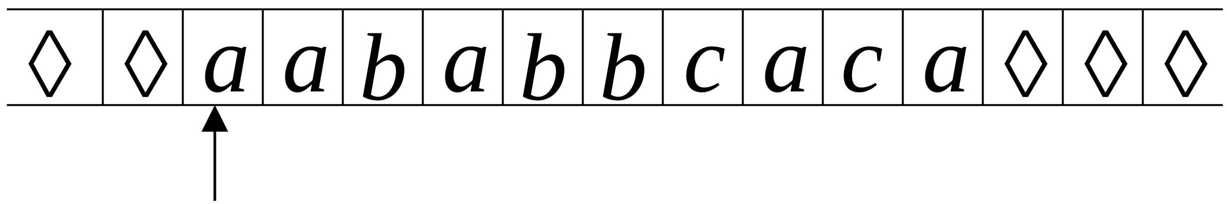
CHAPTER 10

Other Models Of Turing Machines

By R.Ameri

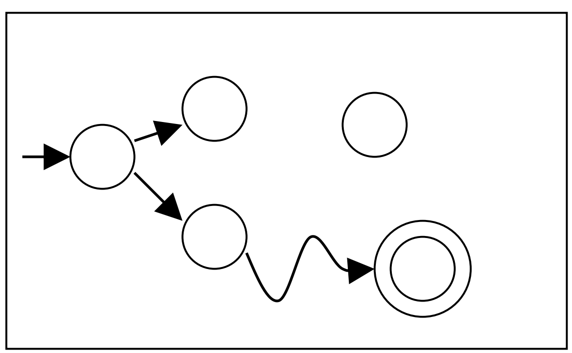
The Standard Model

Infinite Tape



Read-Write Head (Left or Right)

Control Unit



Deterministic

Variations of the Standard Model

Turing machines with:

- Stay-Option
- Multiple Track Tape
- 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

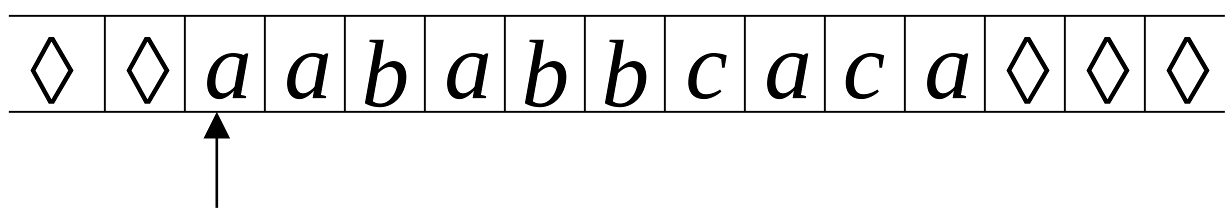
there is a machine M_2 of second class

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

And vice-versa

Turing Machines with Stay-Option

The head can stay in the same position



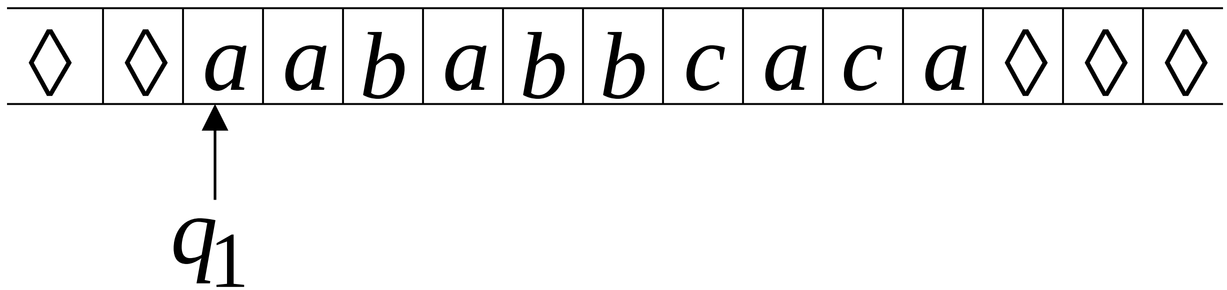
Left, Right, Stay

L,R,S: moves

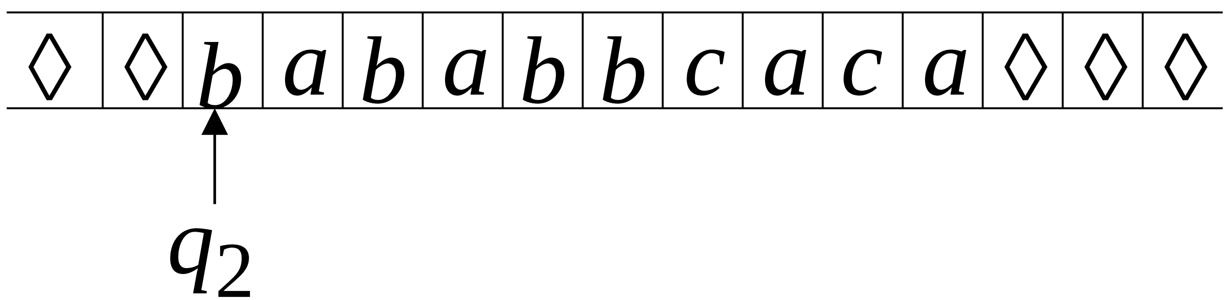
$$\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R, S\}$$

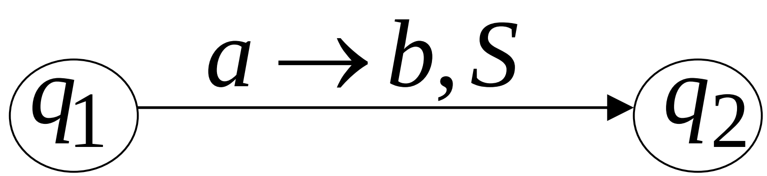
Example:

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

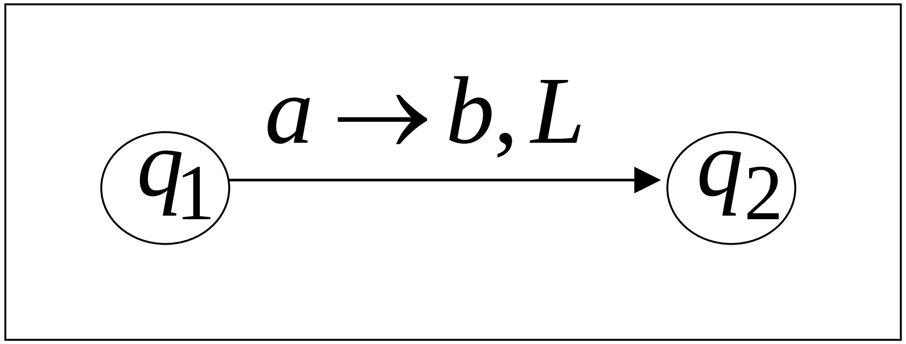
Proof: a Standard machine is also a Stay-Option machine (that never uses the S move)

Proof:

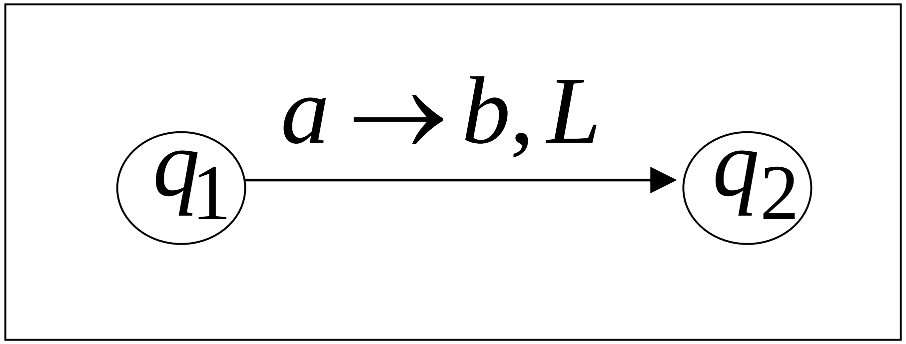
Part 2: Standard Machines are at least as powerful as Stay-Option machines

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

Stay-Option Machine

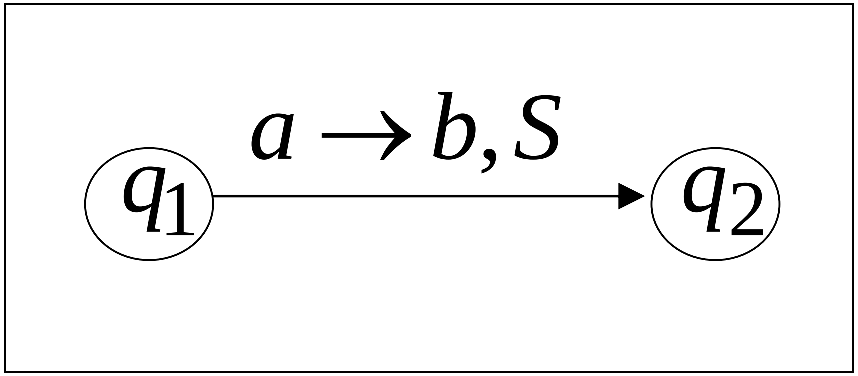


Simulation in Standard Machine

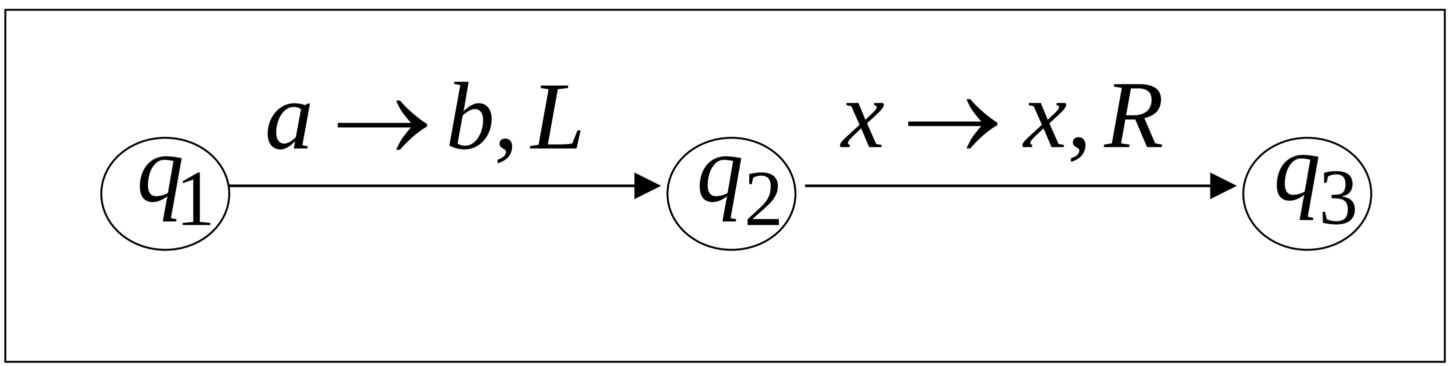


Similar for Right moves

Stay-Option Machine



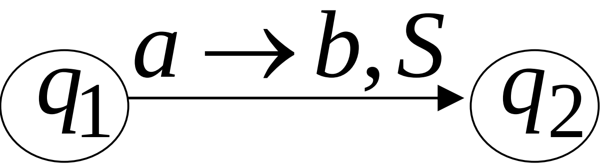
Simulation in Standard Machine



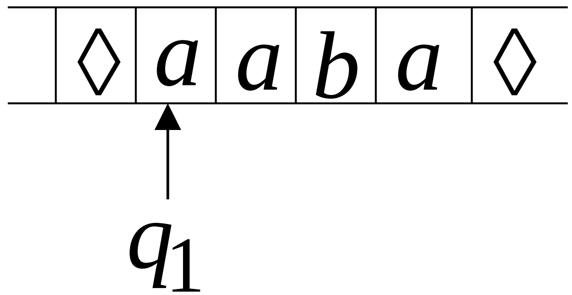
For every symbol x

Example

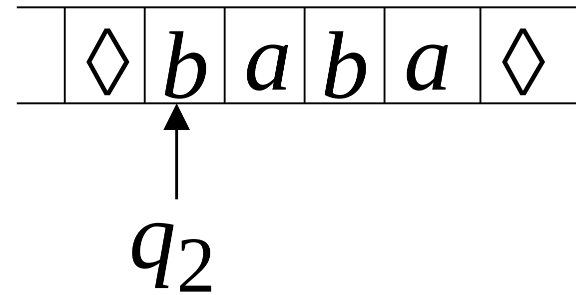
Stay-Option Machine:



1

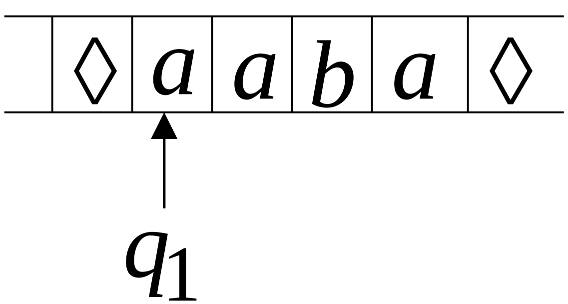


2

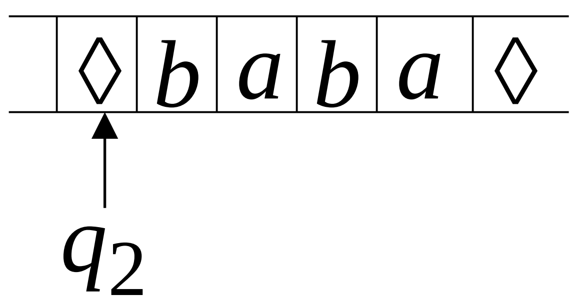


Simulation in Standard Machine:

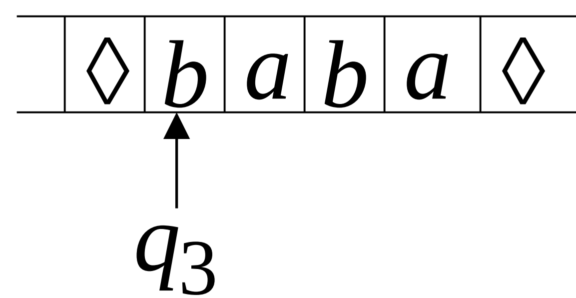
1



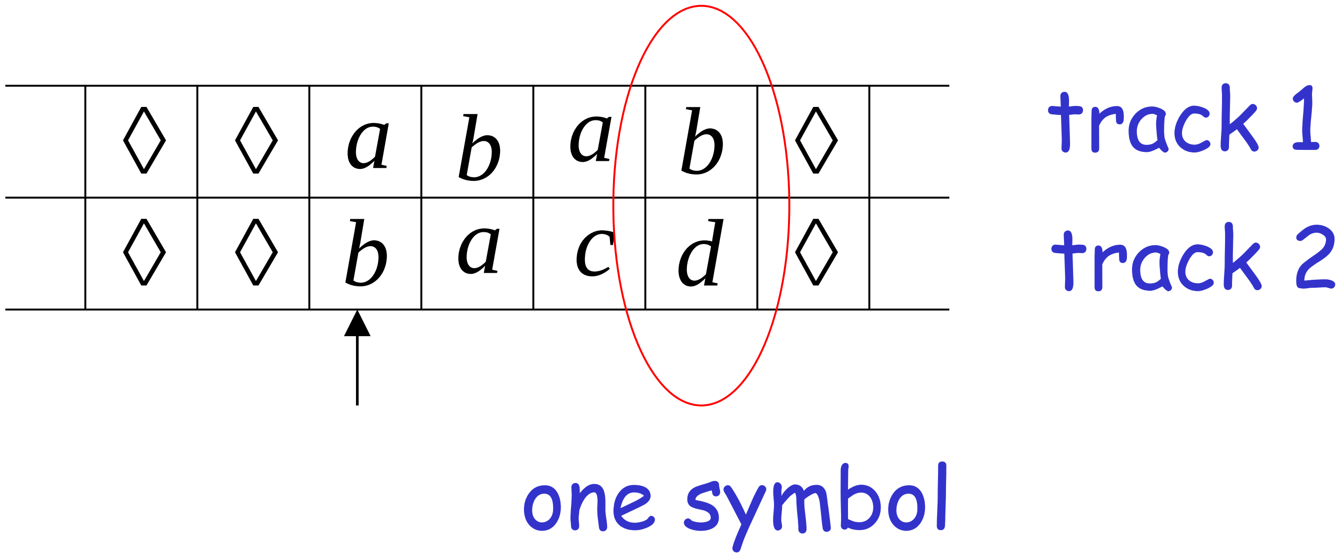
2



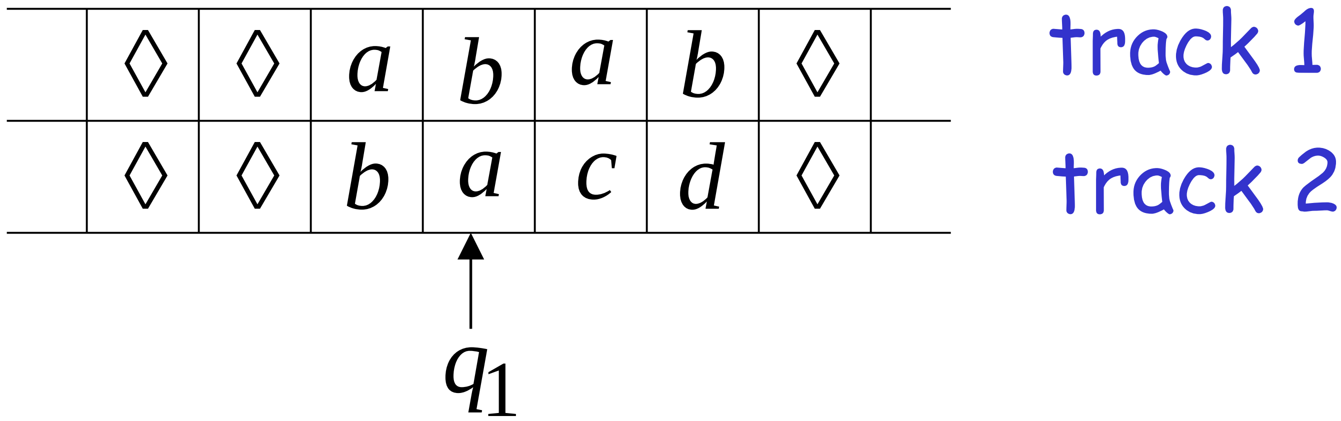
3

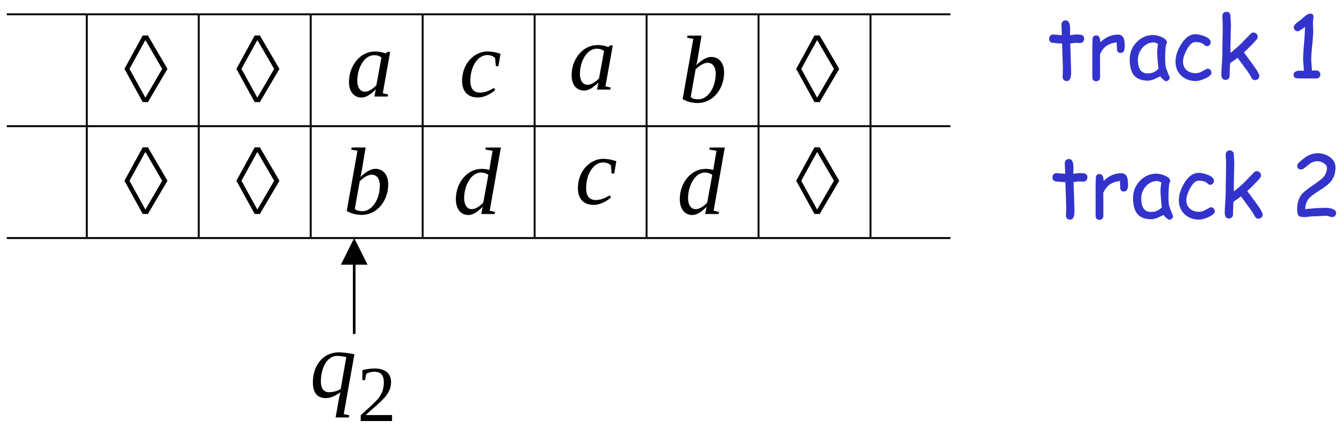


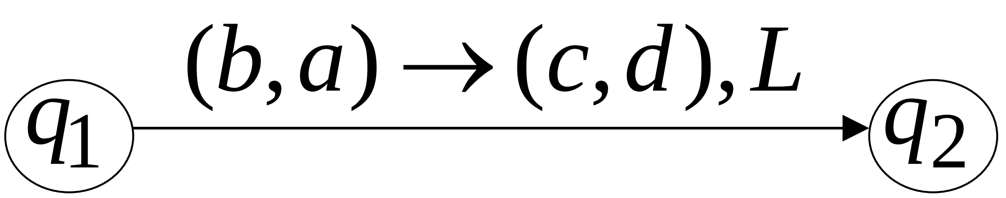
Standard Machine--Multiple Track Tape



$$\delta : Q \times \Gamma^n \rightarrow Q \times \Gamma^n \times \{L, R\}$$

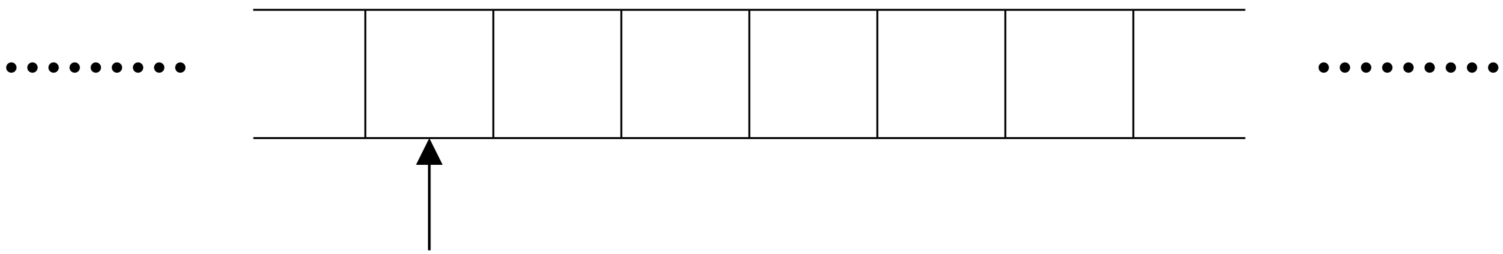

◊ ◊ a b a b ◊
◊ ◊ b a c d ◊
track 1
track 2
q_1


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

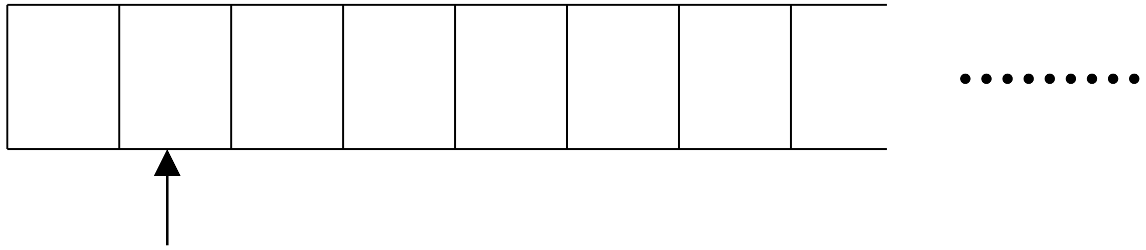

q_1
$(b,a) \rightarrow (c,d), L$
q_2

Semi-infinite tape machines simulate Standard Turing machines:

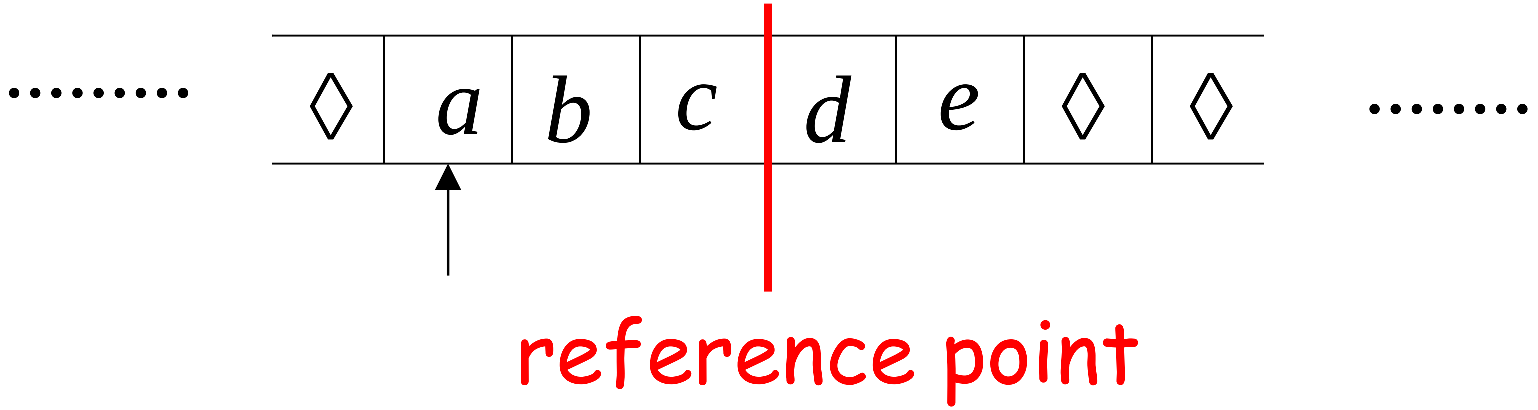
Standard machine



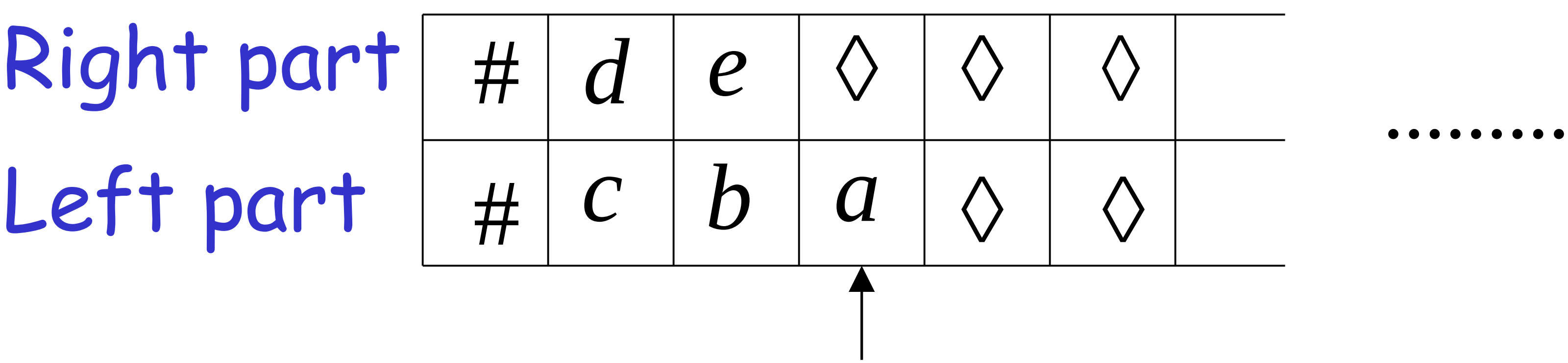
Semi-infinite tape machine



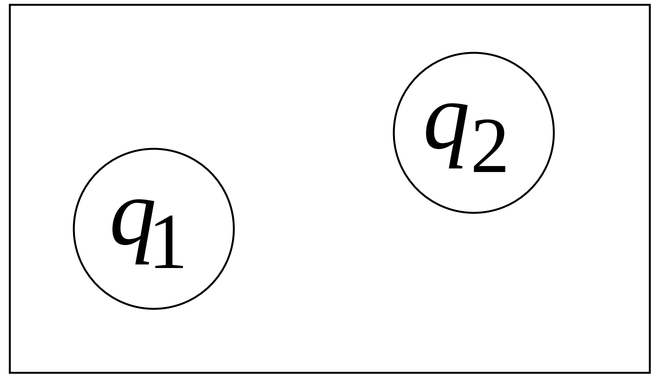
Standard machine



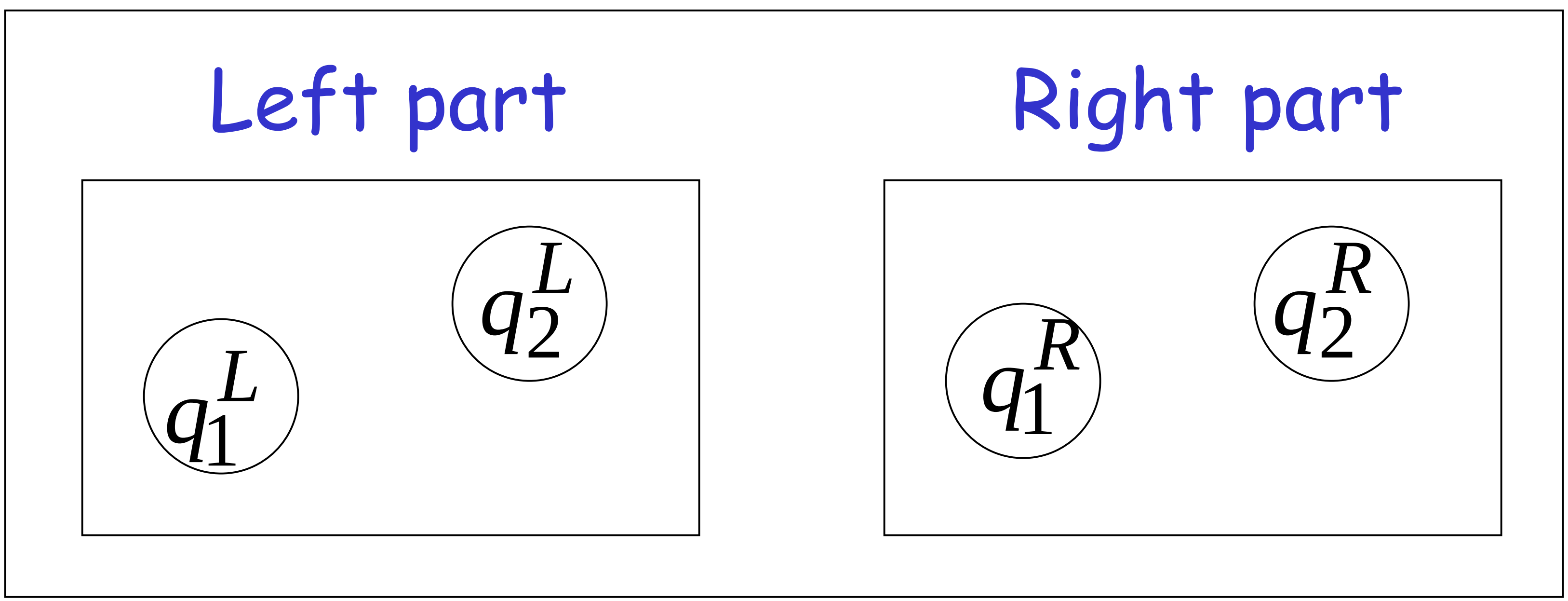
Semi-infinite tape machine with two tracks



Standard machine



Semi-infinite tape machine



Standard machine

$$q_1 \xrightarrow{a \rightarrow g, R} q_2$$

Semi-infinite tape machine

Right part

$$q_1^R \xrightarrow{(a, x) \rightarrow (g, x), R} q_2^R$$

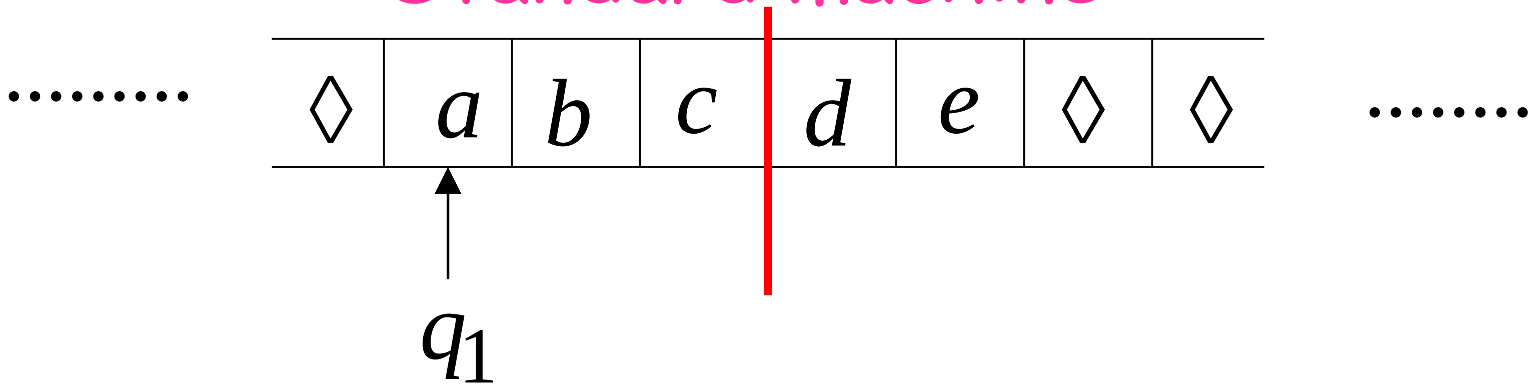
Left part

$$q_1^L \xrightarrow{(x, a) \rightarrow (x, g), L} q_2^L$$

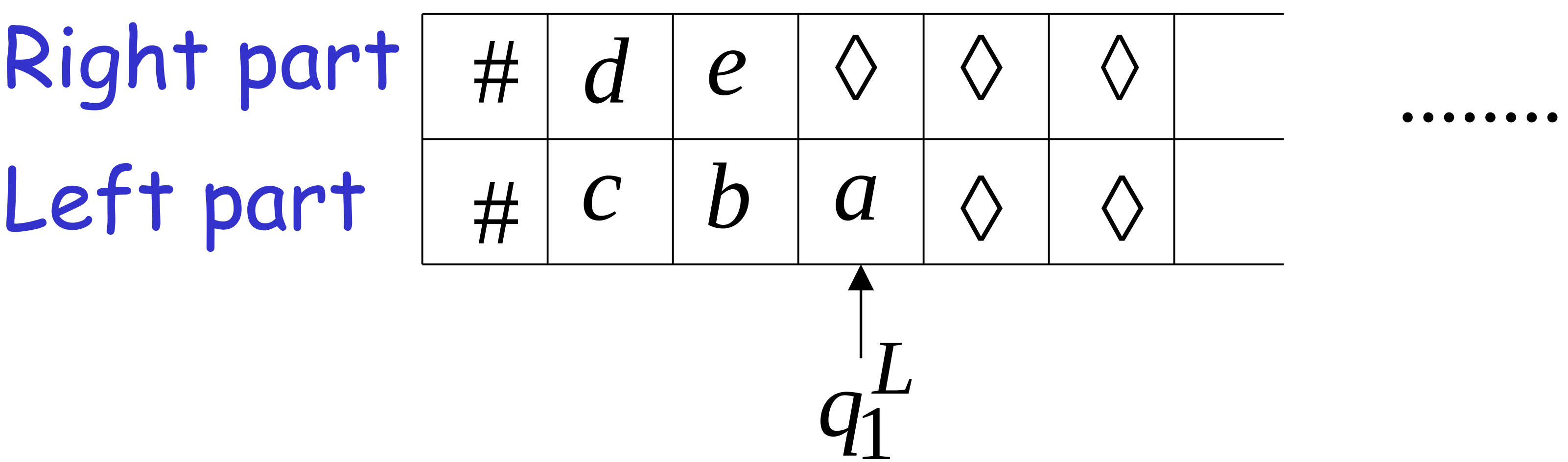
For all symbols x

Time 1

Standard machine

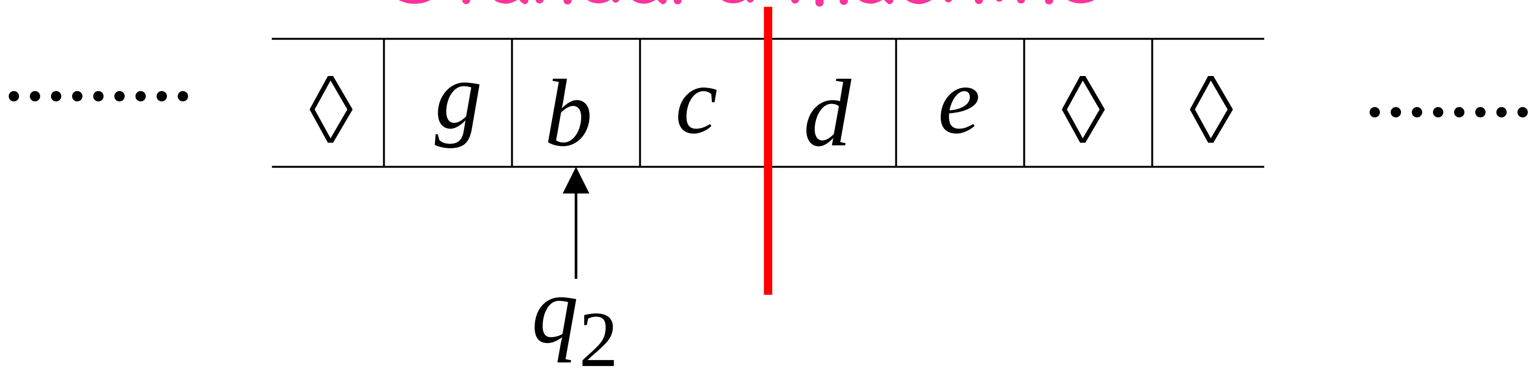


Semi-infinite tape machine



Time 2

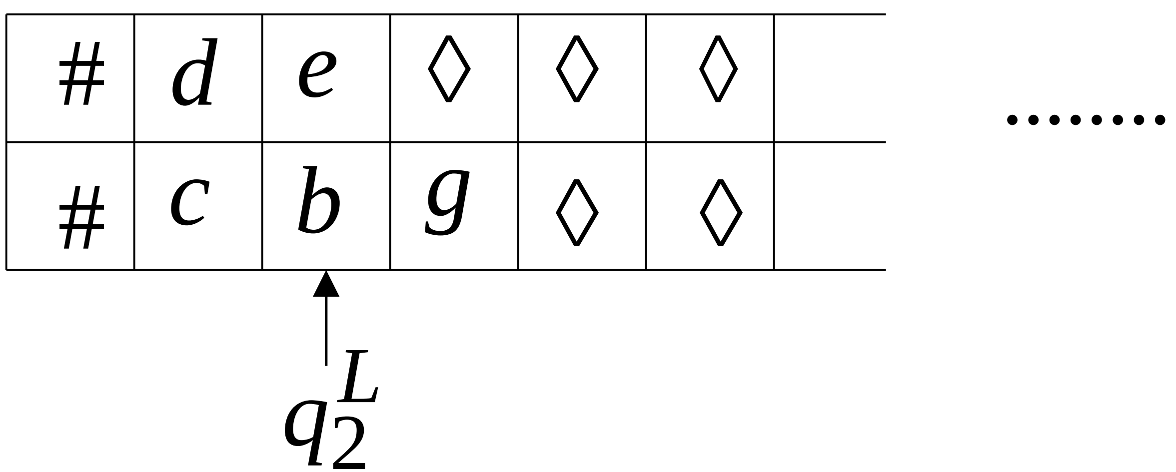
Standard machine



Semi-infinite tape machine

Right part

Left part



At the border:

Semi-infinite tape machine

Right part

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

Left part

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

Semi-infinite tape machine

Time 1

Right part	#	d	e	$\Diamond$	$\Diamond$	$\Diamond$		....
Left part	#	c	b	g	$\Diamond$	$\Diamond$		

q_1^L

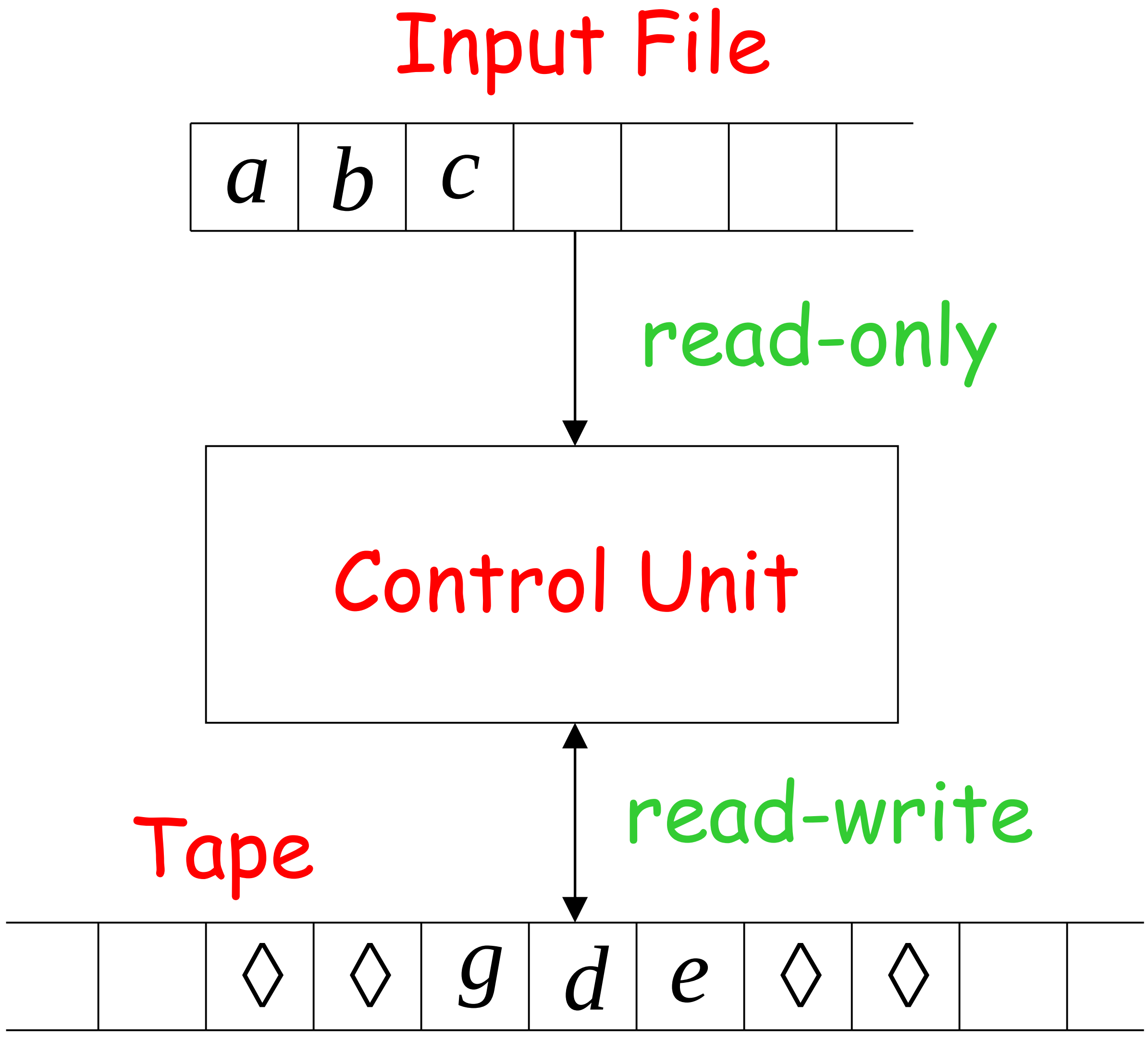
Time 2

Right part	#	d	e	$\Diamond$	$\Diamond$	$\Diamond$		....
Left part	#	c	b	g	$\Diamond$	$\Diamond$		

q_1^R

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

The Off-Line Machine



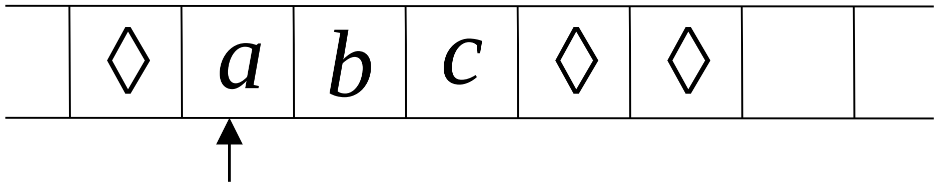
Off-line machines simulate Standard Turing Machines:

Off-line machine:

1. Copy input file to tape

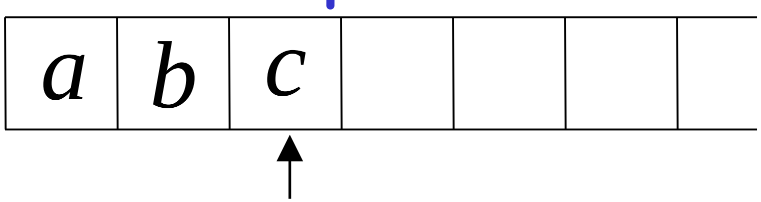
2. Continue computation as in Standard Turing machine

Standard machine

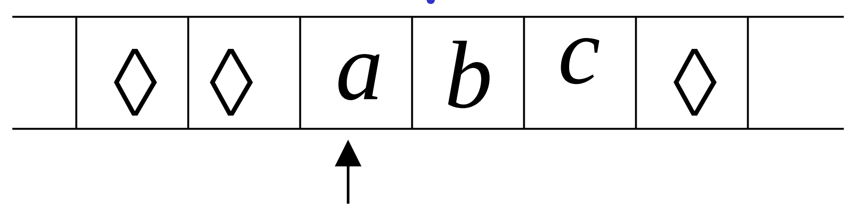


Off-line machine

Input File

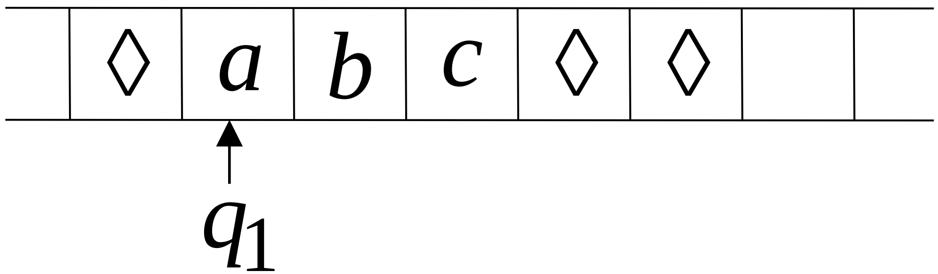


Tape



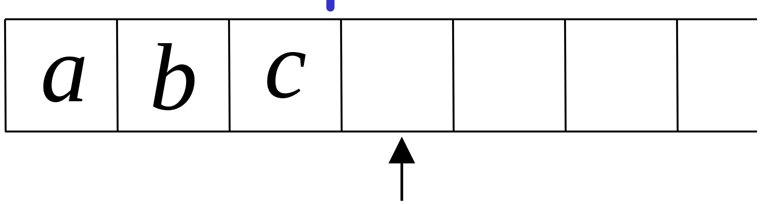
1. Copy input file to tape

Standard machine

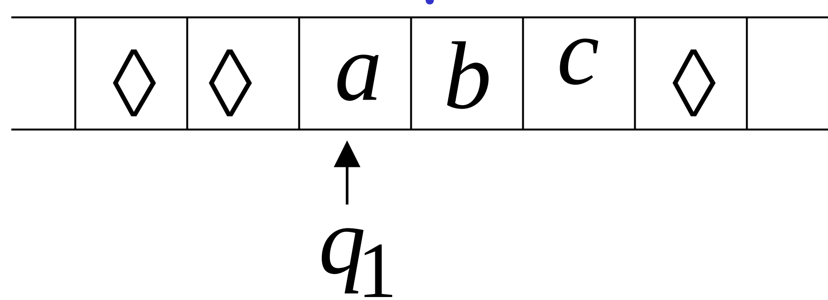


Off-line machine

Input File



Tape



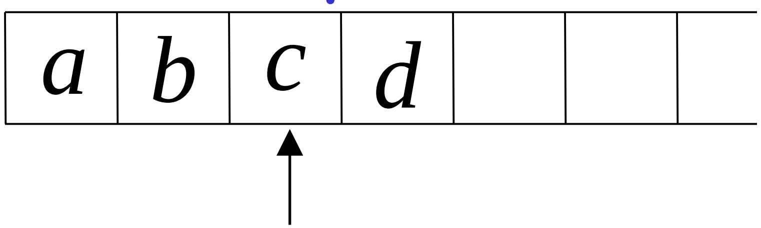
2. Do computations as in Turing machine

Standard Turing machines simulate
Off-line machines:

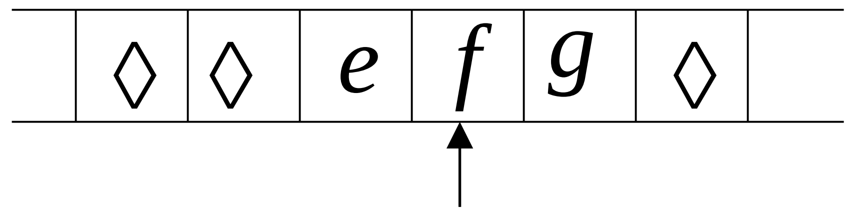
Use a Standard machine with four track tape
to keep track of
the Off-line input file and tape contents

Off-line Machine

Input File



Tape



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

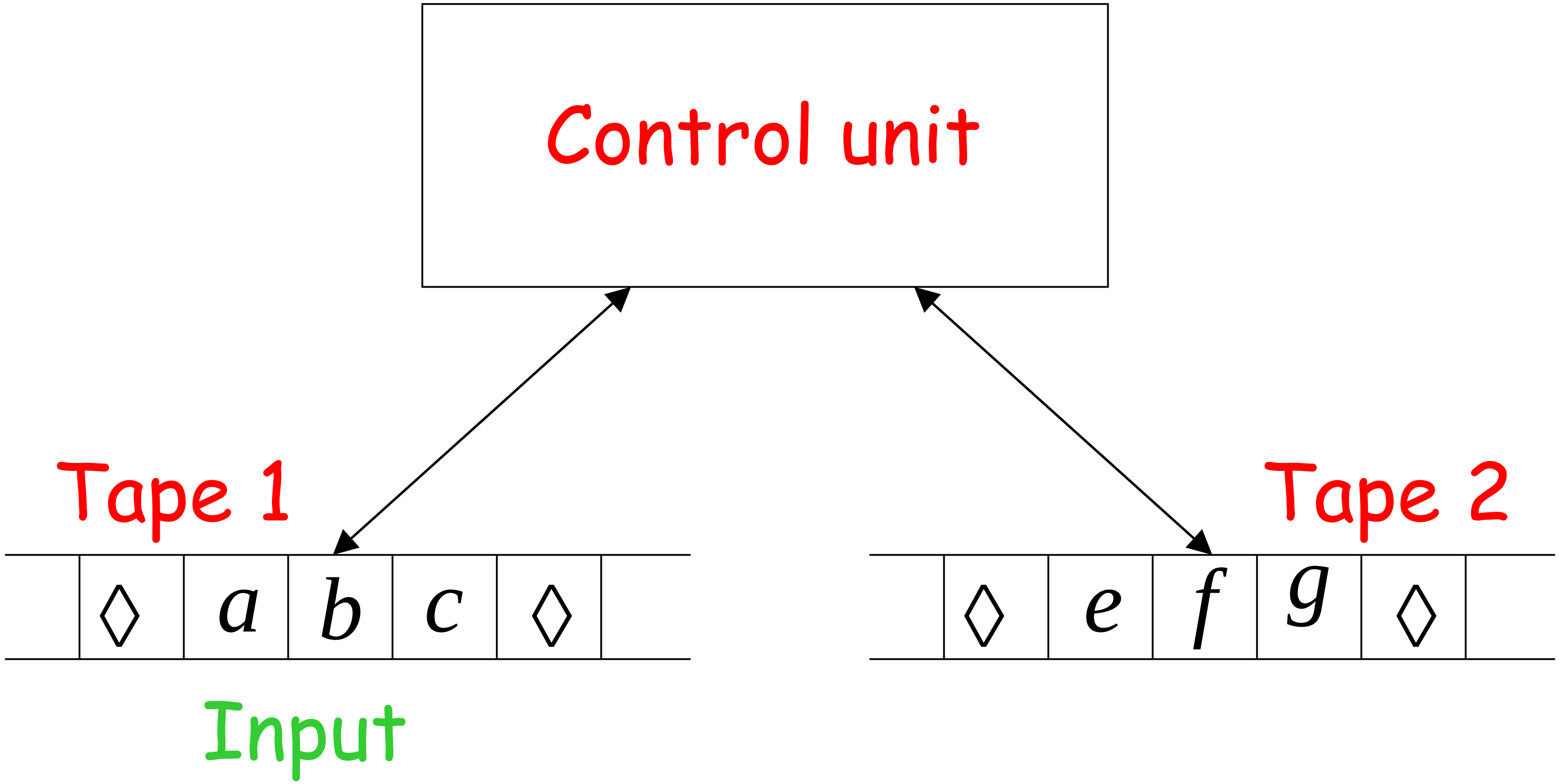
	#	a	b	c	d			Input File
	#	0	0	1	0			head position
		e	f	g				Tape
		0	1	0				head position

Repeat for each state transition:

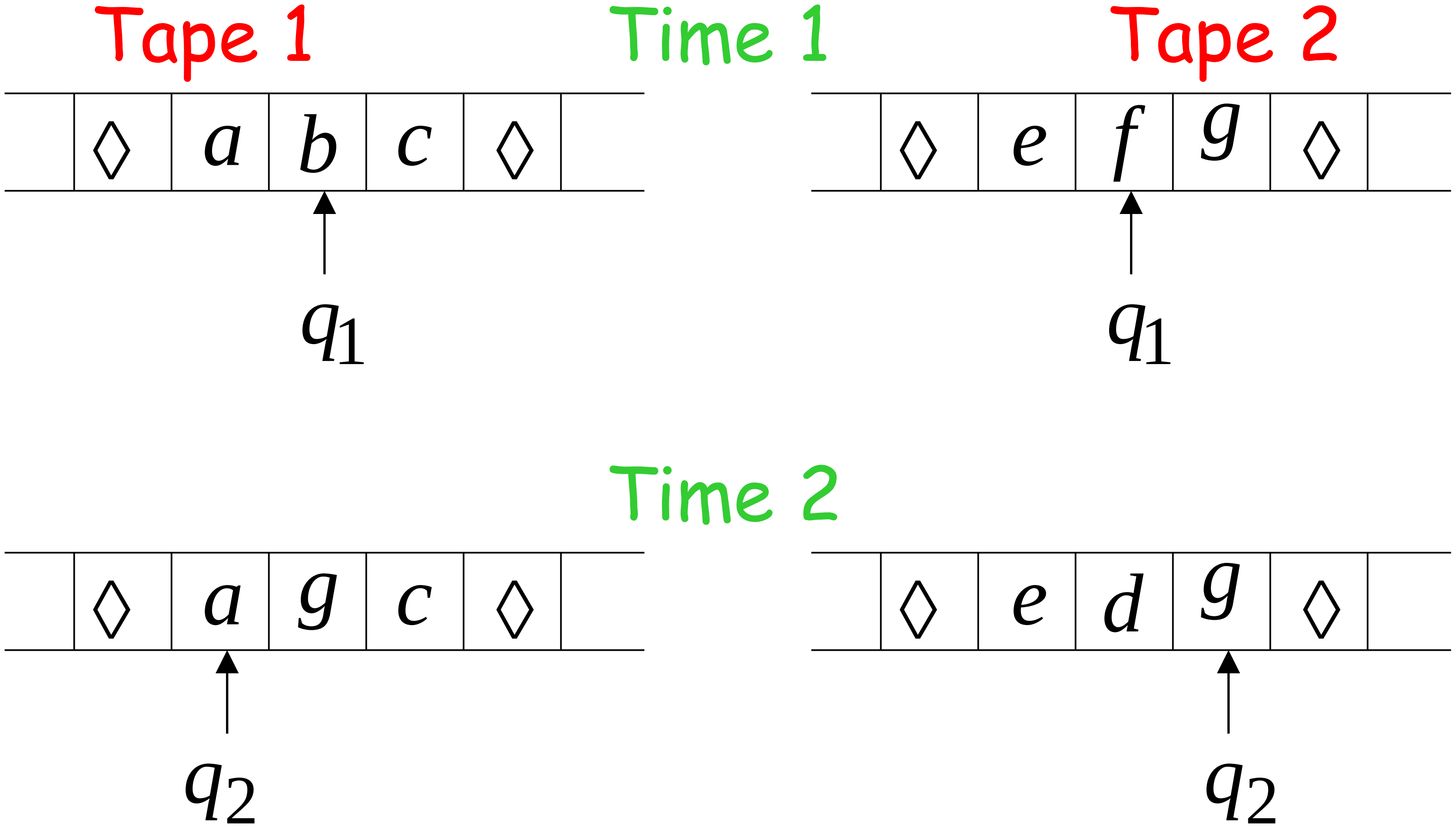
- Return to reference point
- Find current input file symbol
- Find current tape symbol
- Make transition

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

Multitape Turing Machines



$$\delta : Q \times \Gamma^n \to Q \times \Gamma^n \times \{L, R\}^n$$



$$q_1 \xrightarrow{(b,f)\to(g,d),L,R} q_2$$

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

	Tape 1					
	◊	a	b	c	◊	

	Tape 2					
	◊	e	f	g	h	◊

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 times

Two-tape machine:

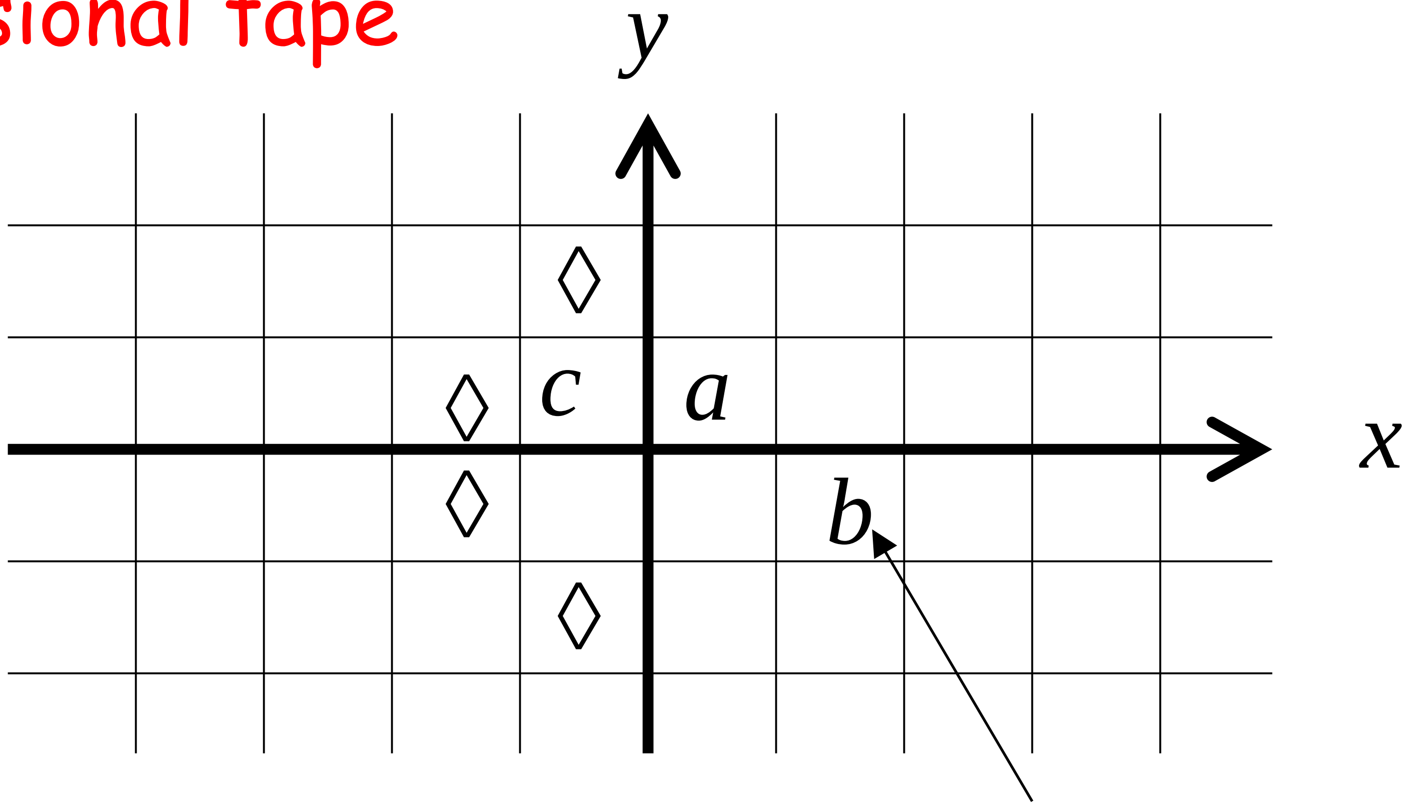
Copy b^n to tape 2 (n steps)

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

U: up D: down

HEAD

Position: +2, -1

$$\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R, U, D\},$$

Multidimensional machines simulate Standard machines:

Use one dimension

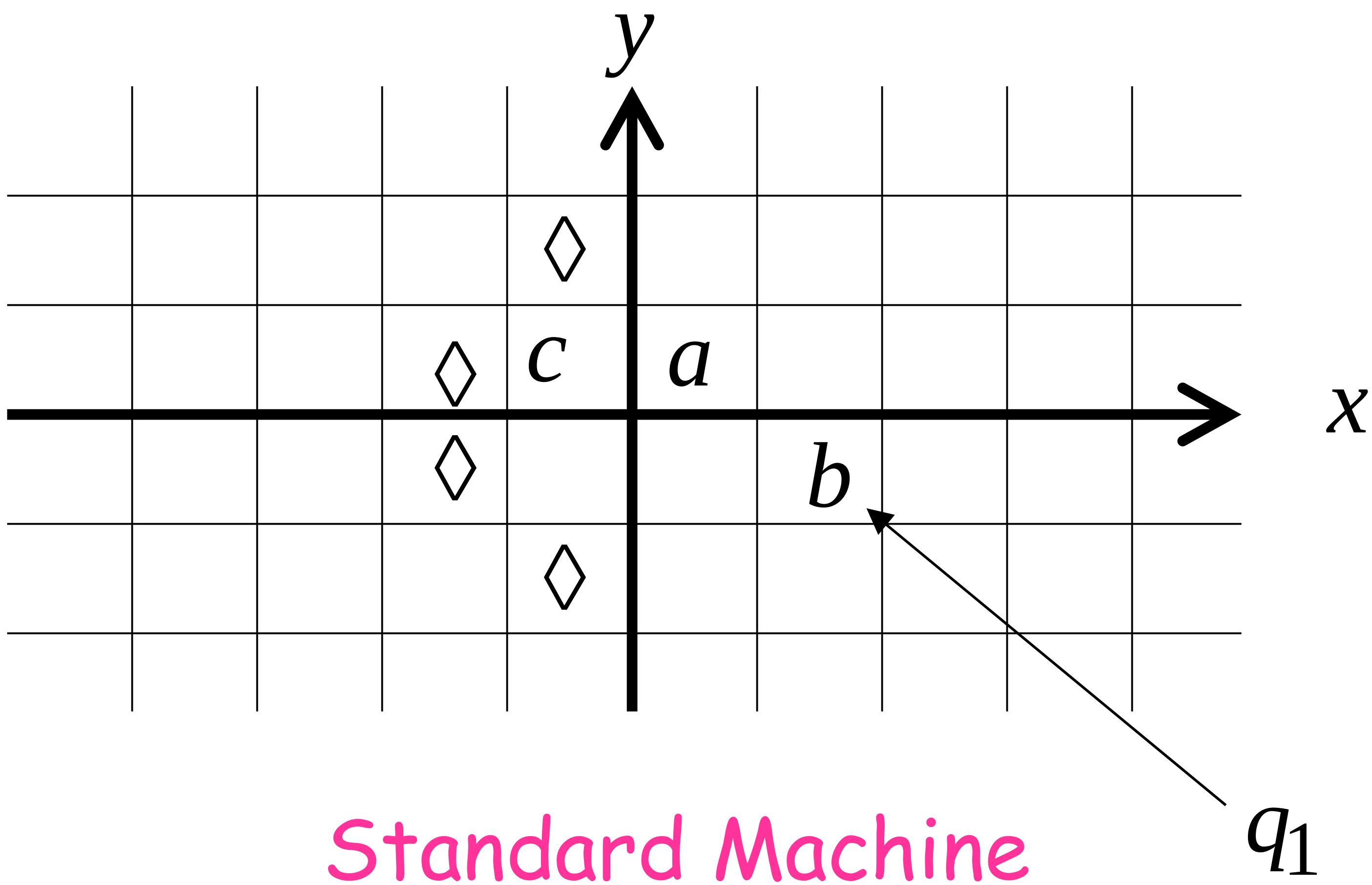
Standard machines simulate Multidimensional machines:

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

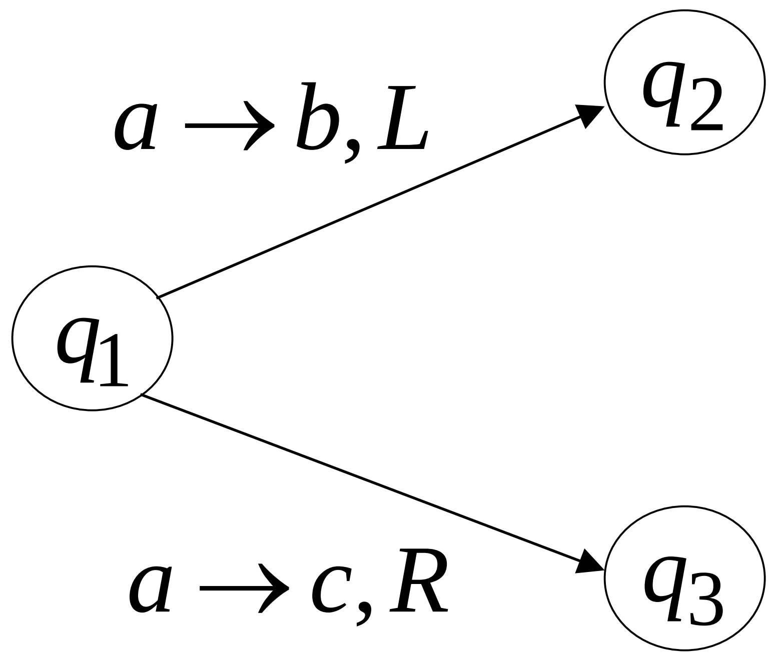
Standard machine:

Repeat for each transition

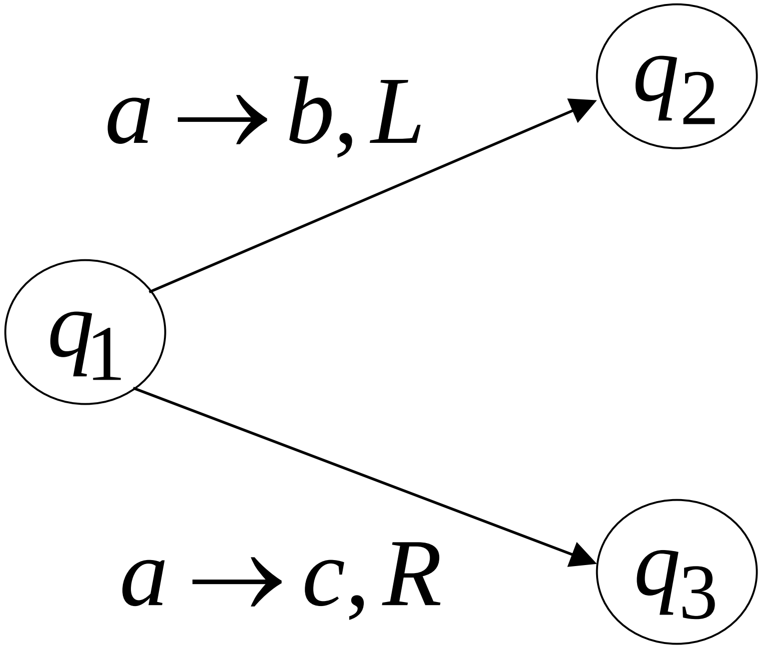
- Update current symbol
- Compute coordinates of next position
- Go to new position

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

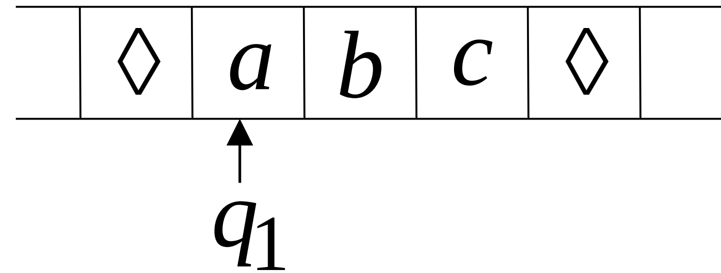
NonDeterministic Turing Machines



Non Deterministic Choice

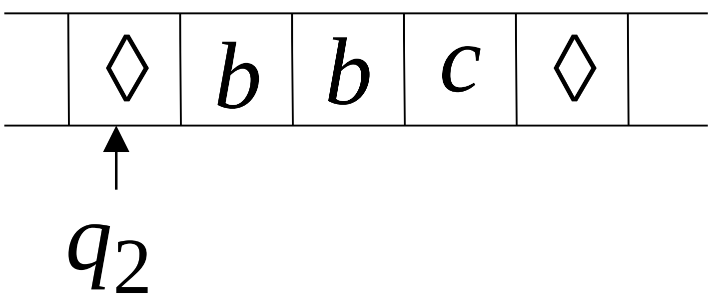


Time 0

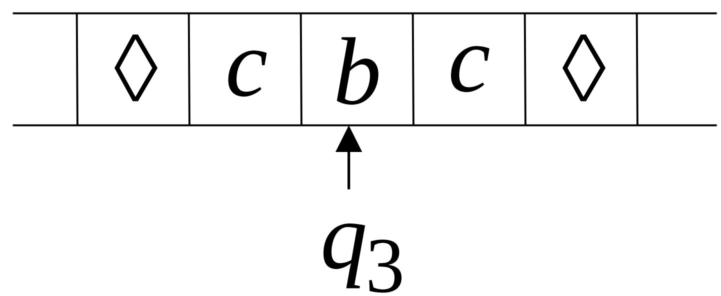


Time 1

Choice 1



Choice 2



Input string w is accepted if this a possible computation

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

Initial configuration

Final Configuration

Final state

$$\delta : Q \times \Gamma \rightarrow 2^{Q \times \Gamma \times \{L, R\}}.$$

NonDeterministic Machines simulate Standard (deterministic) Machines:

Every deterministic machine is also a nondeterministic machine

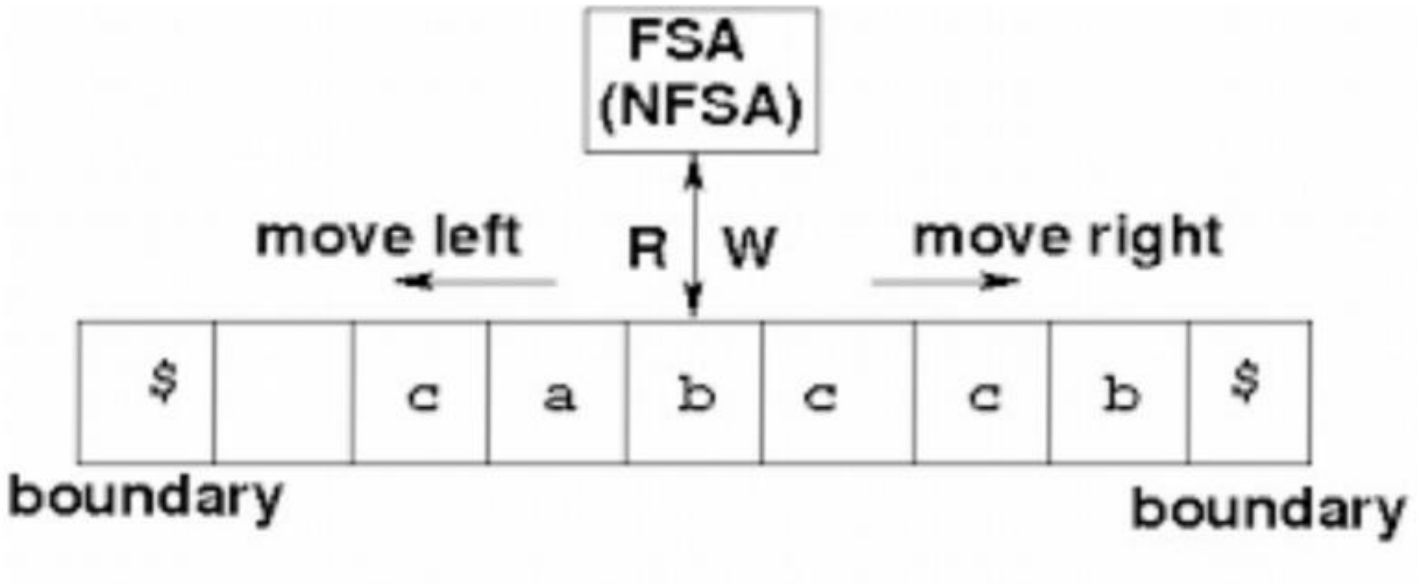
Theorem: NonDeterministic Machines have the same power with Deterministic machines

Remark:

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

Linear Bounded Automata (LBA)

- A non-deterministic Turing machine that uses only the tape space occupied by the input.



Linear Bounded Automata

- Its input alphabet includes two special symbols, serving as left and right endmarkers.
- Linear bounded automata are acceptors for the class of context-sensitive languages.
- linear bounded automata are more powerful than pushdown automata, since neither of the languages is context free.

Linear Bounded Automata

The language $L = \{a^n b^n c^n : n \geq 1\}$

is accepted by some linear bounded automaton. The computation outlined there does not require space outside the original input.

~~a~~bbaab~~a~~bbaab

~~a~~~~b~~baab~~a~~~~b~~baab

abbaababbaab

abbaababbaab

Recursively Enumerable and Recursive Languages

Definition:

A language is **recursively enumerable** or **Turing-recognizable** if some Turing machine accepts it

Let L be a recursively enumerable language

and M the Turing Machine that accepts it

For string w :

if $w \in L$ then M halts in a final state

if $w \notin L$ then M halts in a non-final state

or loops forever

Definition:

A language is **recursive or decidable** if some Turing machine accepts it and halts on any input string

In other words:

A language is recursive if there is a membership algorithm for it

Let L be a recursive language

and M the Turing Machine that accepts it

For string w :

if $w \in L$ then M halts in a final state

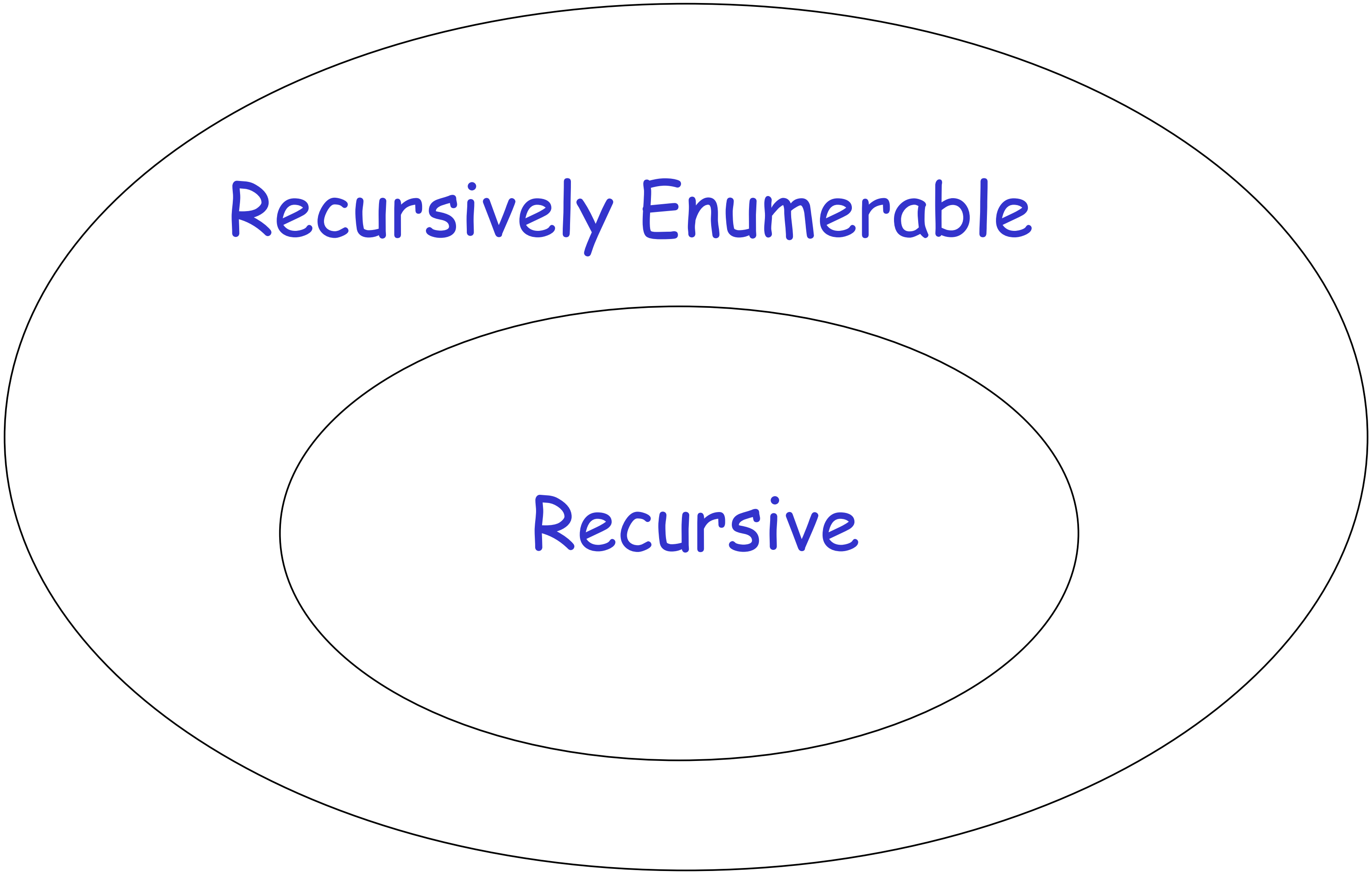
if $w \notin L$ then M halts in a non-final state

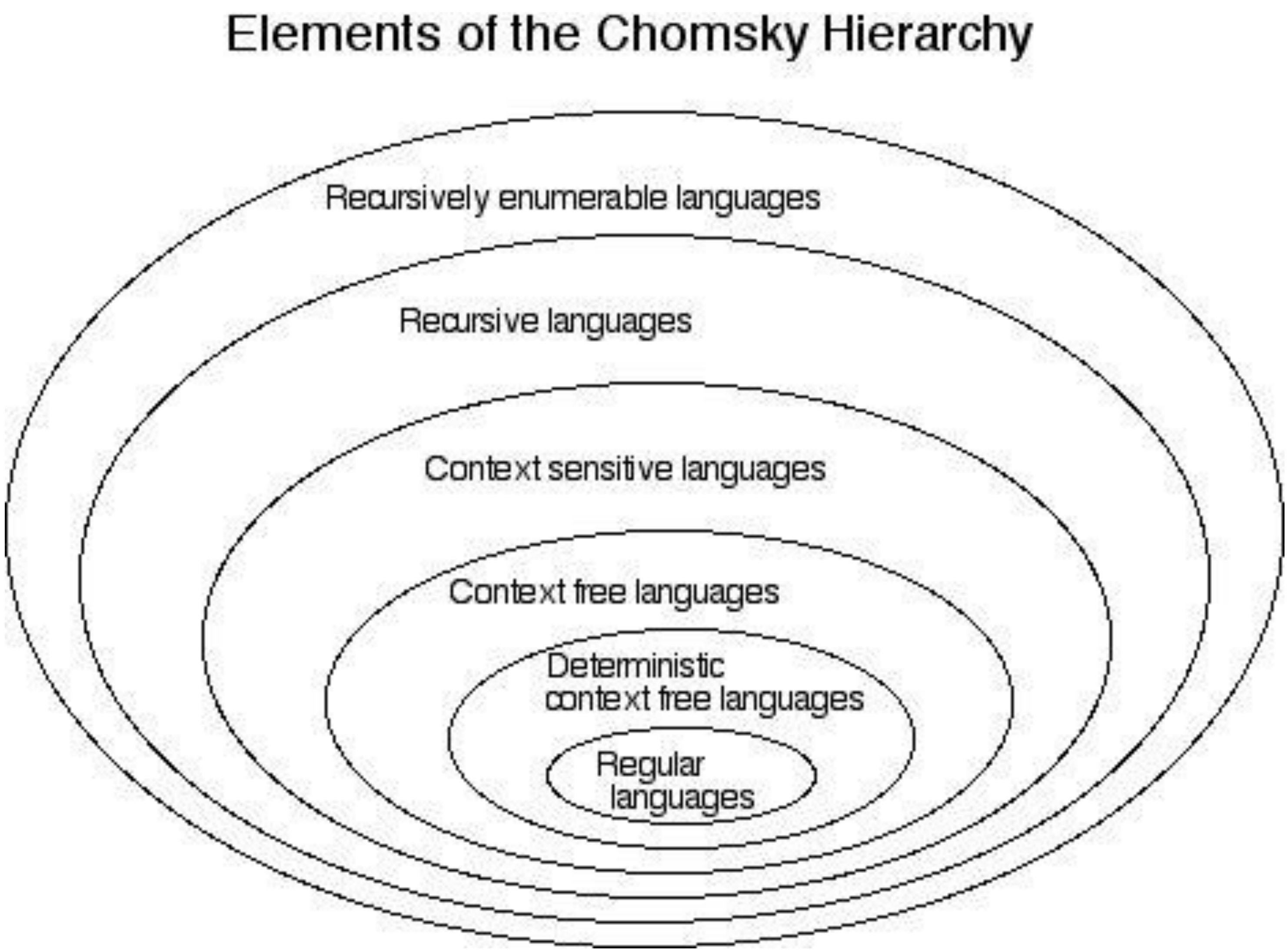
A Turing Machine **decides** a language if it accepts all strings in the language and rejects all strings not in the language

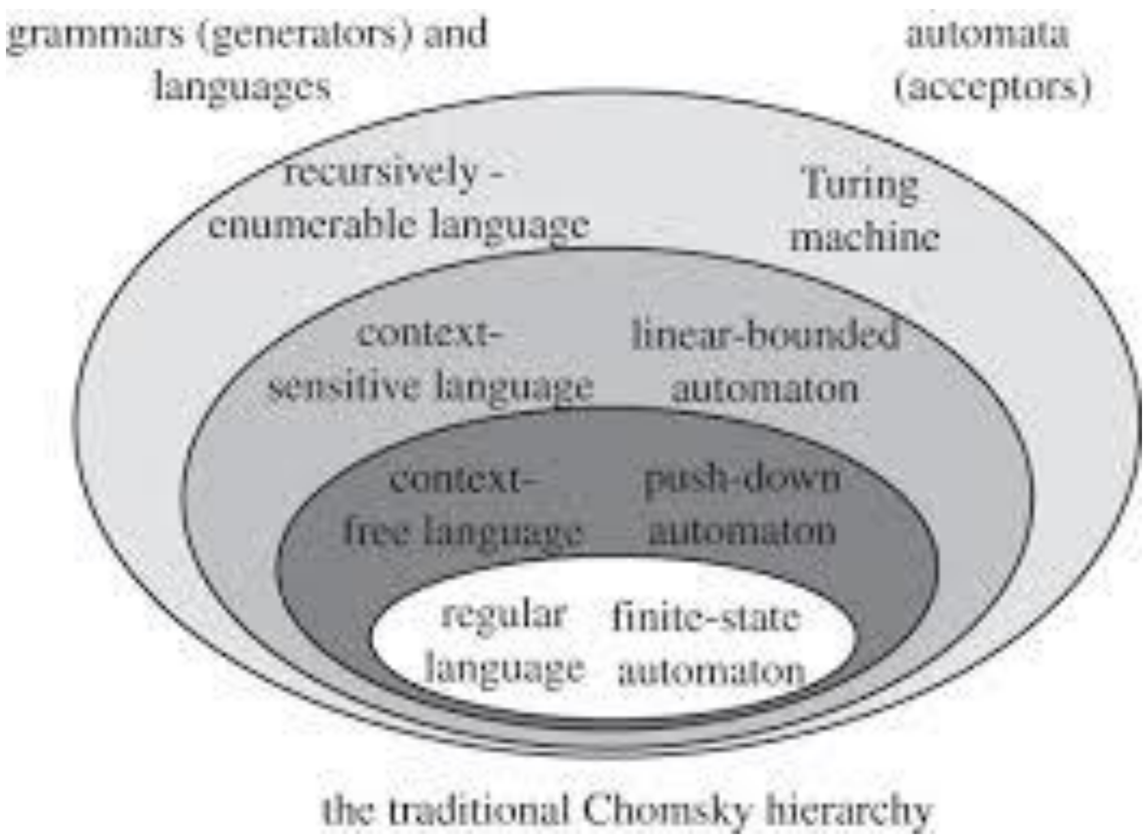
- There is a specific language which is not recursively enumerable (not accepted by any Turing Machine)

- There is a specific language which is recursively enumerable but not recursive

Non Recursively Enumerable




Elements of the Chomsky Hierarchy
Recursively enumerable languages
Recursive languages
Context sensitive languages
Context free languages
Deterministic
context free languages
Regular
languages


grammars (generators) and languages
automata (acceptors)
recursively - enumerable language
Turing machine
context- sensitive language
linear-bounded automaton
context- free language
push-down automaton
regular language
finite-state automaton
the traditional Chomsky hierarchy