2021

Theory of Computation

Kun-Ta Chuang
Department of Computer Science and Information Engineering
National Cheng Kung University

Outline

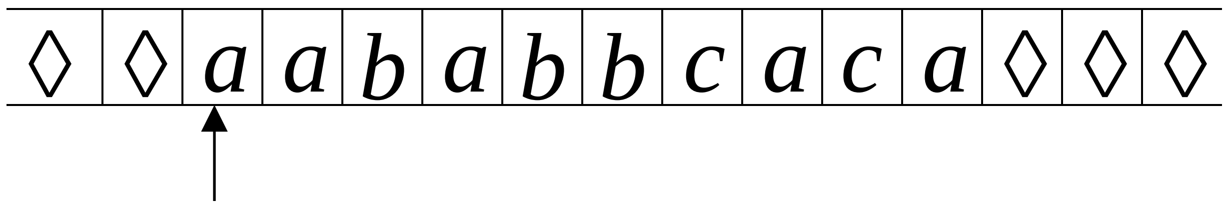
1 Minor Variations on the Turing Machine Theme

2 Turing Machines with More Complex Storage

3 Nondeterministic and Universal Turing Machines

The Standard Model

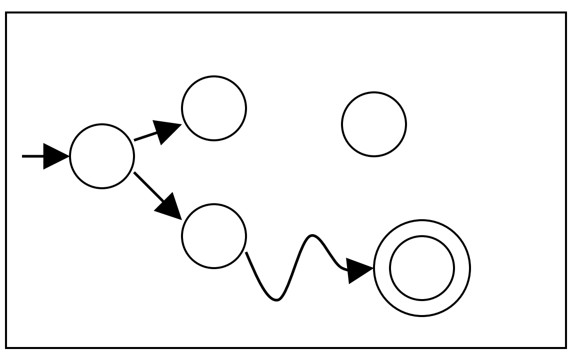
Infinite Tape



Read-Write Head

(Left or Right)

Control Unit



Deterministic

Variations of the Standard Model

Turning machines with:

- Stay-Option
- Semi-Infinite Tape
- Off-Line
- Multitape
- Multidimensional
- Nondeterministic

The variations form different
Turing Machine **Classes**

We want to prove:

Each **Class** has the same
power with the Standard Model

Same Power of two classes means:

Both classes of Turing machines accept the same languages

Same Power of two classes means:

For any machine M_1 of first class

there is a machine M_2 of second class

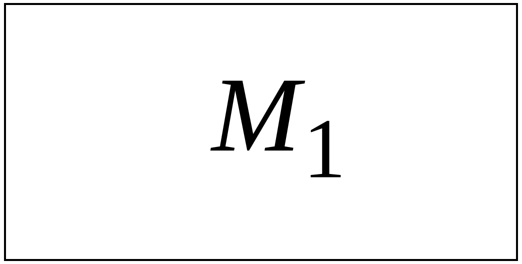
such that: $$L(M_1) = L(M_2)$$

And vice-versa

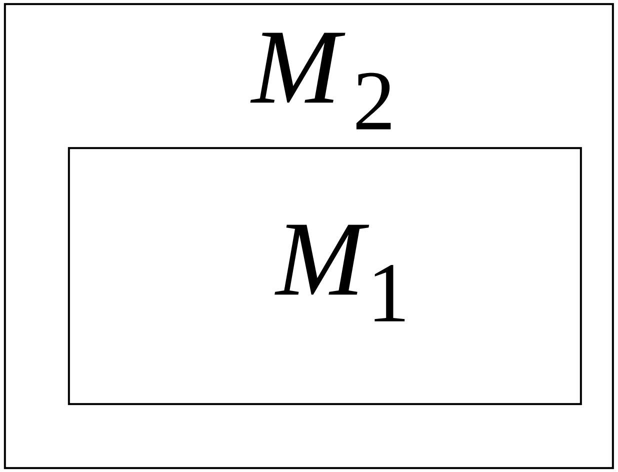
Simulation: a technique to prove same power

Simulate the machine of one class
with a machine of the other class

First Class
Original Machine



Second Class
Simulation Machine



Configurations in the Original Machine correspond to configurations in the Simulation Machine

Instantaneous description

Original Machine: $d_0 \vdash d_1 \vdash \cdots \vdash d_n$

Simulation Machine: $d_0' \overset{*}{\vdash} d_1' \overset{*}{\vdash} \cdots \overset{*}{\vdash} d_n'$

$$
\begin{tikzcd}[column sep=small]
d_0 \arrow[d, leftrightarrow] & \vdash & d_1 \arrow[d, leftrightarrow] & \vdash & \cdots & \vdash & d_n \arrow[d, leftrightarrow] \\
d_0' & \overset{*}{\vdash} & d_1' & \overset{*}{\vdash} & \cdots & \overset{*}{\vdash} & d_n'
\end{tikzcd}
$$

Final Configuration

Original Machine: d_f

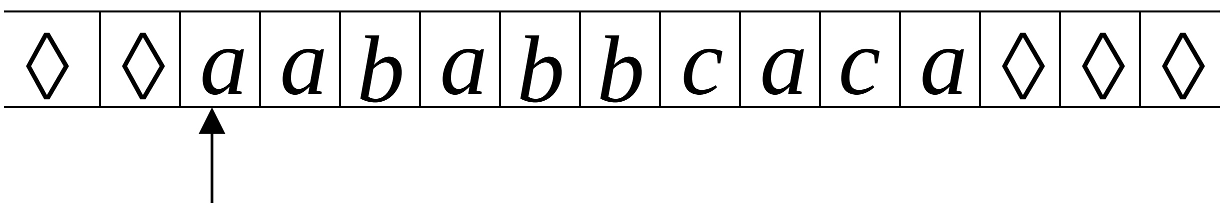
$\updownarrow$

Simulation Machine: d'_f

The Simulation Machine
and the Original Machine
accept the same language

Turing Machines with Stay-Option

The head can stay in the same position

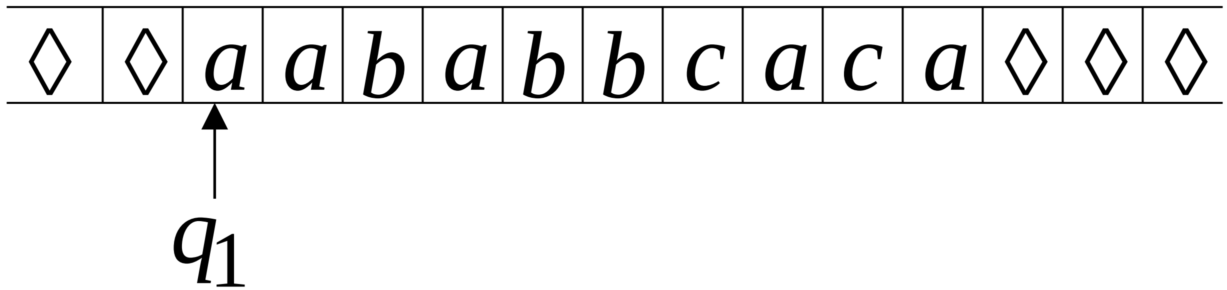


Left, Right, Stay

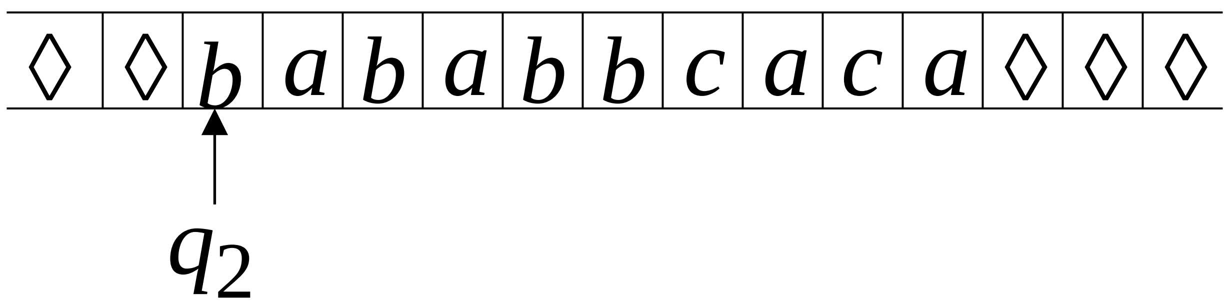
L,R,S: moves

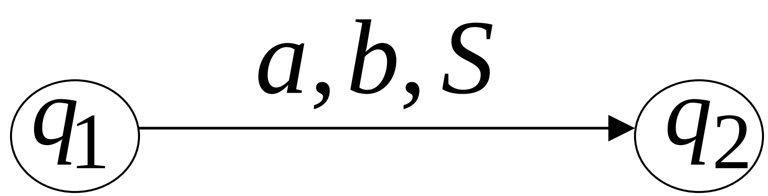
Example:

Time 1



Time 2





Theorem: Stay-Option Machines have the same power with Standard Turing machines

Proof:

Part 1: Stay-Option Machines are at least as powerful as Standard machines

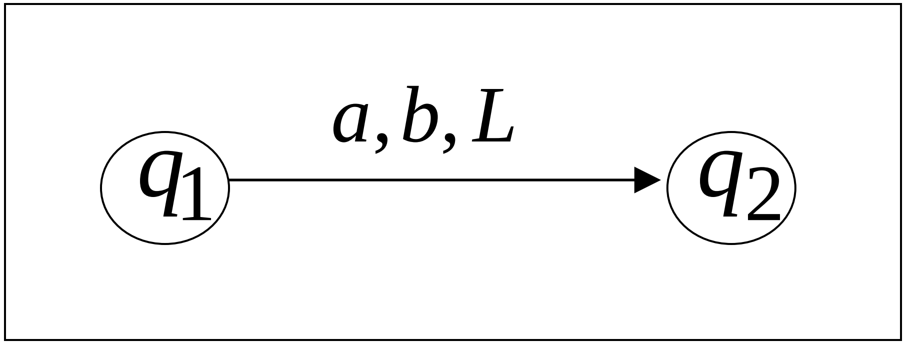
Proof: a Standard machine is also a Stay-Option machine (that never uses the S move)

Proof:

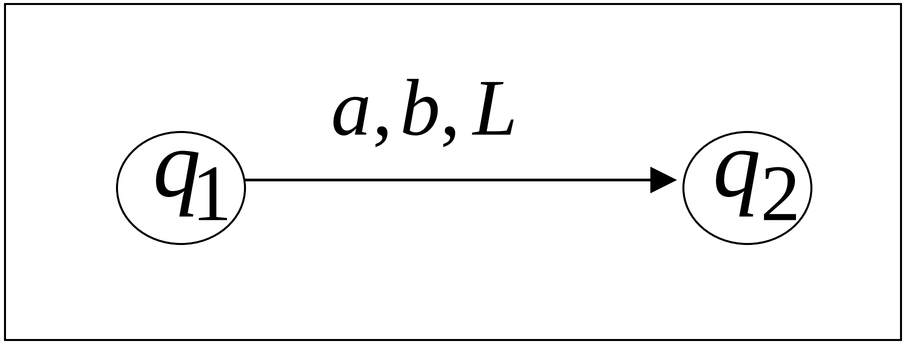
Part 2: Standard Machines
are at least as powerful as
Stay-Option machines

Proof: a standard machine can simulate
a Stay-Option machine

Stay-Option Machine



Simulation in Standard Machine



Similar for Right moves

Stay-Option Machine

$$q_1 \xrightarrow{a,\,b,\,S} q_2$$

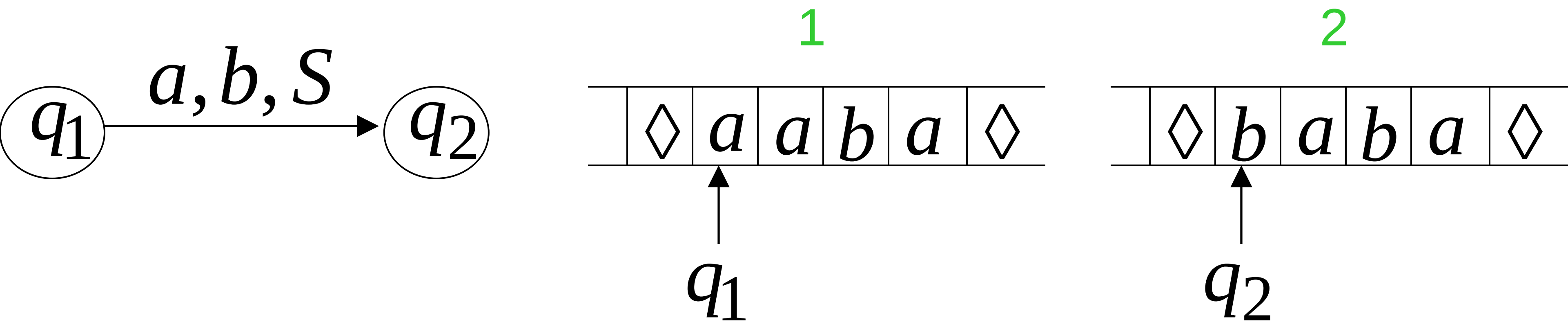
Simulation in Standard Machine

$$q_1 \xrightarrow{a,\,b,\,L} q_3 \xrightarrow{x,\,x,\,R} q_2$$

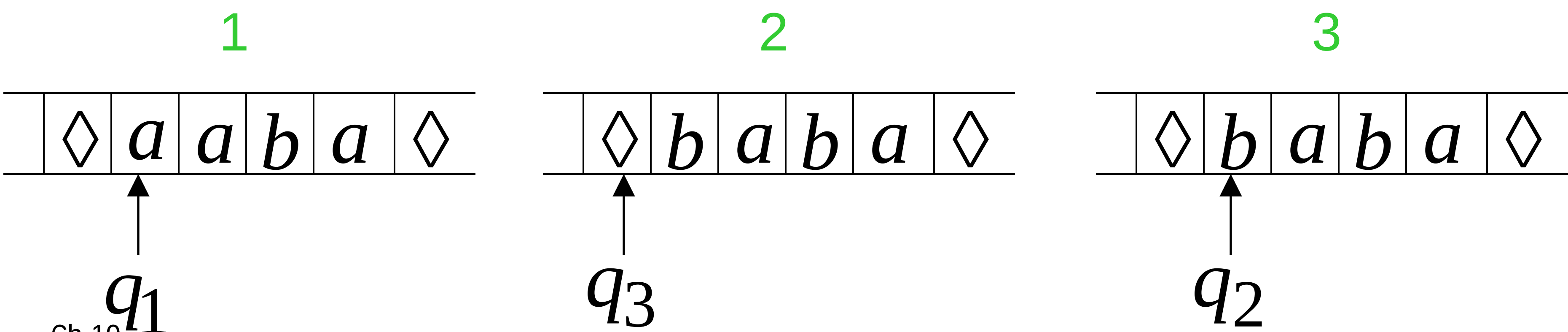
For every symbol x

Example

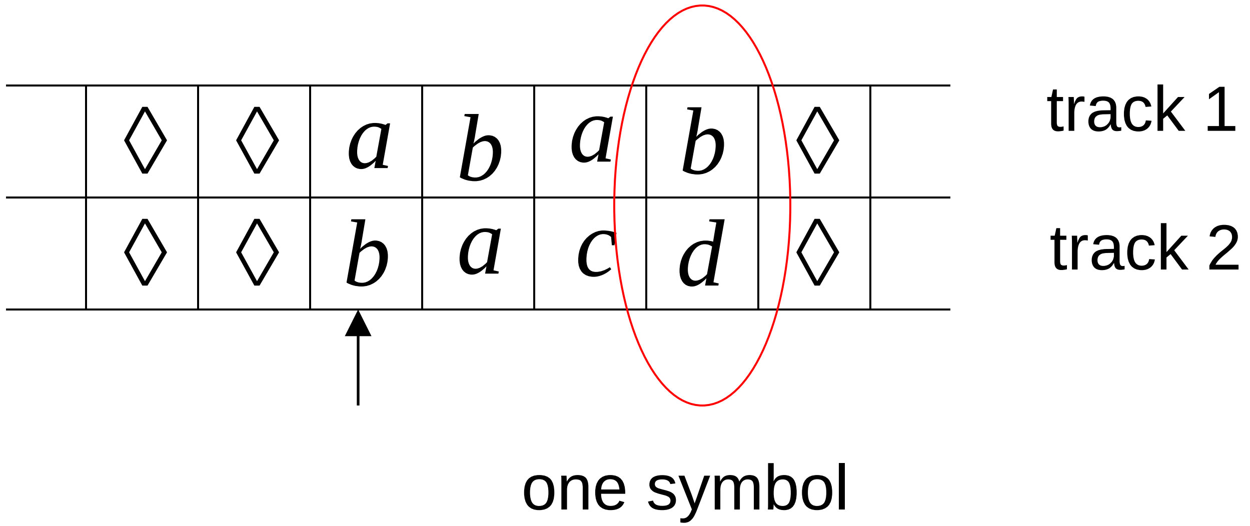
Stay-Option Machine:

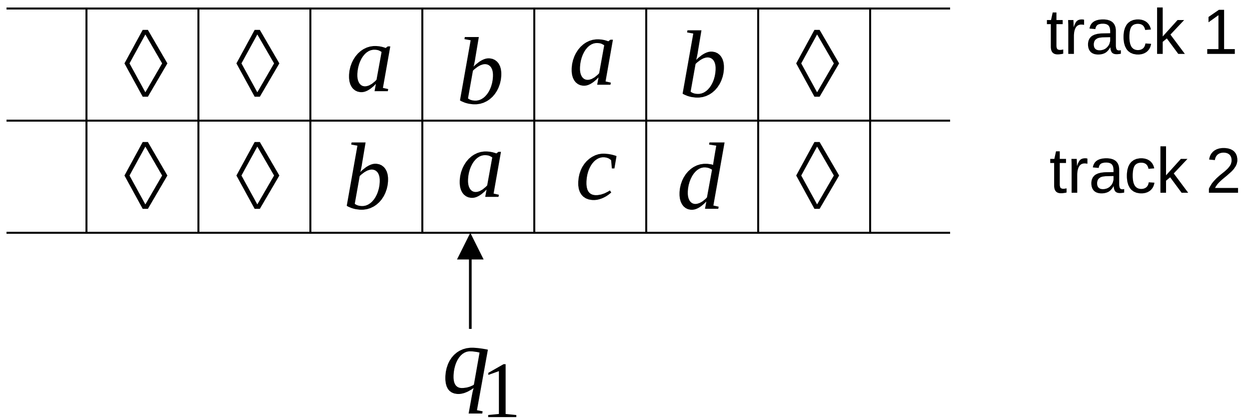


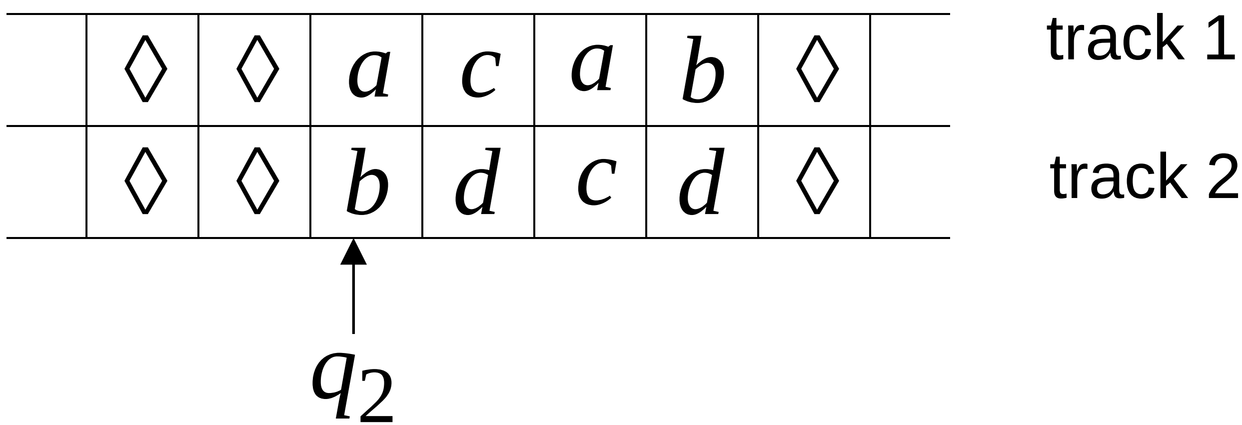
Simulation in Standard Machine:

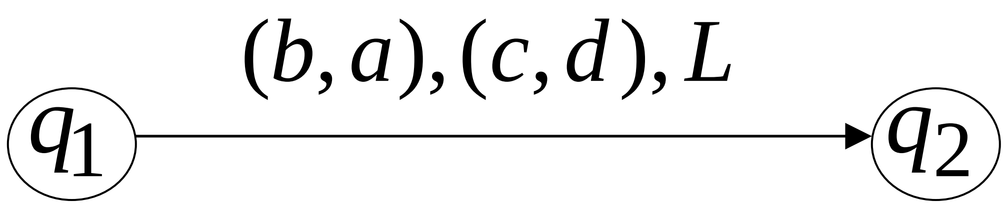


Standard Machine--Multiple Track Tape

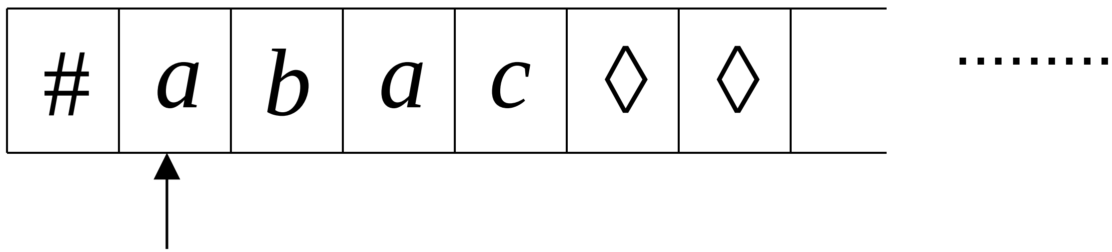



◊ ◊ a b a b ◊
◊ ◊ b a c d ◊
track 1
track 2
q_1


◊ ◊ a c a b ◊
◊ ◊ b d c d ◊
track 1
track 2
q_2


$(b,a),(c,d),L$
q_1
q_2

Semi-Infinite Tape

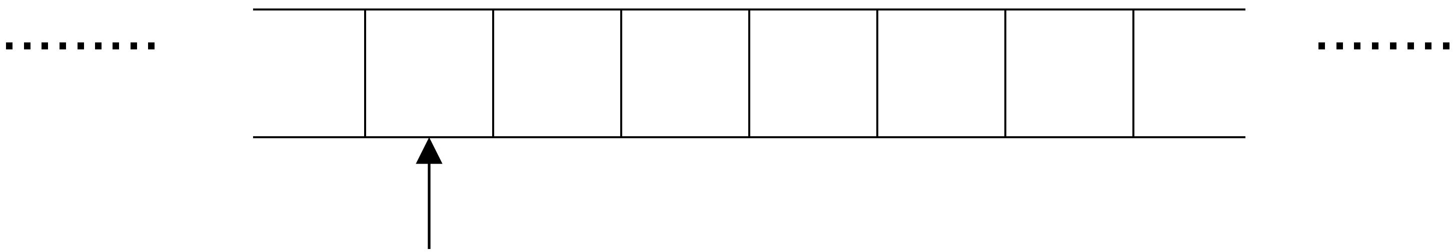


Standard Turing machines simulate Semi-infinite tape machines:

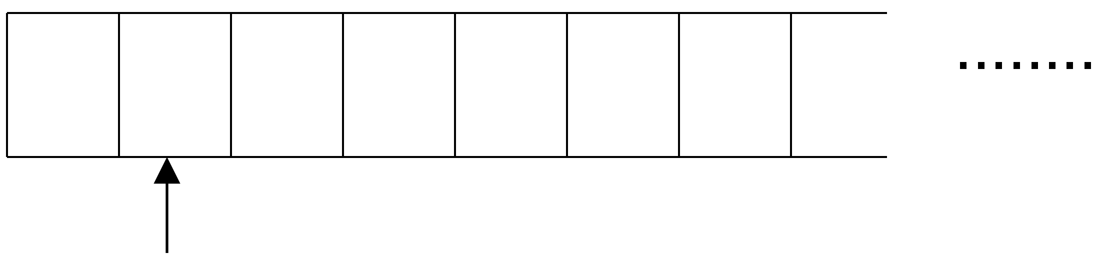
Trivial

Semi-infinite tape machines simulate Standard Turing machines:

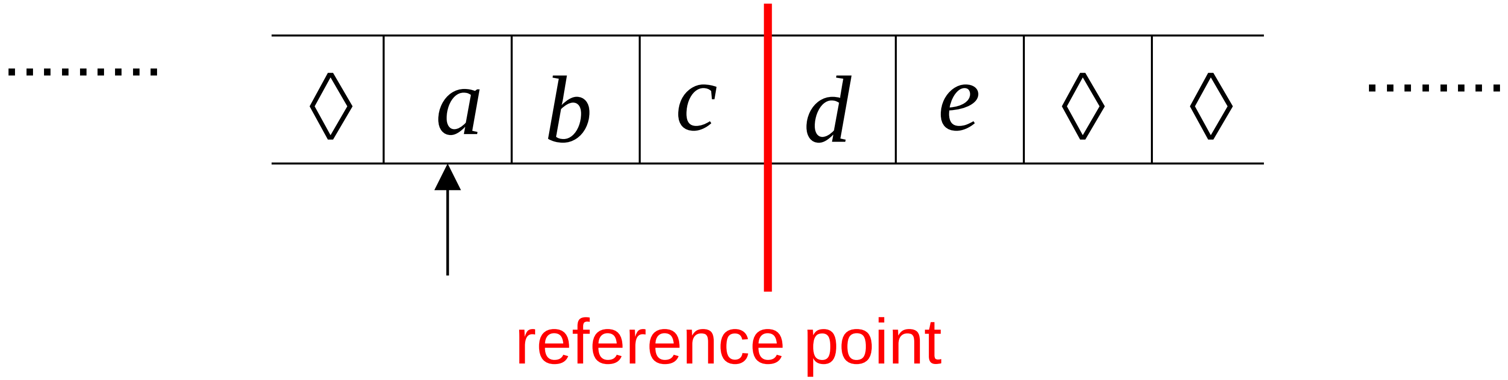
Standard machine



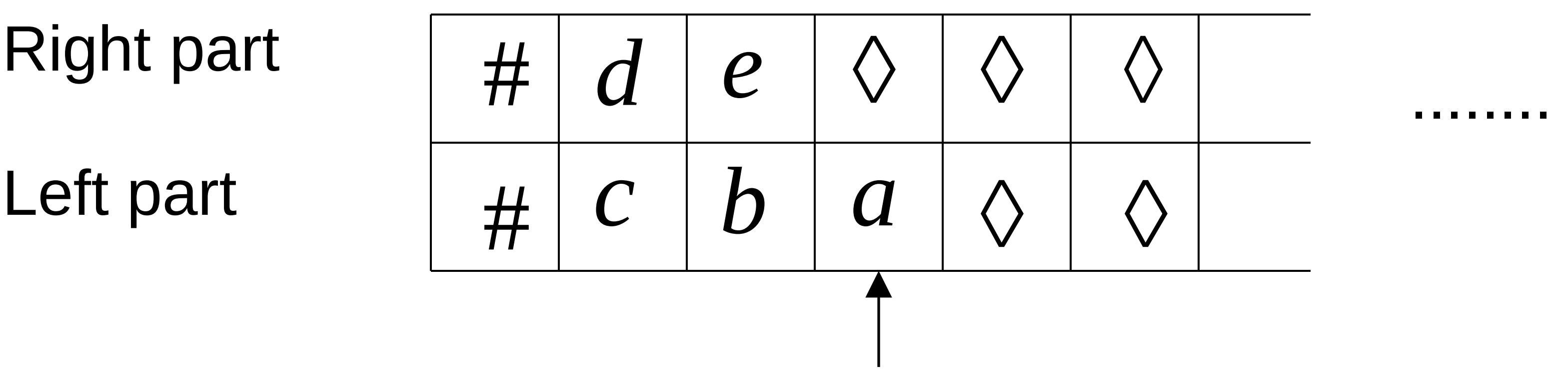
Semi-infinite tape machine



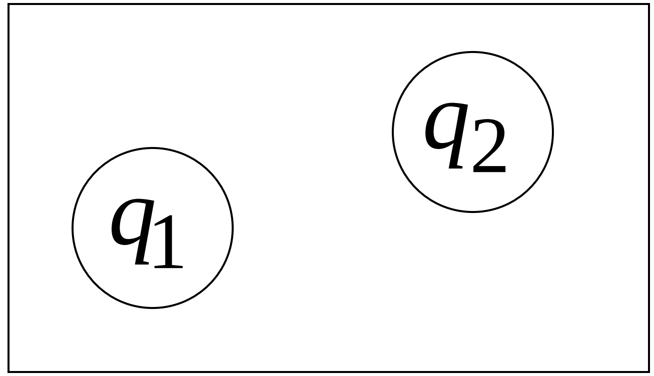
Standard machine



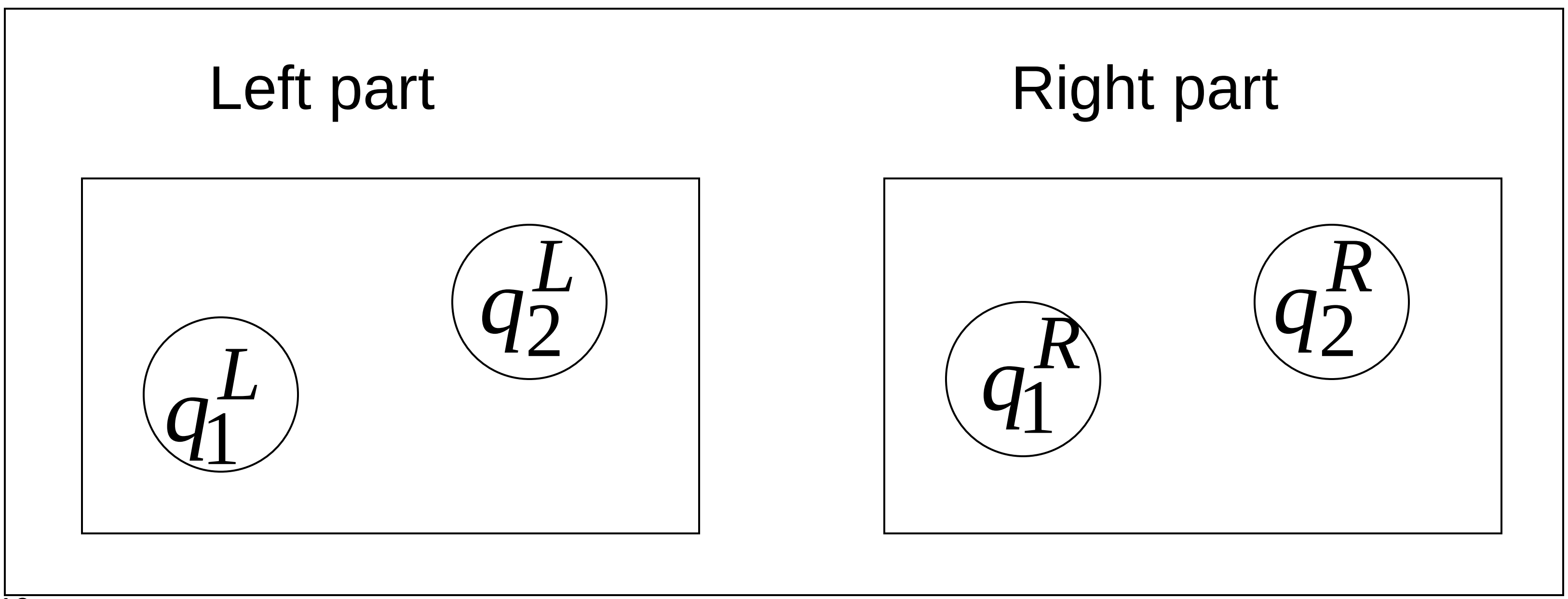
Semi-infinite tape machine with two tracks



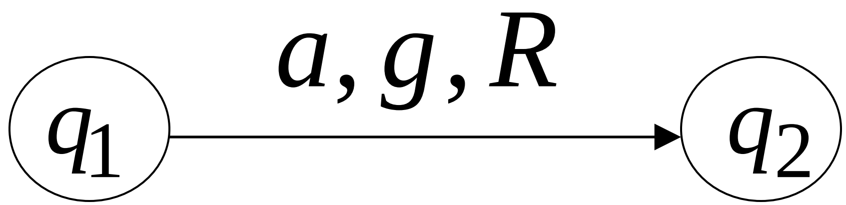
Standard machine



Semi-infinite tape machine

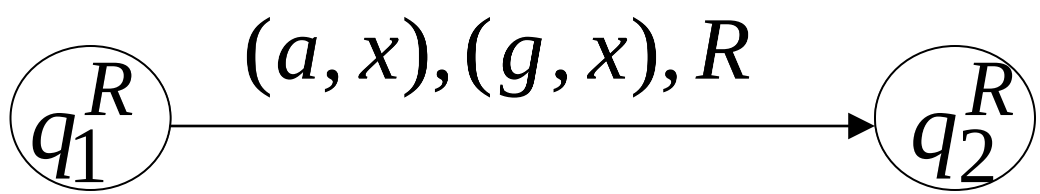


Standard machine

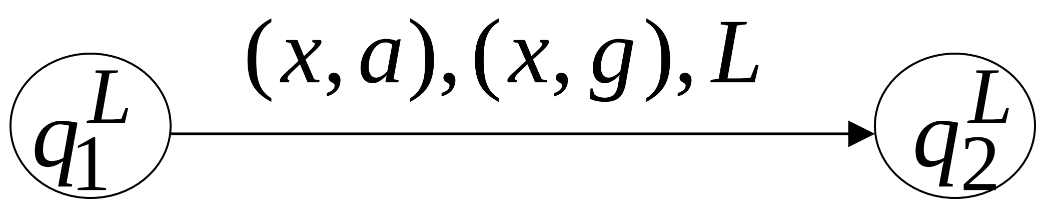


Semi-infinite tape machine

Right part



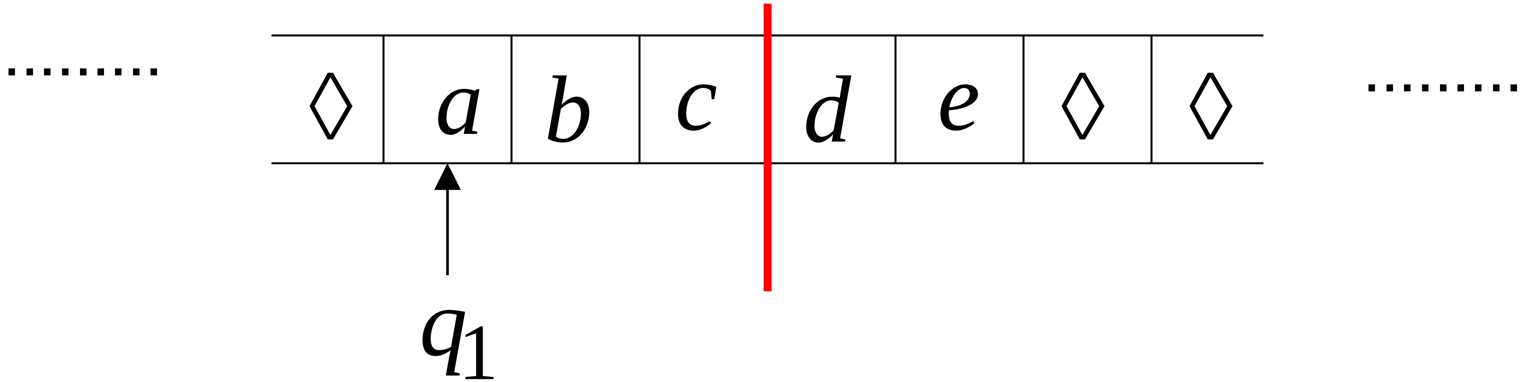
Left part



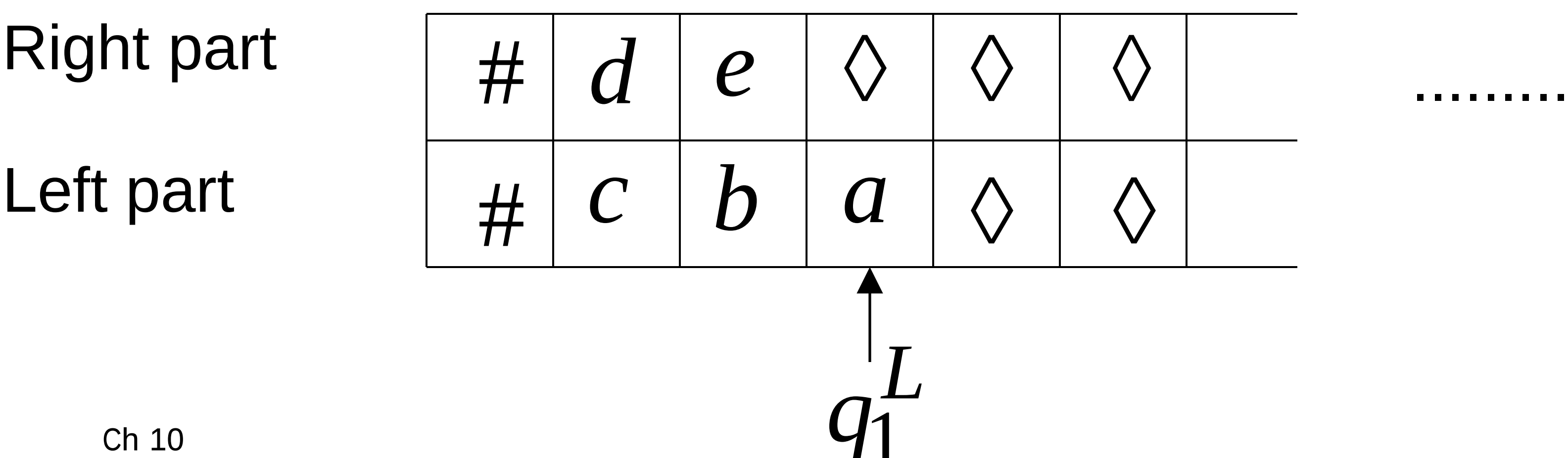
For all symbols x

Time 1

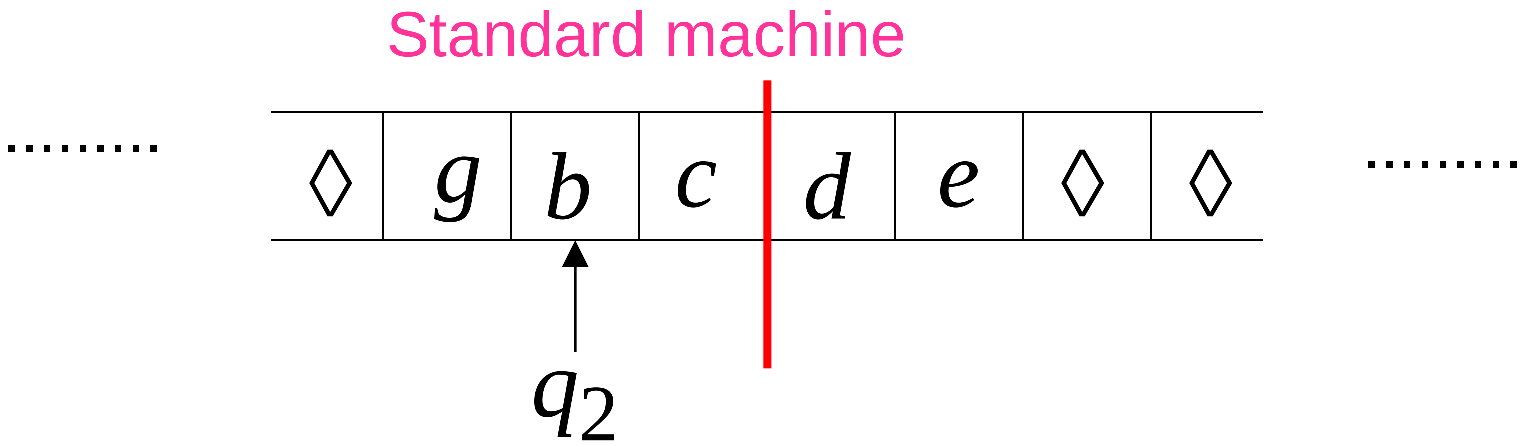
Standard machine



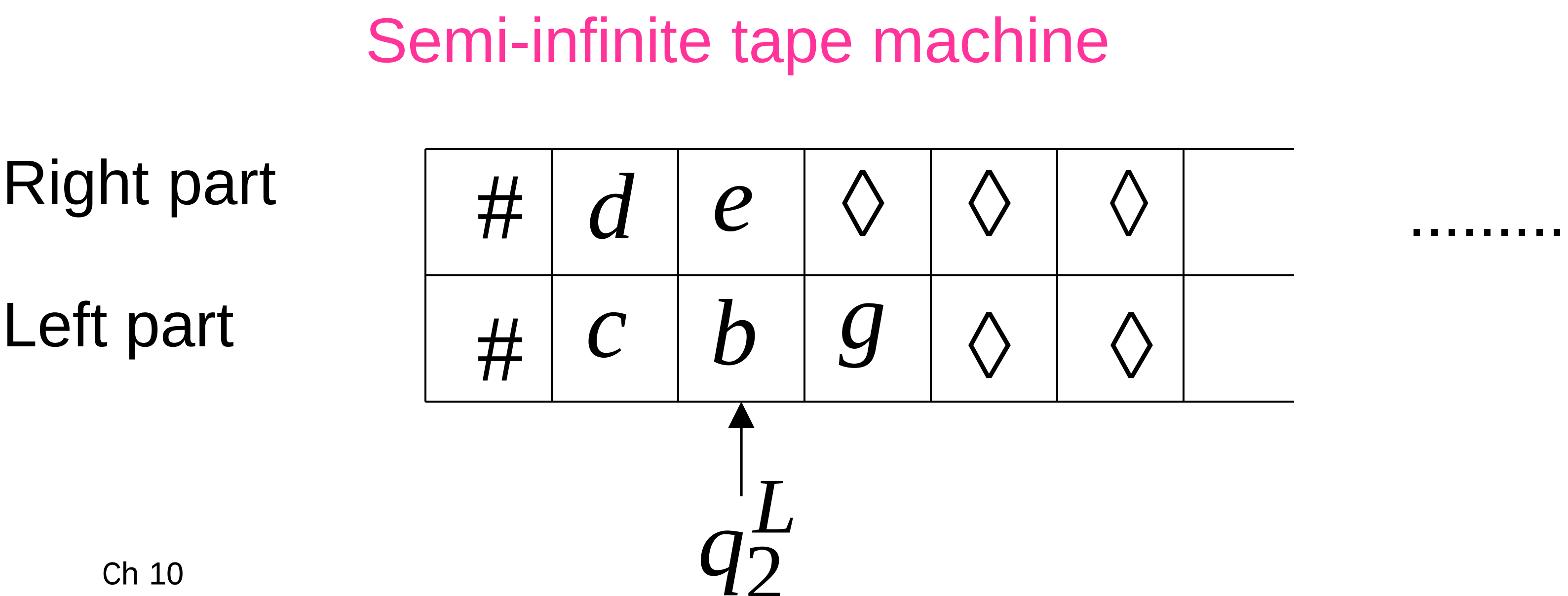
Semi-infinite tape machine



Time 2



Standard machine
....
◊
g
b
c
d
e
◊
◊
....
q_2



Semi-infinite tape machine
Right part
#
d
e
◊
◊
◊
....
Left part
#
c
b
g
◊
◊
q_2^L

At the border:

Semi-infinite tape machine

Right part

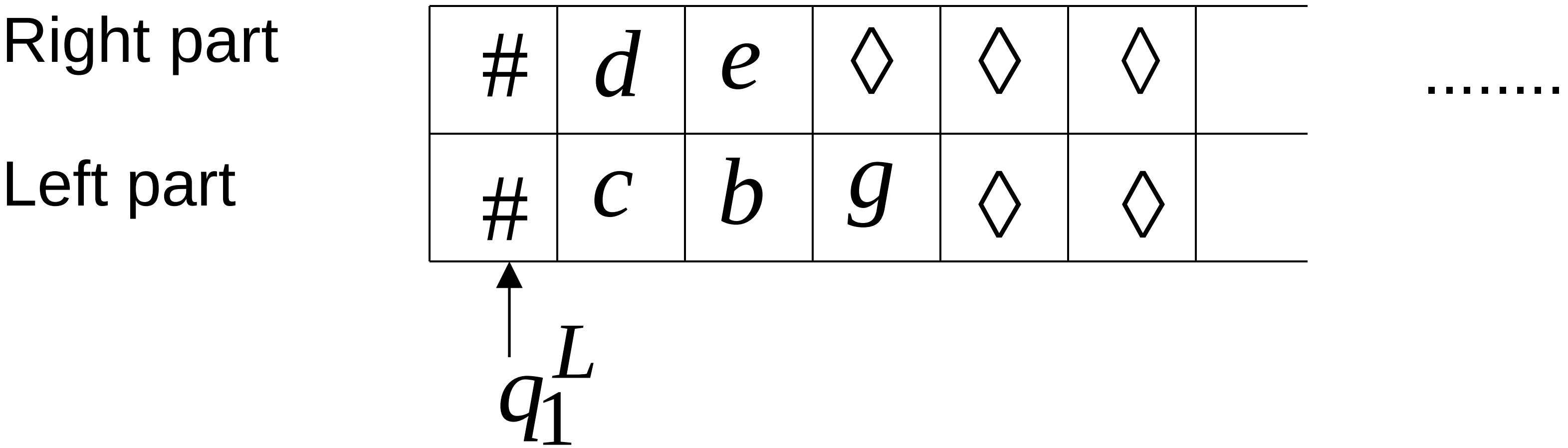
$$q_1^R \xrightarrow{(\#,\#),(\#,\#),R} q_1^L$$

Left part

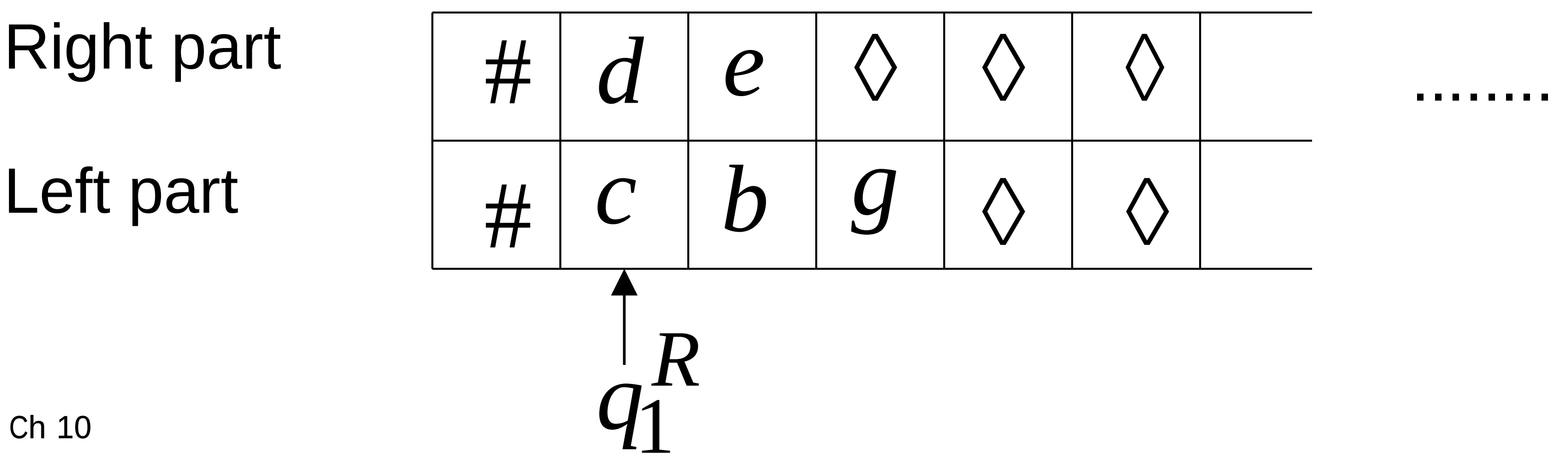
$$q_1^L \xrightarrow{(\#,\#),(\#,\#),R} q_1^R$$

Semi-infinite tape machine

Time 1

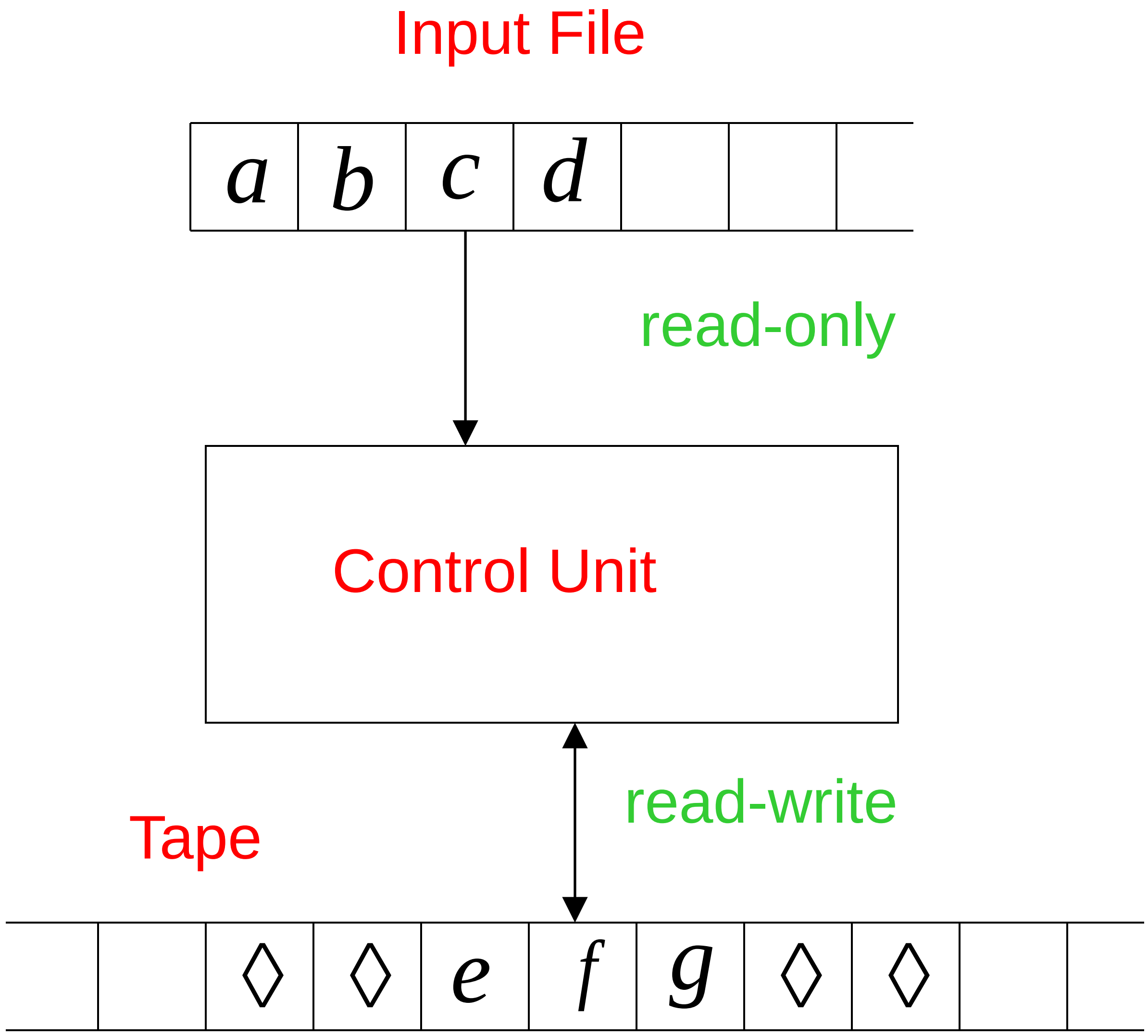


Time 2



Theorem: Semi-infinite tape machines have the same power with Standard Turing machines

The Off-Line Machine



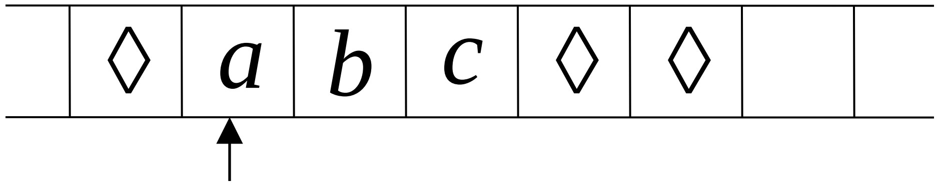
Off-line machines simulate Standard Turing Machines:

Off-line machine:

1. Copy input file to tape

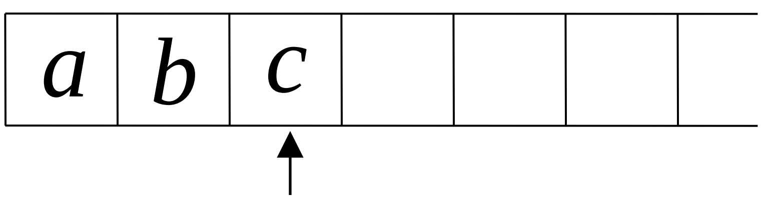
2. Continue computation as in Standard Turing machine

Standard machine

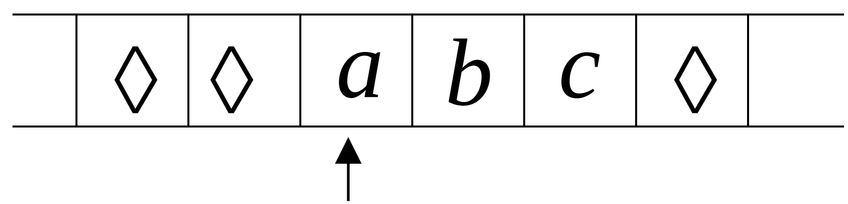


Off-line machine

Input File

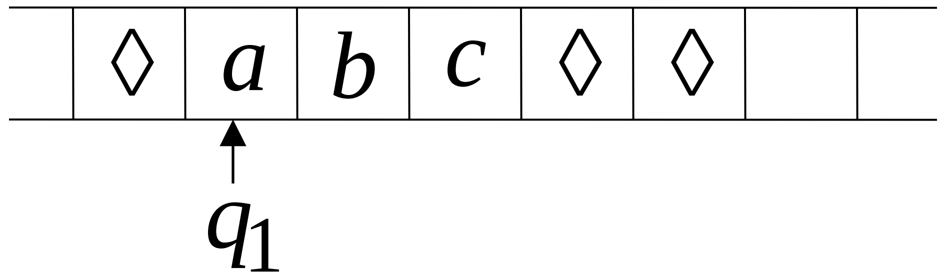


Tape



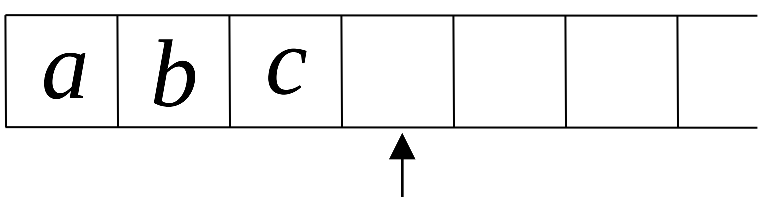
1. Copy input file to tape

Standard machine

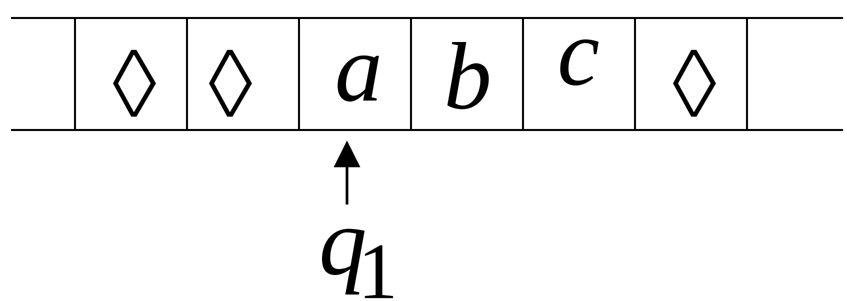


Off-line machine

Input File



Tape

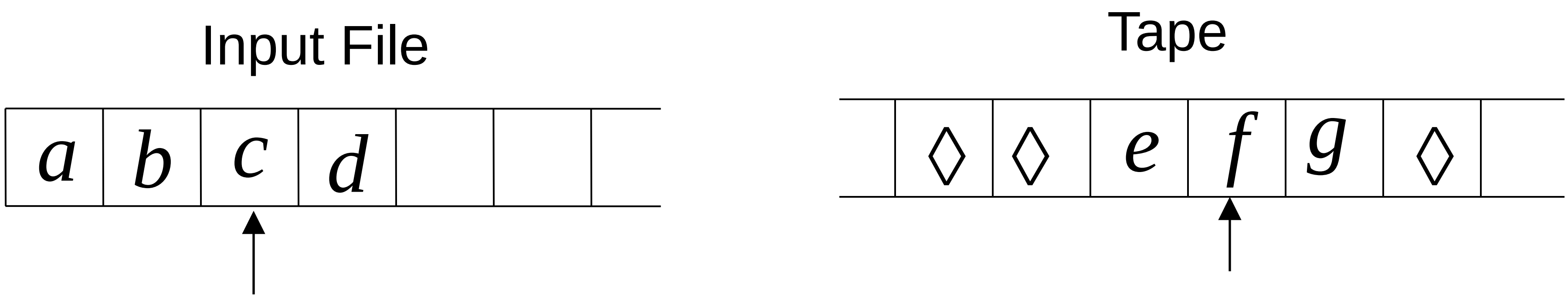


2. Do computations as in Turing machine

Standard Turing machines simulate Off-line machines:

Use a Standard machine with four track tape
to keep track of
the Off-line input file and tape contents

Off-line Machine



Four track tape -- Standard Machine

	#	a	b	c	d			Input File
	#	0	0	1	0			head position
	#	e	f	g				Tape
	#	0	1	0				head position

Reference point

	#	a	b	c	d			Input File
	#	0	0	1	0			head position
	#	e	f	g				Tape
	#	0	1	0				head position

Repeat for each state transition:

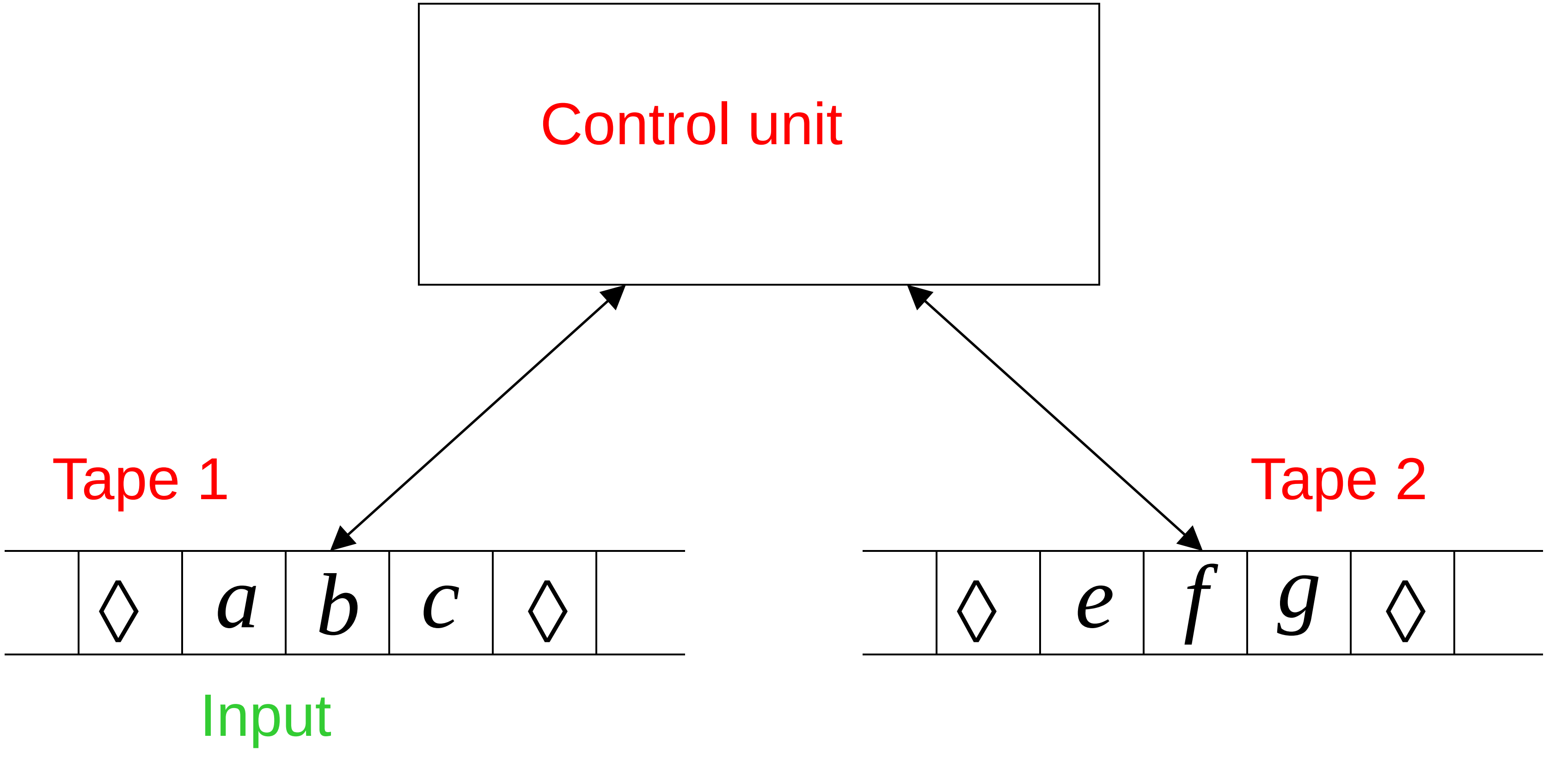
- Return to reference point
- Find current input file symbol
- Find current tape symbol
- Make transition

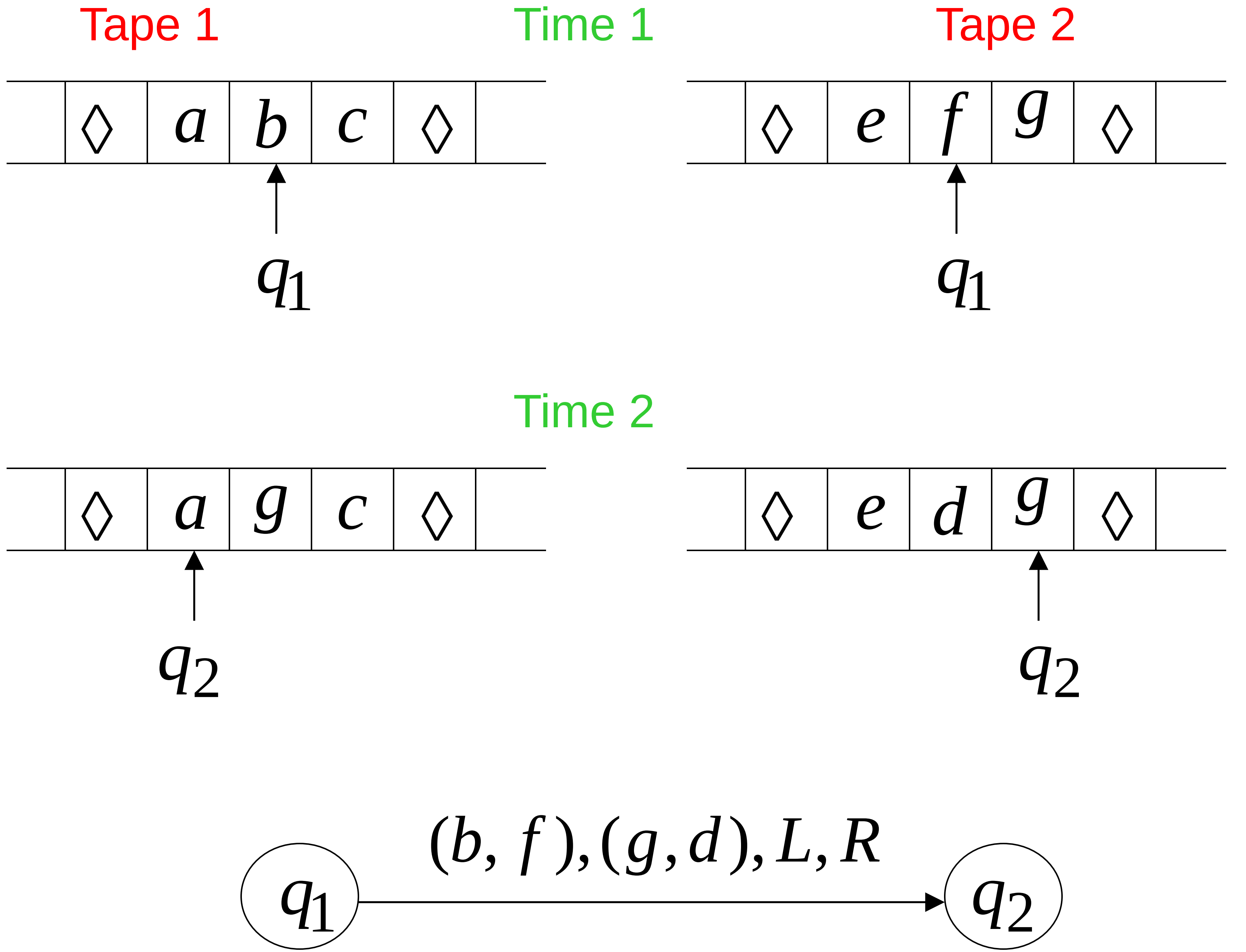
Theorem: Off-line machines
have the same power with
Standard machines

Outline

1. Minor Variations on the Turing Machine Theme
2. Turing Machines with More Complex Storage
3. Nondeterministic and Universal Turing Machines

Multitape Turing Machines




Tape 1
Time 1
Tape 2
◊ a b c ◊
◊ e f g ◊
q_1
q_1
Time 2
◊ a g c ◊
◊ e d g ◊
q_2
q_2
$(b, f), (g, d), L, R$
q_1
q_2

Multitape machines simulate Standard Machines:

Use just one tape

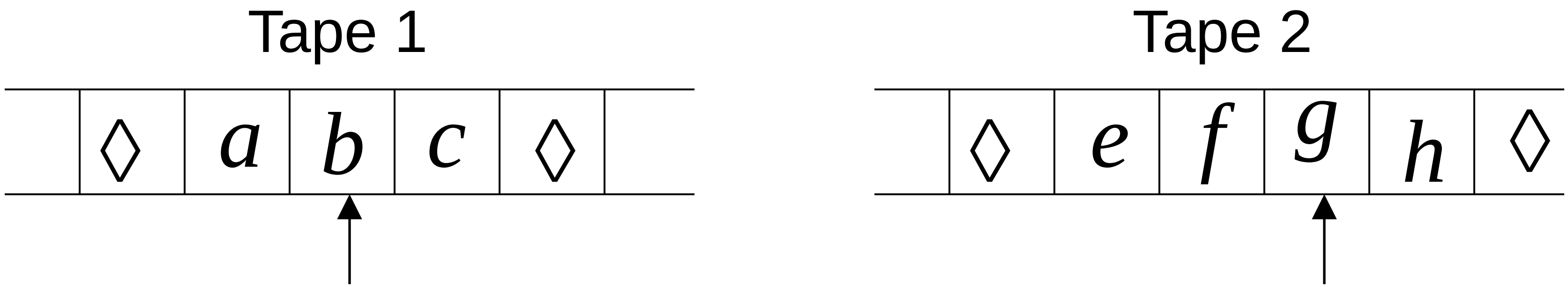
Standard machines simulate Multitape machines:

Standard machine:

- Use a multi-track tape

- A tape of the Multiple tape machine corresponds to a pair of tracks

Multitape Machine



Standard machine with four track tape

	#	a	b	c			Tape 1
	#	0	1	0			head position
	#	e	f	g	h		Tape 2
	#	0	0	1	0		head position

Reference point

	#	a	b	c				Tape 1
	#	0	1	0				head position
	#	e	f	g	h			Tape 2
	#	0	0	1	0			head position

Repeat for each state transition:

- Return to reference point
- Find current symbol in Tape 1
- Find current symbol in Tape 2
- Make transition

Theorem: Multi-tape machines have the same power with Standard Turing Machines

Same power doesn't imply same speed:

Language $L = \{a^n b^n\}$

	Acceptance Time
Standard machine	n^2
Two-tape machine	n

$$L = \{a^n b^n\}$$

Standard machine:

Go back and forth n^2 steps

Two-tape machine:

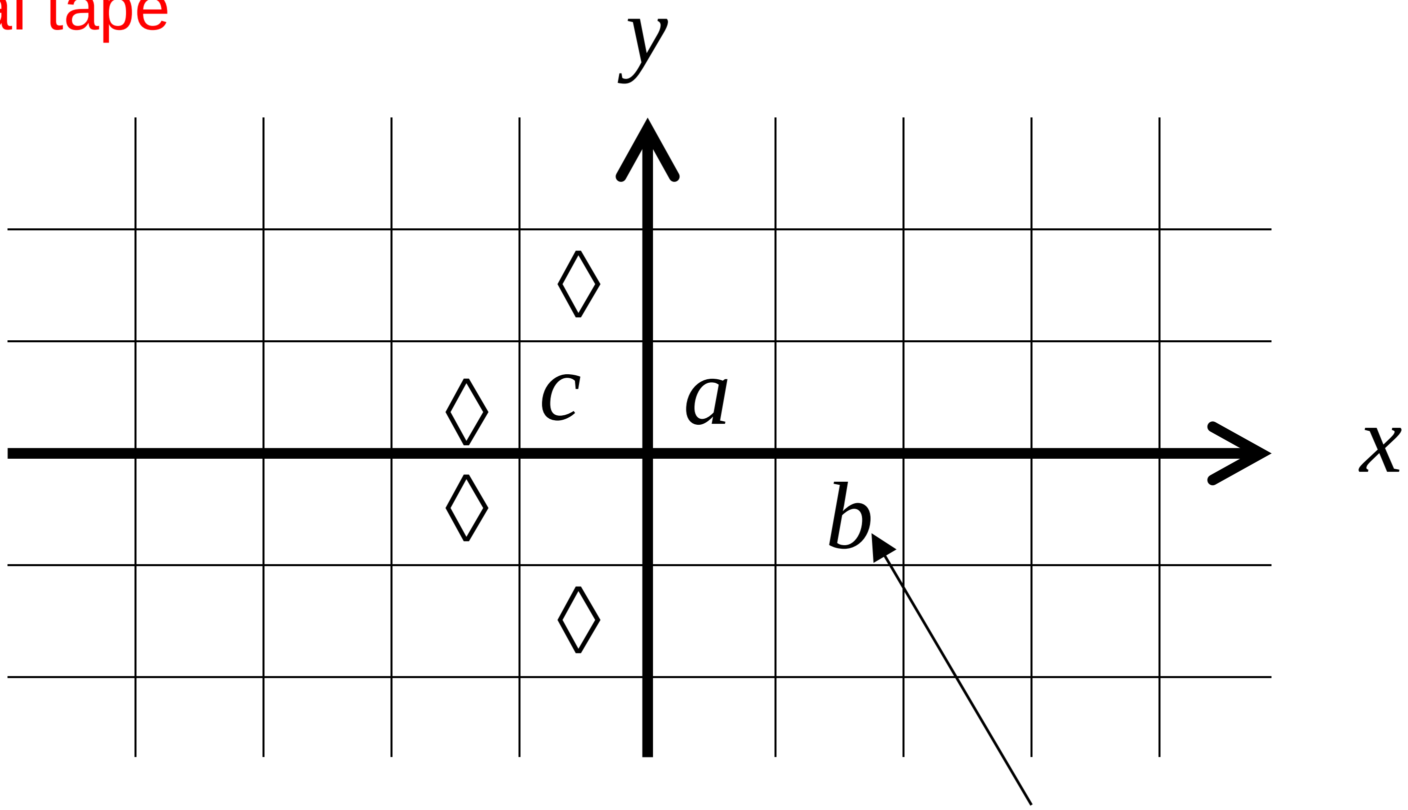
Copy b^n to tape 2 (n steps)

Leave a^n on tape 1 (n steps)

Compare tape 1 and tape 2 (n steps)

MultiDimensional Turing Machines

Two-dimensional tape



MOVES: L,R,U,D

HEAD

U: up D: down

Position: +2, -1

Multidimensional machines simulate Standard machines:

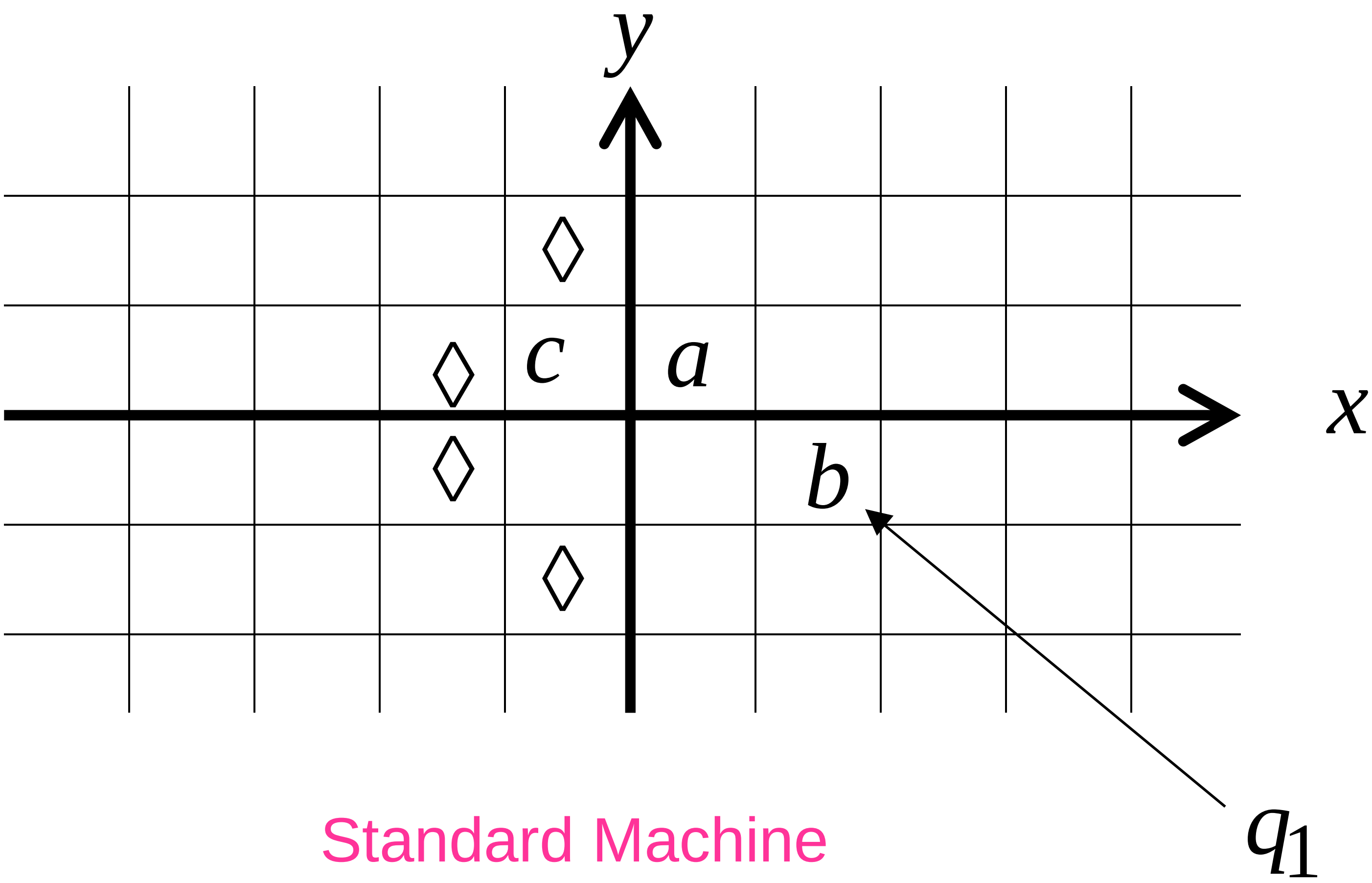
Use one dimension

Standard machines simulate Multidimensional machines:

Standard machine:

- Use a two track tape
- Store symbols in track 1
- Store coordinates in track 2

Two-dimensional machine



Standard Machine

a				b					c		symbols
1	#	1	#	2	#	$-$	1	#	$-$	1	coordinates

q_1

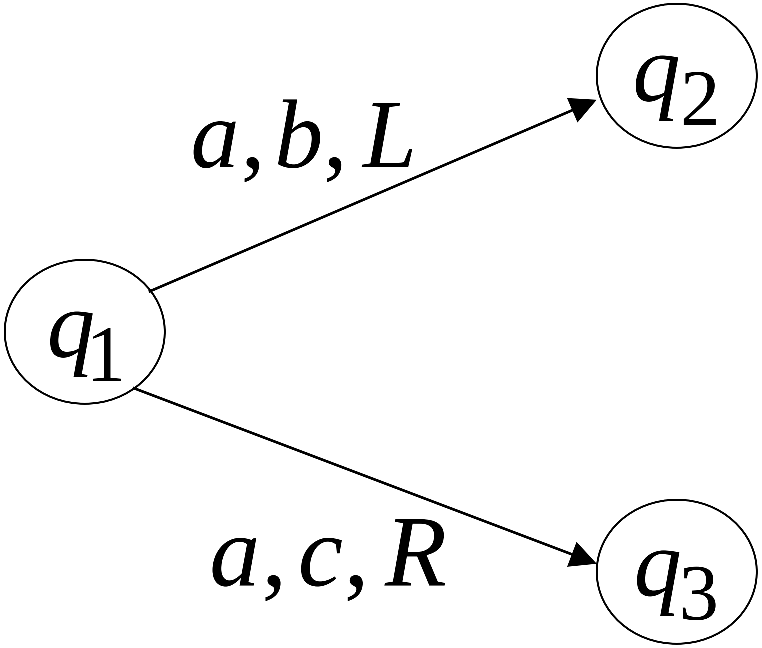
Standard machine:

Repeat for each transition

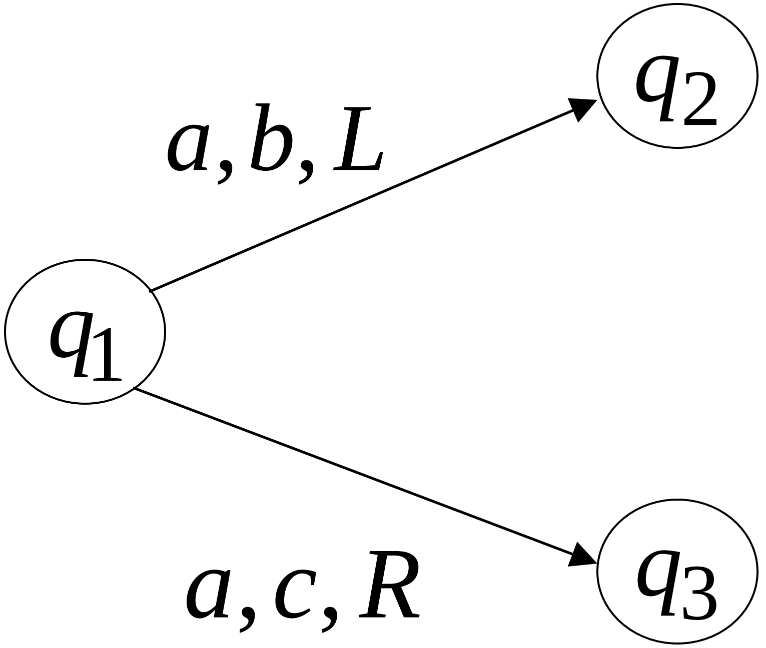
- Update current symbol
- Compute coordinates of next position
- Go to new position

Theorem: MultiDimensional Machines have the same power with Standard Turing Machines

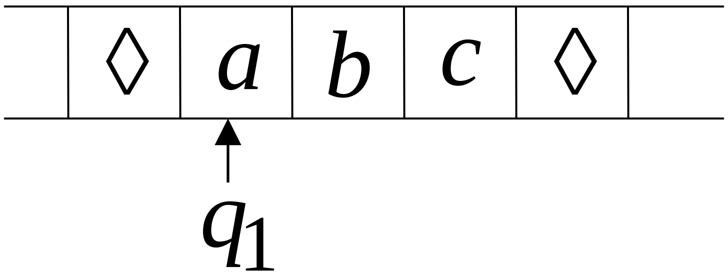
NonDeterministic Turing Machines



Non Deterministic Choice

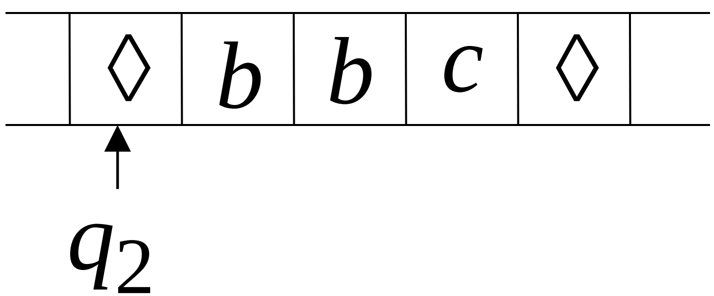


Time 0

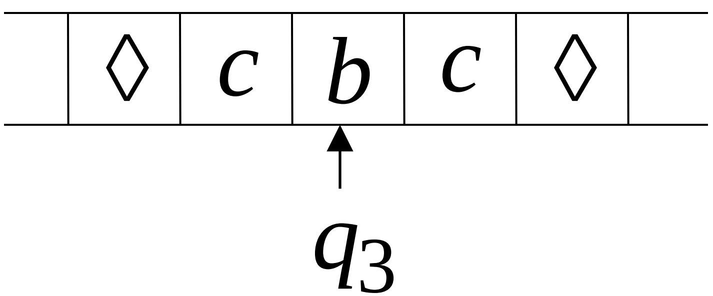


Time 1

Choice 1



Choice 2



Input string w is accepted if this is a possible computation

$$q_0 w \overset{*}{\vdash} x\, q_f\, y$$

Initial configuration

Final Configuration

Final state

Nondeterministic Machines simulate Standard (deterministic) Machines:

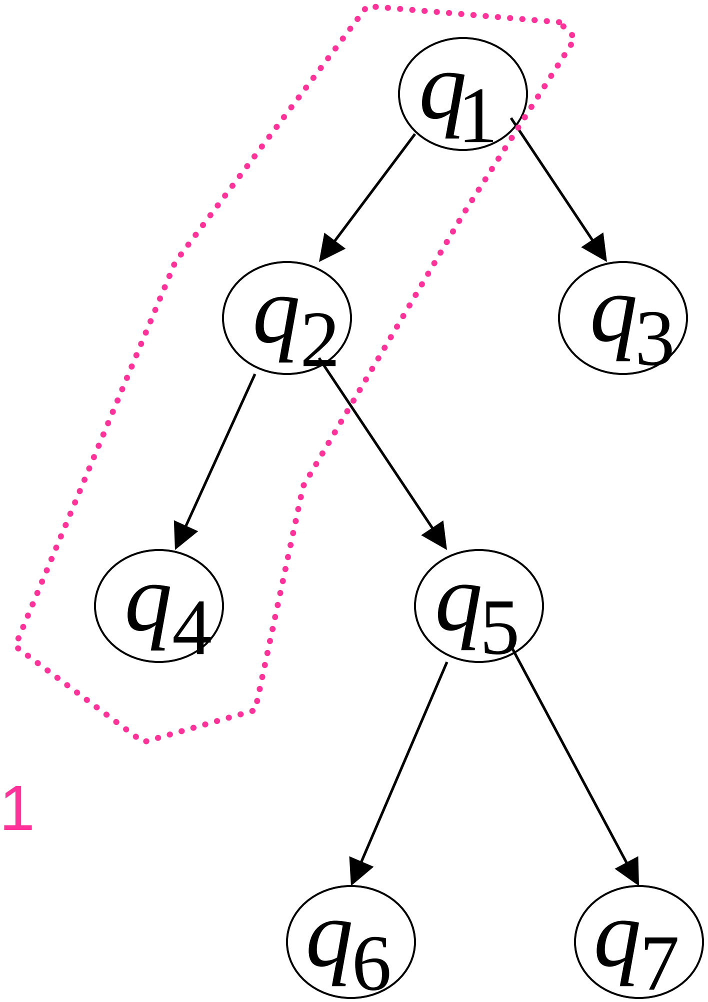
Every deterministic machine
is also a nondeterministic machine

Deterministic machines simulate NonDeterministic machines:

Deterministic machine:

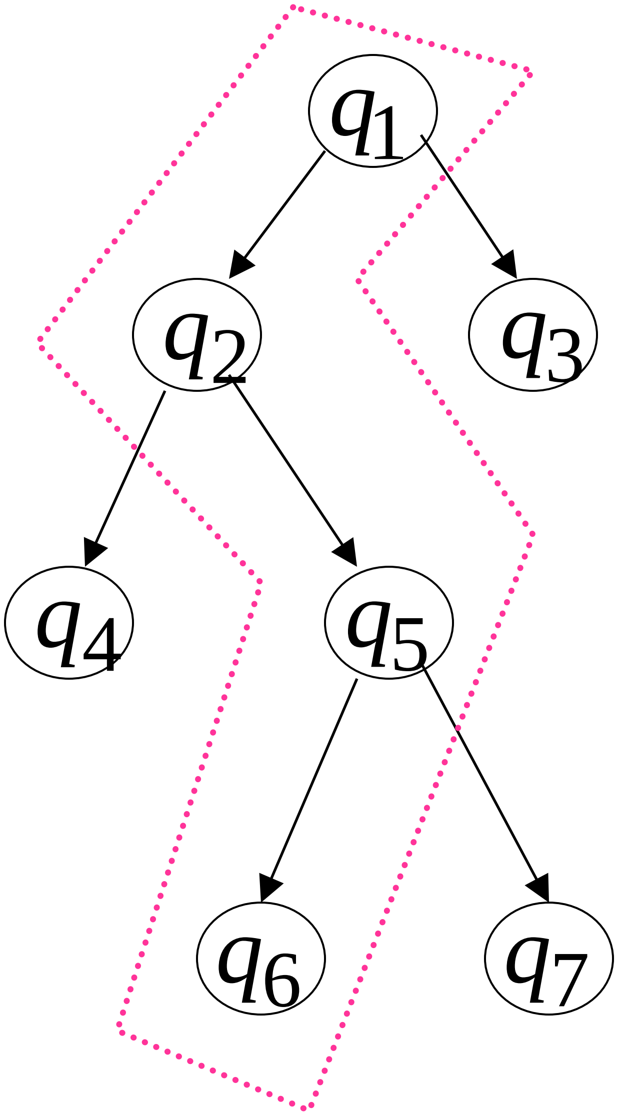
Keeps track of all possible computations

Non-Deterministic Choices



Computation 1

Non-Deterministic Choices



Computation 2

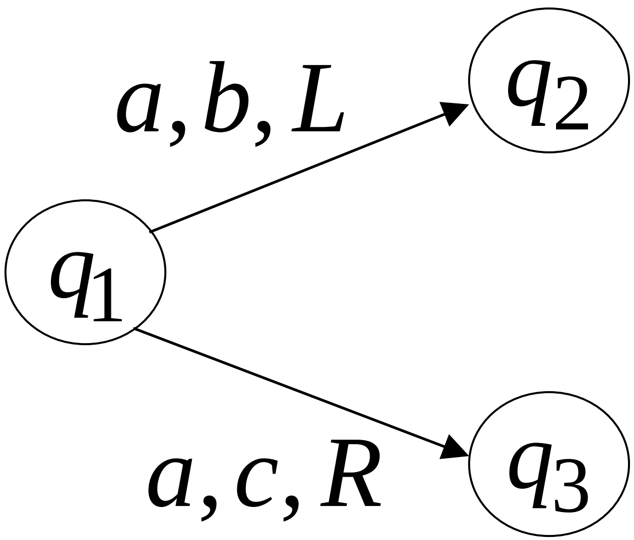
Simulation

Deterministic machine:

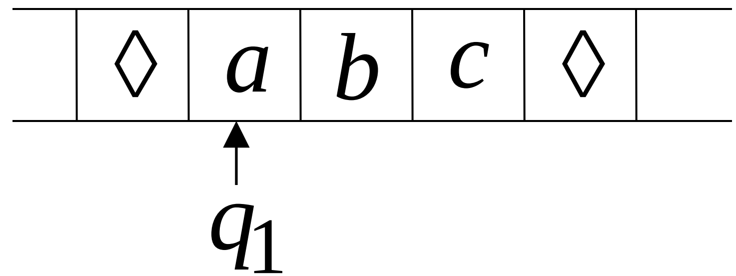
- Keeps track of all possible computations

- Stores computations in a
  two-dimensional tape

NonDeterministic machine



Time 0



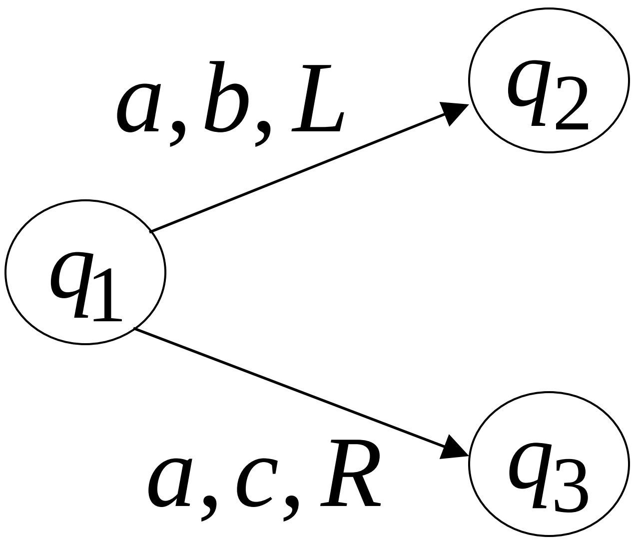
Deterministic machine

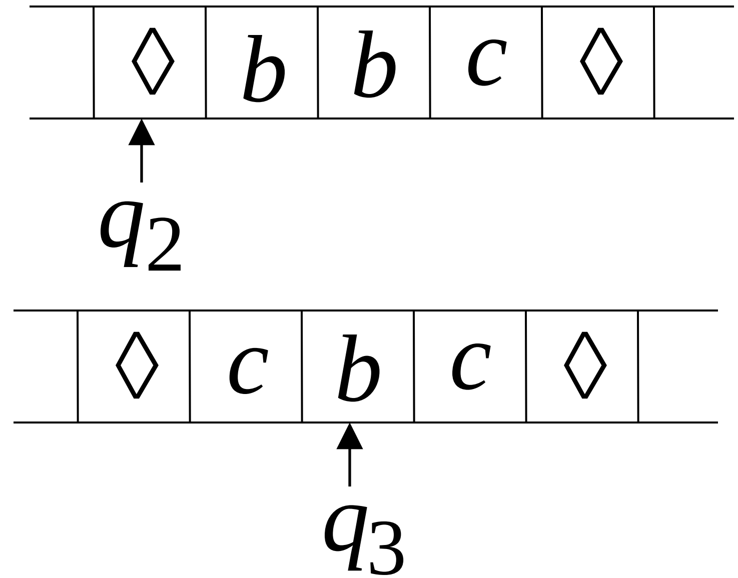
	#	#	#	#	#	#	
	#	a	b	c	#		
	#	q_1			#		
	#	#	#	#	#		

Computation 1

NonDeterministic machine

Time 1





Choice 1

Choice 2

Deterministic machine

	#	#	#	#	#	#	
#		b	b	c	#		
#	q_2				#		
#		c	b	c	#		
#			q_3		#		

Computation 1

Computation 2

Repeat

- Execute a step in each computation:

- If there are two or more choices in current computation:
  1. Replicate configuration
  2. Change the state in the replication

Theorem: NonDeterministic Machines have the same power with Deterministic machines

Remark:

The simulation in the Deterministic machine takes time exponential time compared to the NonDeterministic machine

A Universal Turing Machine

A limitation of Turing Machines:

$$\text{Turing Machines are } \underbrace{\text{“hardwired”}}_{\substack{\text{they execute} \\ \text{only one program}}}$$

Real Computers are re-programmable

Solution: Universal Turing Machine

Attributes:

- Reprogrammable machine
- Simulates any other Turing Machine

Universal Turing Machine

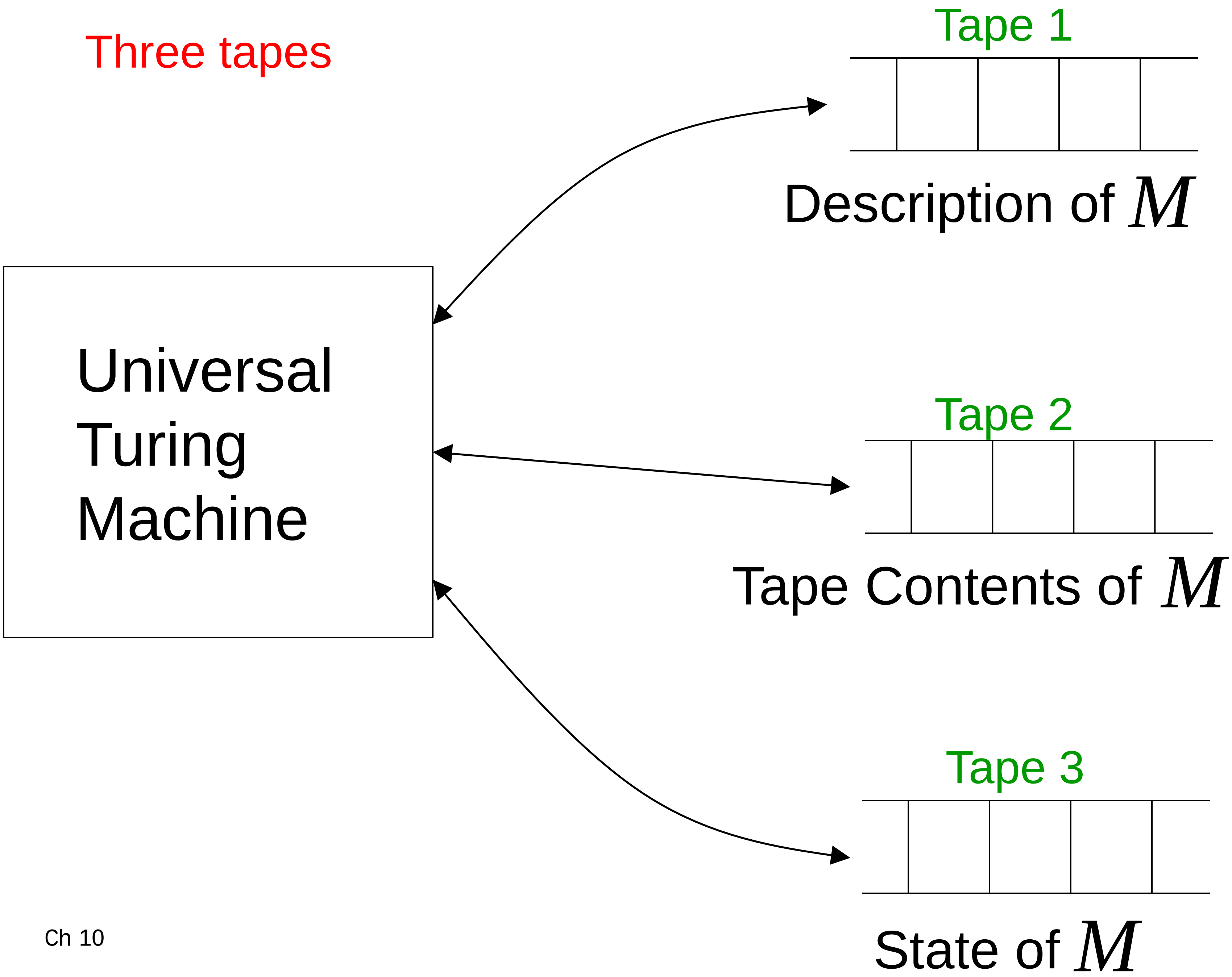
simulates any other Turing Machine M

Input of Universal Turing Machine:

Description of transitions of M

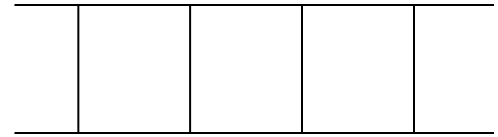
Initial tape contents of M

Three tapes



Universal Turing Machine
Tape 1
Description of M
Tape 2
Tape Contents of M
Tape 3
State of M

Tape 1



Description of M

We describe Turing machine M
as a string of symbols:

We encode M as a string of symbols

Alphabet Encoding

Symbols:	a	b	c	d	$\cdots$
Encoding:	1	11	111	1111	

State Encoding

States: q_1 q_2 q_3 q_4 $\cdots$

Encoding: 1 11 111 1111

Head Move Encoding

Move: L R

Encoding: 1 11

Transition Encoding

Transition: $\delta(q_1, a) = (q_2, b, L)$

Encoding: $1\,0\,1\,0\,11\,0\,11\,0\,1$

separator

Machine Encoding

Transitions:

$$\delta(q_1, a) = (q_2, b, L) \qquad \delta(q_2, b) = (q_3, c, R)$$

Encoding:

$$1\,0\,1\,0\,11\,0\,11\,0\,1 \quad 00 \quad 11\,0\,11\,0\,111\,0\,111\,0\,11$$

separator

Tape 1 contents of Universal Turing Machine:

encoding of the simulated machine M
as a binary string of 0's and 1's

A Turing Machine is described
with a binary string of 0's and 1's

Therefore:

The set of Turing machines forms a language:

each string of the language is
the binary encoding of a Turing Machine

Language of Turing Machines

L = { 010100101, (Turing Machine 1)

00100100101111, (Turing Machine 2)

111010011110010101,

…… ……

…… }

Countable Sets

Infinite sets are either:

Countable

or

Uncountable

Countable set:

Any finite set

or

Any *Countably infinite* set:

There is a one to one correspondence
between
elements of the set
and
Natural numbers

Example: The set of even integers is countable

Even integers: $0, \; 2, \; 4, \; 6, \; \ldots$

Correspondence:

Positive integers: $1, \; 2, \; 3, \; 4, \; \ldots$

$2n$ corresponds to $n+1$

Example: The set of rational numbers is countable

Rational numbers: $\frac{1}{2}, \frac{3}{4}, \frac{7}{8}, \ldots$

Naïve Proof

Rational numbers: $\frac{1}{1}, \frac{1}{2}, \frac{1}{3}, \ldots$

Correspondence:

$\downarrow \quad \downarrow \quad \downarrow$

Positive integers: $1, 2, 3, \ldots$

Doesn't work:

we will never count numbers with nominator 2: $\frac{2}{1}, \frac{2}{2}, \frac{2}{3}, \ldots$

Better Approach

$$
\begin{array}{ccccc}
\frac{1}{1} & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \cdots \\[2em]
\frac{2}{1} & \frac{2}{2} & \frac{2}{3} & \cdots & \\[2em]
\frac{3}{1} & \frac{3}{2} & \cdots & & \\[2em]
\frac{4}{1} & \cdots & & &
\end{array}
$$

$$
\begin{array}{ccccc}
\frac{1}{1} & \longrightarrow \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \cdots \\
\frac{2}{1} & \frac{2}{2} & \frac{2}{3} & \cdots & \\
\frac{3}{1} & \frac{3}{2} & \cdots & & \\
\frac{4}{1} & \cdots & & &
\end{array}
$$

$$
\begin{tikzcd}
\frac{1}{1} \arrow[r] & \frac{1}{2} \arrow[dl] & \frac{1}{3} & \frac{1}{4} & \cdots \\
\frac{2}{1} & \frac{2}{2} & \frac{2}{3} & \cdots & \\
\frac{3}{1} & \frac{3}{2} & \cdots & & \\
\frac{4}{1} & \cdots & & &
\end{tikzcd}
$$

$$\begin{array}{ccccc}
\frac{1}{1} & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \cdots \\[2em]
\frac{2}{1} & \frac{2}{2} & \frac{2}{3} & \cdots & \\[2em]
\frac{3}{1} & \frac{3}{2} & \cdots & & \\[2em]
\frac{4}{1} & \cdots & & &
\end{array}$$

$$
\begin{array}{ccccc}
\frac{1}{1} & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \cdots \\[2em]
\frac{2}{1} & \frac{2}{2} & \frac{2}{3} & \cdots & \\[2em]
\frac{3}{1} & \frac{3}{2} & \cdots & & \\[2em]
\frac{4}{1} & \cdots & & &
\end{array}
$$

$$
\begin{array}{ccccc}
\frac{1}{1} & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \cdots \\[2em]
\frac{2}{1} & \frac{2}{2} & \frac{2}{3} & \cdots & \\[2em]
\frac{3}{1} & \frac{3}{2} & \cdots & & \\[2em]
\frac{4}{1} & \cdots & & &
\end{array}
$$

Rational Numbers:	$\frac{1}{1}$,	$\frac{1}{2}$,	$\frac{2}{1}$,	$\frac{1}{3}$,	$\frac{2}{2}$,	$\cdots$
Correspondence:	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	
Positive Integers:	1,	2,	3,	4,	5,	$\ldots$

We proved:

the set of rational numbers is countable
by describing an enumeration procedure

Definition

Let S be a set of strings

An **enumeration procedure** for S is a Turing Machine that generates all strings of S one by one

and

Each string is generated in finite time

strings $s_1, s_2, s_3, \ldots \in S$

Enumeration Machine for S

output

(on tape)

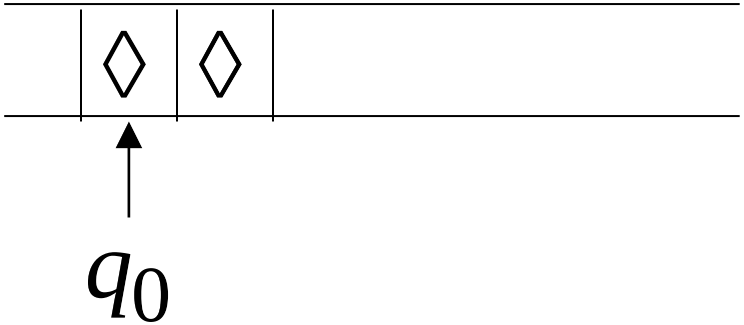
$s_1, s_2, s_3, \ldots$

Finite time: $t_1, t_2, t_3, \ldots$

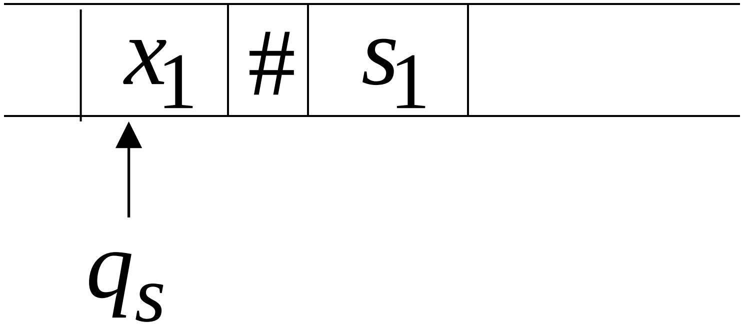
Enumeration Machine

Configuration

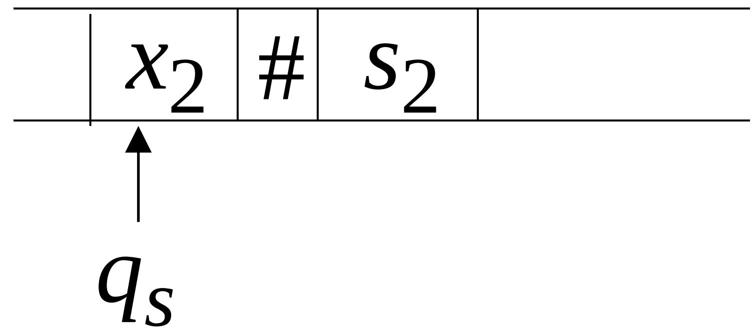
Time 0



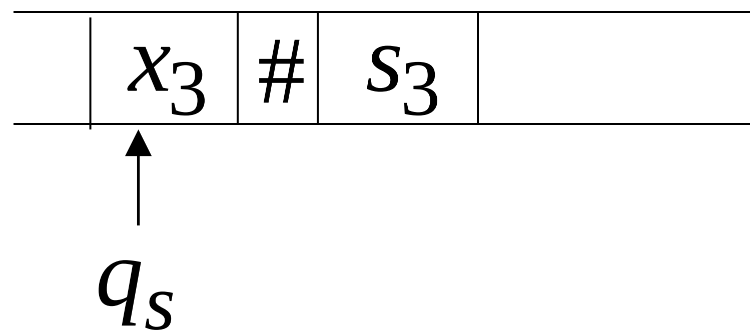
Time t_1



Time t_2



Time t_3



Observation:

If for a set there is an enumeration procedure, then the set is countable

Example:

The set of all strings $\{a,b,c\}^+$ is countable

Proof:

We will describe an enumeration procedure

Naive procedure:

Produce the strings in lexicographic order:

$$a$$

$$aa$$

$$aaa$$

$$aaaa$$

$$\ldots\ldots$$

Doesn't work:

strings starting with b
will never be produced

Better procedure: **Proper Order**

1. Produce all strings of length 1

2. Produce all strings of length 2

3. Produce all strings of length 3

4. Produce all strings of length 4

....

Produce strings in **Proper Order**:

a
b
c
length 1

aa
ab
ac
ba
bb
bc
ca
cb
cc
length 2

aaa
aab
aac
$\ldots\ldots$
length 3

Theorem 10.3:

The set of all Turing Machines
is countable

Proof: Any Turing Machine can be encoded
with a binary string of 0's and 1's

Find an enumeration procedure
for the set of Turing Machine strings

Enumeration Procedure:

Repeat

1. Generate the next binary string
   of 0's and 1's in proper order

2. Check if the string describes a
   Turing Machine
   - if **YES:** print string on output tape
   - if **NO:** ignore string

Uncountable Sets

Definition: A set is uncountable if it is not countable

Theorem:

Let S be an infinite countable set

The powerset 2^S of S is uncountable

Proof:

Since S is countable, we can write

$$S = \{s_1, s_2, s_3, \ldots\}$$

Elements of S

Elements of the powerset have the form:

$$\{s_1, s_3\}$$

$$\{s_5, s_7, s_9, s_{10}\}$$

$$\ldots\ldots$$

We encode each element of the power set with a binary string of 0's and 1's

Powerset element	Encoding				
	s_1	s_2	s_3	s_4	$\cdots$
$\{s_1\}$	1	0	0	0	$\cdots$
$\{s_2, s_3\}$	0	1	1	0	$\cdots$
$\{s_1, s_3, s_4\}$	1	0	1	1	$\cdots$

Let's assume (for contradiction)
that the powerset is countable.

Then: we can enumerate
the elements of the powerset

Powerset element

Encoding

t_1	1	0	0	0	0	$\cdots$
t_2	1	1	0	0	0	$\cdots$
t_3	1	1	0	1	0	$\cdots$
t_4	1	1	0	0	1	$\cdots$
$\cdots$						

Take the powerset element whose bits are the complements in the diagonal

$$
\begin{array}{llllll}
t_1 & 1 & 0 & 0 & 0 & 0 \ \cdots \\
t_2 & 1 & 1 & 0 & 0 & 0 \ \cdots \\
t_3 & 1 & 1 & 0 & 1 & 0 \ \cdots \\
t_4 & 1 & 1 & 0 & 0 & 1 \ \cdots
\end{array}
$$

New element: $0011\ldots$

(binary complement of diagonal)

The new element must be some t_i of the powerset

However, that's impossible:

from definition of t_i

the i^{th} bit of t_i must be the complement of itself

Contradiction!!!

Since we have a contradiction:

The powerset 2^S of S is uncountable

An Application: Languages

Example Alphabet : $\{a,b\}$

The set of all Strings:

$$S=\{a,b\}^*=\{\lambda,a,b,aa,ab,ba,bb,aaa,aab,\ldots\}$$

infinite and countable

Example Alphabet : $\{a,b\}$

The set of all Strings:

$$S=\{a,b\}^*=\{\lambda,a,b,aa,ab,ba,bb,aaa,aab,\ldots\}$$

infinite and countable

A language is a subset of S :

$$L=\{aa,ab,aab\}$$

Example Alphabet : $\{a, b\}$

The set of all Strings:

$$S = \{a, b\}^* = \{\lambda, a, b, aa, ab, ba, bb, aaa, aab, \ldots\}$$

infinite and countable

The powerset of S contains all languages:

$$2^S = \{\underbrace{\{\lambda\}}_{L_1}, \underbrace{\{a\}}_{L_2}, \underbrace{\{a, b\}}_{L_3} \underbrace{\{aa, ab, aab\}}_{L_4}, \ldots\}$$

uncountable

Languages: uncountable

$$L_1 \quad L_2 \quad L_3 \quad \cdots \quad L_k \quad \cdots$$

$$\downarrow \quad \downarrow \quad \downarrow \quad \quad \downarrow$$

$$M_1 \quad M_2 \quad M_3 \quad \quad ?$$

Turing machines: countable

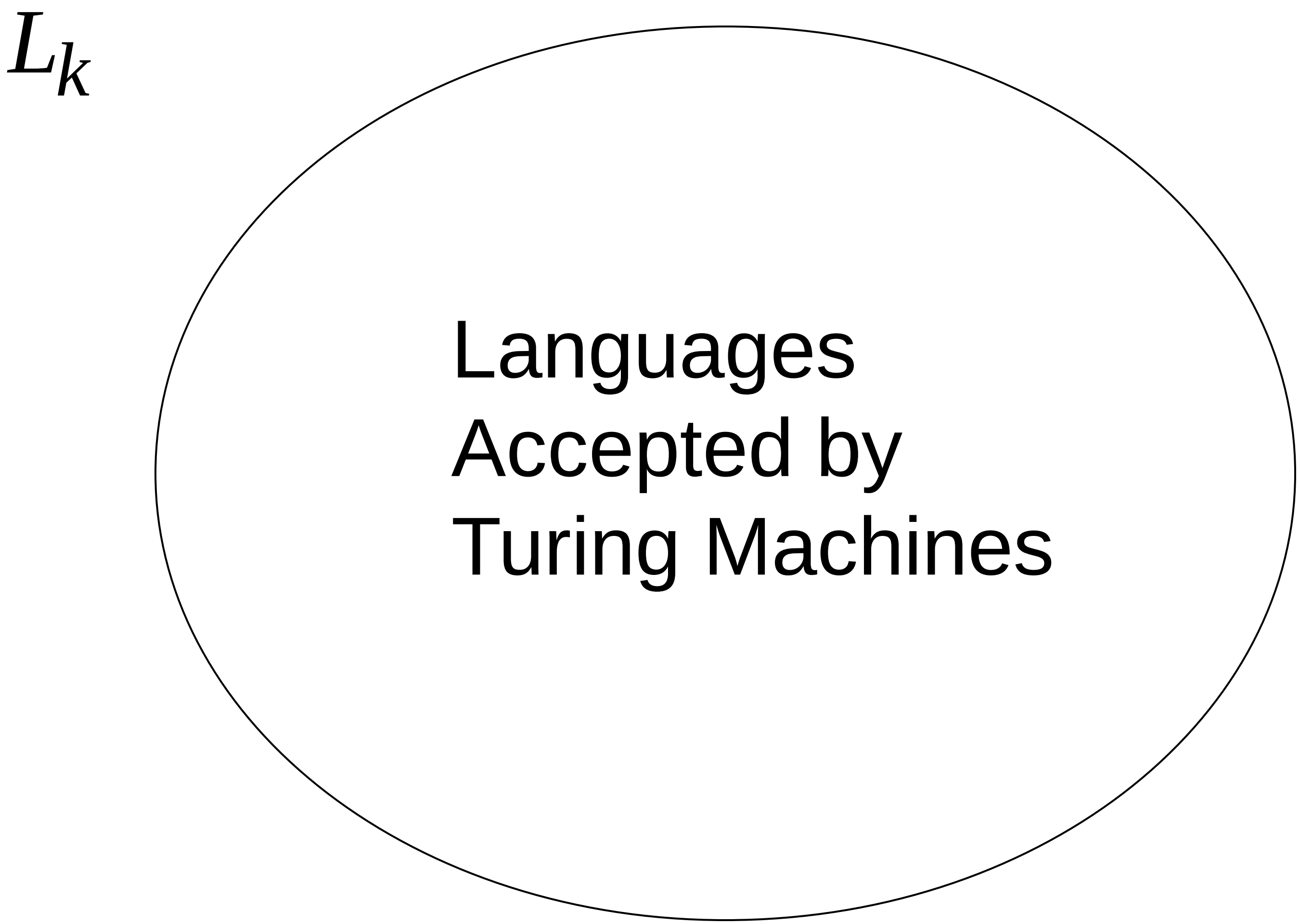
There are more languages
than Turing Machines

Conclusion:

There are some languages not accepted by Turing Machines

(These languages cannot be described by algorithms)

Languages not accepted by Turing Machines

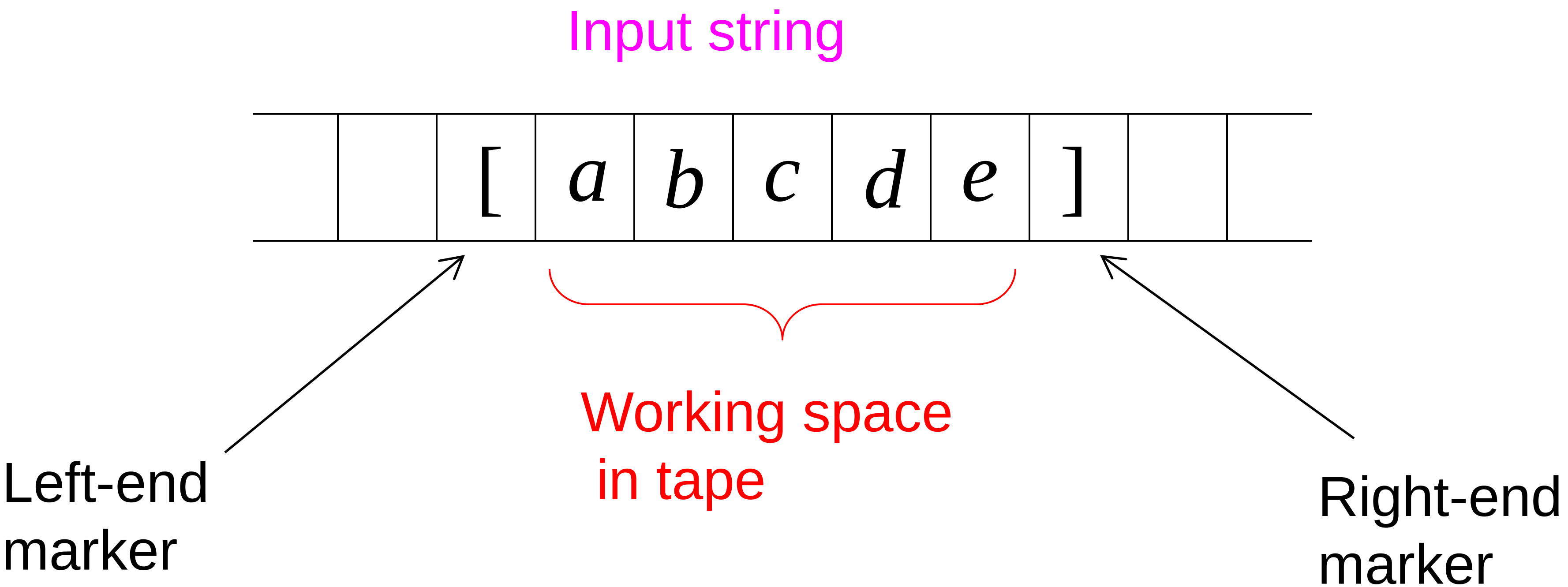


Linear Bounded Automata
LBAs

Linear Bounded Automata (LBAs)
are the same as Turing Machines
with one difference:

The input string tape space
is the only tape space allowed to use

Linear Bounded Automaton (LBA)



All computation is done between end markers

Example languages accepted by LBAs:

$$L = \{a^n b^n c^n\}$$

$$L = \{a^{n!}\}$$

LBA’s have more power than NPDA’s

LBA’s have also less power
than Turing Machines