Automate,

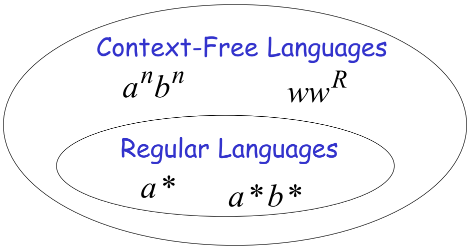
Calculability,

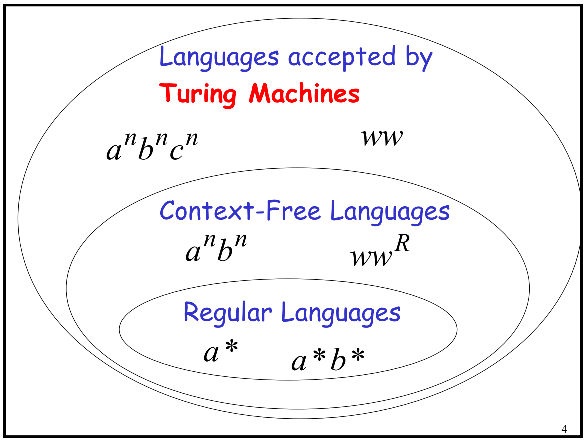
Complexity

Turing Machines

The Language Hierarchy

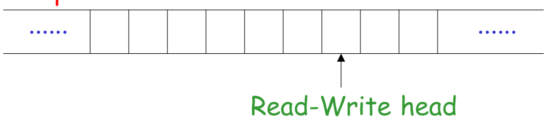
$a^n b^n c^n$ **?** ww **?**



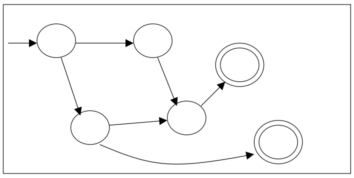


A Turing Machine

Tape

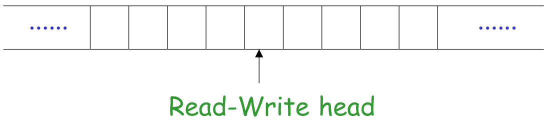


Control Unit

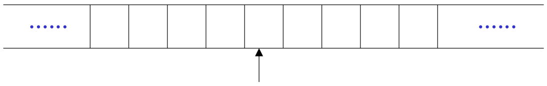


The Tape

No boundaries -- infinite length



The head moves Left or Right



Read-Write head

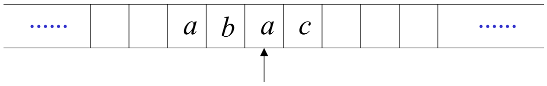
The head at each transition (time step):

1. Reads a symbol
2. Writes a symbol
3. Moves Left or Right

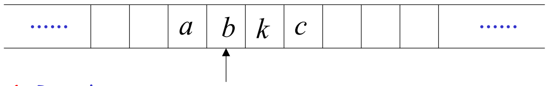
7

Example:

Time 0



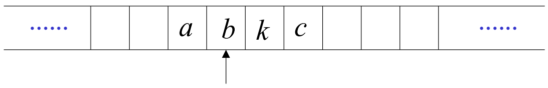
Time 1



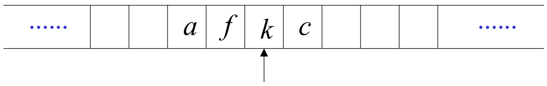
1. Reads a
2. Writes k
3. Moves Left

8

Time 1



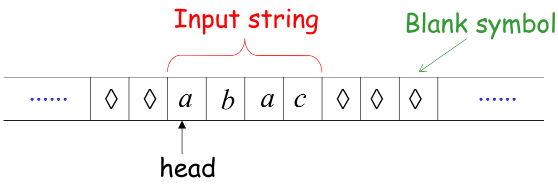
Time 2



1. Reads b
2. Writes f
3. Moves Right

9

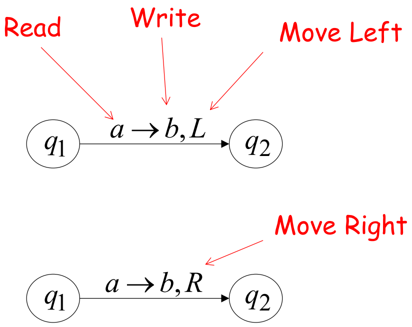
The Input String



Head starts at the leftmost position of the input string

10

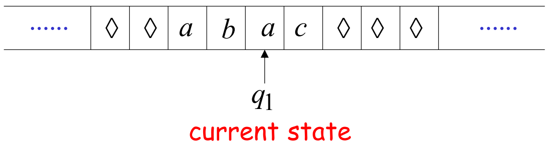
States & Transitions

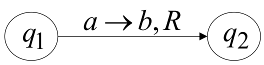


11

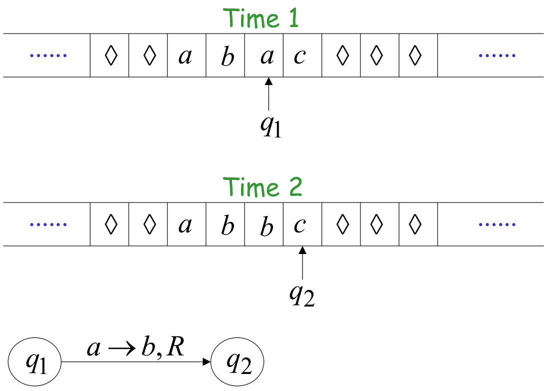
Example:

Time 1



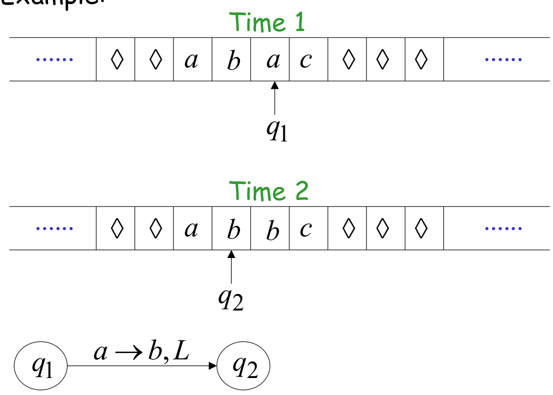


12



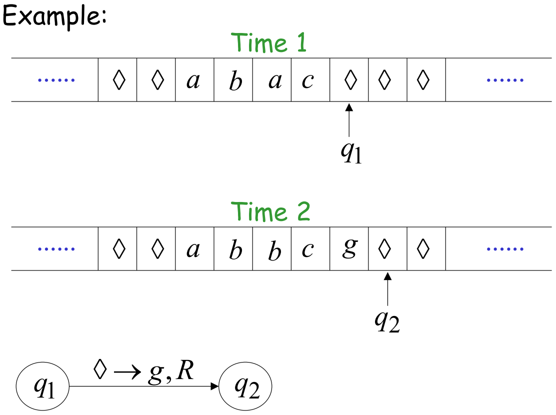
13

Example:



14

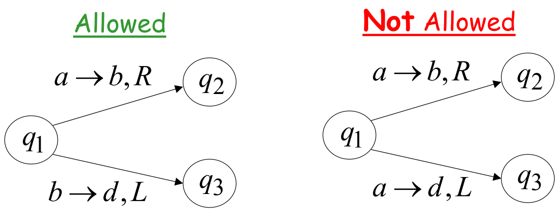
Example:



15

Determinism

Turing Machines are deterministic

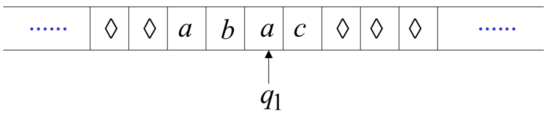


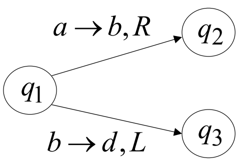
No lambda transitions allowed

16

Partial Transition Function

Example:





Allowed:

No transition
for input symbol c

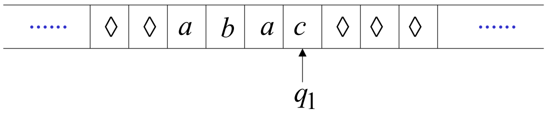
17

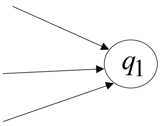
Halting

The machine ***halts*** in a state if there is no transition to follow

18

Halting Example 1:

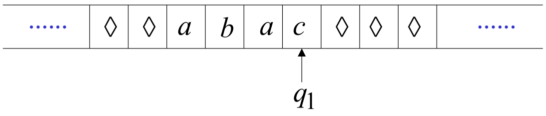


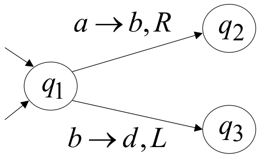


No transition from q_1

HALT!!!

Halting Example 2:



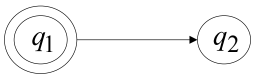


No possible transition from q_1 and symbol c

HALT!!!

Accepting States

Allowed



Not Allowed

- Accepting states have no outgoing transitions
- The machine halts and accepts

Acceptance

Accept Input string ⇒ If machine halts in an accept state

Reject Input string ⇒ If machine halts in a non-accept state
or
If machine enters an *infinite loop*

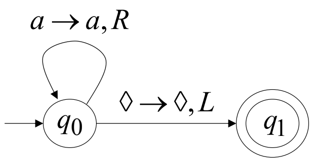
Observation:

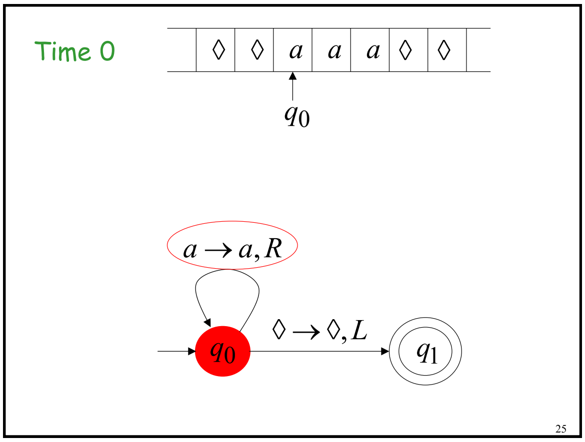
In order to accept an input string, it is not necessary to scan all the symbols in the string

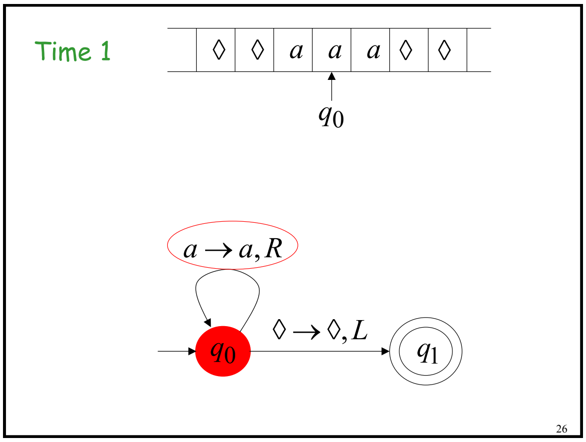
Turing Machine Example

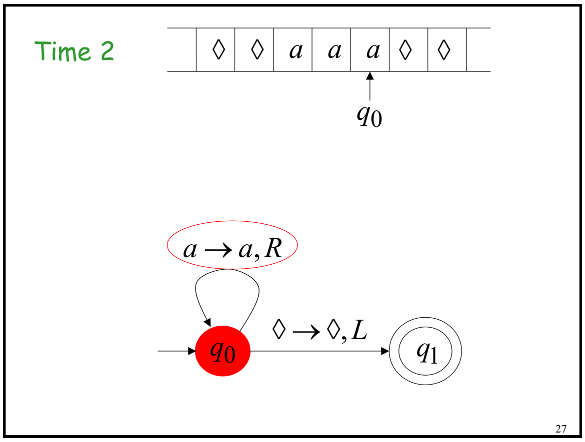
Input alphabet $\Sigma = \{a, b\}$

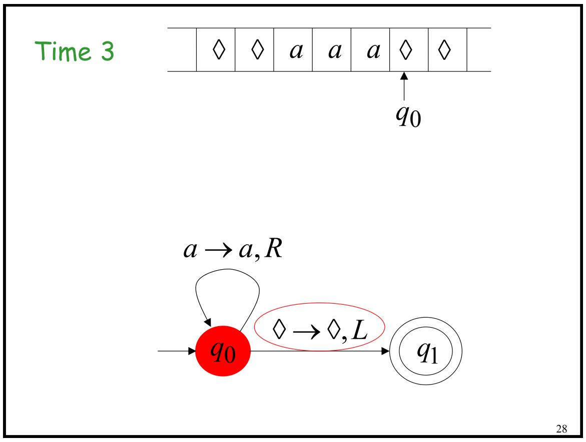
Accepts the language: a^*

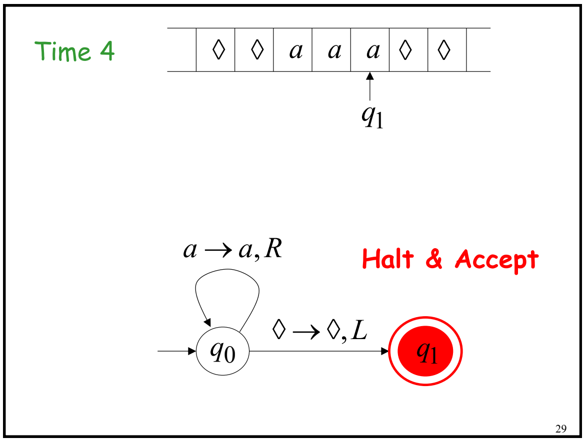


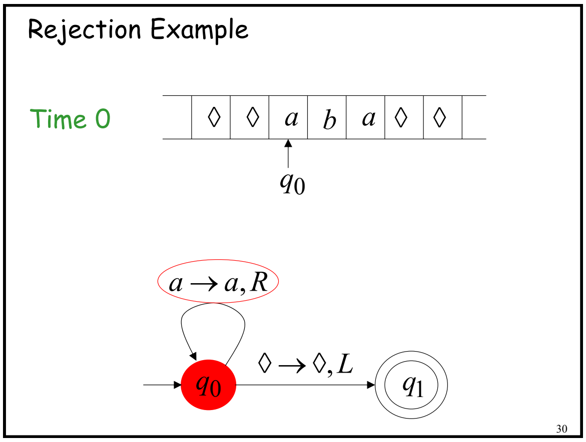

Time 0
◊ ◊ a a a ◊ ◊
q0
a → a, R
◊ → ◊, L
q0
q1
25


Time 1
◊ ◊ a a a ◊ ◊
q0
a → a, R
◊ → ◊, L
q0
q1
26

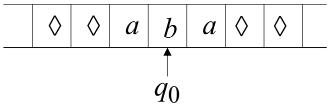

Time 2
◊ ◊ a a a ◊ ◊
q0
a → a, R
◊ → ◊, L
q0
q1
27


Time 3
◊ ◊ a a a ◊ ◊
q0
a → a, R
◊ → ◊, L
q0
q1
28


Time 4
◊ ◊ a a a ◊ ◊
q1
a → a, R
Halt & Accept
◊ → ◊, L
q0
q1
29

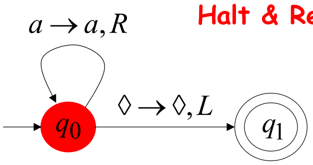

Rejection Example
Time 0
◊ ◊ a b a ◊ ◊
q0
a → a, R
◊ → ◊, L
q0
q1
30

Time 1



No possible Transition

Halt & Reject

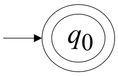


31

A simpler machine for same language

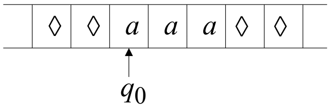
but for input alphabet $\Sigma = \{a\}$

Accepts the language: a^*

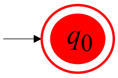


32

Time 0



Halt & Accept



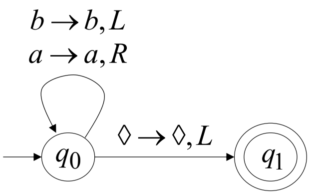
Not necessary to scan input

33

Infinite Loop Example

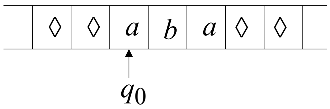
A Turing machine

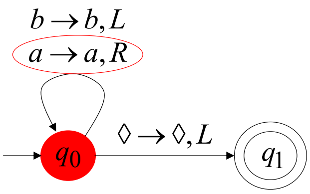
for language $a^* + b(a+b)^*$



34

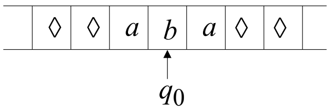
Time 0

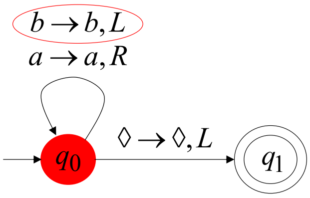




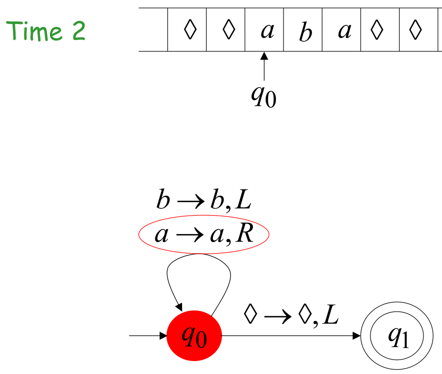
35

Time 1

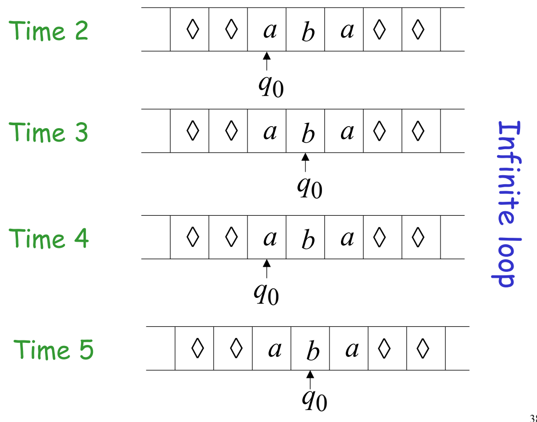




36



37



38

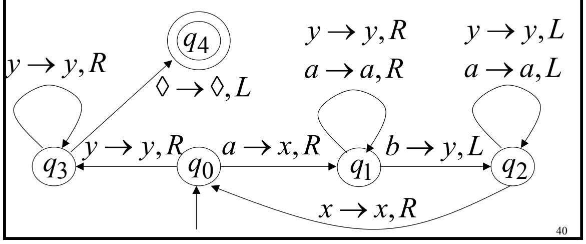
Because of the infinite loop:

- The accepting state cannot be reached
- The machine never halts
- The input string is rejected

39

Another Turing Machine Example

Turing machine for the language $\{a^n b^n\}$ $n \geq 1$



40

Basic Idea:

Match a's with b's:

Repeat:

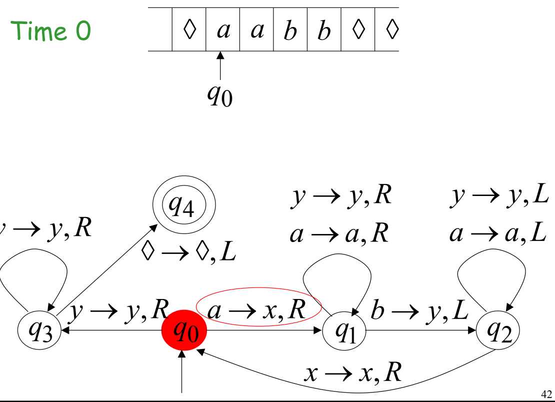
replace leftmost a with x

find leftmost b and replace it with y

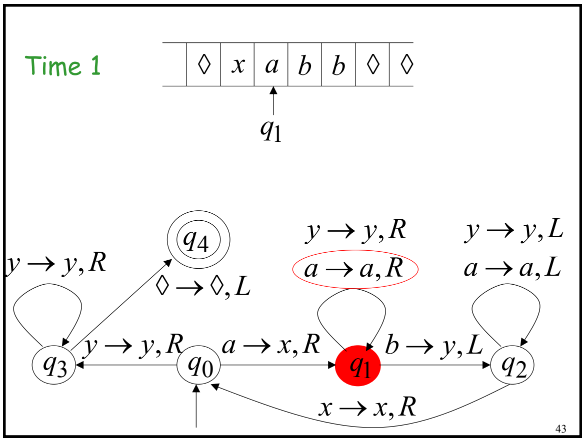
Until there are no more a's or b's

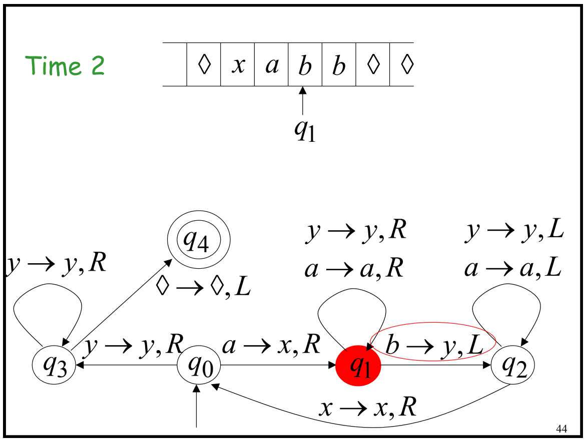
If there is a remaining a or b reject

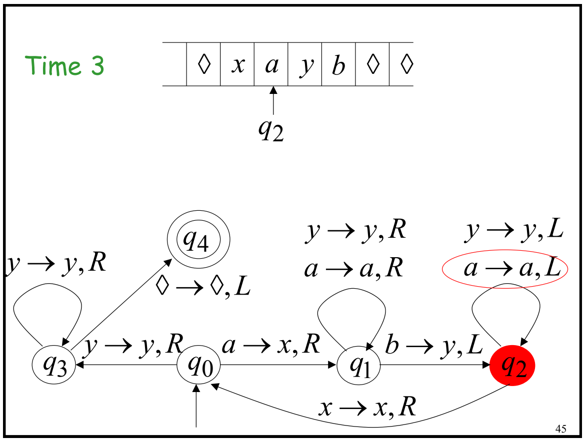
41

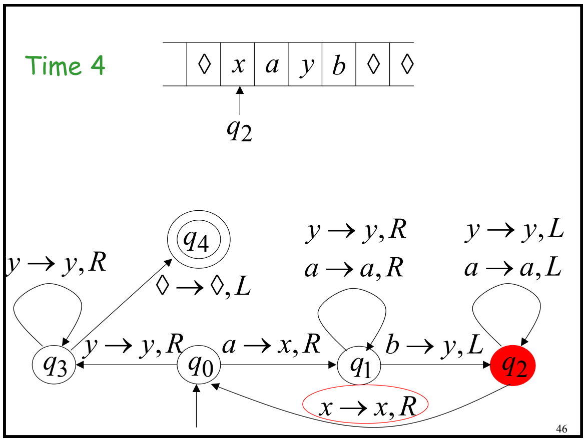


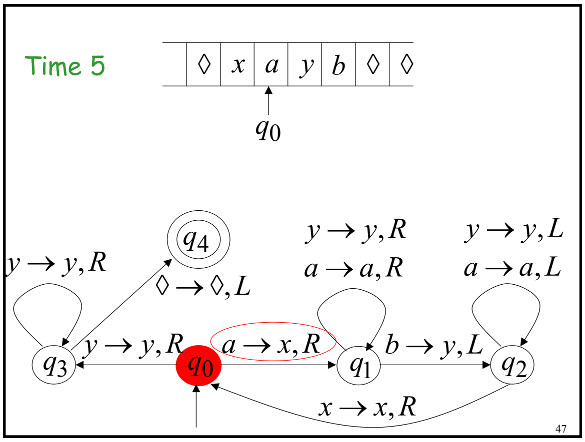
42

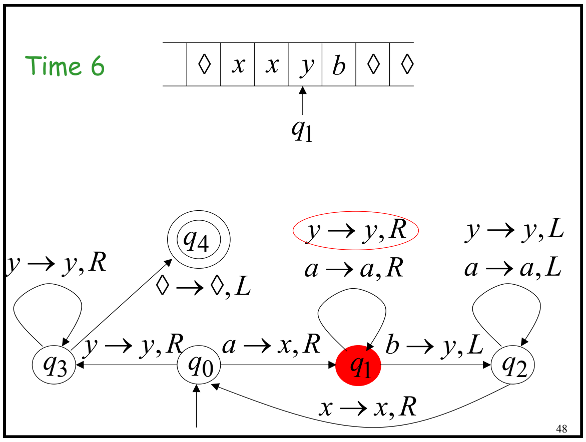

Time 1
◊ x a b b ◊ ◊
q1
q4
y → y, R
◊ → ◊, L
y → y, R
a → a, R
y → y, L
a → a, L
q3 q0 q1 q2
y → y, R
a → x, R
b → y, L
x → x, R
43

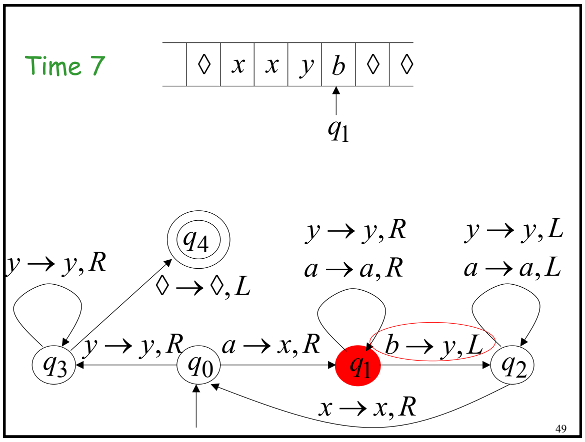

Time 2
◊ x a b b ◊ ◊
q1
q4
y → y, R
◊ → ◊, L
y → y, R
a → a, R
y → y, L
a → a, L
q3 q0 q1 q2
y → y, R
a → x, R
b → y, L
x → x, R
44

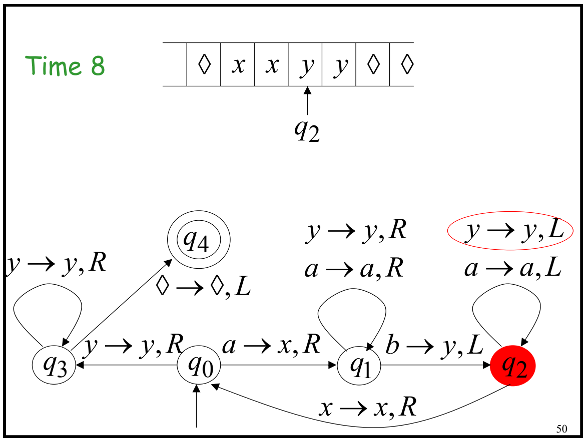

Time 3
◊ x a y b ◊ ◊
q2
q4
y → y, R
◊ → ◊, L
y → y, R
a → a, R
y → y, L
a → a, L
q3 q0 q1 q2
y → y, R
a → x, R
b → y, L
x → x, R
45

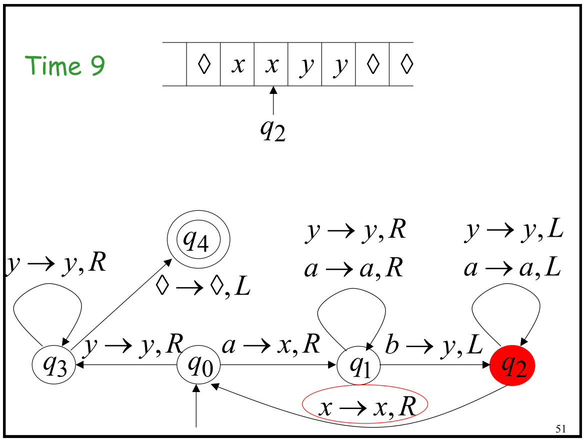

Time 4
◊ x a y b ◊ ◊
q2
q4
y → y, R
◊ → ◊, L
y → y, R
a → a, R
y → y, L
a → a, L
q3 q0 q1 q2
y → y, R
a → x, R
b → y, L
x → x, R
46

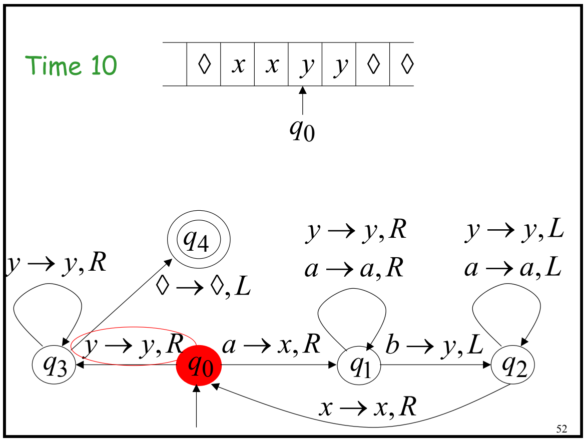

Time 5
◊ x a y b ◊ ◊
q0
q4
y → y, R
◊ → ◊, L
y → y, R
a → a, R
y → y, L
a → a, L
q3 q0 q1 q2
y → y, R
a → x, R
b → y, L
x → x, R
47

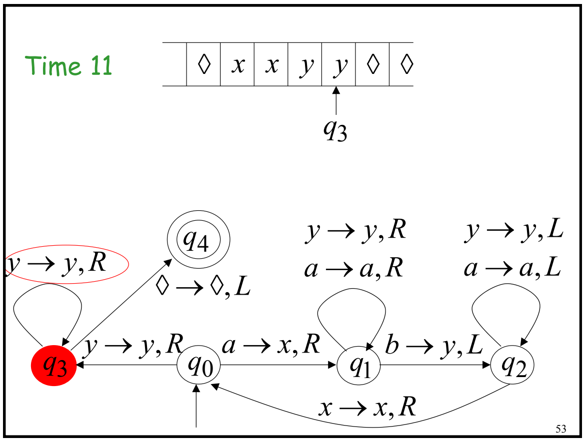

Time 6
◊ x x y b ◊ ◊
q1
q4
y → y, R
◊ → ◊, L
y → y, R
a → a, R
y → y, L
a → a, L
q3 q0 q1 q2
y → y, R
a → x, R
b → y, L
x → x, R
48

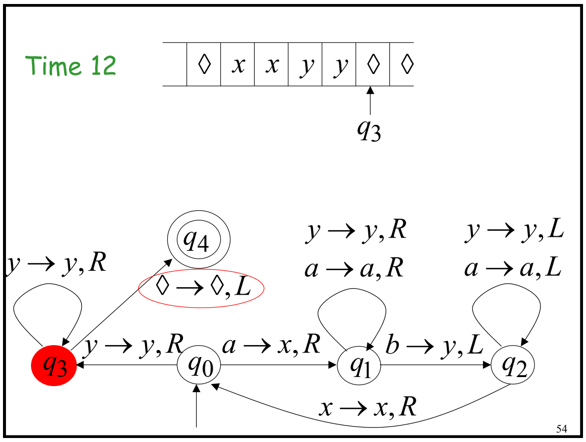

Time 7
◊ x x y b ◊ ◊
q1
q4
y → y, R
a → a, R
y → y, L
a → a, L
y → y, R
◊ → ◊, L
y → y, R
a → x, R
b → y, L
q3 q0 q1 q2
x → x, R
49


Time 8
◊ x x y y ◊ ◊
q2
q4
y → y, R
a → a, R
y → y, L
a → a, L
y → y, R
◊ → ◊, L
y → y, R
a → x, R
b → y, L
q3 q0 q1 q2
x → x, R
50

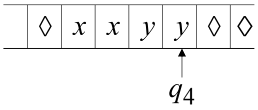

Time 9
◊ x x y y ◊ ◊
q2
q4
y → y, R
a → a, R
y → y, L
a → a, L
y → y, R
◊ → ◊, L
y → y, R
a → x, R
b → y, L
q3 q0 q1 q2
x → x, R
51


Time 10
◊ x x y y ◊ ◊
q0
q4
y → y, R
a → a, R
y → y, L
a → a, L
y → y, R
◊ → ◊, L
y → y, R
a → x, R
b → y, L
q3 q0 q1 q2
x → x, R
52

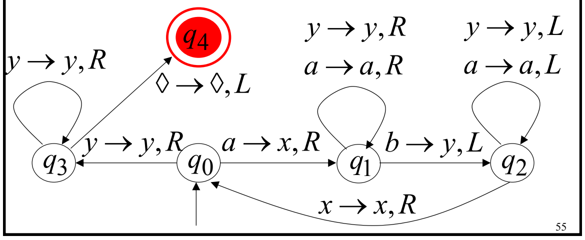

Time 11
◊ x x y y ◊ ◊
q3
q4
y → y, R
a → a, R
y → y, L
a → a, L
y → y, R
◊ → ◊, L
y → y, R
a → x, R
b → y, L
q3 q0 q1 q2
x → x, R
53


Time 12
◊ x x y y ◊ ◊
q3
q4
y → y, R
a → a, R
y → y, L
a → a, L
y → y, R
◊ → ◊, L
y → y, R
a → x, R
b → y, L
q3 q0 q1 q2
x → x, R
54

Time 13



Halt & Accept



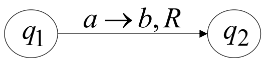
Observation:

If we modify the machine for the language $\{a^n b^n\}$

we can easily construct a machine for the language $\{a^n b^n c^n\}$

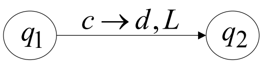
Formal Definitions for Turing Machines

Transition Function



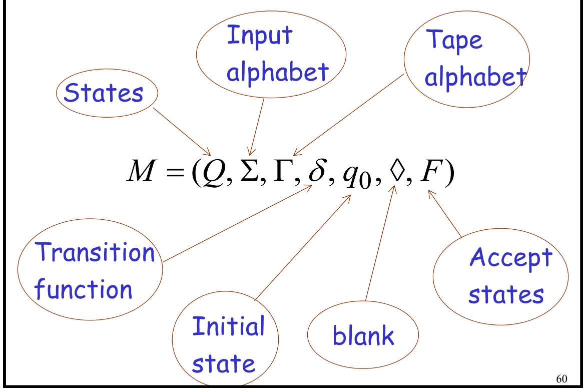
$$\delta(q_1, a) = (q_2, b, R)$$

Transition Function

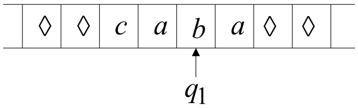


$$\delta(q_1, c) = (q_2, d, L)$$

Turing Machine:



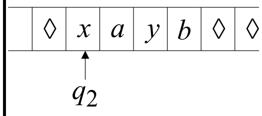
Configuration



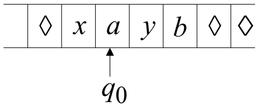
Instantaneous description: $ca\ q_1\ ba$

61

Time 4



Time 5

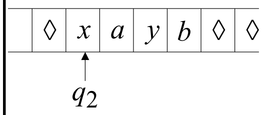


A Move: $q_2\ xayb \succ x\ q_0\ ayb$

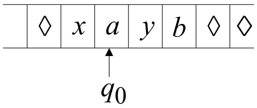
(yields in one mode)

62

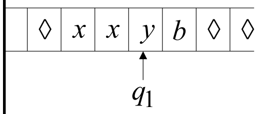
Time 4



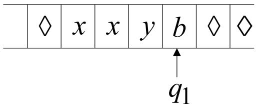
Time 5



Time 6



Time 7



A computation

$$q_2\ xayb \succ x\ q_0\ ayb \succ xx\ q_1\ yb \succ xxy\ q_1\ b$$

63

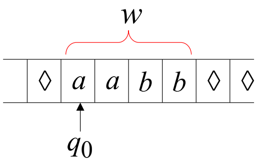
$$q_2\ xayb \succ x\ q_0\ ayb \succ xx\ q_1\ yb \succ xxy\ q_1\ b$$

Equivalent notation: $q_2\ xayb \overset{*}{\succ} xxy\ q_1\ b$

64

Initial configuration: $q_0\ w$

Input string



65

The Accepted Language

For any Turing Machine M

$$L(M) = \{w :\ \ q_0\ w \overset{*}{\succ} x_1\ q_f\ x_2\}$$

Initial state

Accept state

66

If a language L is accepted
by a Turing machine M
then we say that L is:

- Turing Recognizable

Other names used:

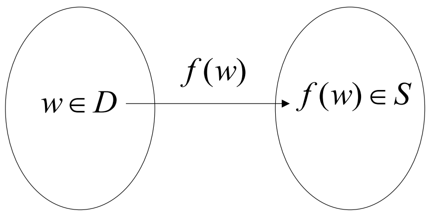
- Turing Acceptable
- Recursively Enumerable

Computing Functions with Turing Machines

A function $f(w)$ has:

Domain: D

Result Region: S



A function may have many parameters:

Example: Addition function

$$f(x, y) = x + y$$

Integer Domain

Decimal: 5

Binary: 101

Unary: 11111

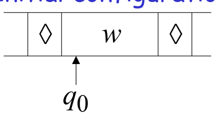
We prefer **unary** representation:

easier to manipulate with Turing machines

Definition:

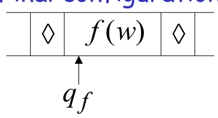
A function f is computable if
there is a Turing Machine M such that:

Initial configuration



initial state

Final configuration



accept state

For all $w \in D$ Domain

In other words:

A function f is computable if
there is a Turing Machine M such that:

$$q_0\, w \overset{*}{\succ} q_f\, f(w)$$

Initial Configuration — Final Configuration

For all $w \in D$ Domain

Example

The function $f(x,y)=x+y$ is computable

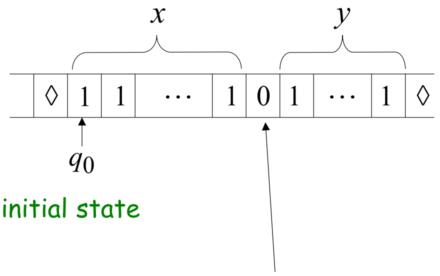
x, y are integers

Turing Machine:

Input string: $x0y$ unary

Output string: $xy0$ unary

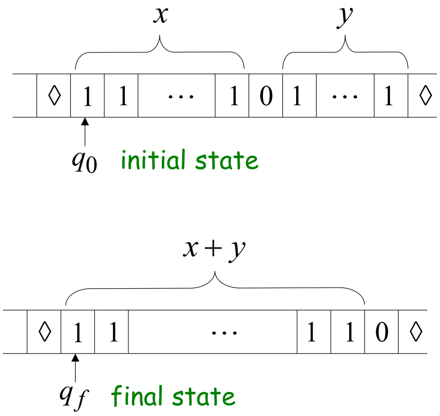
Start



initial state

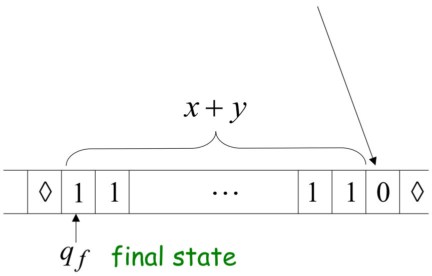
The 0 is the delimiter that
separates the two numbers

Start



Finish

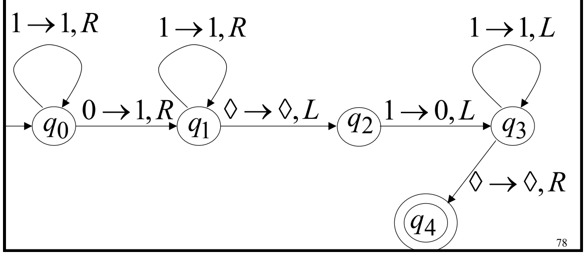
The 0 here helps when we use
the result for other operations



Finish

final state

Turing machine for function $f(x,y)=x+y$



Execution Example:

Time 0

$x = 11$ (=2)

$y = 11$ (=2)

	x			y	
◊	1	1	0	1	1	◊

$\uparrow$ q_0

Final Result

$x + y$

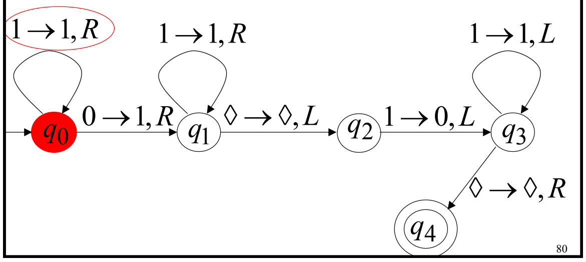
◊	1	1	1	1	0	◊

$\uparrow$ q_4

Time 0

◊	1	1	0	1	1	◊

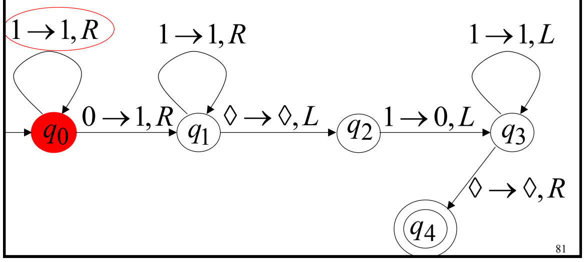
$\uparrow$ q_0



Time 1

◊	1	1	0	1	1	◊

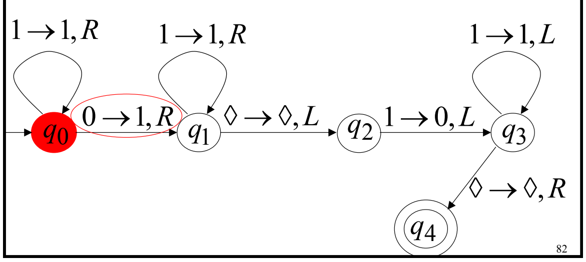
$\uparrow$ q_0



Time 2

◊	1	1	0	1	1	◊

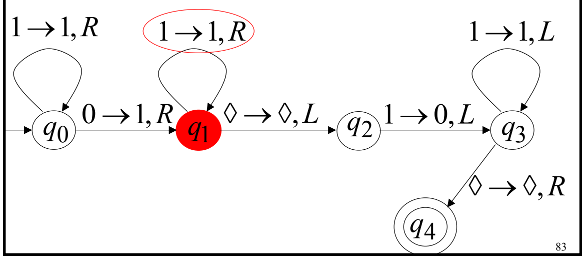
$\uparrow$ q_0



Time 3

◊	1	1	1	1	1	◊

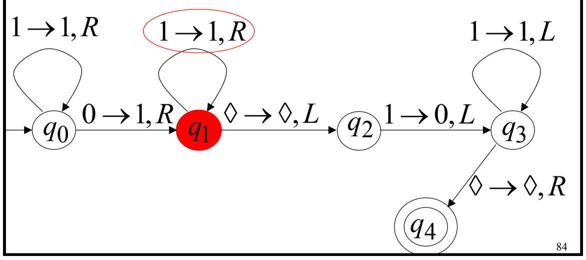
$\uparrow$ q_1

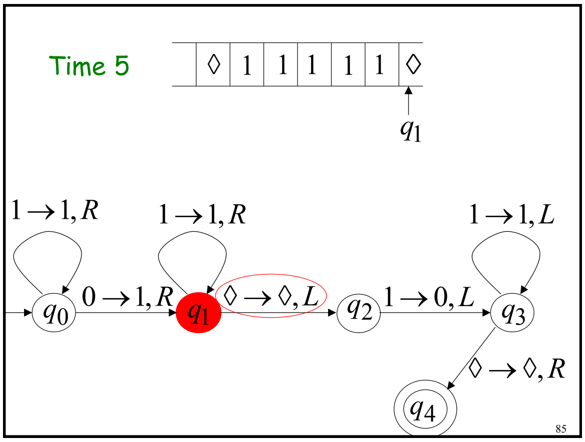


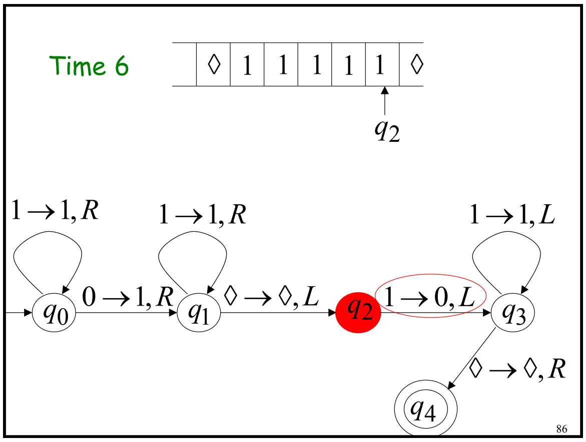
Time 4

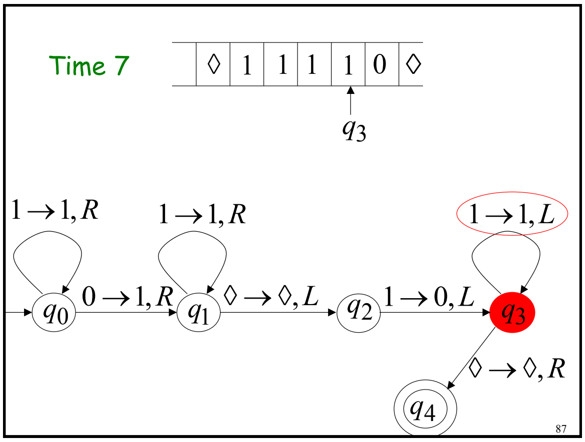
◊	1	1	1	1	1	◊

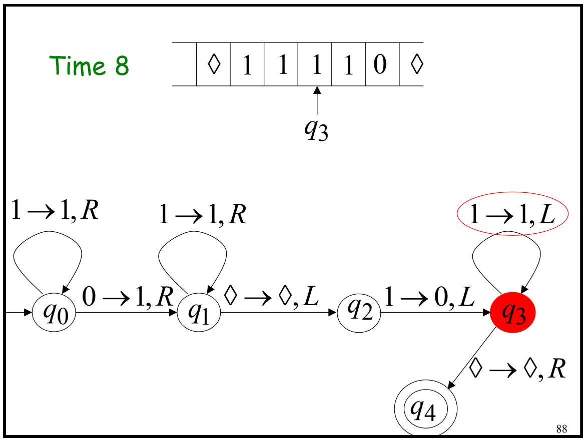
$\uparrow$ q_1

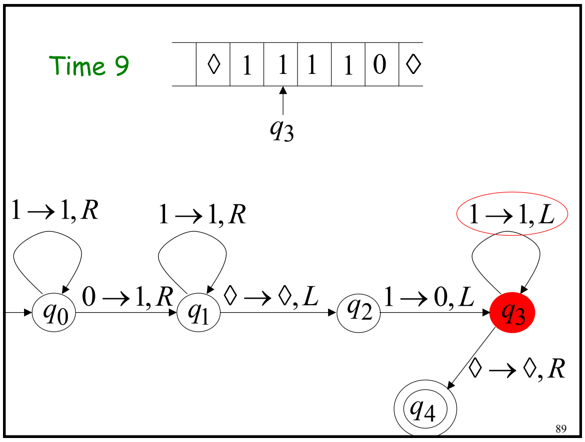


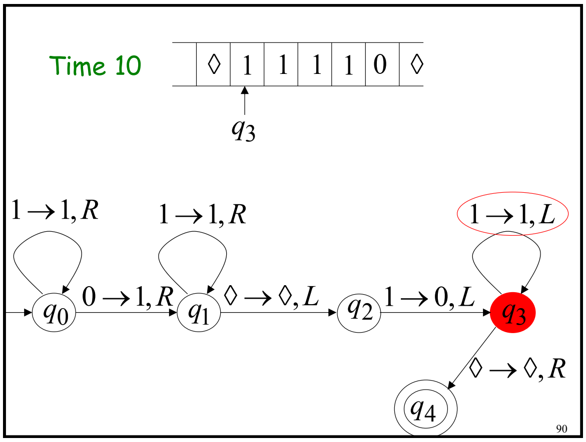

Time 5
◊ 1 1 1 1 1 ◊
q_1
$1 \to 1, R$
$1 \to 1, R$
$1 \to 1, L$
$0 \to 1, R$
$◊ \to ◊, L$
$1 \to 0, L$
$◊ \to ◊, R$
q_0 q_1 q_2 q_3 q_4
85


Time 6
◊ 1 1 1 1 1 ◊
q_2
$1 \to 1, R$
$1 \to 1, R$
$1 \to 1, L$
$0 \to 1, R$
$◊ \to ◊, L$
$1 \to 0, L$
$◊ \to ◊, R$
q_0 q_1 q_2 q_3 q_4
86

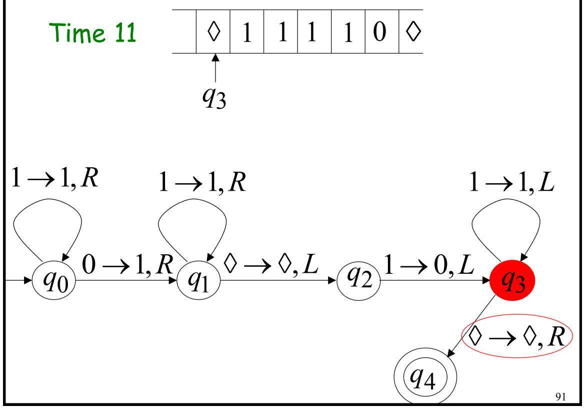

Time 7
◊ 1 1 1 1 0 ◊
q_3
$1 \to 1, R$
$1 \to 1, R$
$1 \to 1, L$
$0 \to 1, R$
$◊ \to ◊, L$
$1 \to 0, L$
$◊ \to ◊, R$
q_0 q_1 q_2 q_3 q_4
87


Time 8
◊ 1 1 1 1 0 ◊
q_3
$1 \to 1, R$
$1 \to 1, R$
$1 \to 1, L$
$0 \to 1, R$
$◊ \to ◊, L$
$1 \to 0, L$
$◊ \to ◊, R$
q_0 q_1 q_2 q_3 q_4
88

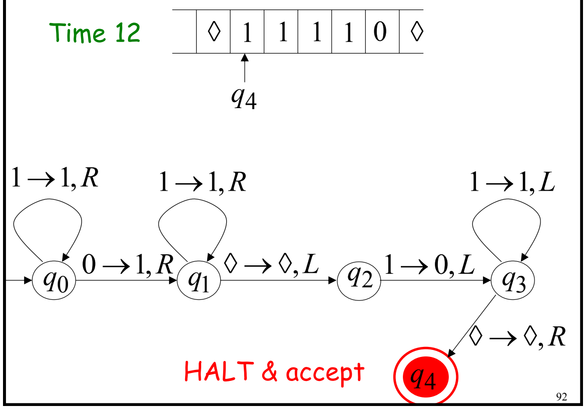

Time 9
◊ 1 1 1 1 0 ◊
q_3
$1 \to 1, R$
$1 \to 1, R$
$1 \to 1, L$
$0 \to 1, R$
$◊ \to ◊, L$
$1 \to 0, L$
$◊ \to ◊, R$
q_0 q_1 q_2 q_3 q_4
89


Time 10
◊ 1 1 1 1 0 ◊
q_3
$1 \to 1, R$
$1 \to 1, R$
$1 \to 1, L$
$0 \to 1, R$
$◊ \to ◊, L$
$1 \to 0, L$
$◊ \to ◊, R$
q_0 q_1 q_2 q_3 q_4
90

Time 11



Time 12



Another Example

The function $f(x) = 2x$ is computable

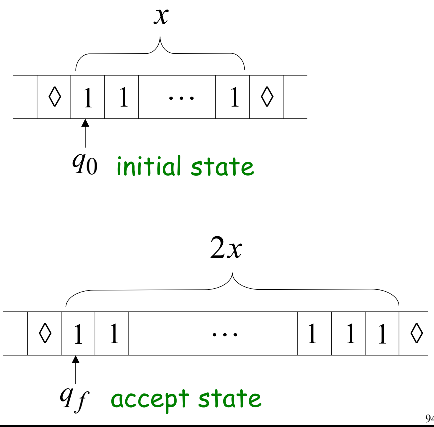
x is integer

Turing Machine:

Input string: x unary

Output string: xx unary

Start



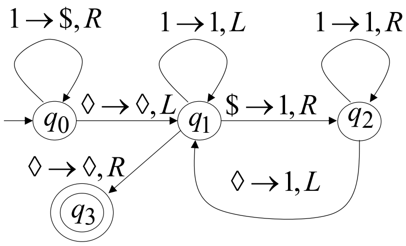
Finish

Turing Machine Pseudocode for $f(x) = 2x$

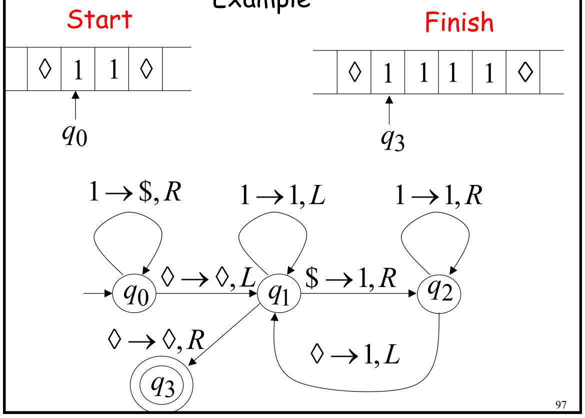
- Replace every 1 with \$
- Repeat:
  - Find rightmost \$, replace it with 1
  - Go to right end, insert 1

Until no more \$ remain

Turing Machine for $f(x) = 2x$



Example



Another Example

The function is computable

$$f(x, y) = \begin{cases} 1 & \text{if } x > y \\ 0 & \text{if } x \le y \end{cases}$$

Input: $x0y$

Output: 1 or 0

Turing Machine Pseudocode:

- Repeat

  Match a 1 from x with a 1 from y

  Until all of x or y is matched

- If a 1 from x is not matched

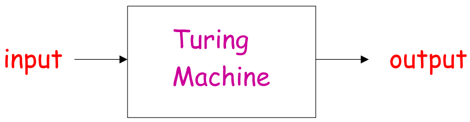
  erase tape, write 1 $\quad (x > y)$

  else

  erase tape, write 0 $\quad (x \le y)$

Combining Turing Machines

Block Diagram



Example:

$$f(x, y) = \begin{cases} x + y & \text{if } x > y \\ 0 & \text{if } x \le y \end{cases}$$