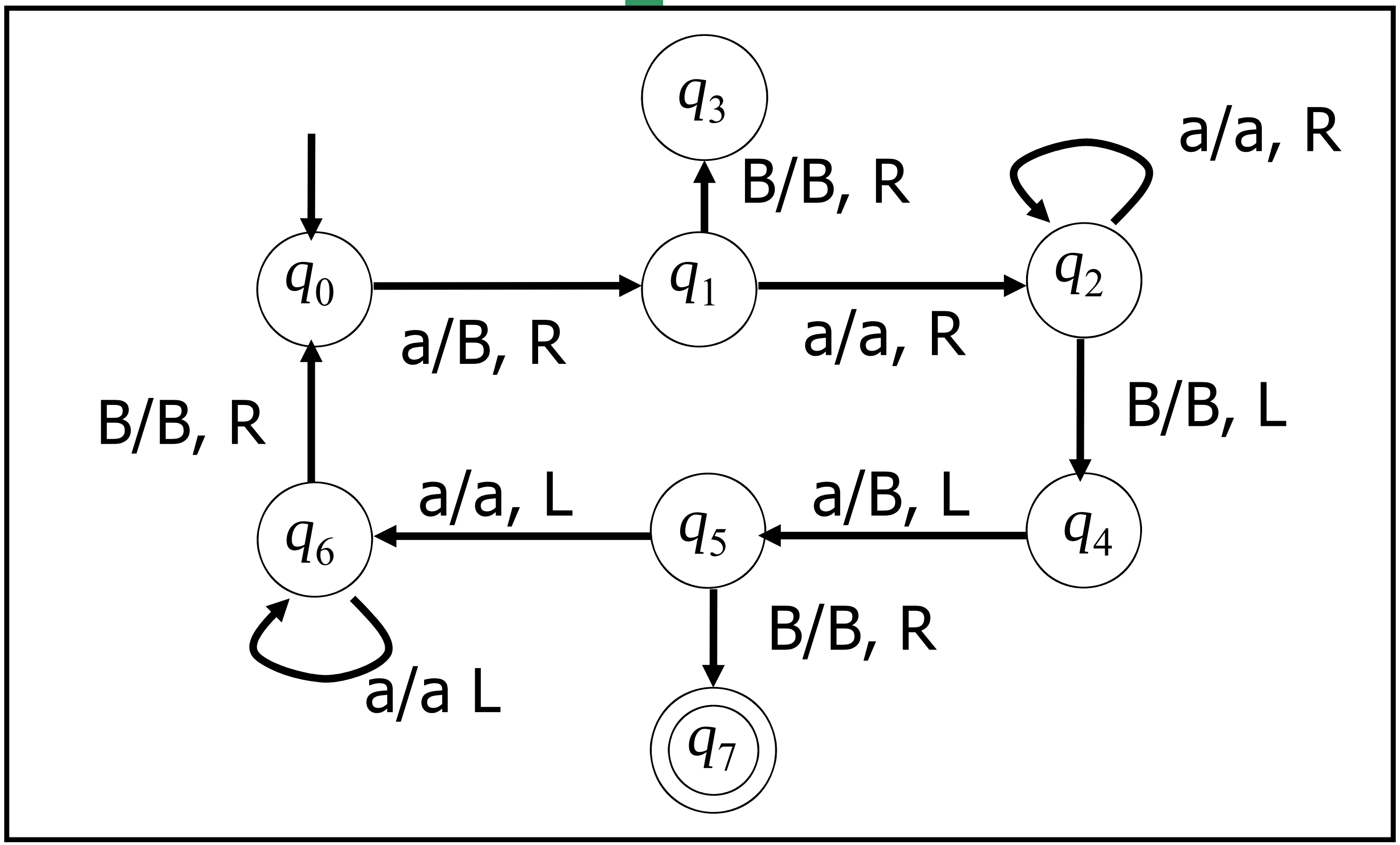
Dividing x by 2 (1-4)

B	B	a	a	a	a	a	B	B	B	B



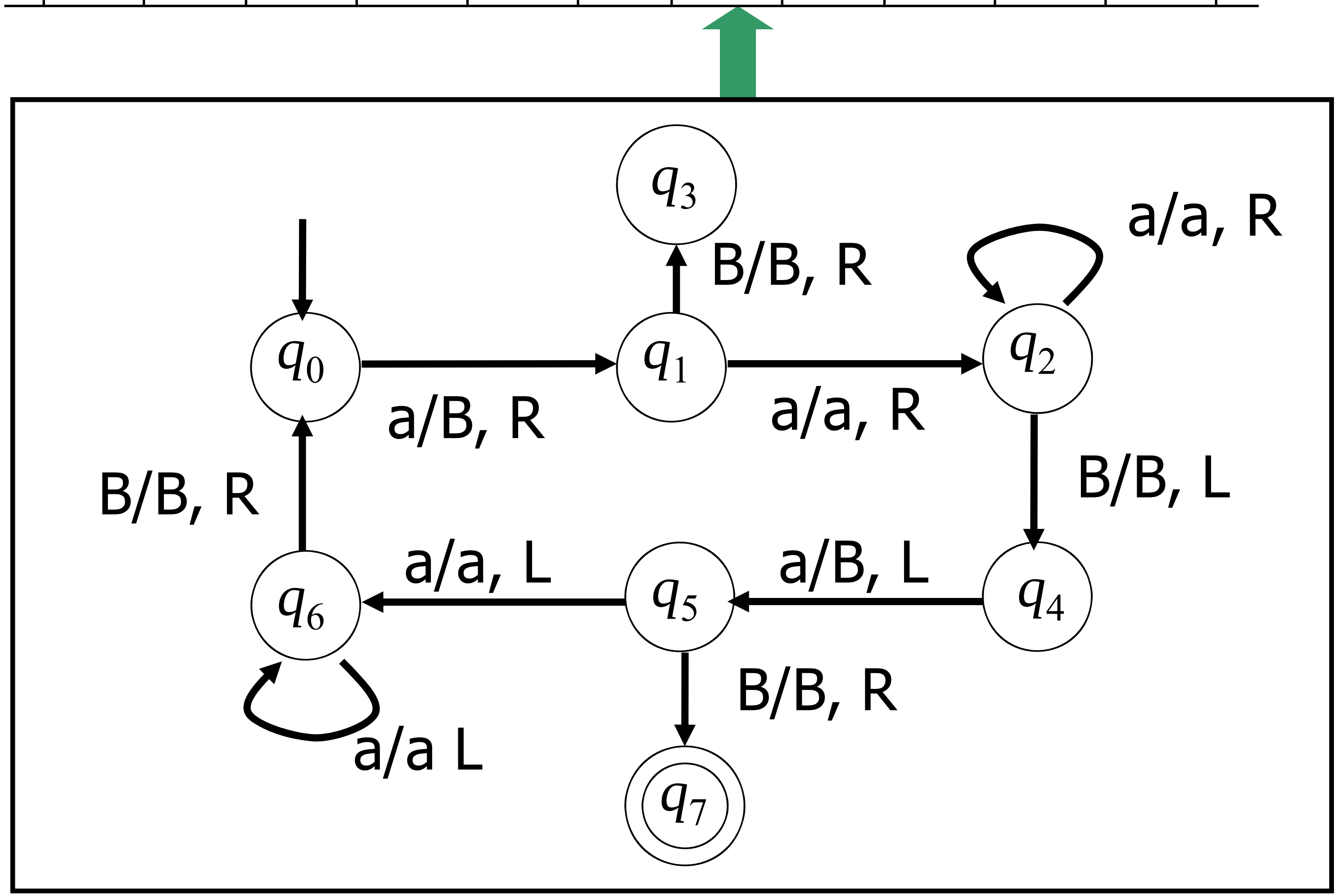
Dividing x by 2 (1-5)

B	B	a	a	a	a	a	B	B	B	B



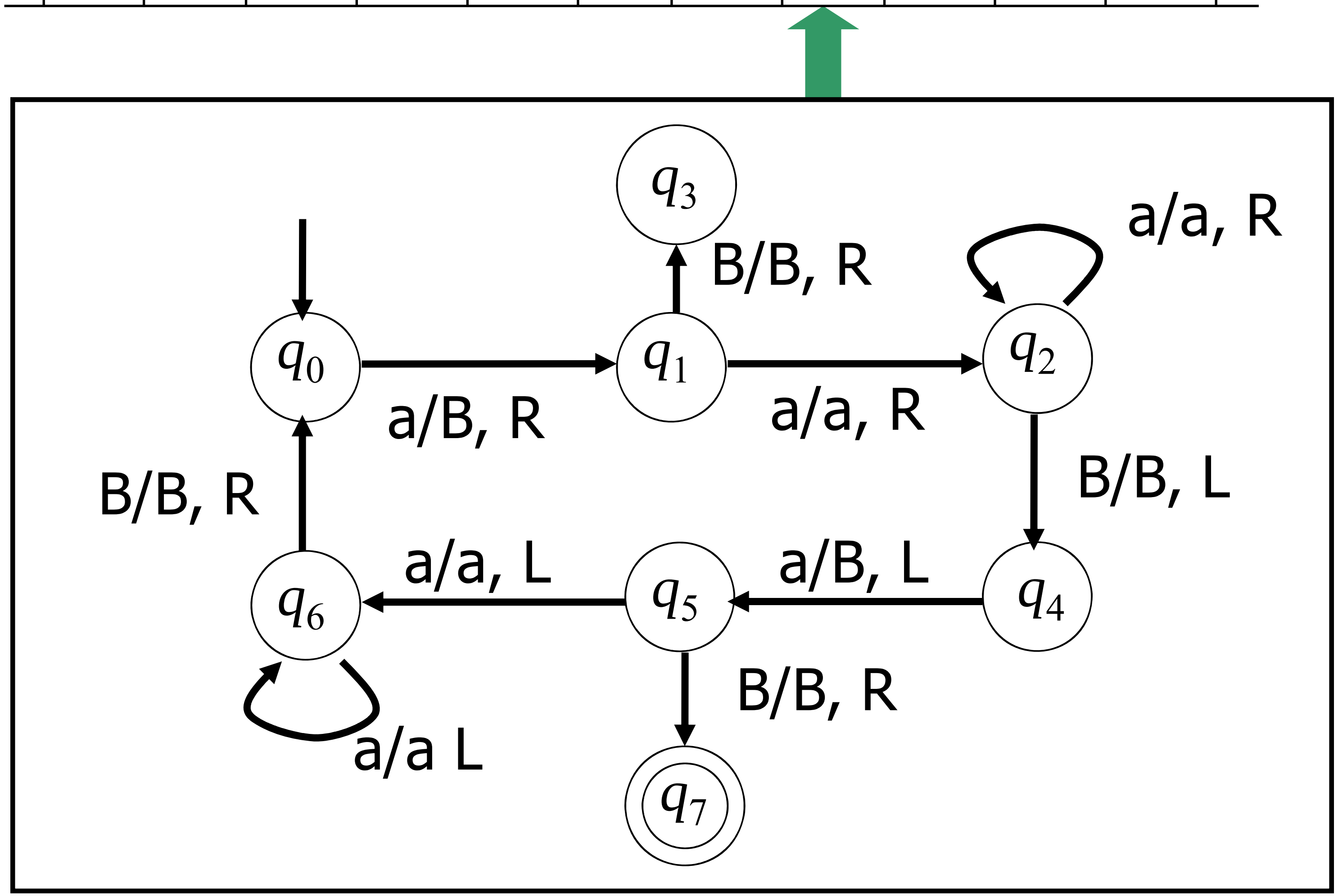
Dividing x by 2 (1-6)

B	B	a	a	a	a	a	B	B	B	B



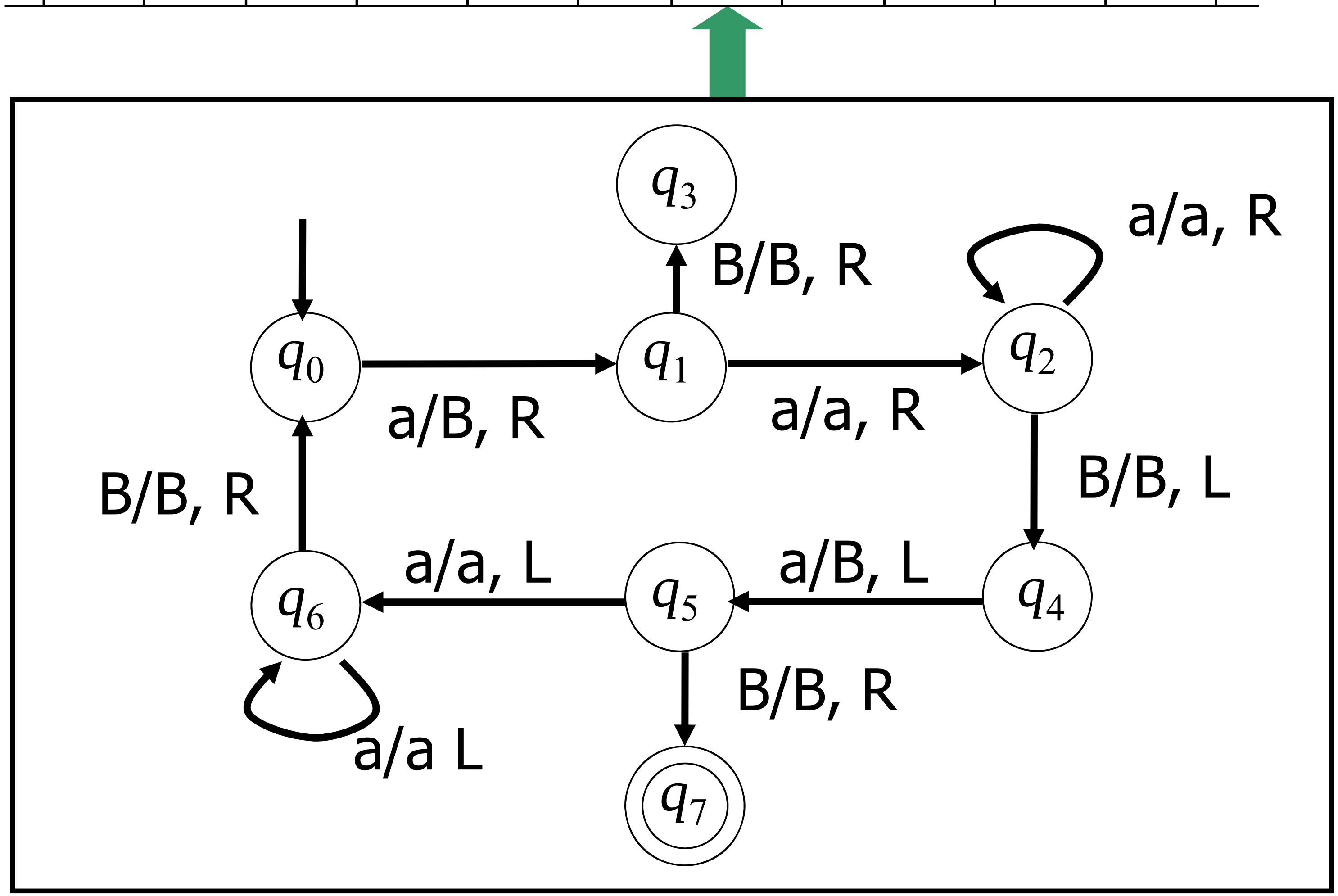
Dividing x by 2 (1-7)

B	B	a	a	a	a	a	B	B	B	B

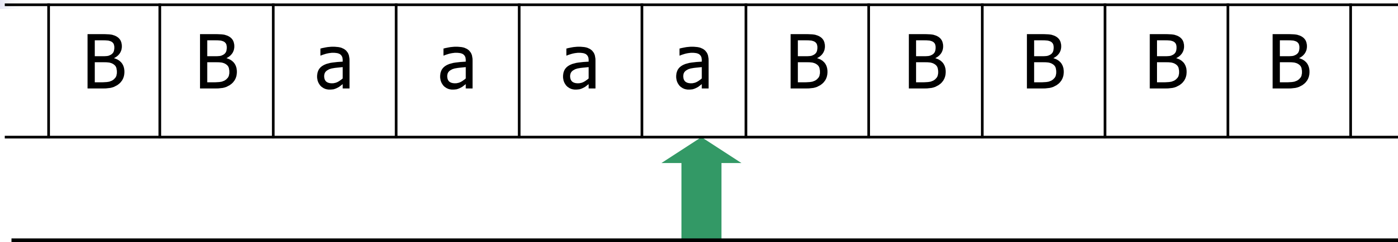


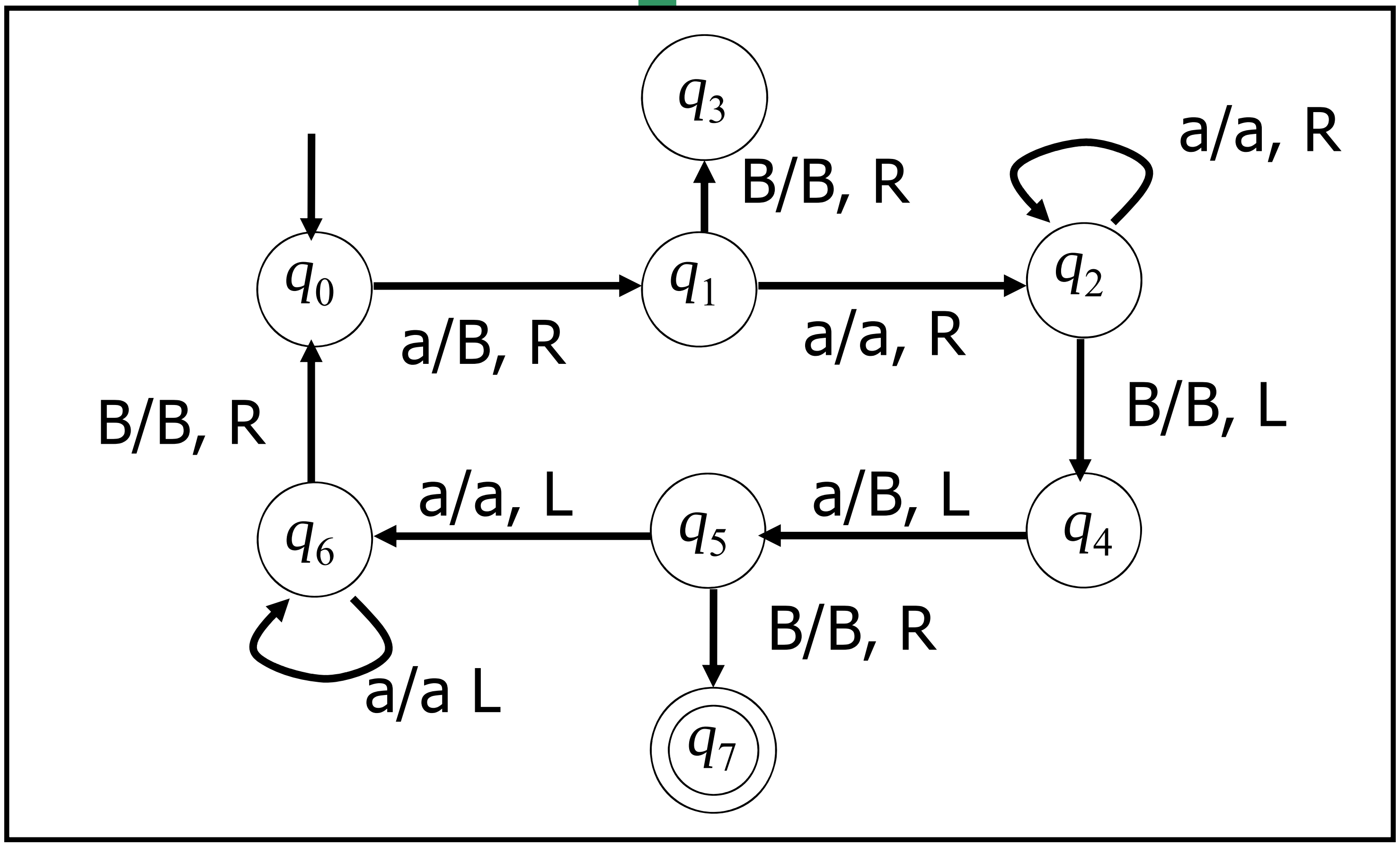
Dividing x by 2 (1-8)

B	B	a	a	a	a	a	B	B	B	B



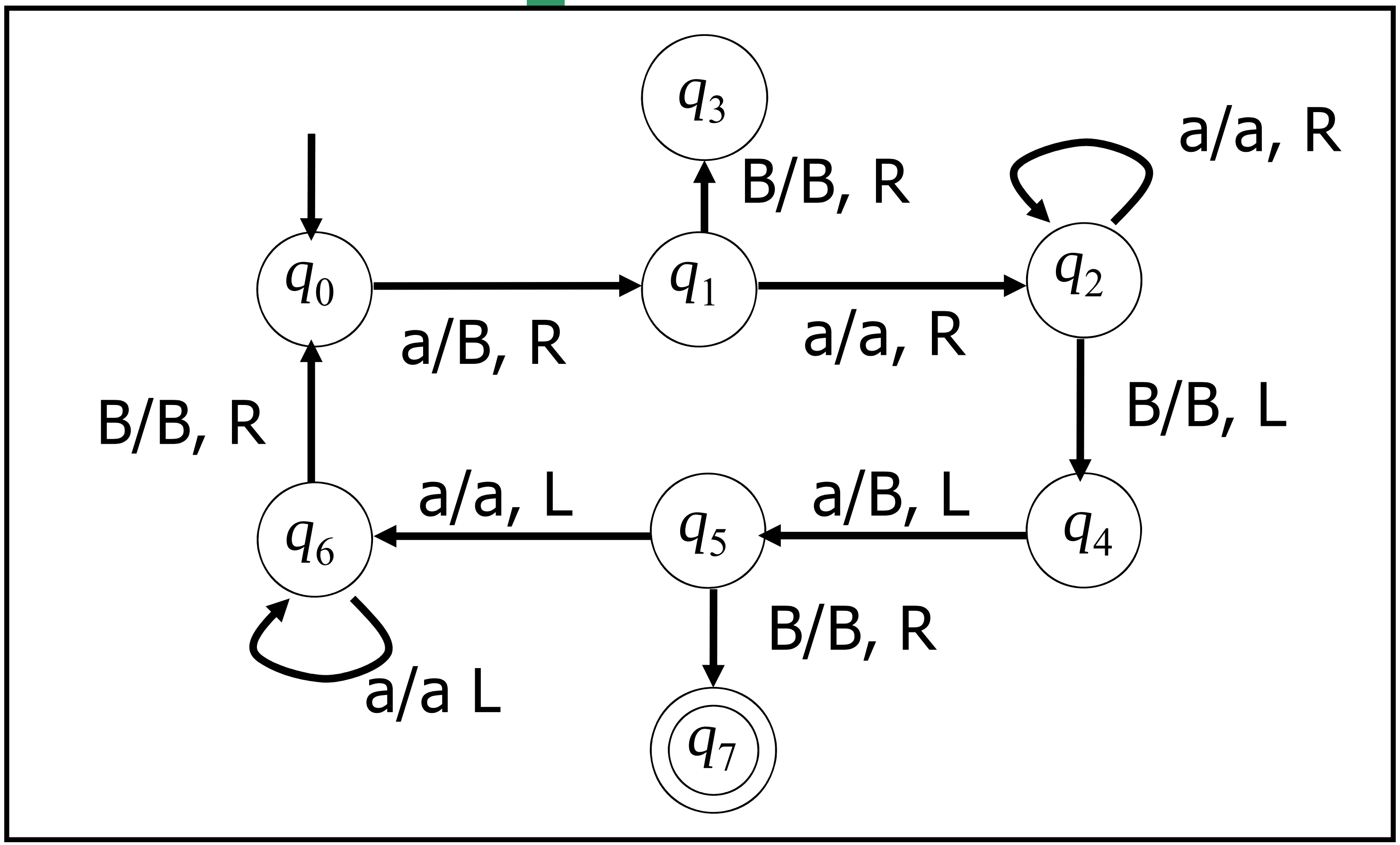
Dividing x by 2 (1-9)





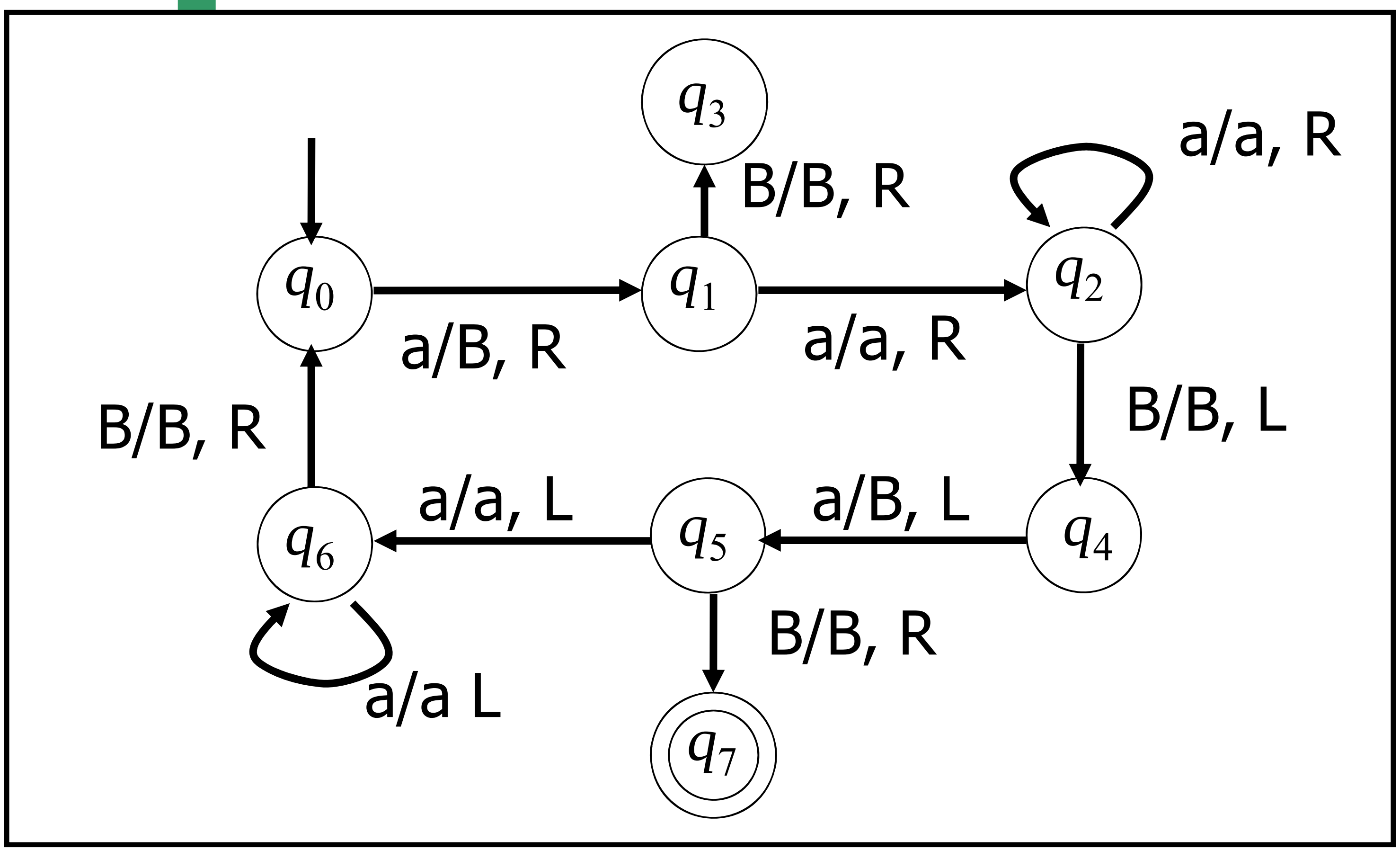
Dividing x by 2 (1-10)

B	B	a	a	a	a	B	B	B	B	B

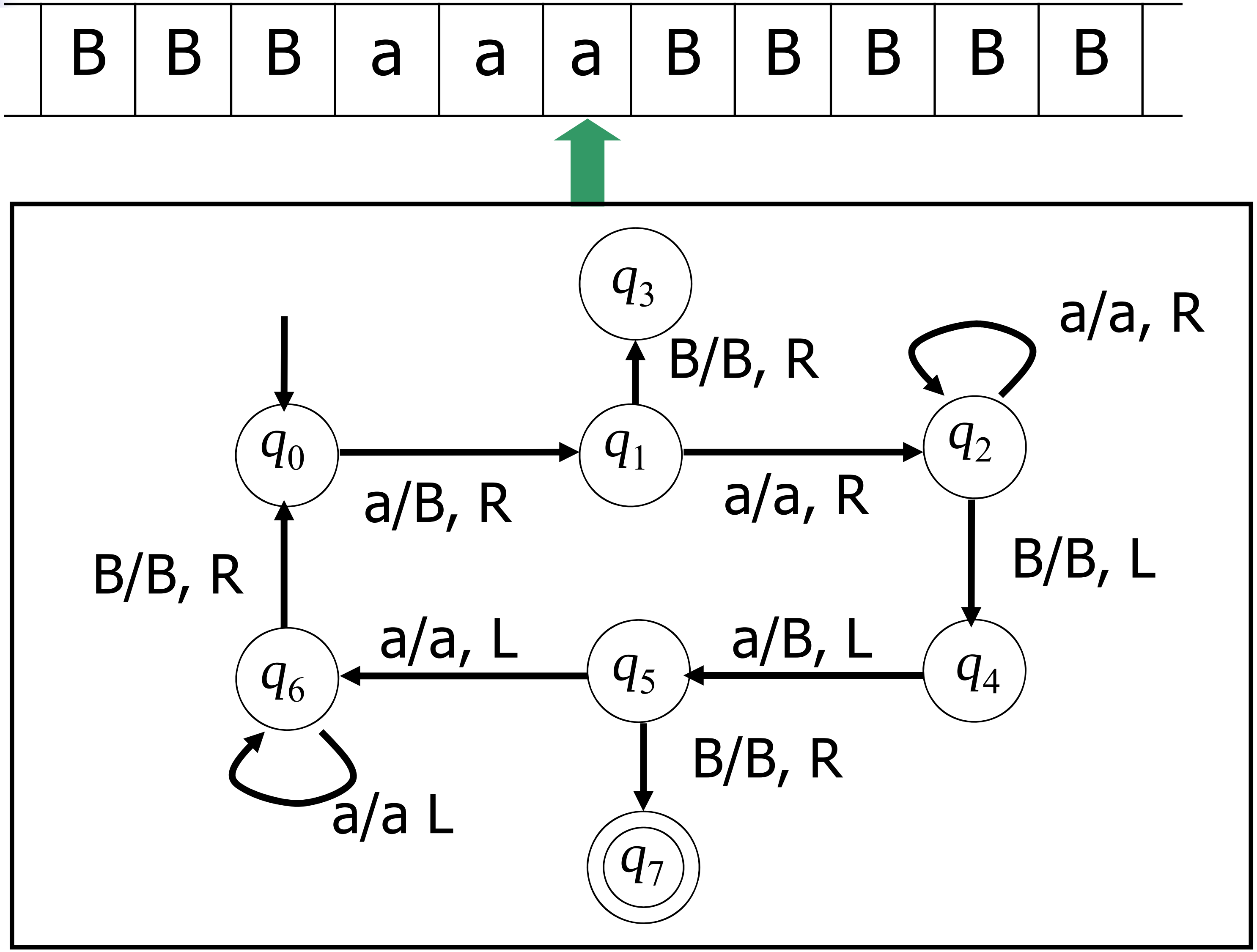


Dividing x by 2 (1-11)

B	B	a	a	a	a	B	B	B	B	B

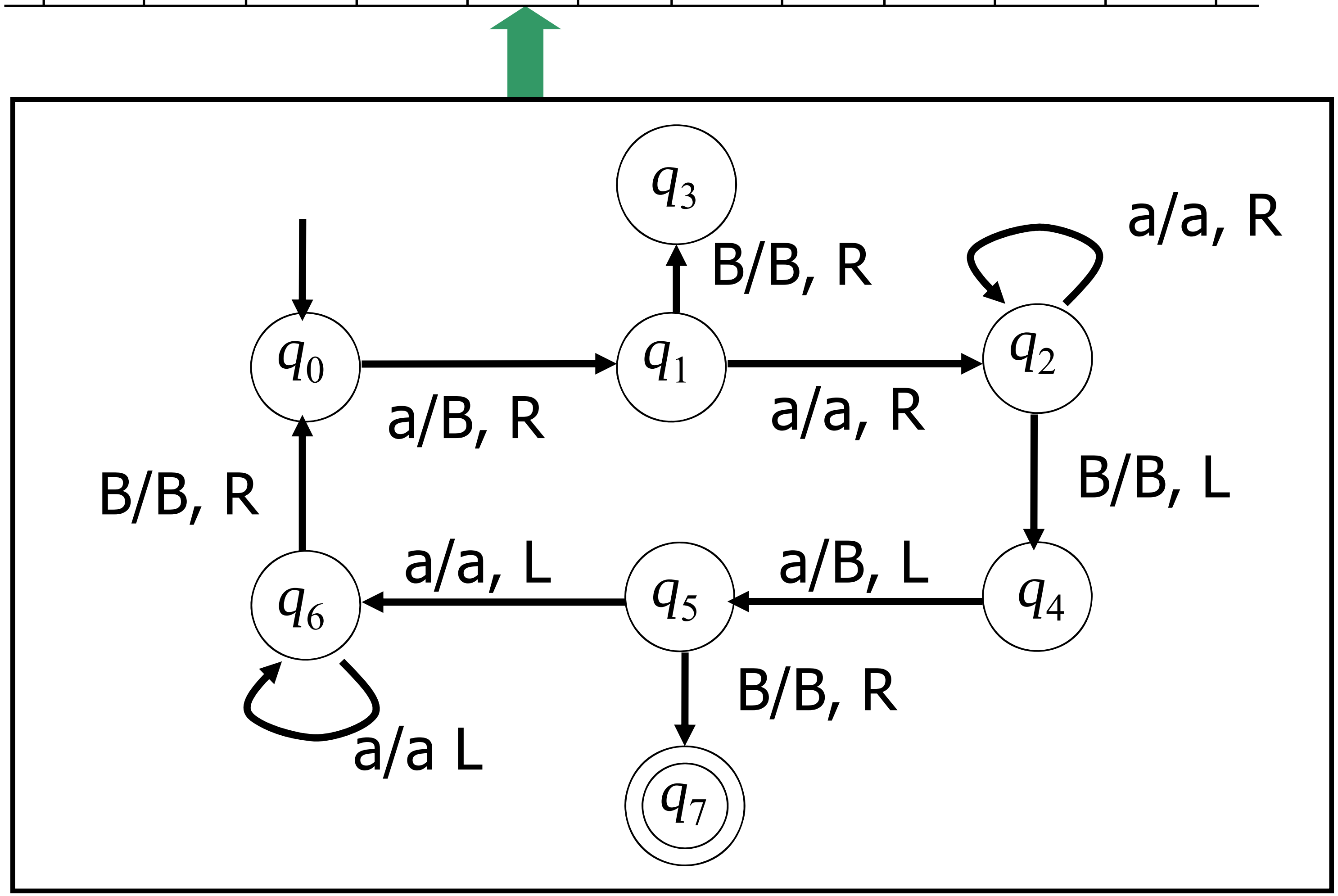


Dividing x by 2 (1-12)



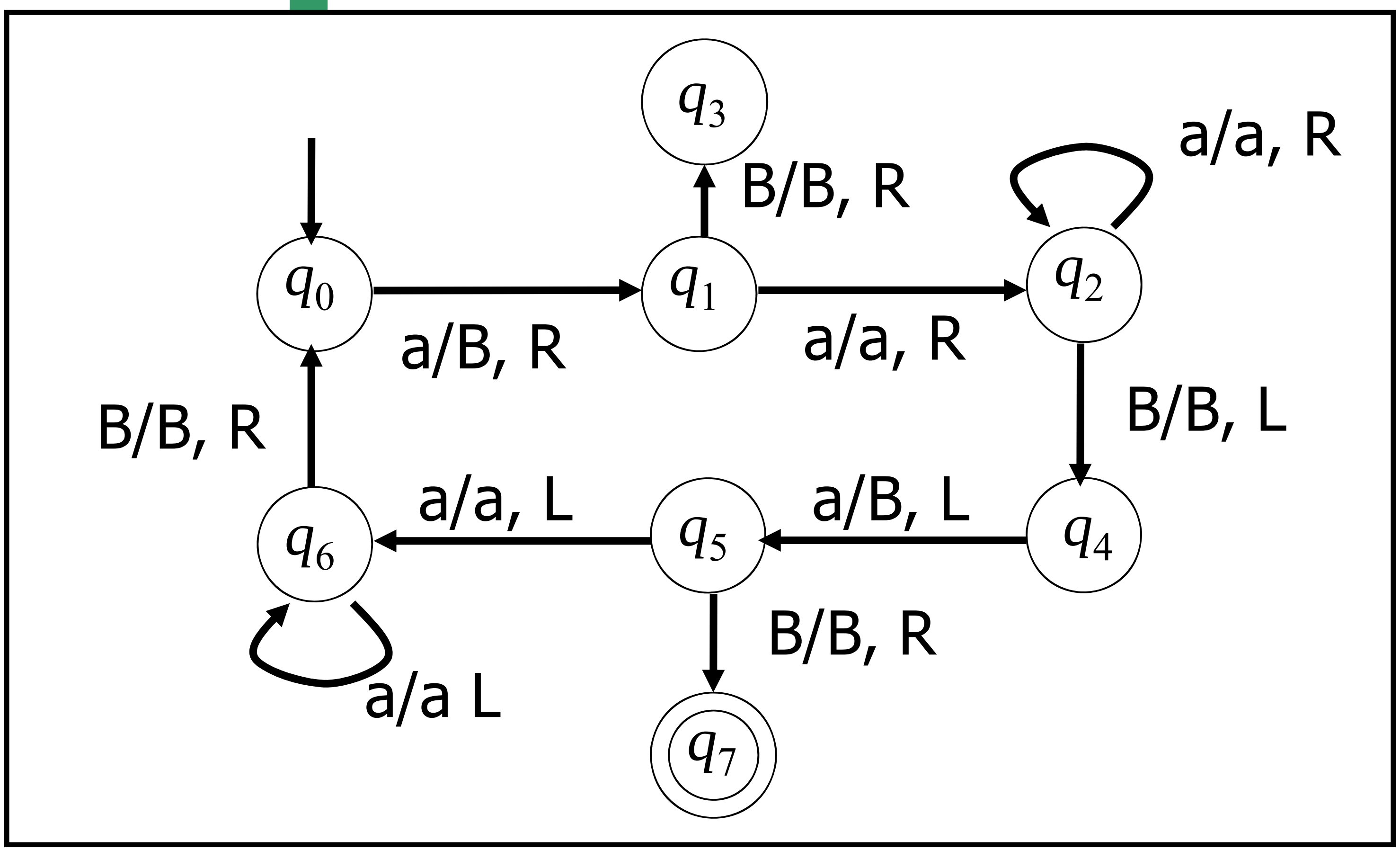
Dividing x by 2 (1-13)

B	B	B	a	a	B	B	B	B	B	B



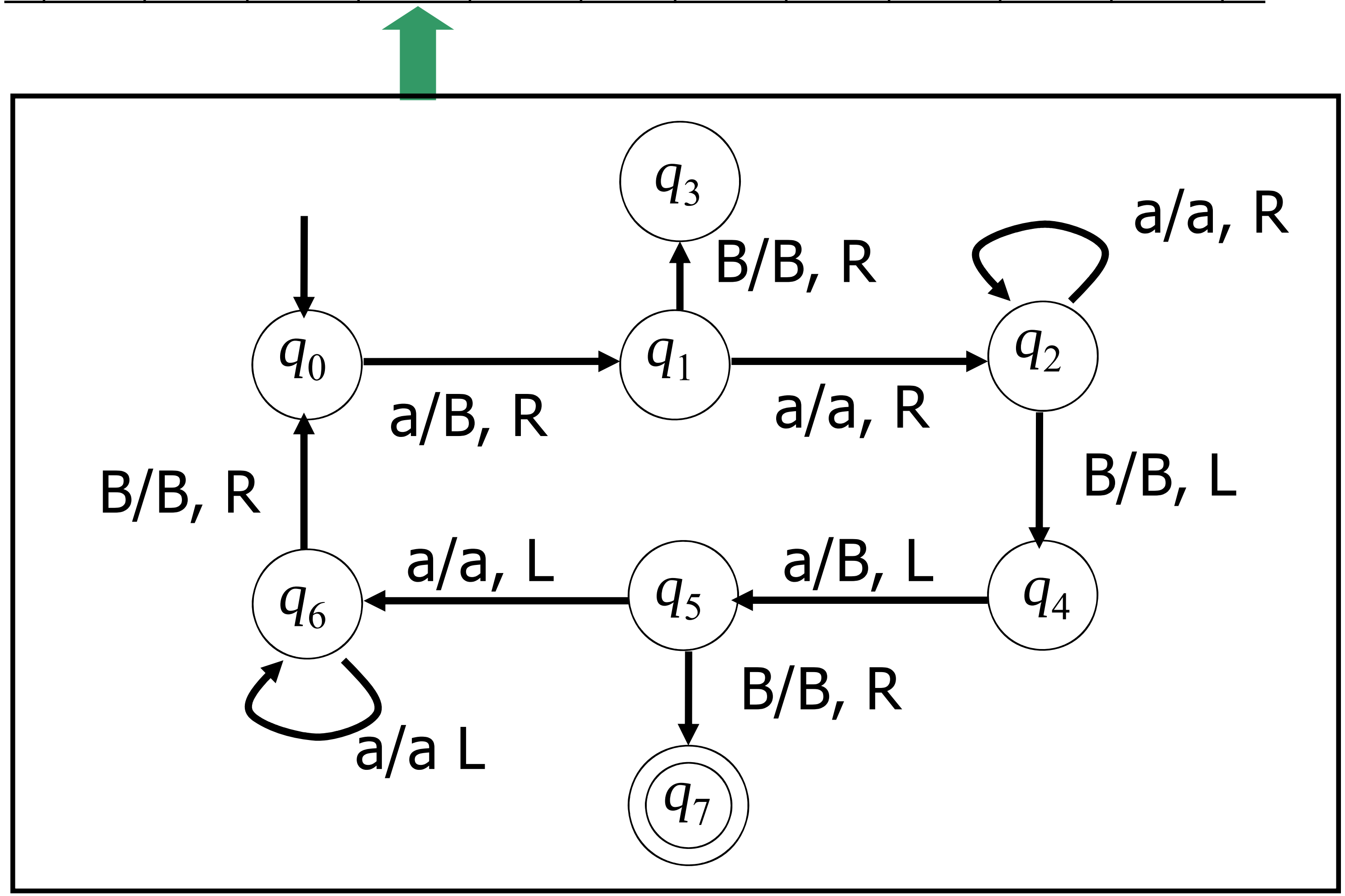
Dividing x by 2 (1-14)

B	B	B	a	a	B	B	B	B	B	B



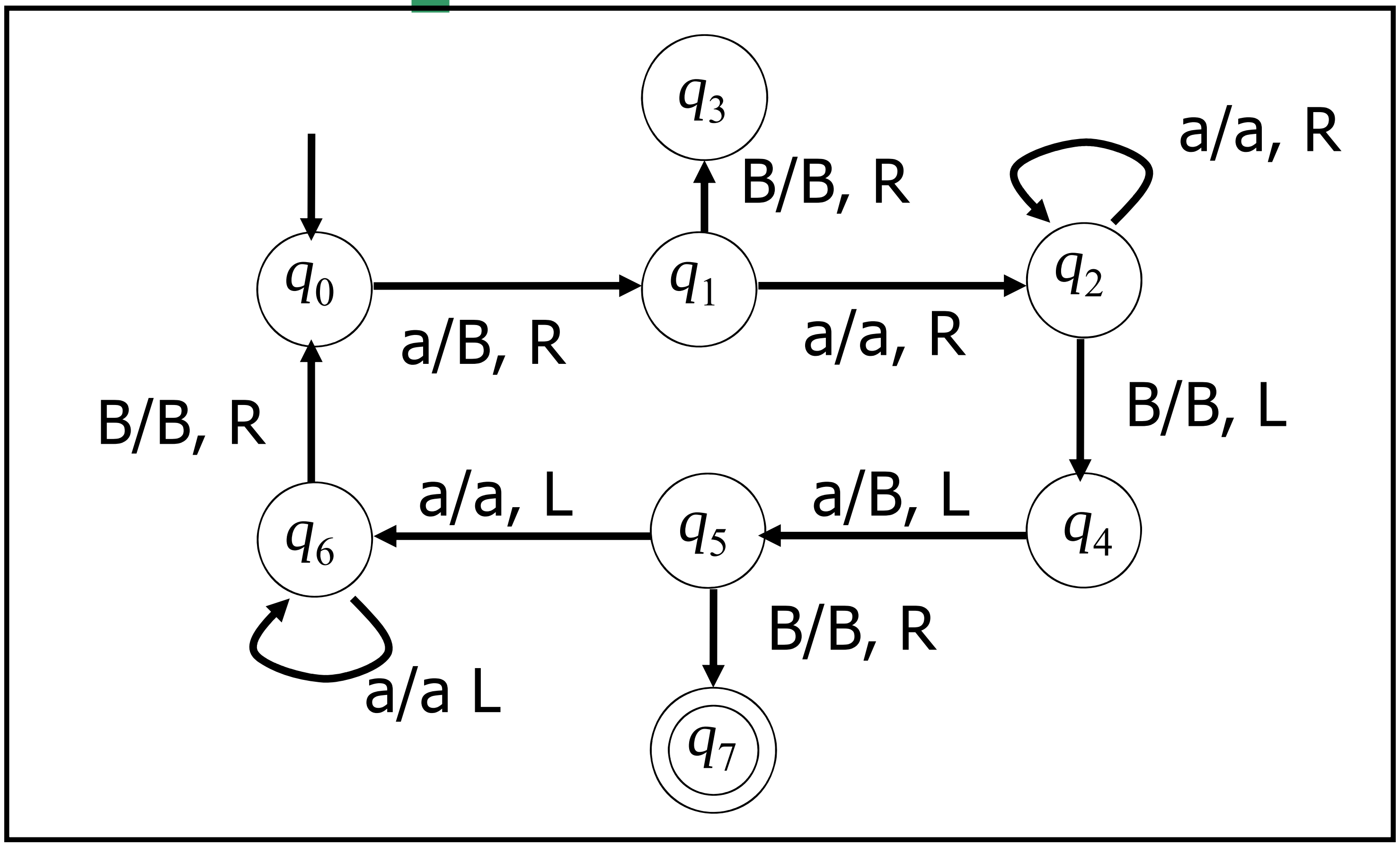
Dividing x by 2 (1-15)

B	B	B	a	a	B	B	B	B	B	B



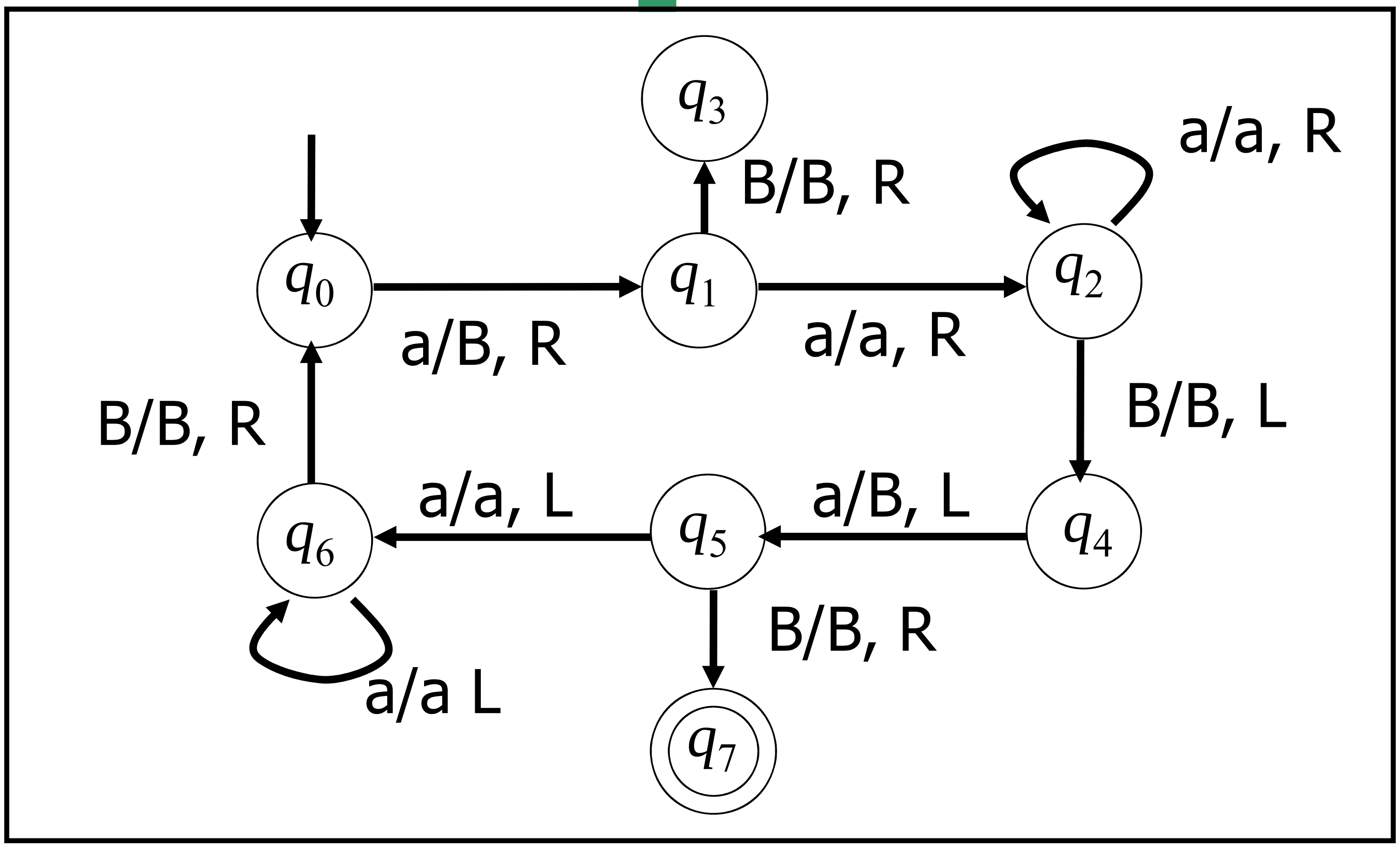
Dividing x by 2 (1-16)

B	B	B	B	a	B	B	B	B	B	B

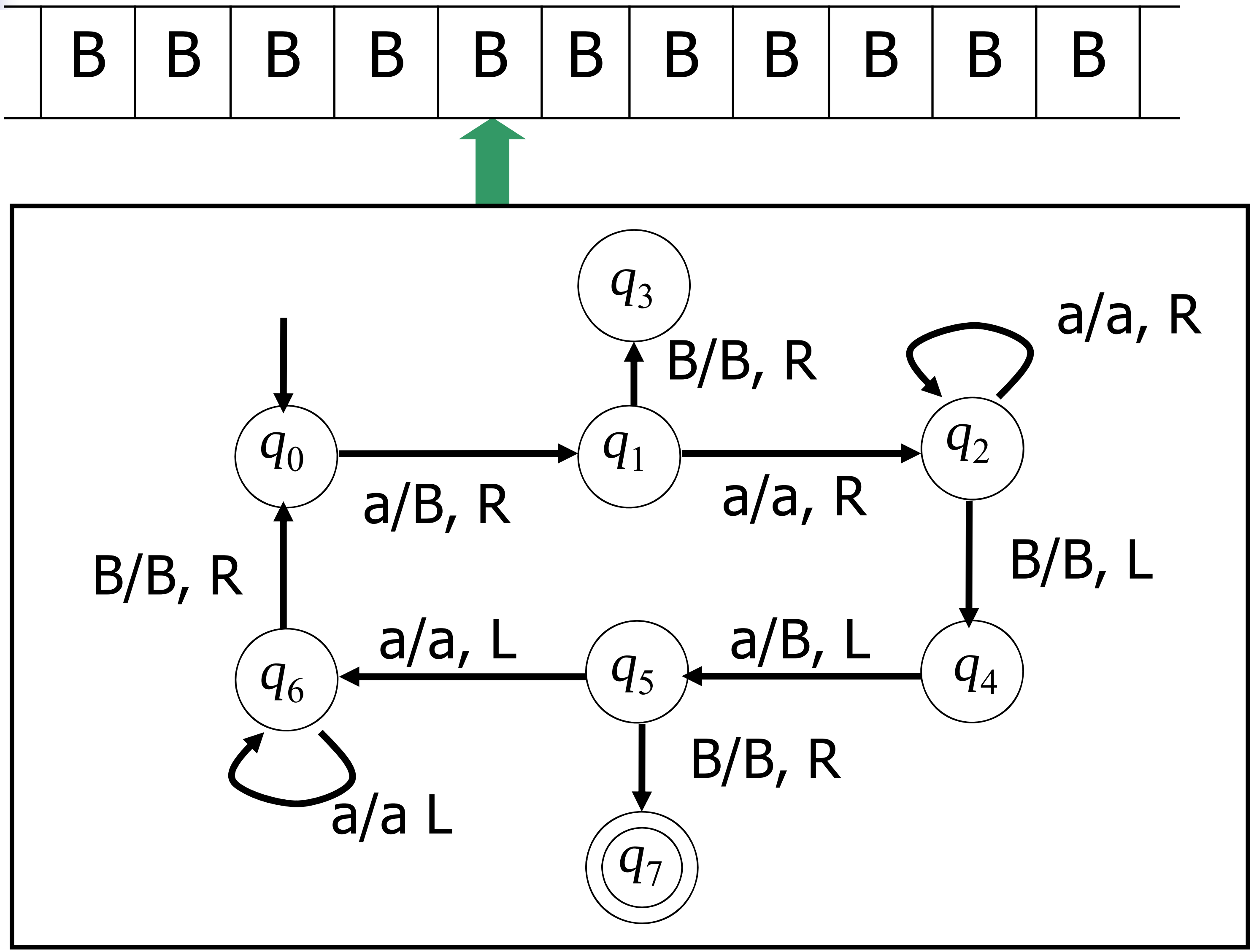


Dividing x by 2 (1-17)

B	B	B	B	a	B	B	B	B	B	B

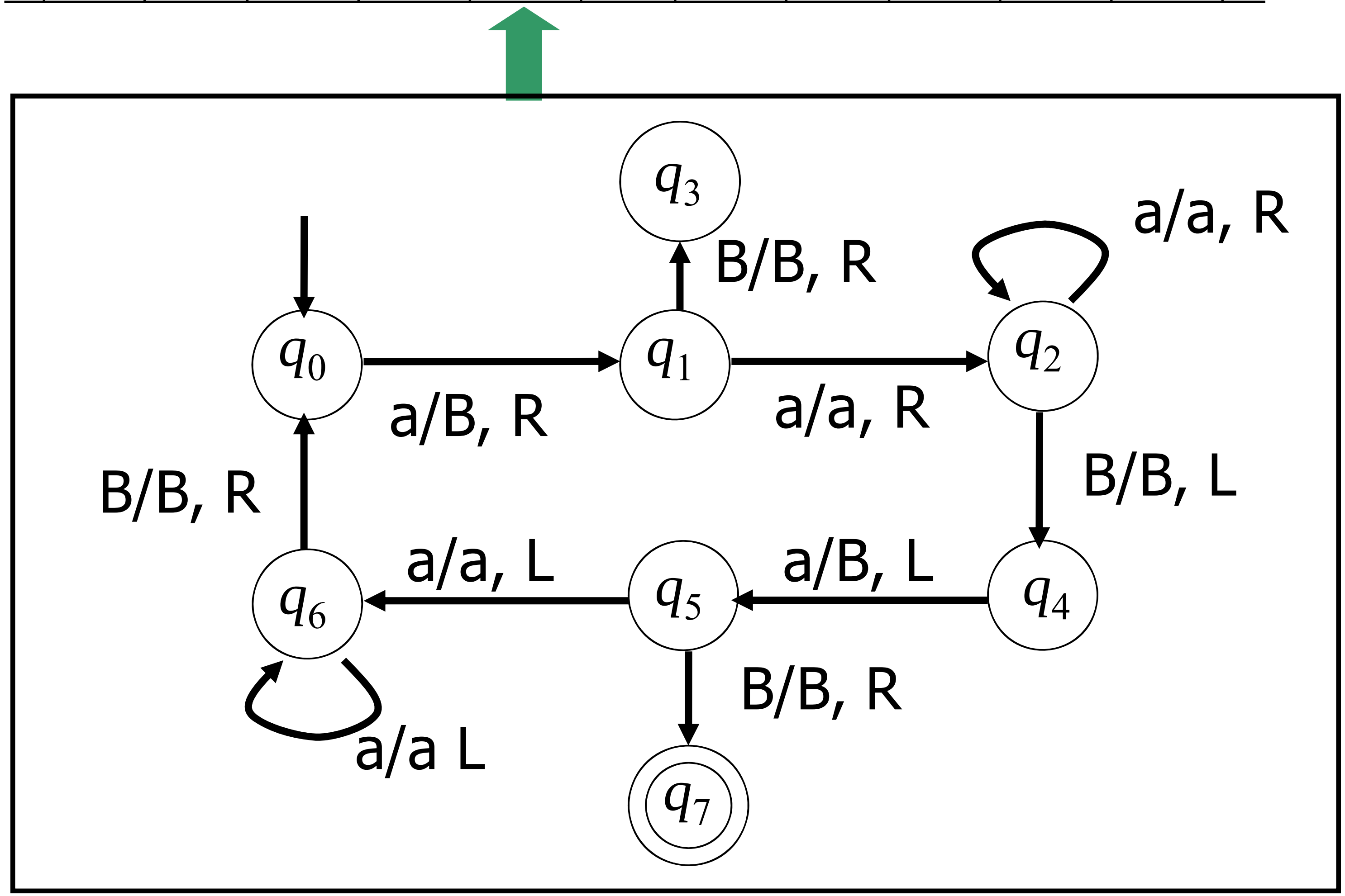


Dividing x by 2 (1-18)



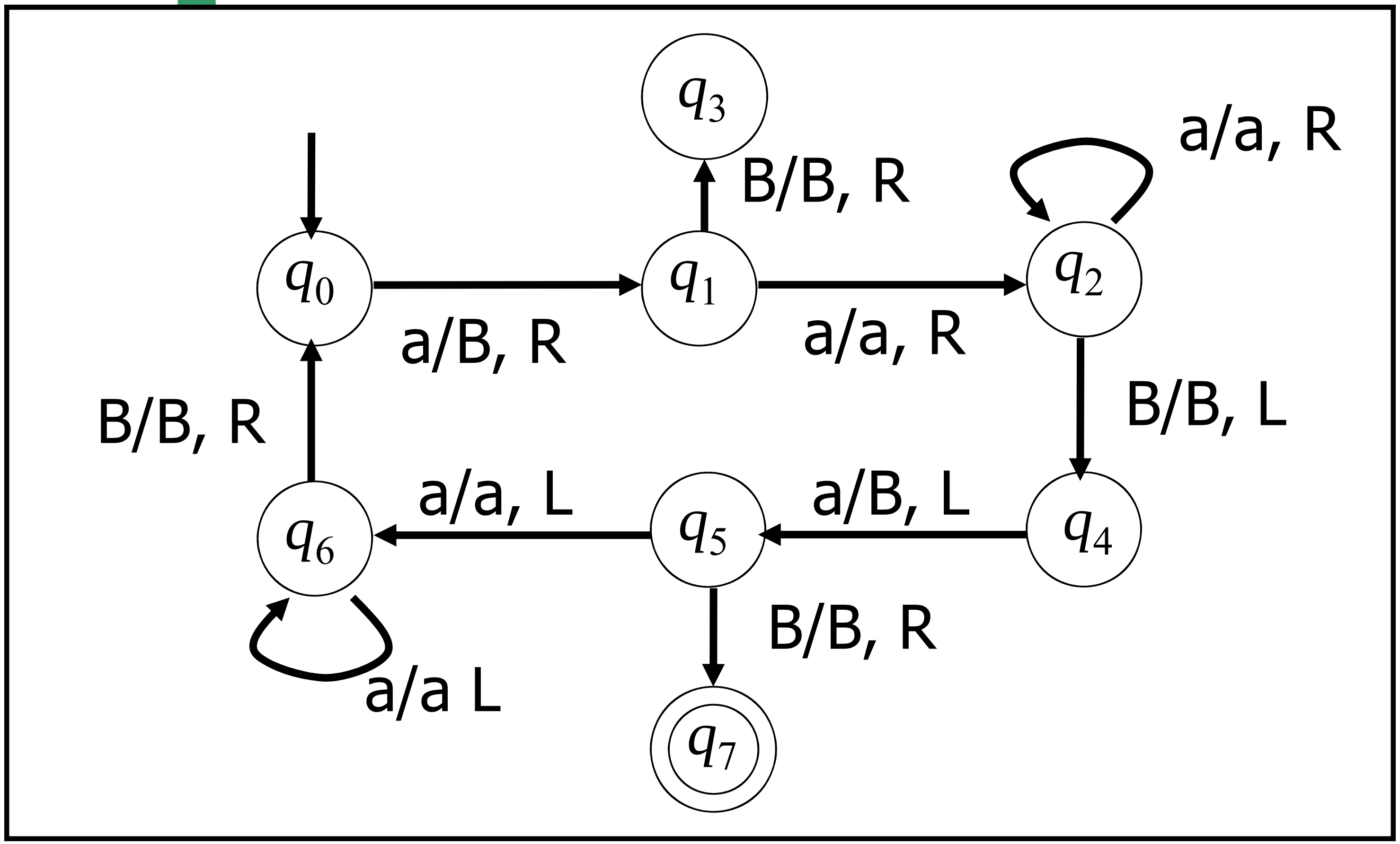
Dividing x by 2 (1-19)

B	B	B	B	B	B	B	B	B	B	B



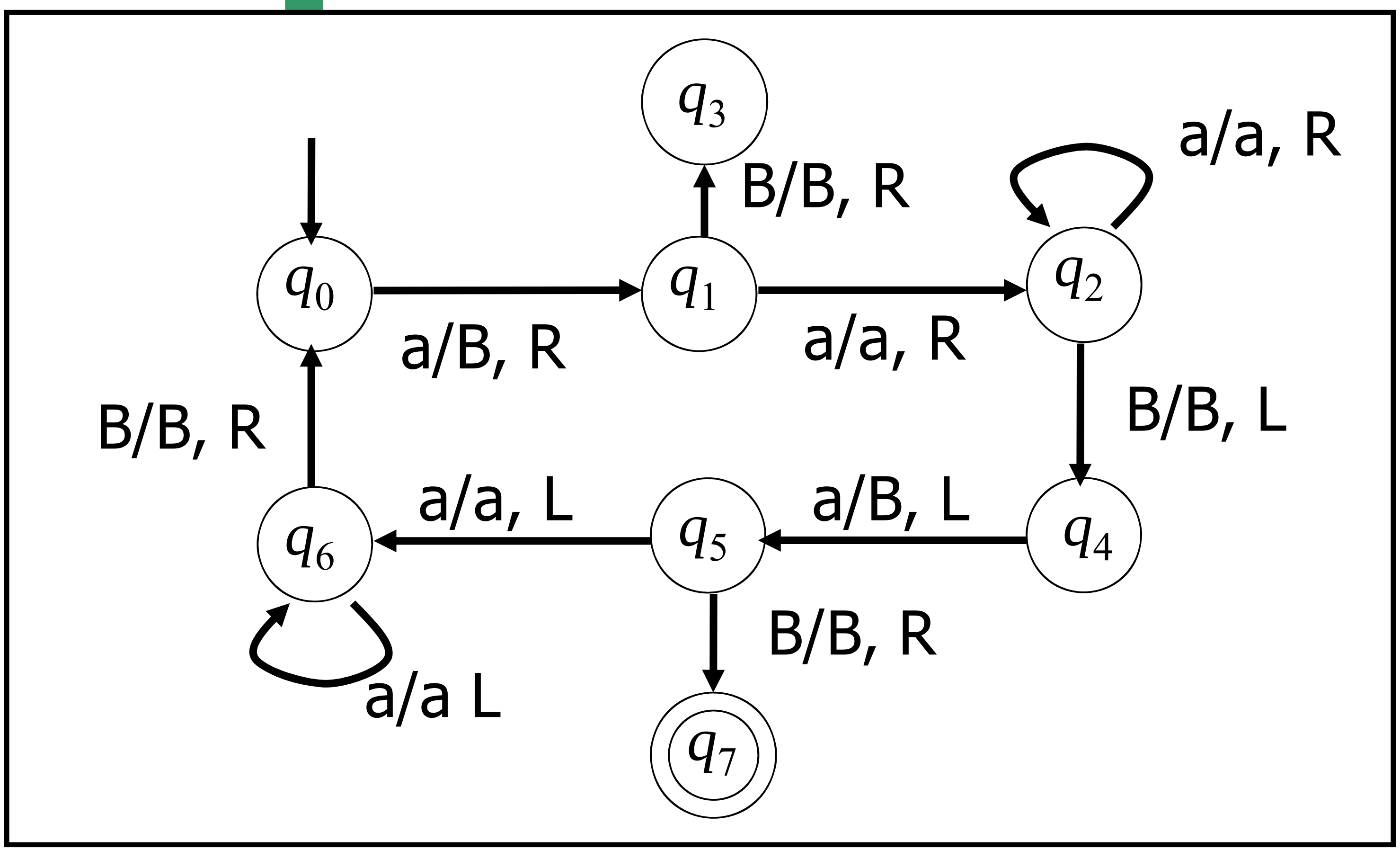
Dividing x by 2 (2-1)

B	a	a	a	a	a	a	a	B	B	B



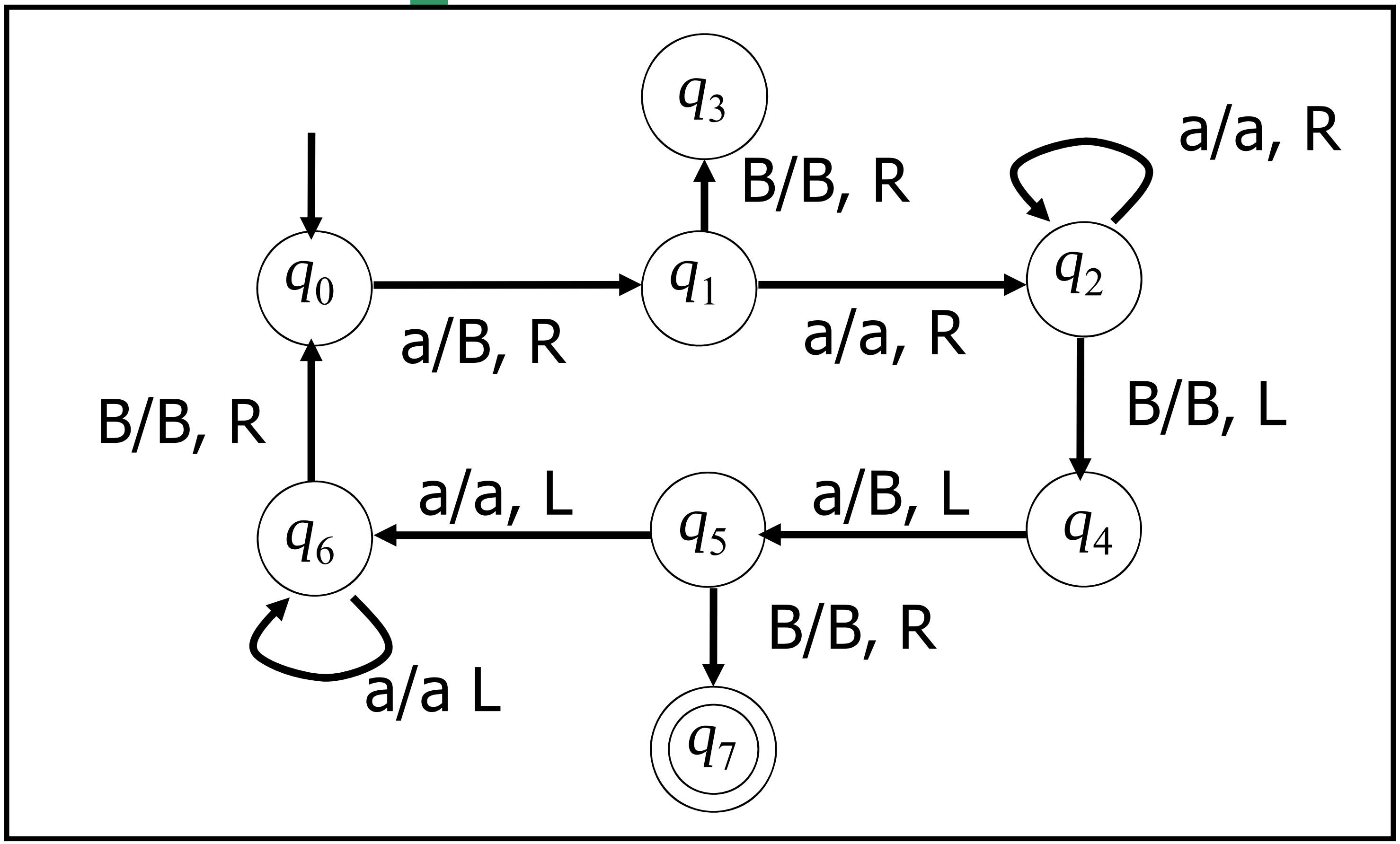
Dividing x by 2 (2-2)

B	B	a	a	a	a	a	a	B	B	B

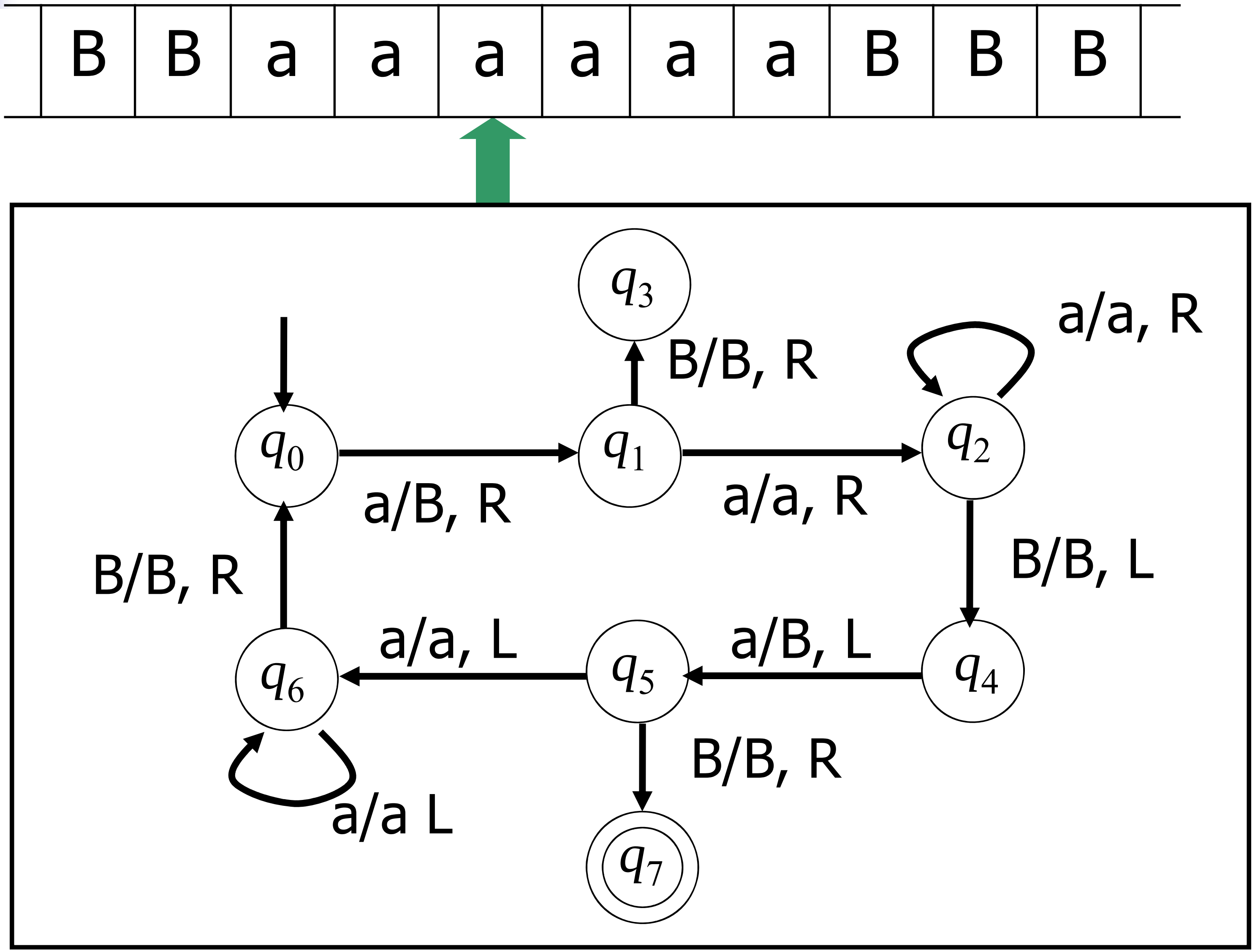


Dividing x by 2 (2-3)

B	B	a	a	a	a	a	a	B	B	B

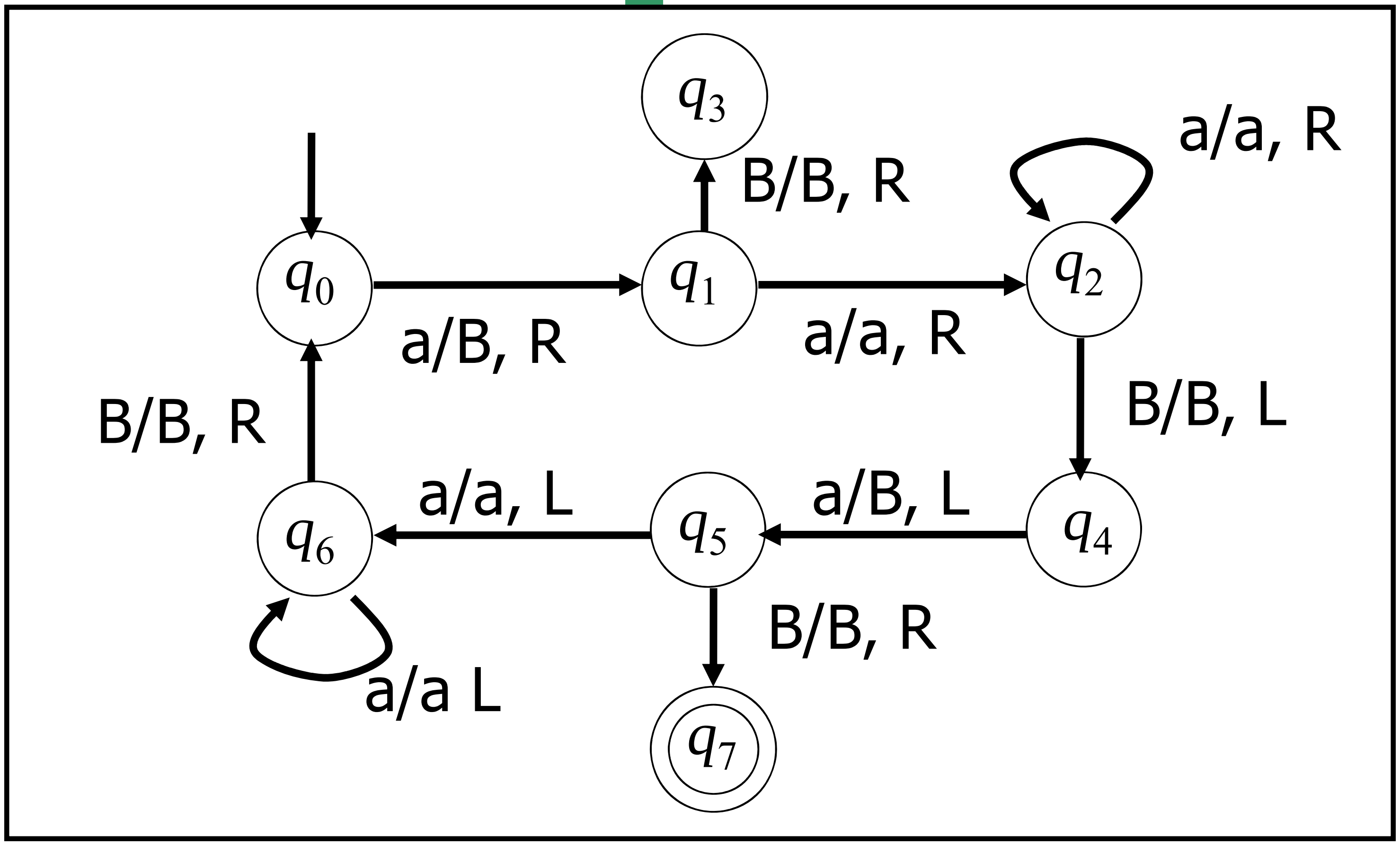


Dividing x by 2 (2-4)



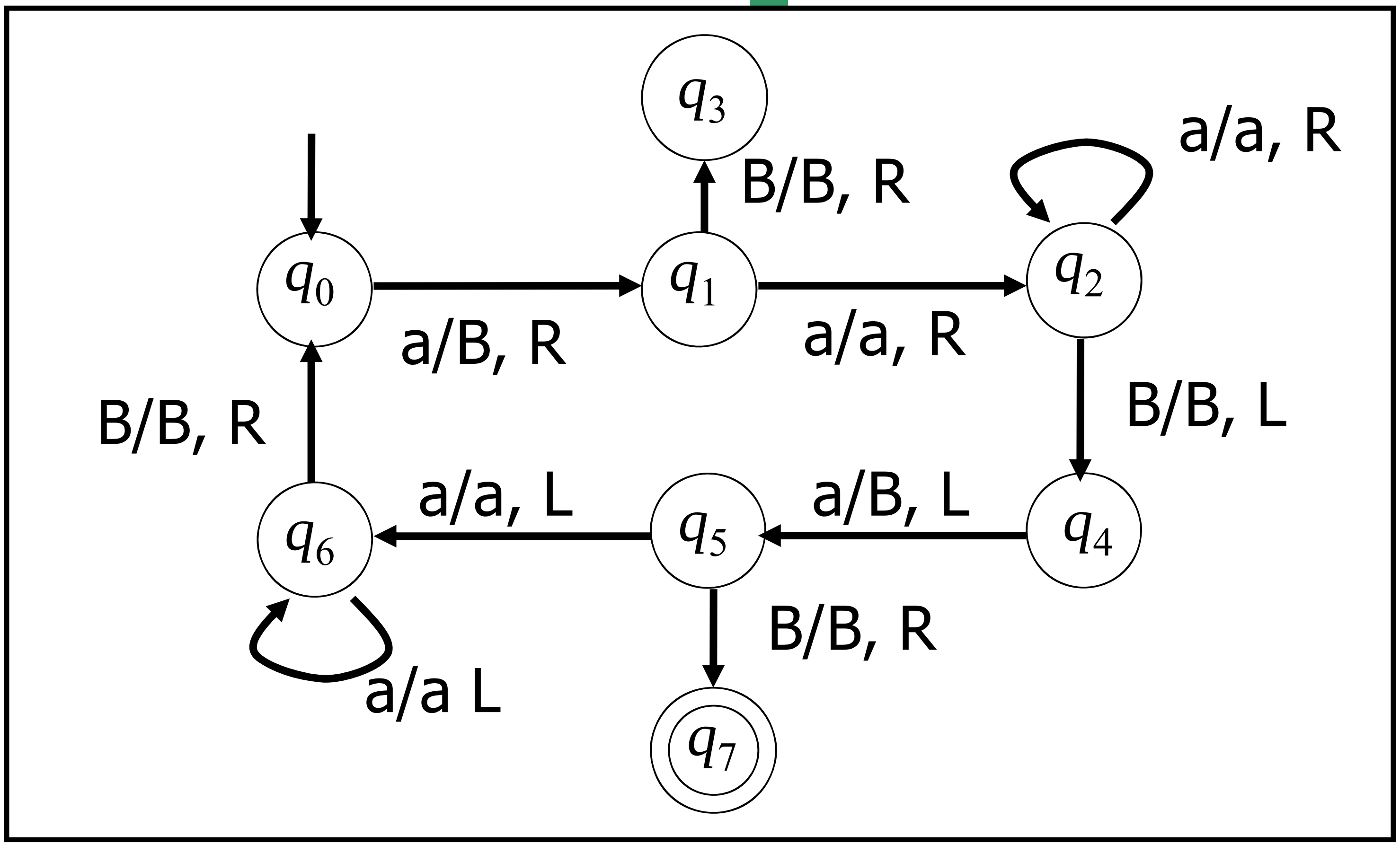
Dividing x by 2 (2-5)

B	B	a	a	a	a	a	a	B	B	B

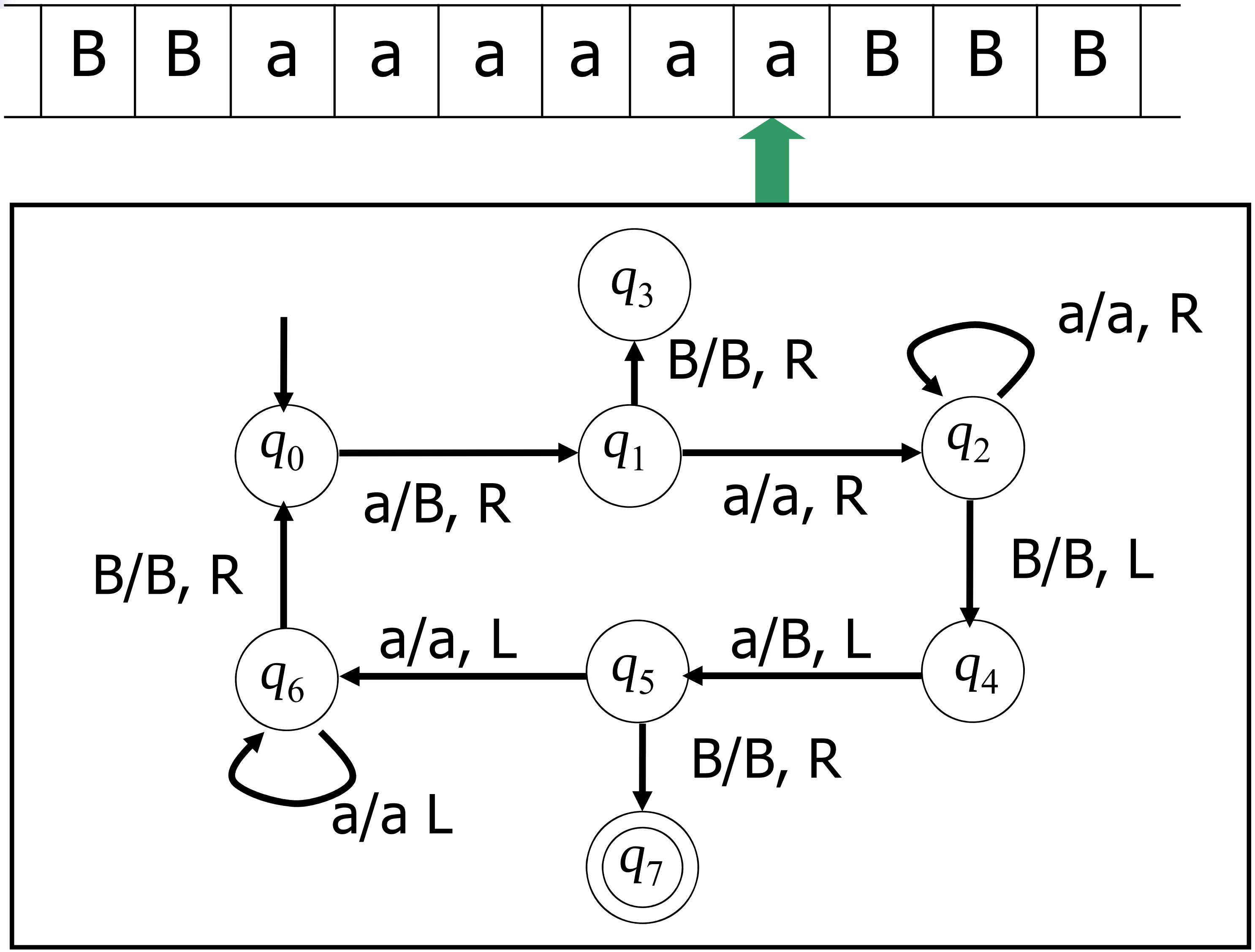


Dividing x by 2 (2-6)

B	B	a	a	a	a	a	a	B	B	B

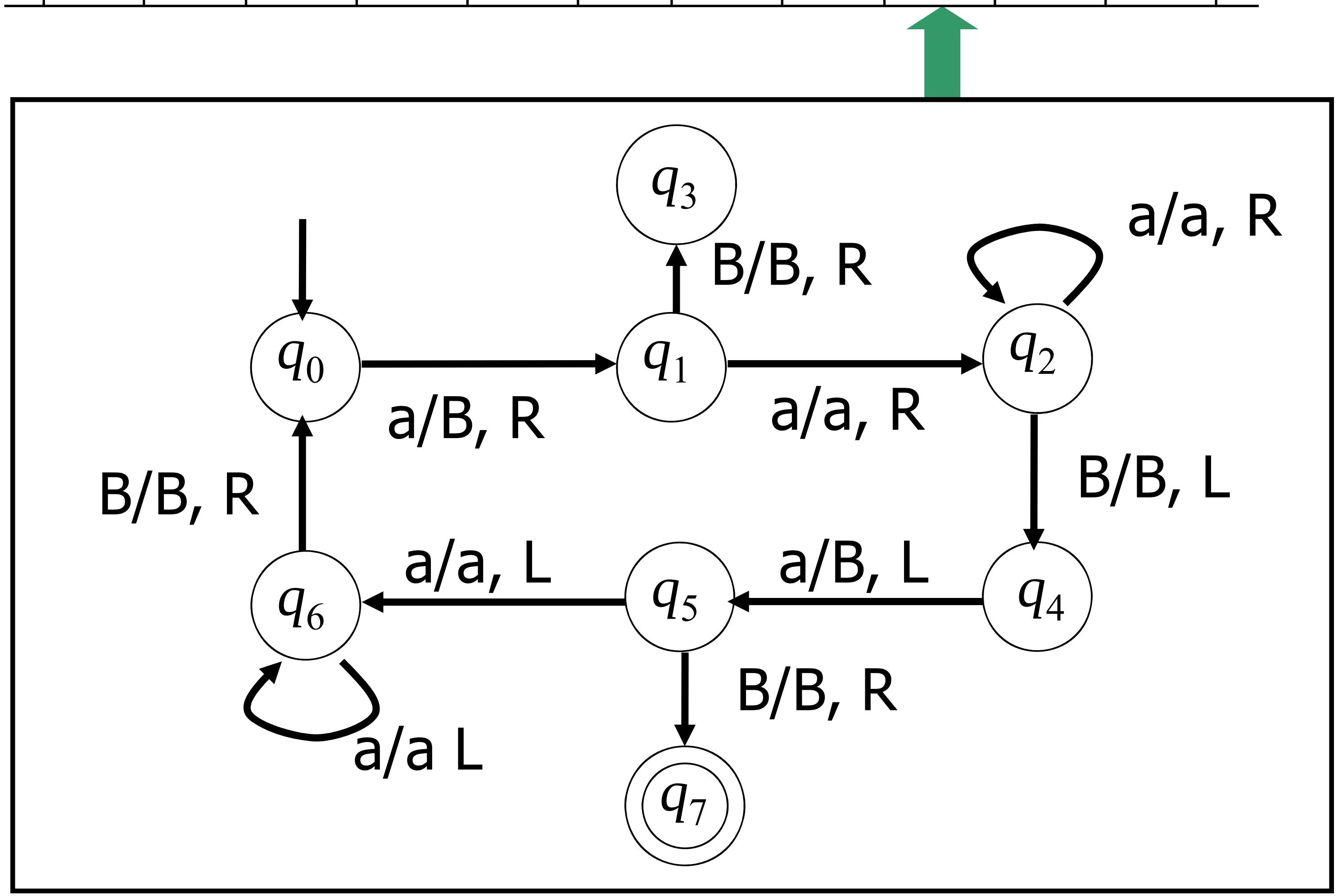


Dividing x by 2 (2-7)

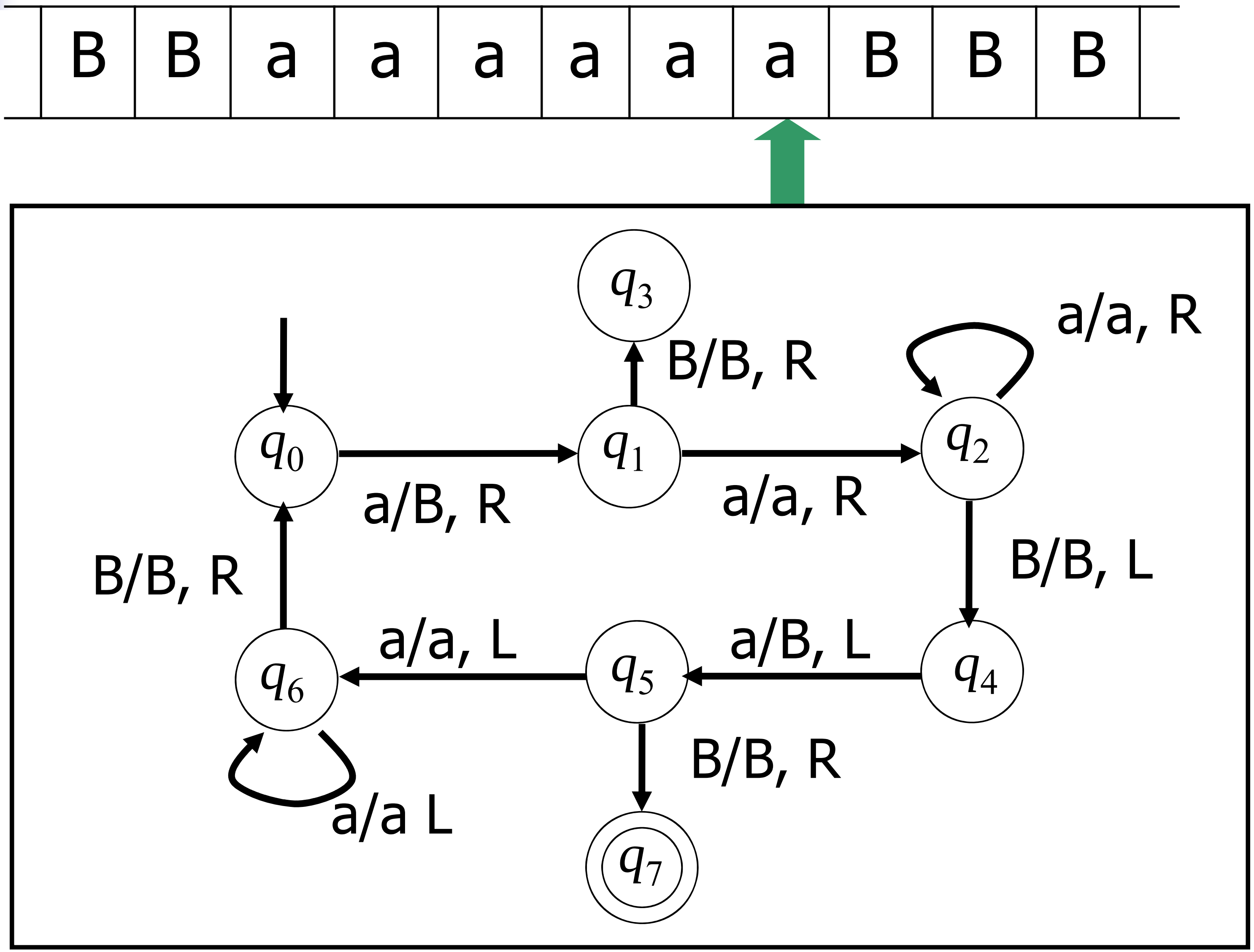


Dividing x by 2 (2-8)

B	B	a	a	a	a	a	a	B	B	B

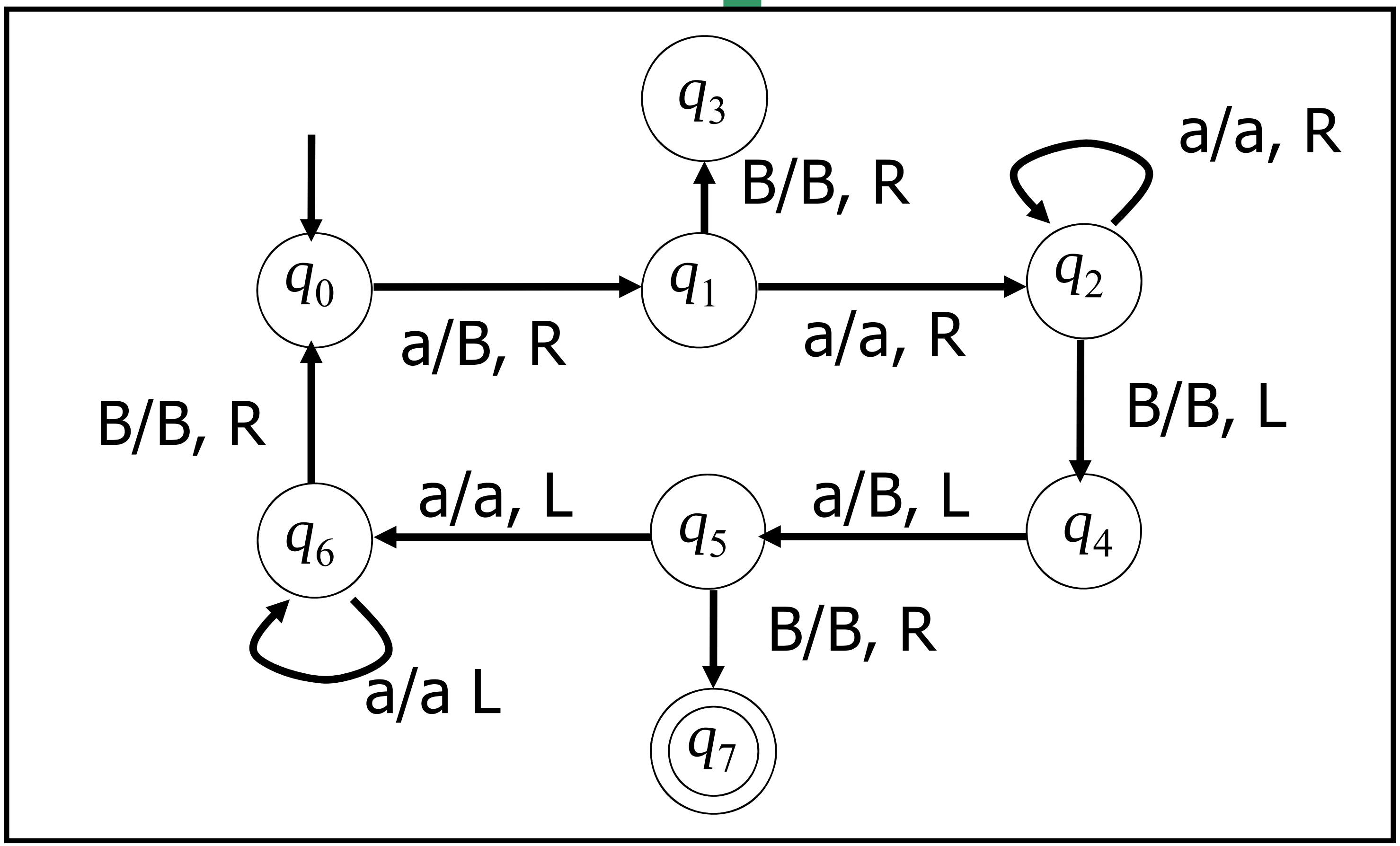


Dividing x by 2 (2-9)



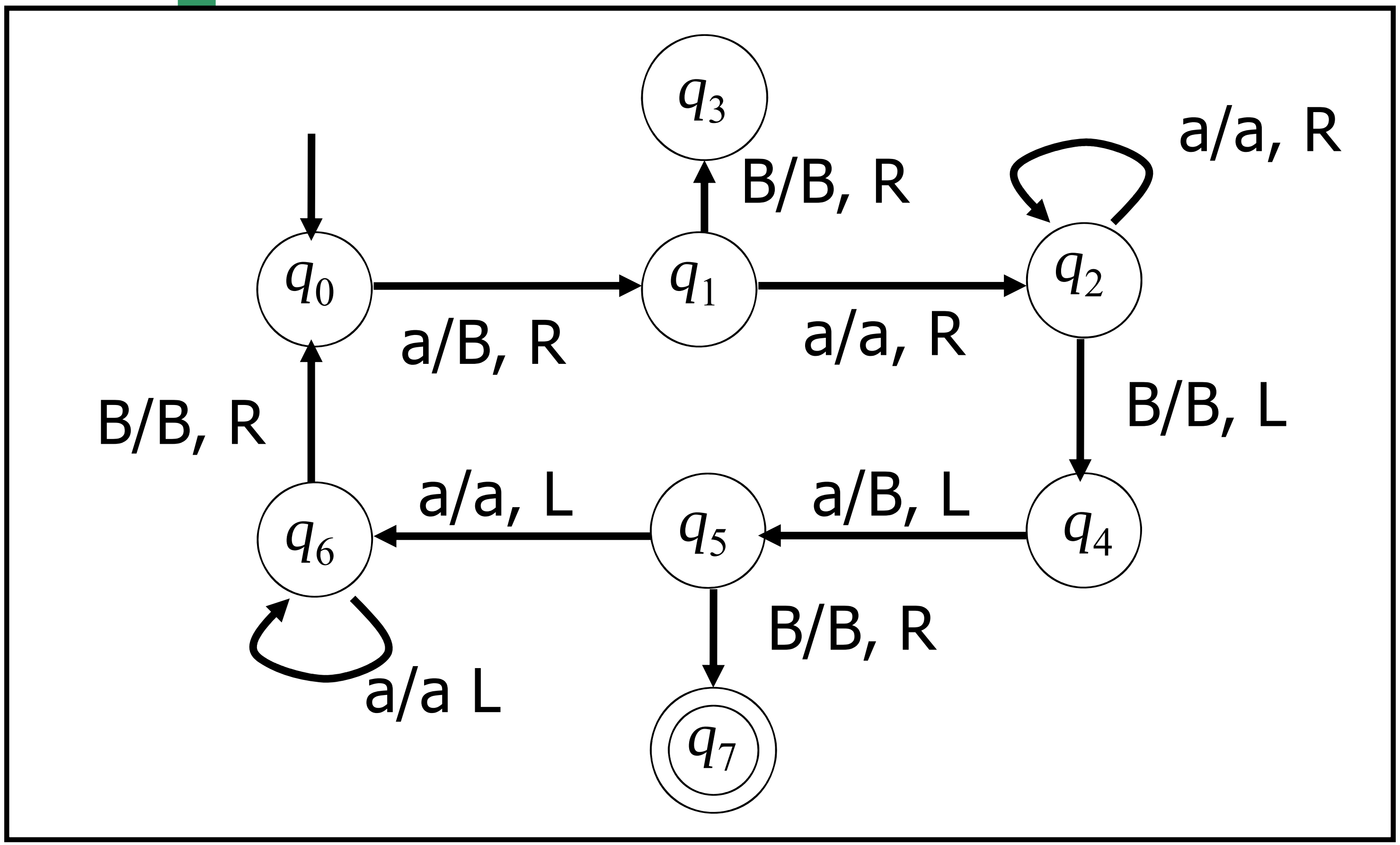
Dividing x by 2 (2-10)

B	B	a	a	a	a	a	B	B	B	B

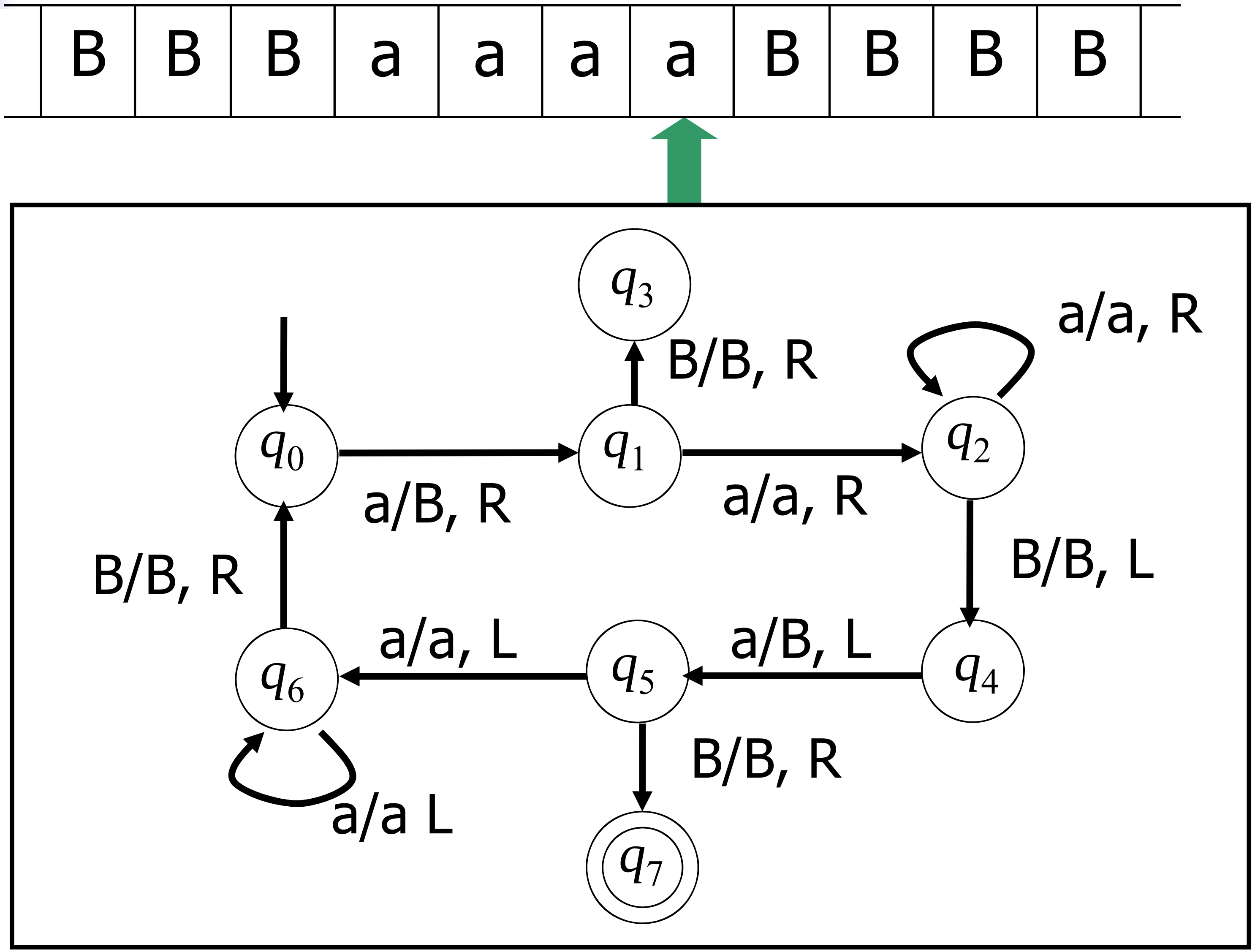


Dividing x by 2 (2-11)

B	B	a	a	a	a	a	B	B	B	B

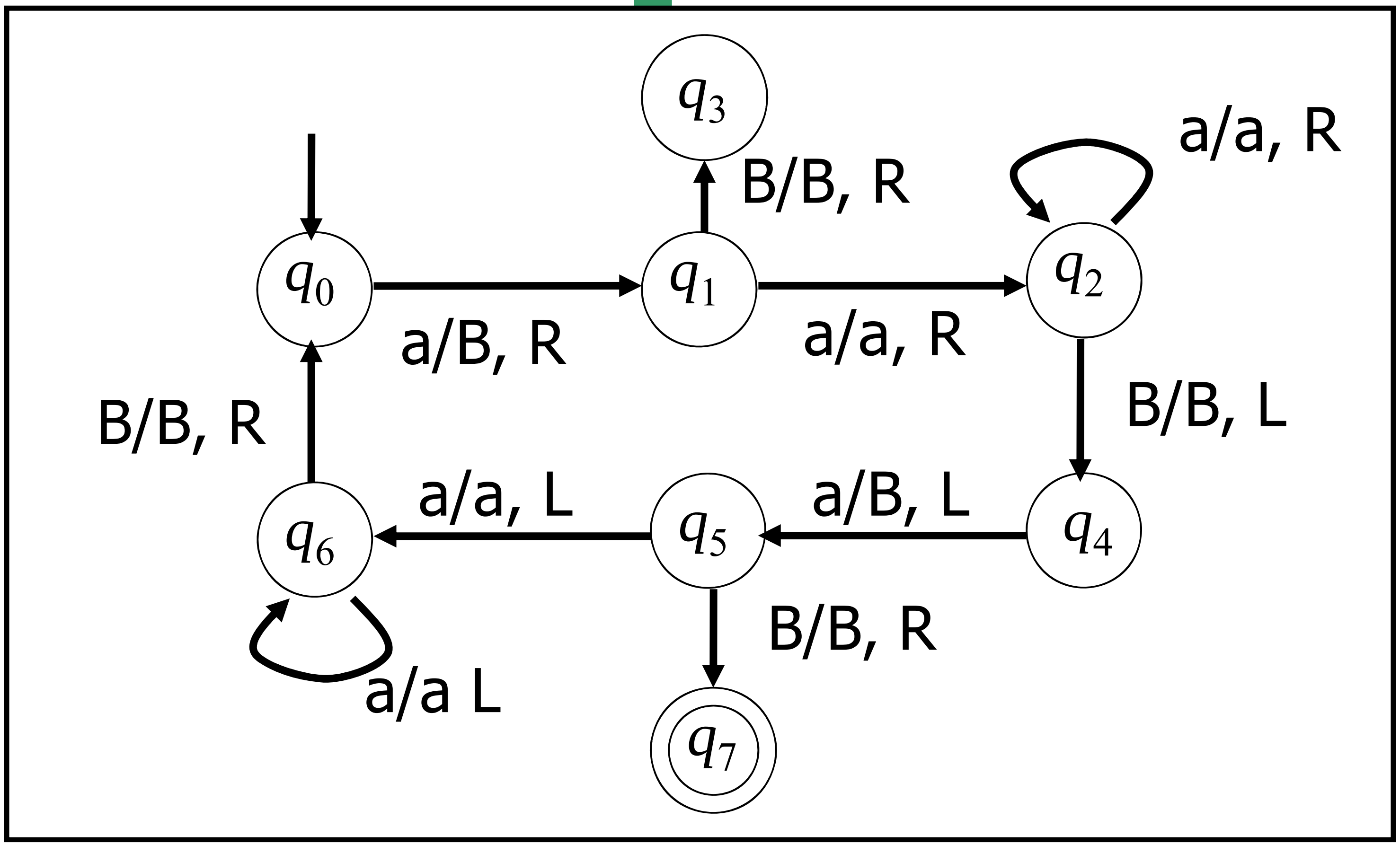


Dividing x by 2 (2-12)

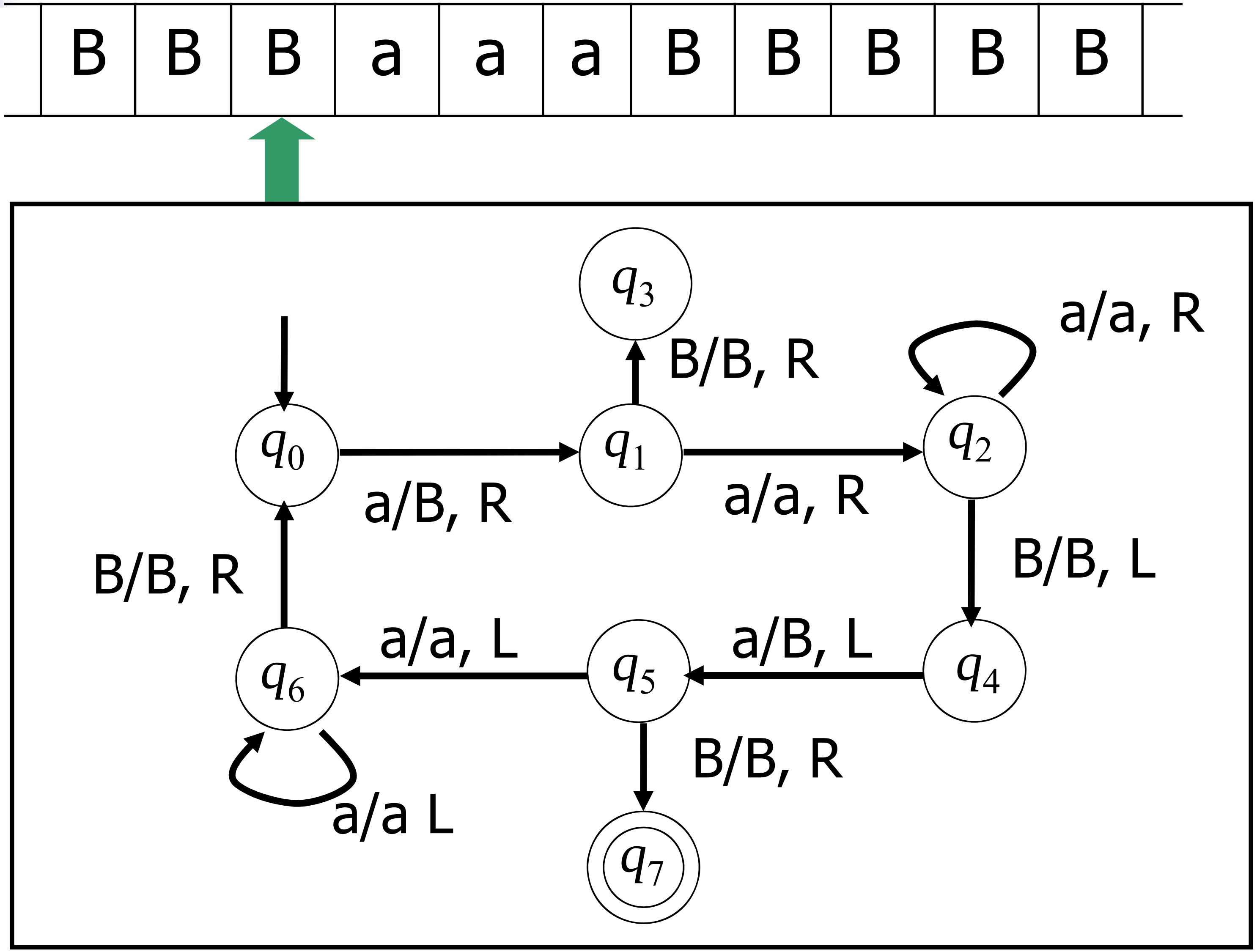


Dividing x by 2 (2-13)

B	B	B	a	a	a	B	B	B	B	B

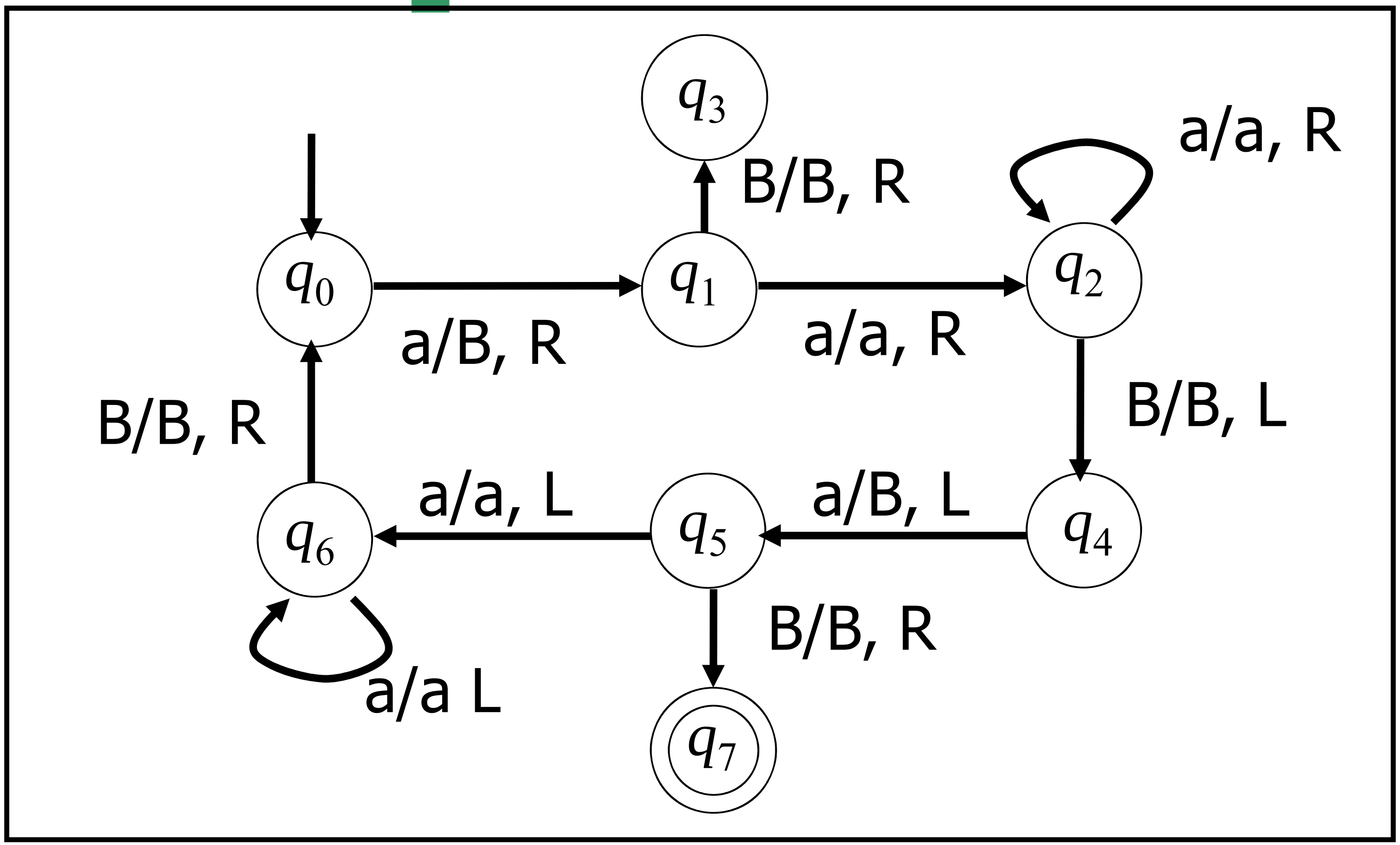


Dividing x by 2 (2-14)



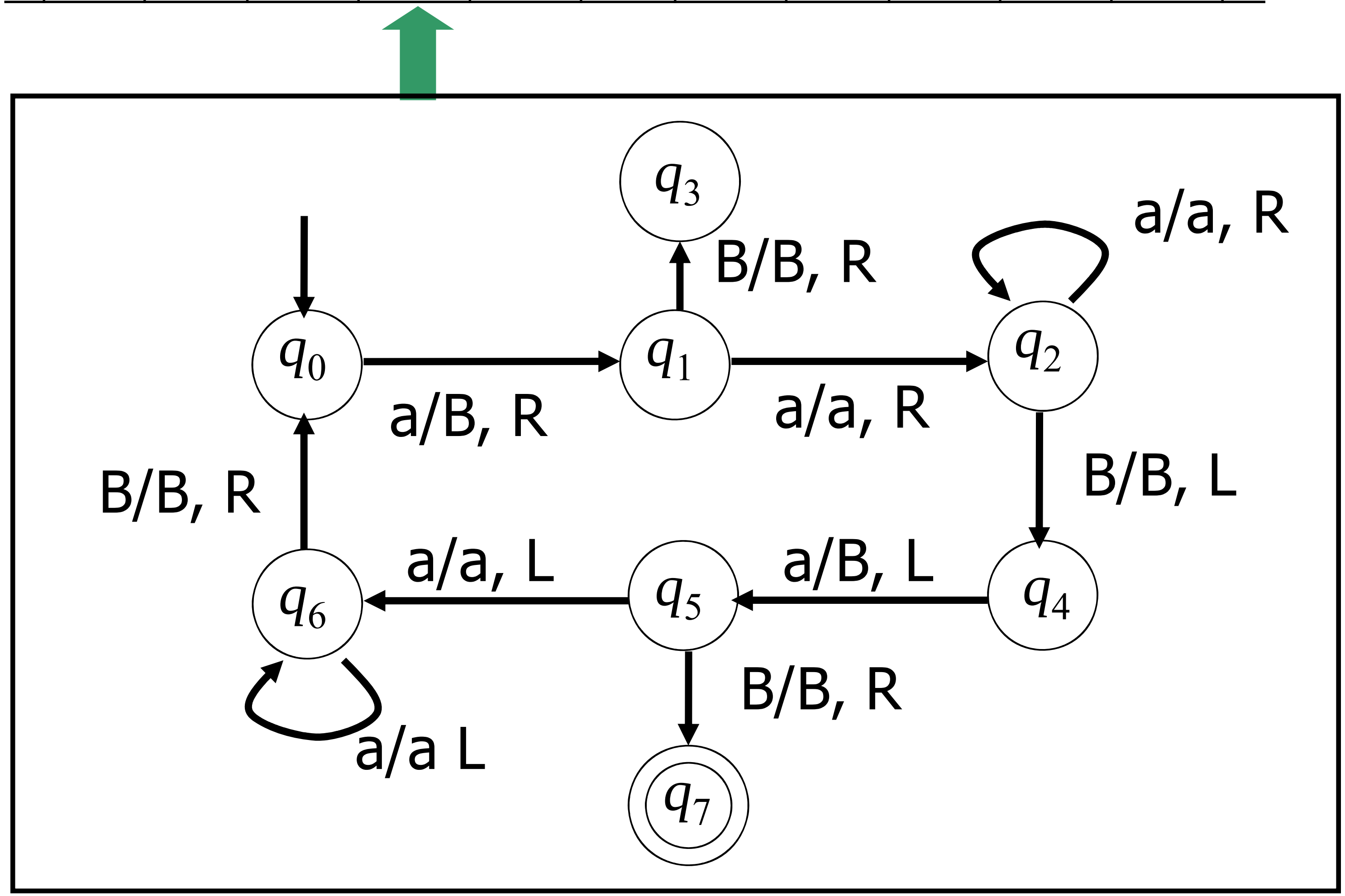
Dividing x by 2 (2-15)

B	B	B	a	a	a	B	B	B	B	B



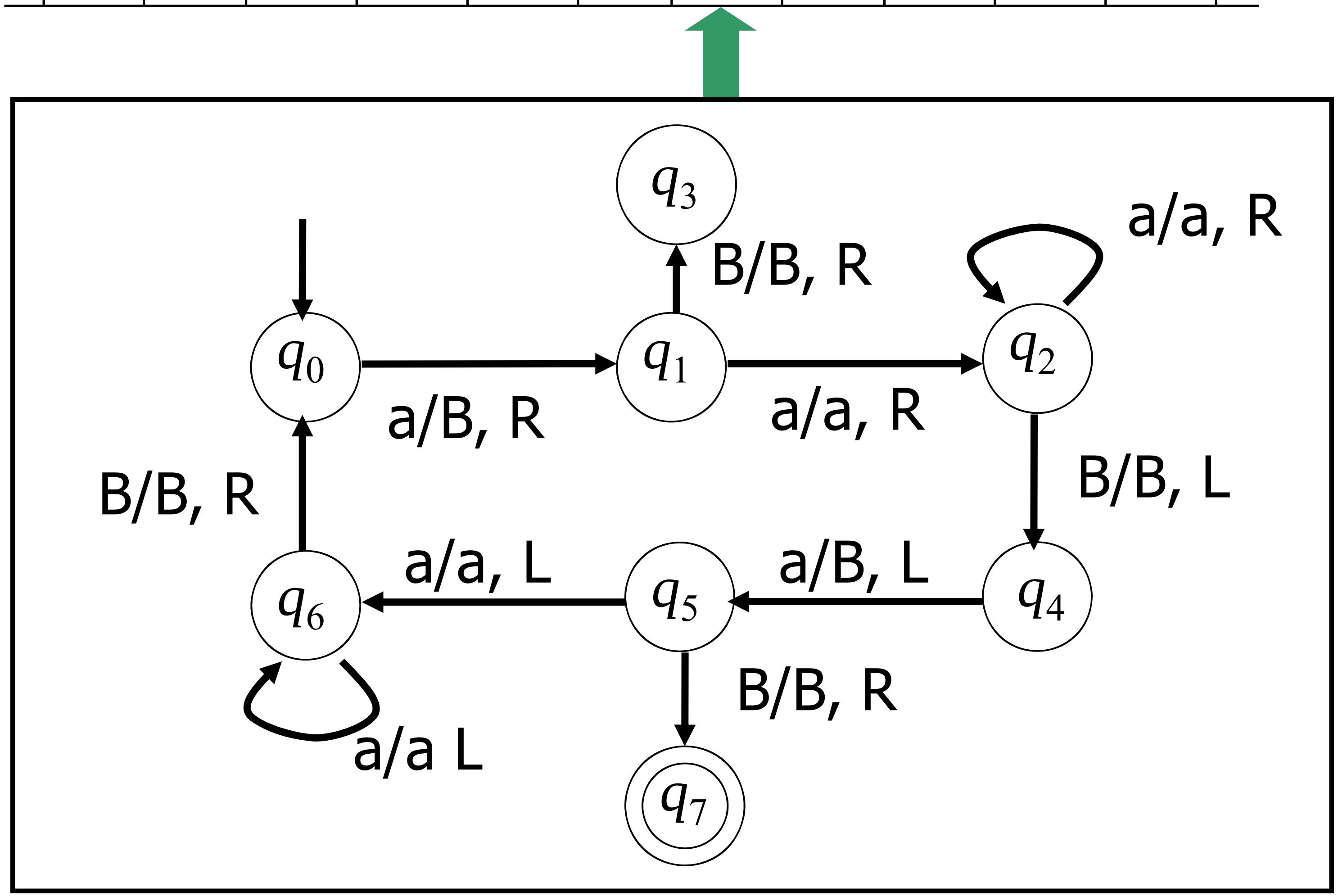
Dividing x by 2 (2-16)

B	B	B	B	a	a	B	B	B	B	B



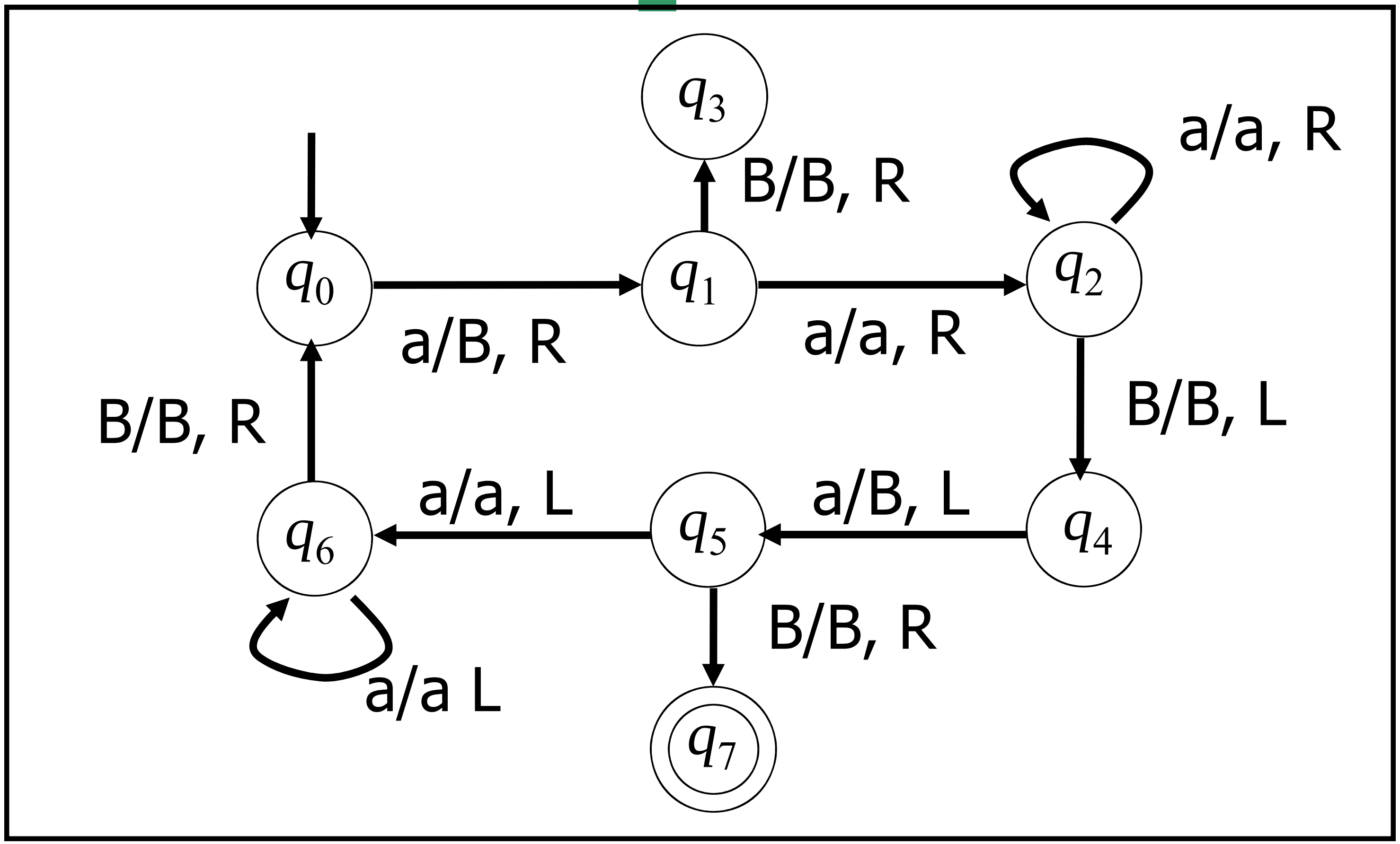
Dividing x by 2 (2-17)

B	B	B	B	a	a	B	B	B	B	B

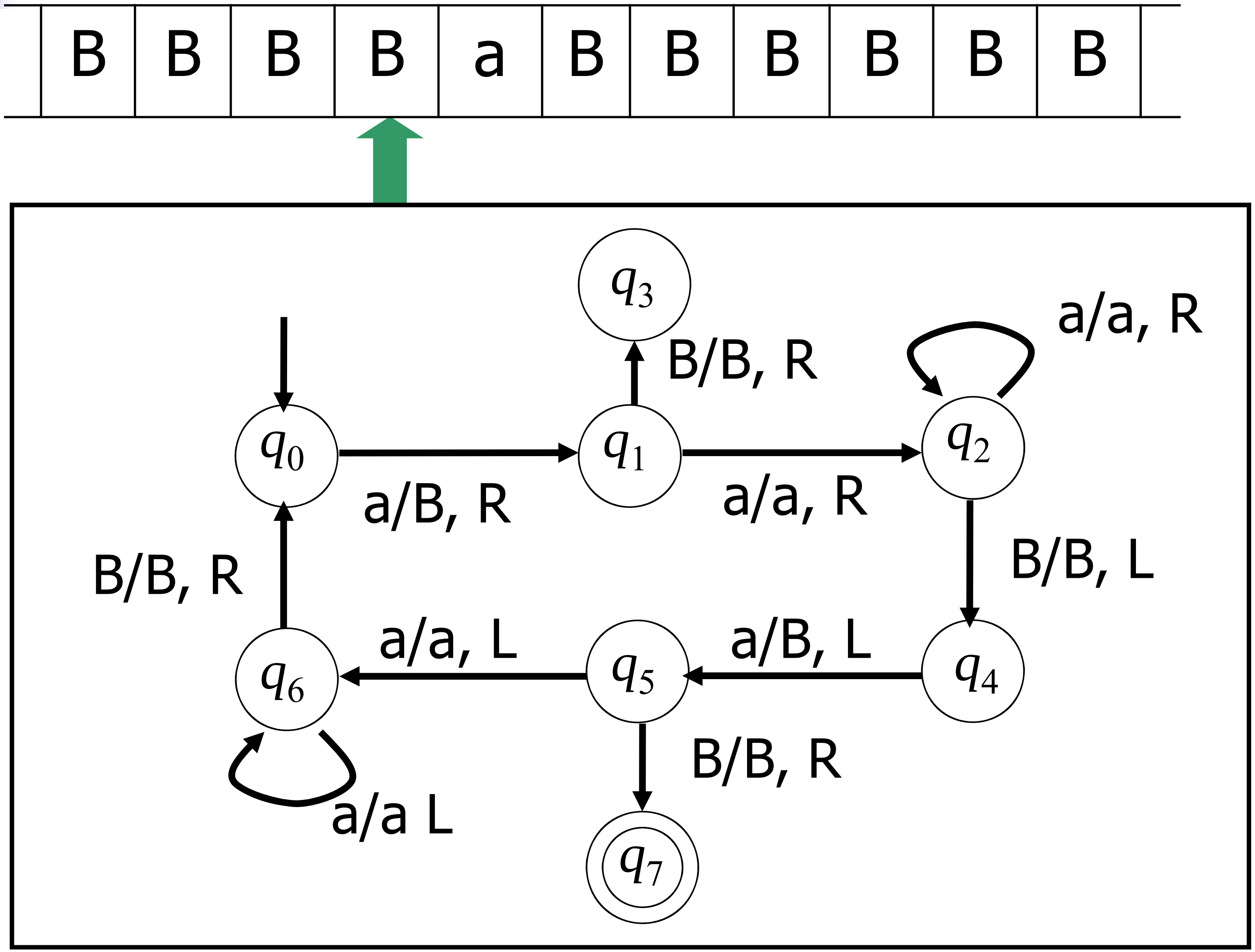


Dividing x by 2 (2-18)

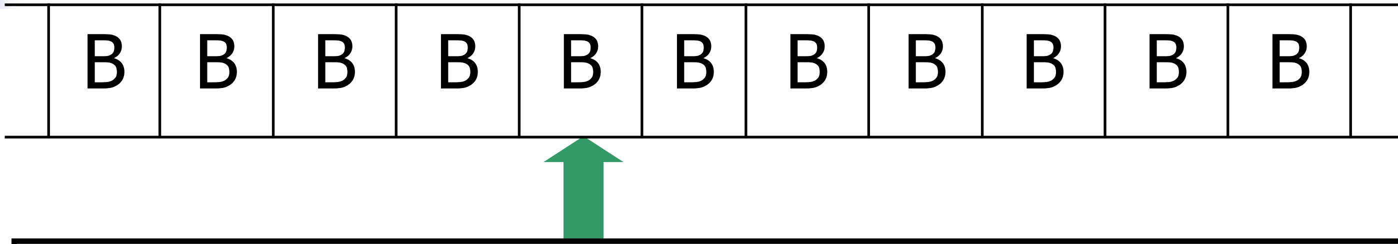
B	B	B	B	a	B	B	B	B	B	B

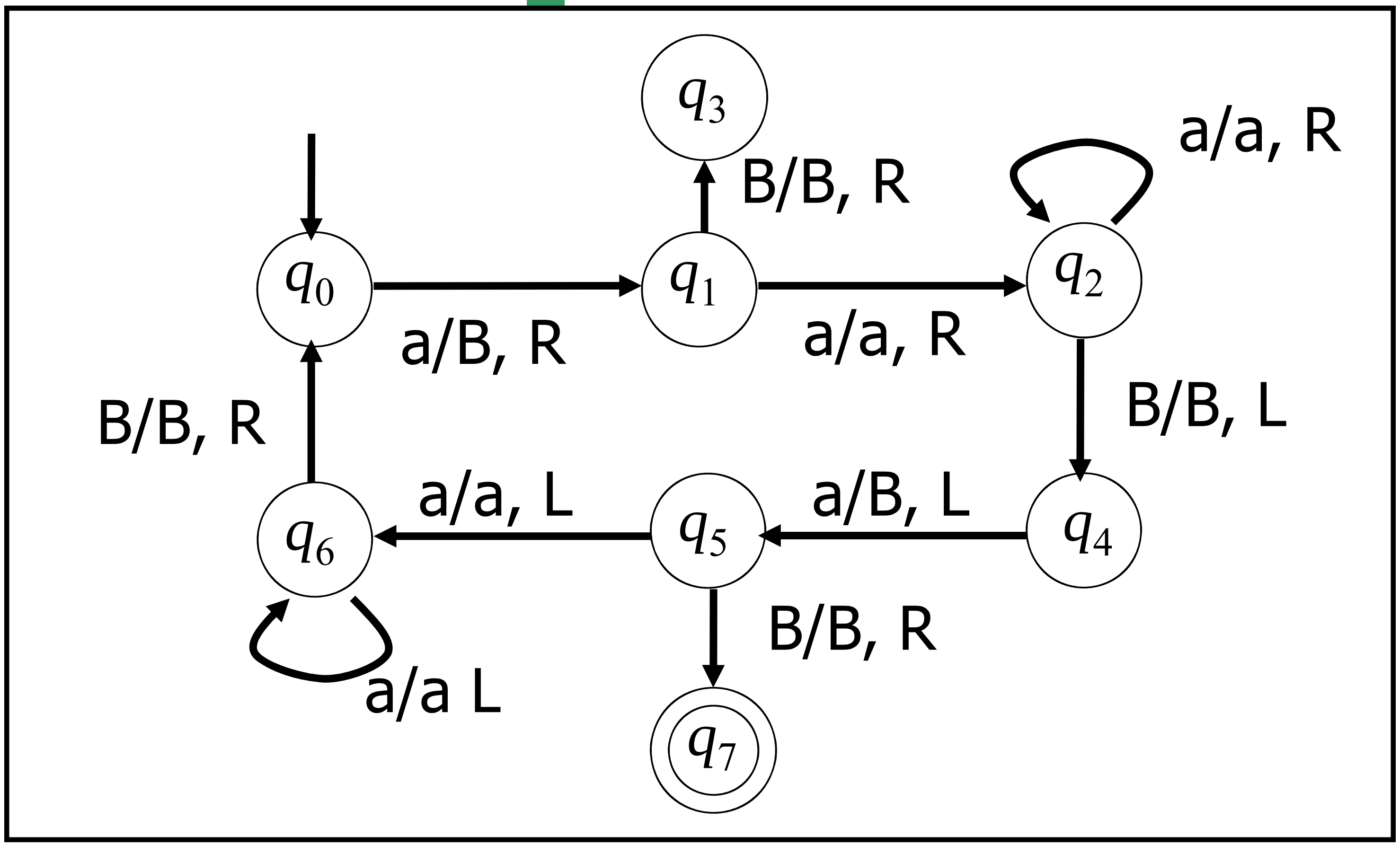


Dividing x by 2 (2-19)



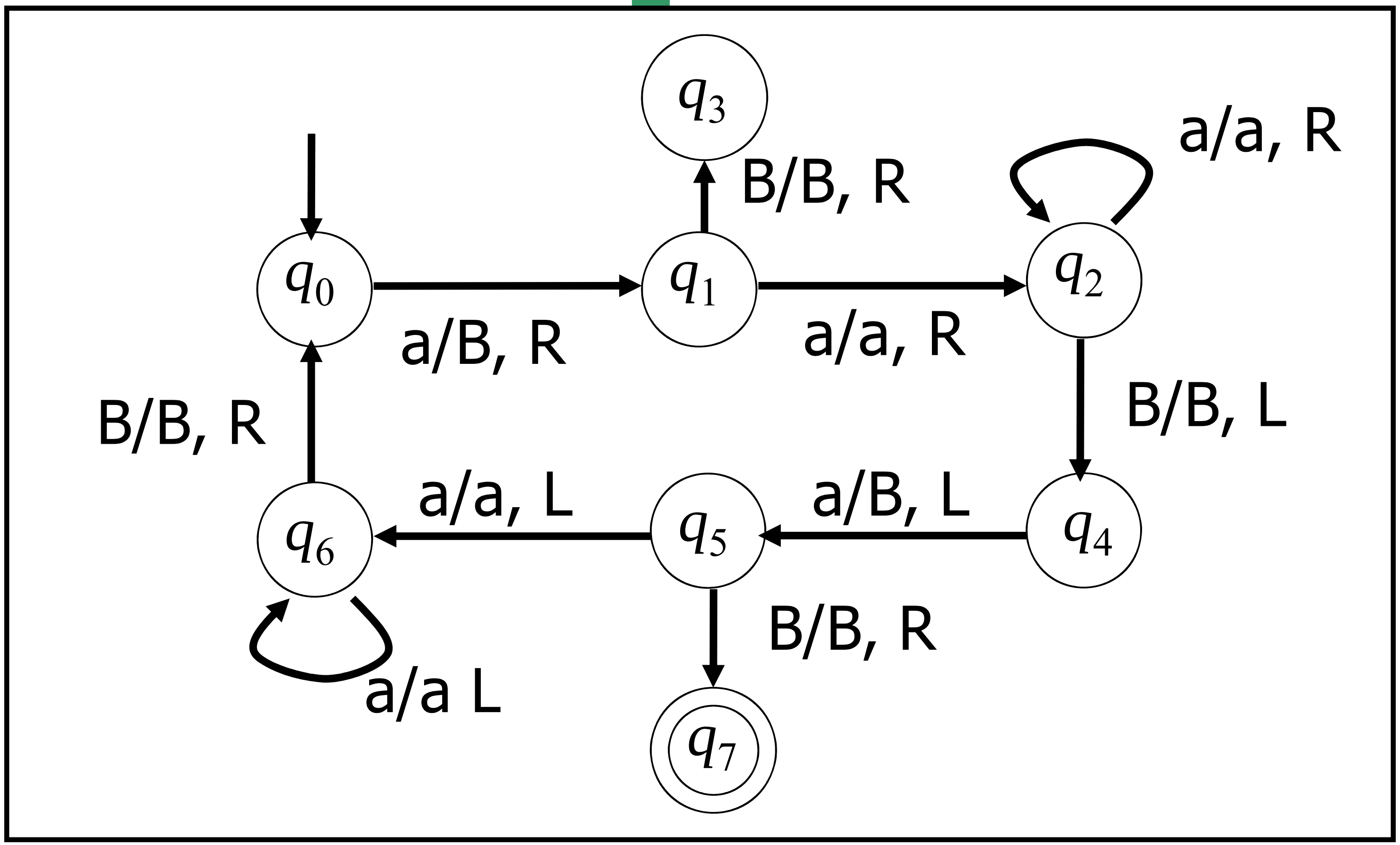
Dividing x by 2 (2-20)





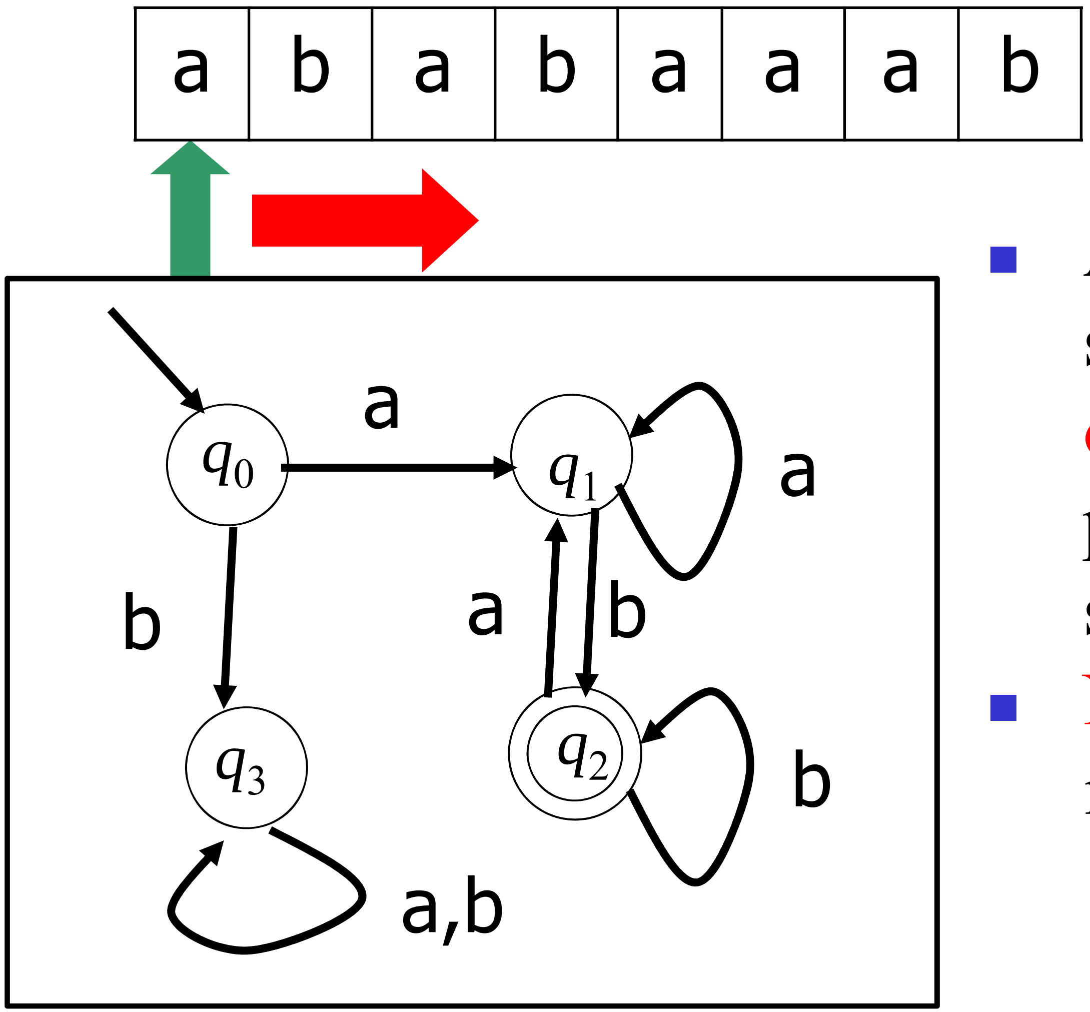
Dividing x by 2 (2-21)

B	B	B	B	B	B	B	B	B	B	B



Strings and Automata

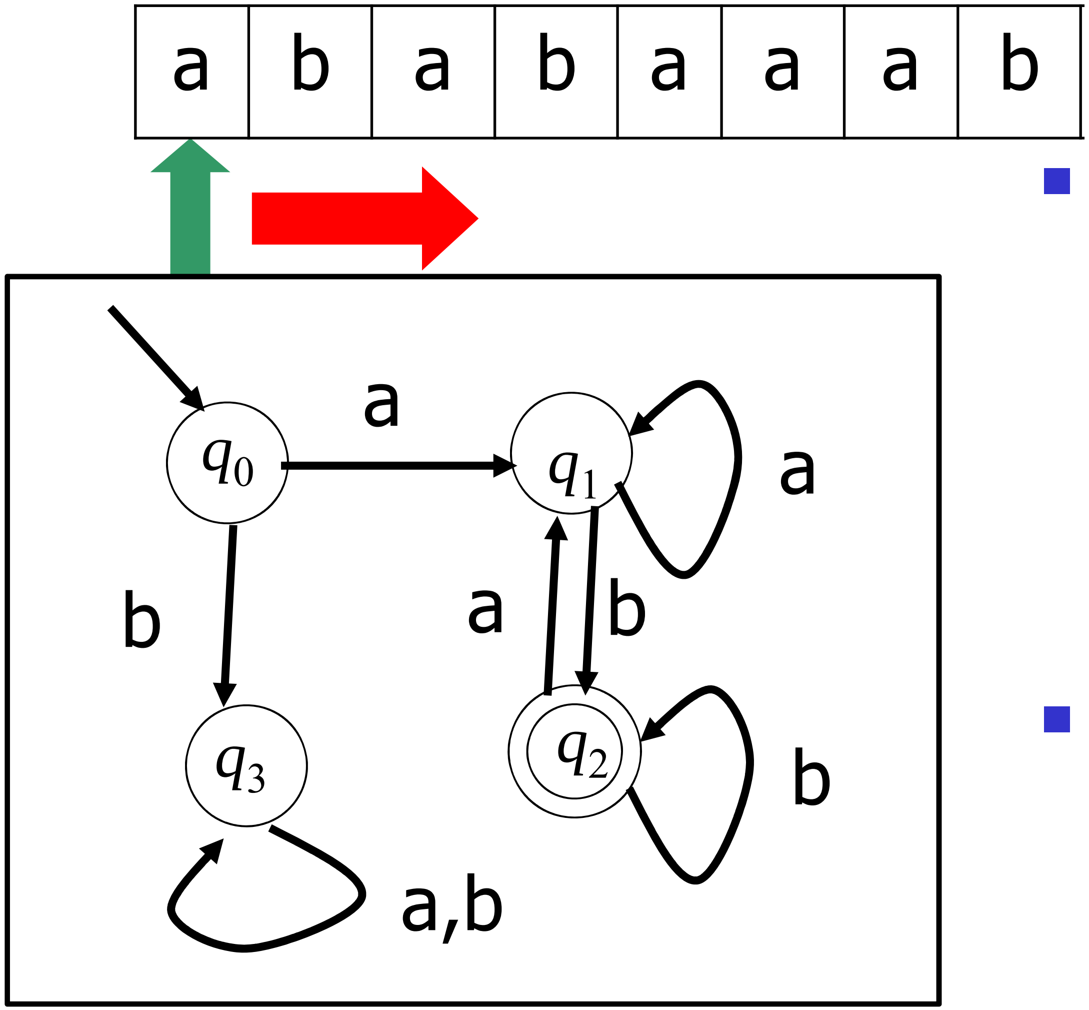
A Simple Instance of TM



- A change of observed squares left to right only, together with a possible change of state of mind.
- No change of symbols in the squares.

Machines of this type are called finite state automata.

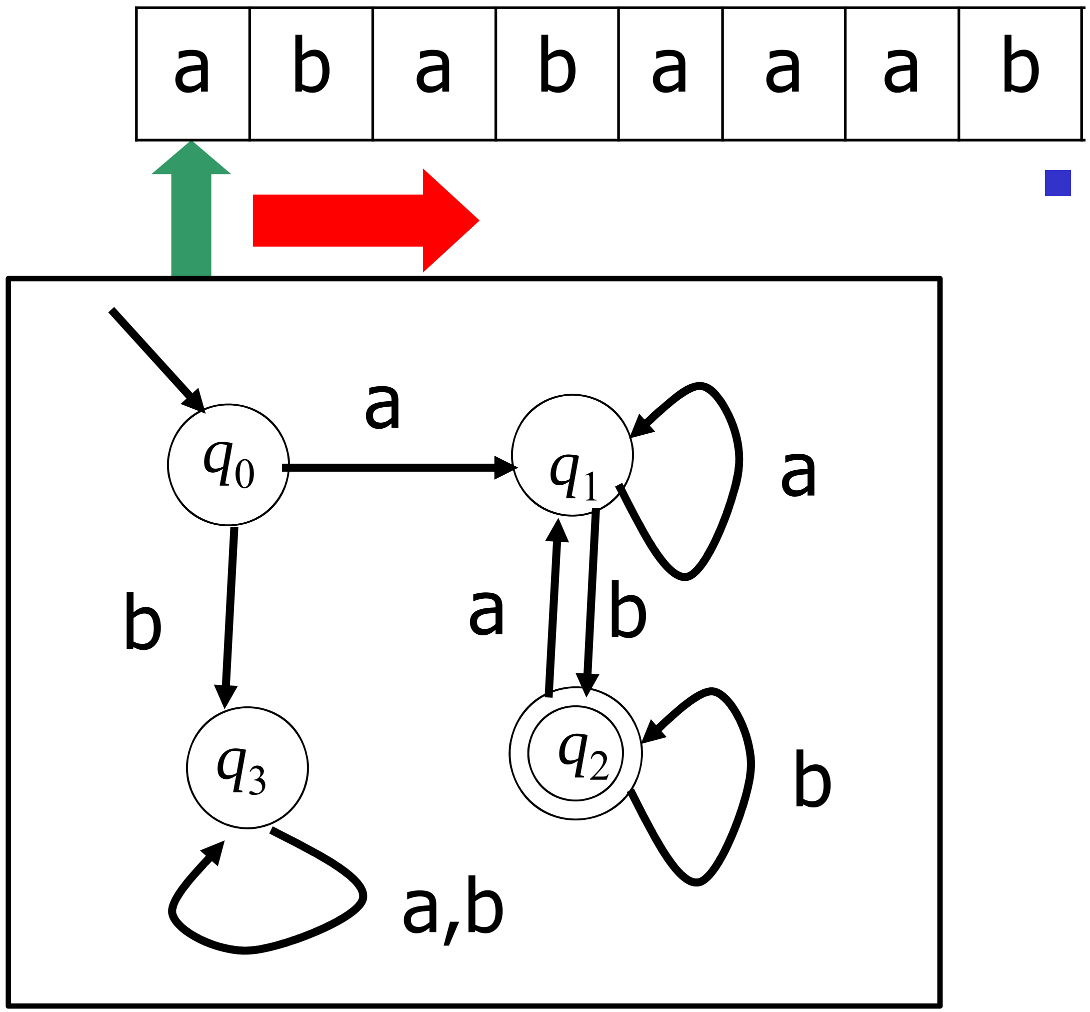
A Simple Restriction



- The input string is accepted by the finite state automaton iff the transition ends at a finial state.
- The set of all strings accepted by the automaton is a formal language.

$L(M)$ = {aab, abb,aaab, aabb,abab, ,... }

A Simple Restriction



- The automaton is represented in the form of a table.

	F	a	b
q_0		q_1	q_3
q_1		q_1	q_2
q_2	v	q_1	q_2
q_3		q_3	q_3

Representation of Finite State Automata

- Mathematically, a finite state automaton is represented in the form $M=(\Sigma, S, \delta, s_0, F)$

  where

  Σ is the alphabet,

  S is a set of states,

  $\delta : S \times \Sigma \to S$ is a transition function represented as a transition table,

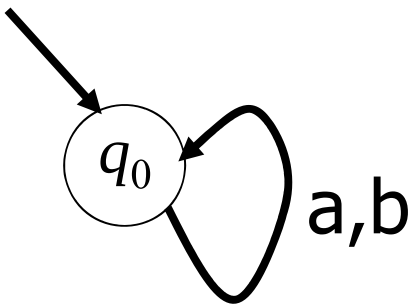
  $q_0 \in S$ is an initial state,

  $F \subset S$ is a set of final states.

	F	a_1	$\ldots$	a_n
q_0				
$\cdots$				
q_m				

Finite Automata of One State

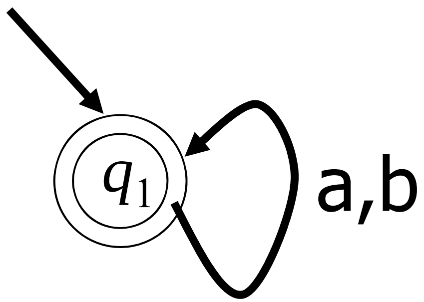
M_0



	F	a	b
q_0		q_0	q_0

$$L(M_0) = \varnothing$$

M_1

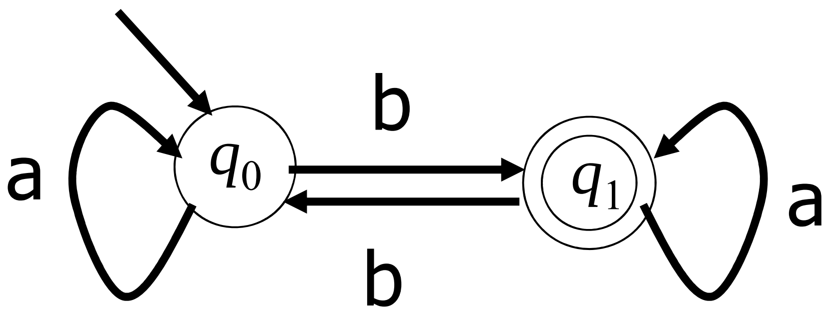


	F	a	b
q_0	v	q_0	q_0

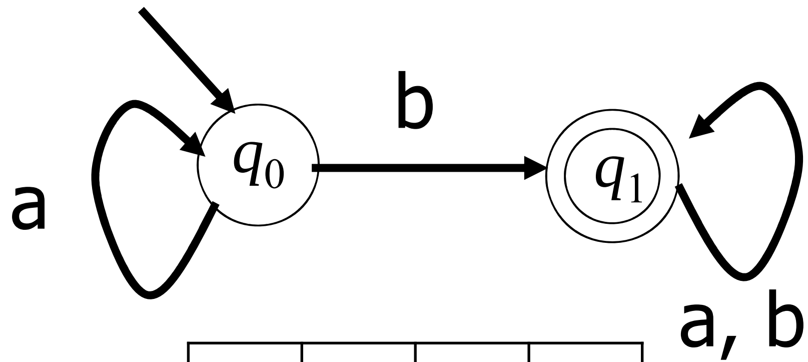
$$L(M_0) = \Sigma^*$$

Exercise 1

- Explain the languages in English accepted by the following FAs.



	F	a	b
q_0		q_0	q_1
q_1	v	q_1	q_0



	F	a	b
q_0		q_0	q_1
q_1	v	q_1	q_1

Exercise 2

- For each of the following languages, give an FA which accepts it.

  $L_1 = \{\ w \mid w$ starts with a and ends with b$\}$

  $L_2 = \{\ w \mid w$ is constructed by repeating ba more than once$\}$

  $L_4 = \{\ w \mid w$ has more than two a's $\}$

Pumping Lemma

Theorem If a language L is accepted by a finite state automaton, there is $N \geq 0$ such that for every string w with $|w| \geq N$ in L can be divided into three strings $w = xyz$ so that

$$y \neq \varepsilon,\ |xz| \leq N, \text{ and } x\underbrace{y\ldots y}_{k}z \in L \text{ for all } k \geq 0.$$

Example

For $L_1 = \{ w \mid w$ starts with a and ends with b$\}$, $N = 3$, and $x =$ a, $z =$ b for every w with $|w| \geq 3$.

For $L_2 = \{ w \mid w$ is constructed by repeating ba more than once$\}$, $N = 4$, $x =$ b, $z =$ a for every w with $|w| \geq 4$.

Application of Pumping Lemma

- No finite automaton accepts the language

$$L_3 = \{ w \mid w = \mathrm{a}^n\mathrm{b}^n \quad n \geq 1 \} .$$

  - c^n means $\underbrace{\mathrm{c}\,\mathrm{c}\ldots\mathrm{c}}_{n}$.