

# Turing's Thesis

# Turing's thesis (1930):

Any computation carried out  
by mechanical means  
can be performed by a Turing Machine

## Algorithm:

An algorithm for a problem is a Turing Machine which solves the problem

The algorithm describes the steps of the mechanical means

This is easily translated to computation steps of a Turing machine

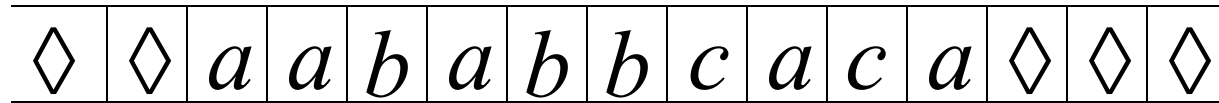
When we say: There exists an algorithm

We mean: There exists a Turing Machine  
that executes the algorithm

# Variations of the Turing Machine

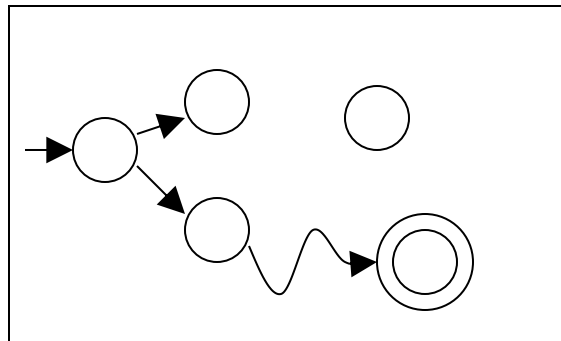
# The Standard Model

## Infinite Tape



Read-Write Head (Left or Right)

## Control Unit



Deterministic

# Variations of the Standard Model

- Turing machines with:
- Stay-Option
  - Semi-Infinite Tape
  - Off-Line
  - Multitape
  - Multidimensional
  - Nondeterministic

Different Turing Machine **Classes**

Same Power of two machine classes:

both classes accept the  
same set of languages

We will prove:

each new class has the same power  
with Standard Turing Machine

(accept Turing-Recognizable Languages)



## Same Power of two classes means:

for any machine  $M_1$  of first class

there is a machine  $M_2$  of second class

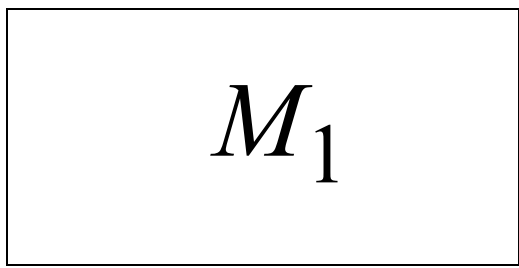
such that:  $L(M_1) = L(M_2)$

and vice-versa

**Simulation:** A technique to prove same power.  
Simulate the machine of one class  
with a machine of the other class

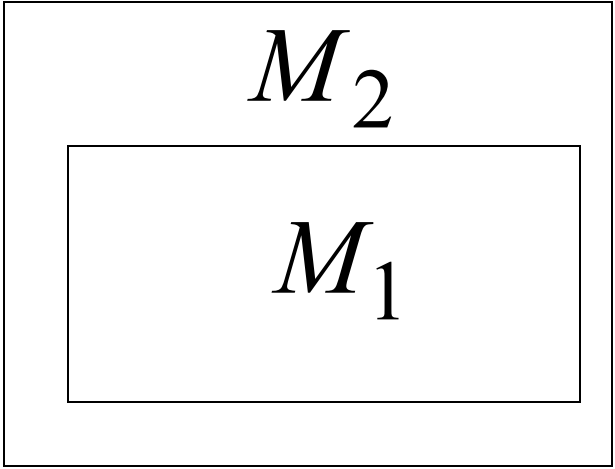
First Class

Original Machine



Second Class

Simulation Machine



simulates  $M_1$

Configurations in the Original Machine  $M_1$   
 have corresponding configurations  
 in the Simulation Machine  $M_2$

Original Machine:  $M_1$

$d_0$	$\phi$	$d_1$	$\phi$	$\Lambda$	$\phi$	$d_n$
↕		↕				↕

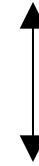
Simulation Machine:  $M_2$

$d'_0$	$\phi$	$d'_1$	$\phi$	$\Lambda$	$\phi$	$d'_n$
	*		*			*

# Accepting Configuration

Original Machine:

$d_f$



Simulation Machine:

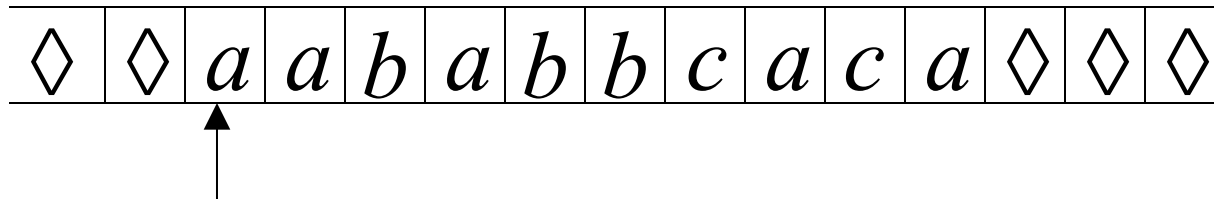
$d'_f$

the Simulation Machine  
and the Original Machine  
accept the same strings

$$L(M_1) = L(M_2)$$

# Turing Machines with Stay-Option

The head can stay in the same position

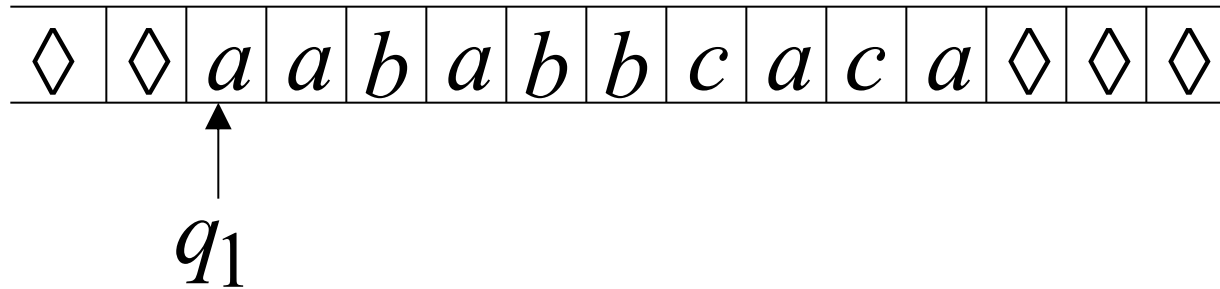


Left, Right, Stay

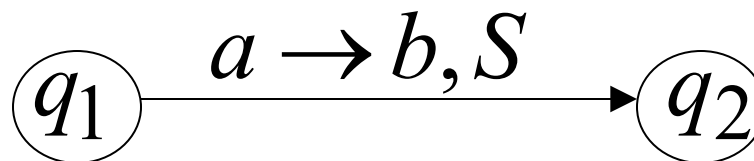
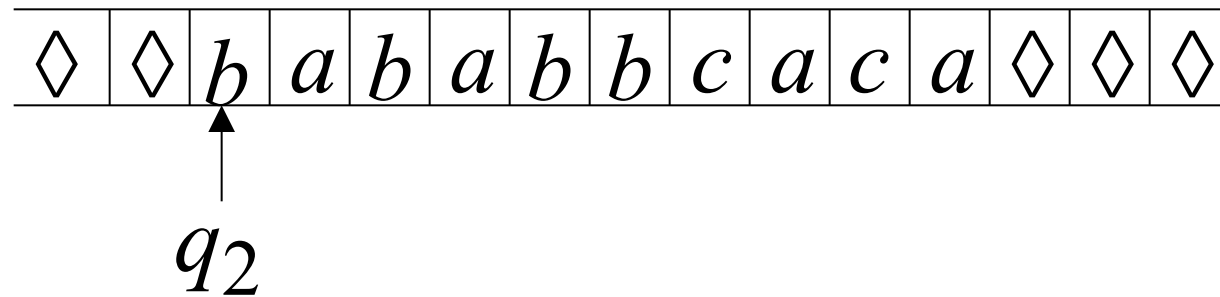
L,R,S: possible head moves

Example:

Time 1



Time 2



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

- Proof:**
1. Stay-Option Machines  
simulate Standard Turing machines
  2. Standard Turing machines  
simulate Stay-Option machines

# 1. Stay-Option Machines

simulate Standard Turing machines

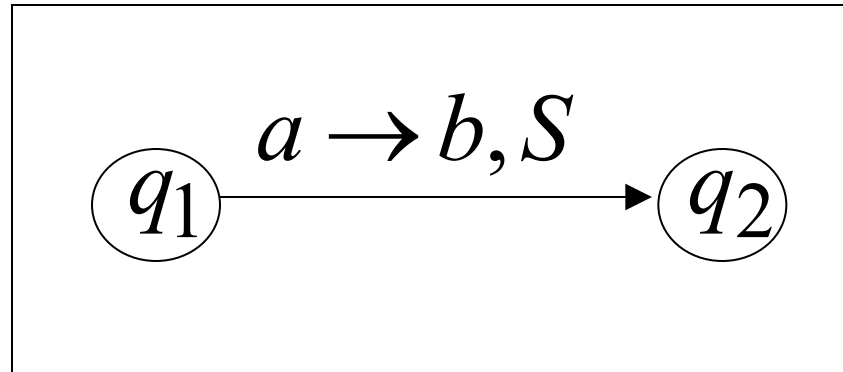
**Trivial:** any standard Turing machine  
is also a Stay-Option machine



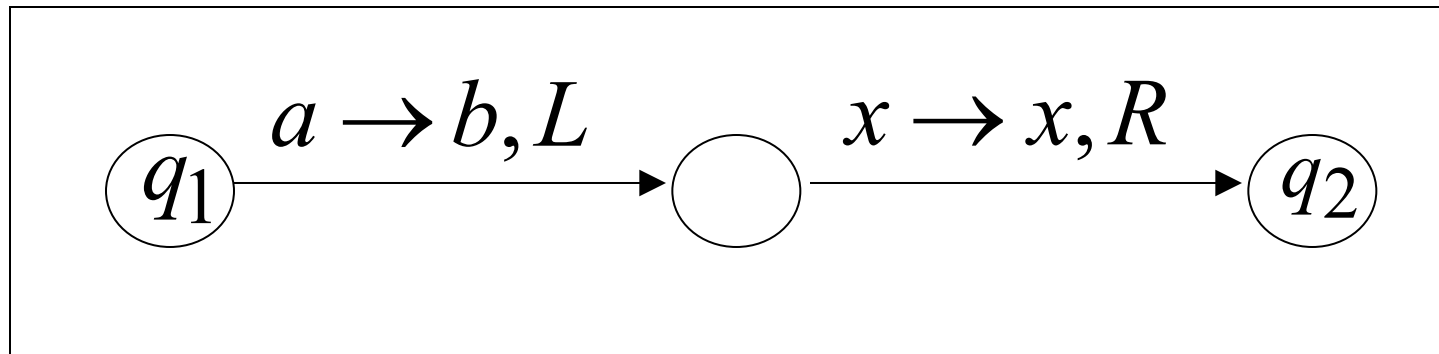
## 2. Standard Turing machines simulate Stay-Option machines

We need to simulate the **stay** head option  
with two head moves, one **left** and one **right**

# Stay-Option Machine



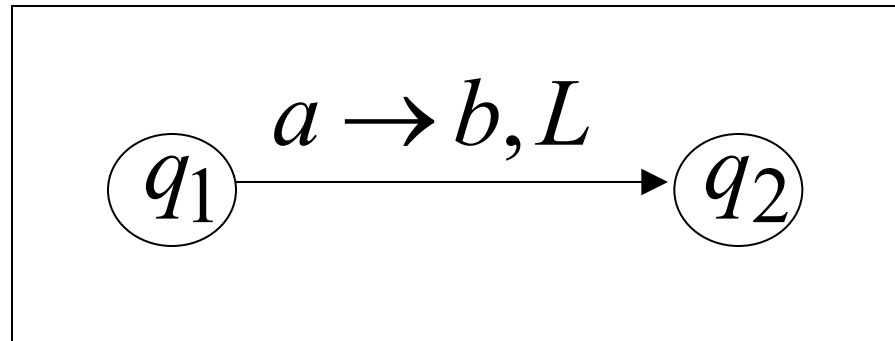
## Simulation in Standard Machine



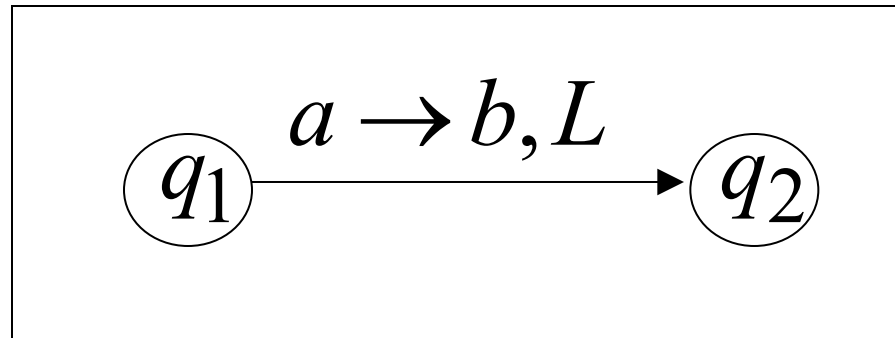
For every possible tape symbol  $x$

For other transitions nothing changes

## Stay-Option Machine



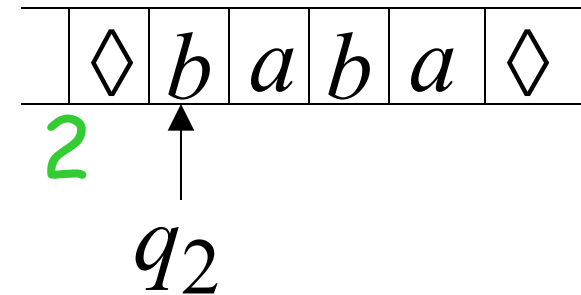
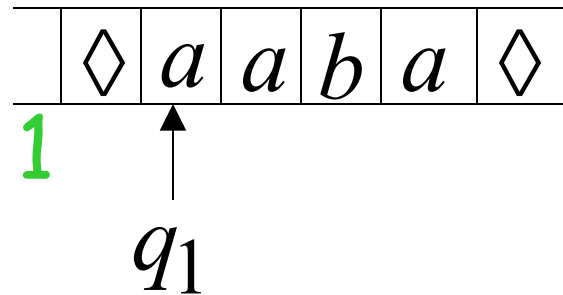
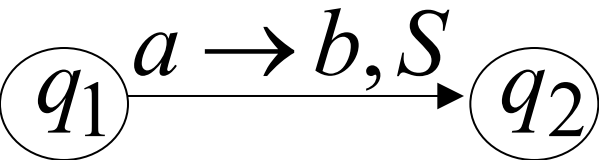
## Simulation in Standard Machine



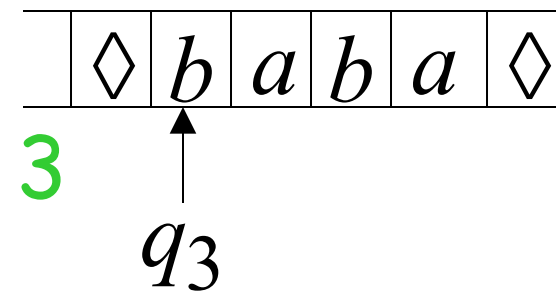
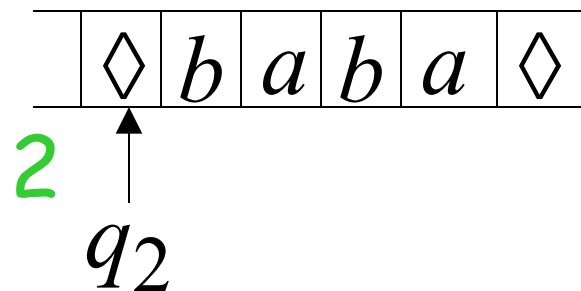
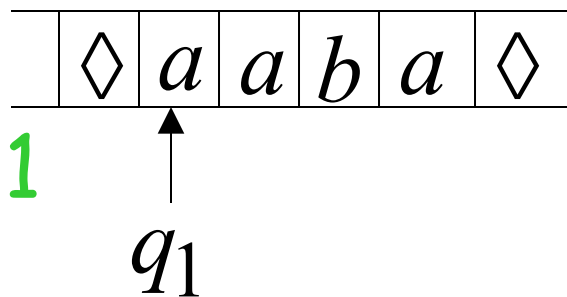
Similar for Right moves

# example of simulation

## Stay-Option Machine:



## Simulation in Standard Machine:

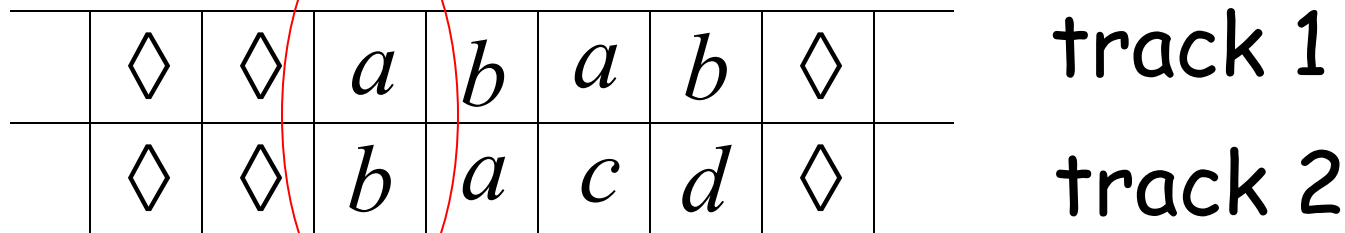


END OF PROOF

# Multiple Track Tape

A useful trick to perform more complicated simulations

One Tape



One head

One symbol (*a*, *b*)

	◇	◇	<i>a</i>	<i>b</i>	<i>a</i>	<i>b</i>	◇	
	◇	◇	<i>b</i>	<i>a</i>	<i>c</i>	<i>d</i>	◇	

track 1

track 2

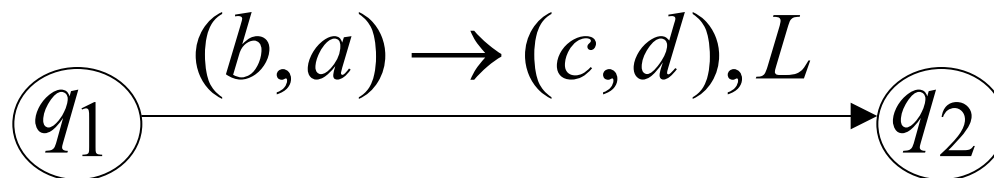
$q_1$

	◇	◇	<i>a</i>	<i>c</i>	<i>a</i>	<i>b</i>	◇	
	◇	◇	<i>b</i>	<i>d</i>	<i>c</i>	<i>d</i>	◇	

track 1

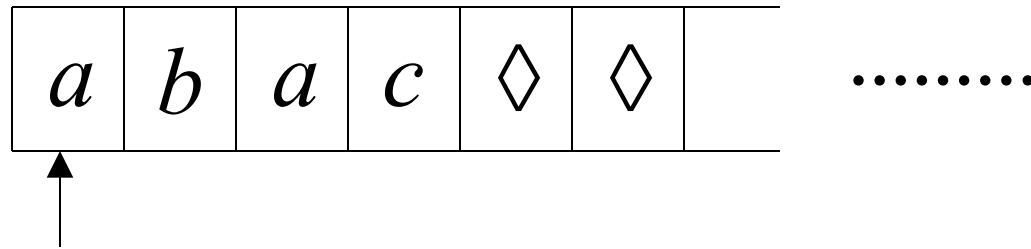
track 2

$q_2$



# Semi-Infinite Tape

The head extends infinitely only to the right



- Initial position is the leftmost cell
- When the head moves left from the border, it returns to the same position

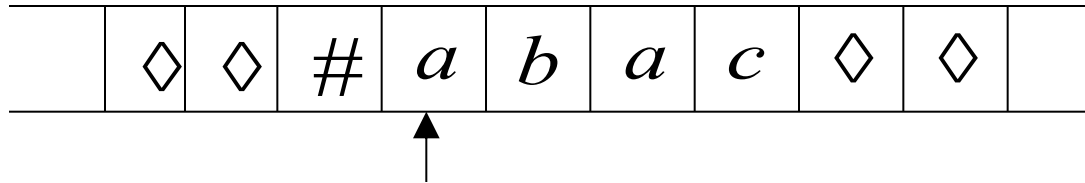
**Theorem:** Semi-Infinite machines  
have the same power with  
Standard Turing machines

**Proof:** 1. Standard Turing machines  
simulate Semi-Infinite machines

2. Semi-Infinite Machines  
simulate Standard Turing machines



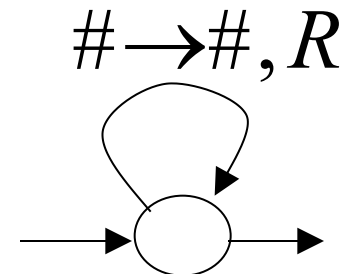
# 1. Standard Turing machines simulate Semi-Infinite machines:



## Standard Turing Machine

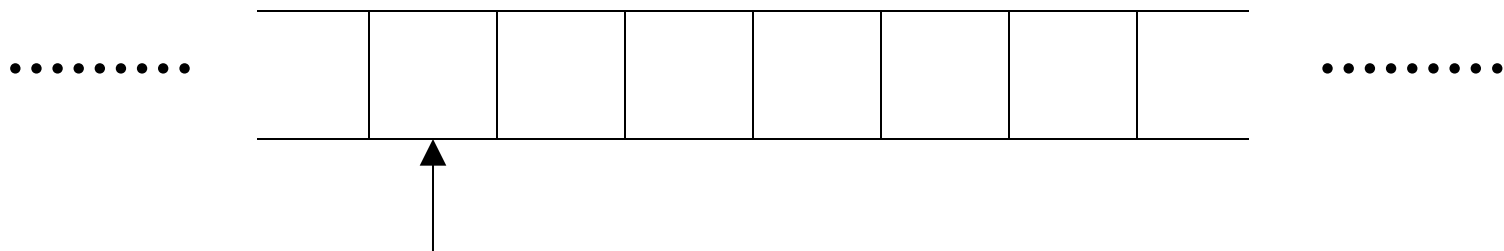
a. insert special symbol  $\#$   
at left of input string

b. Add a self-loop  
to every state  
(except states with no  
outgoing transitions)

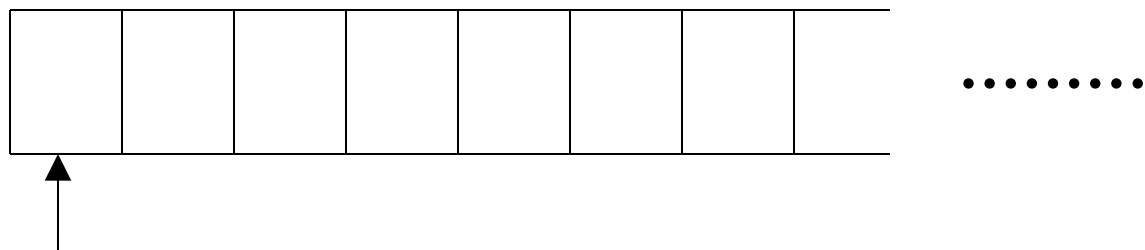


## 2. Semi-Infinite tape machines simulate Standard Turing machines:

Standard machine

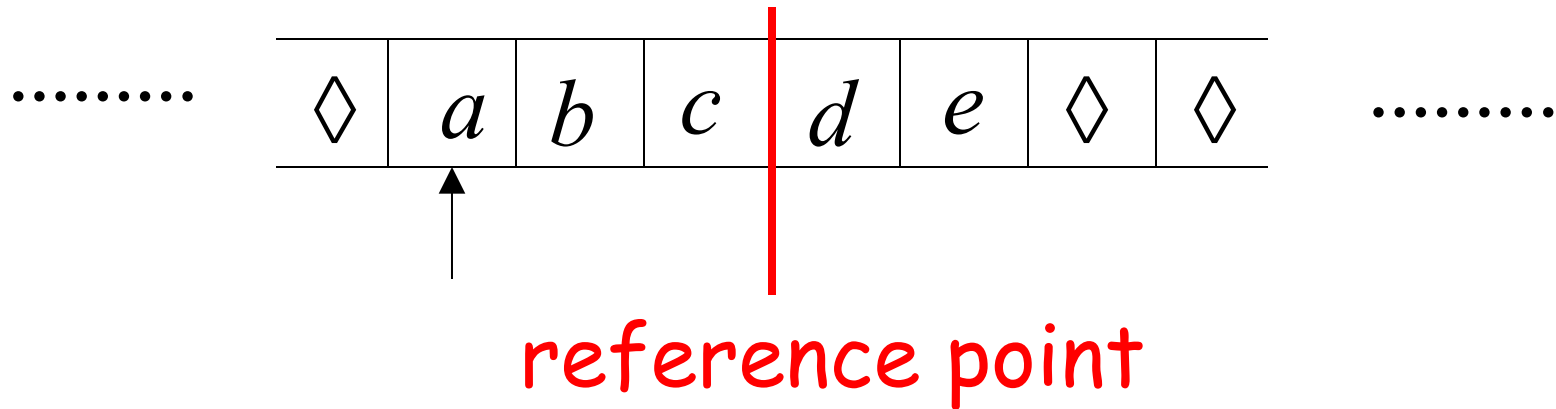


Semi-Infinite tape machine

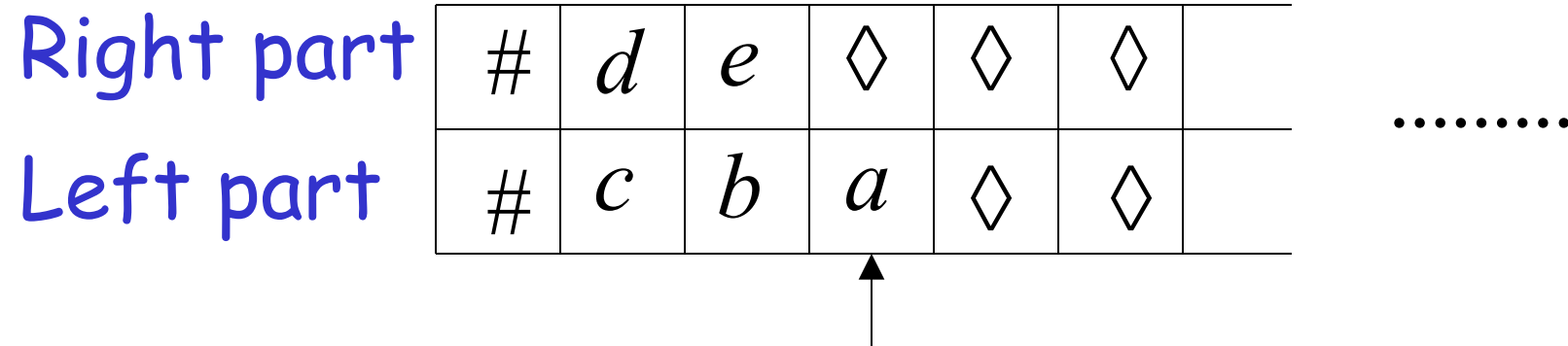


Squeeze infinity of both directions  
in one direction

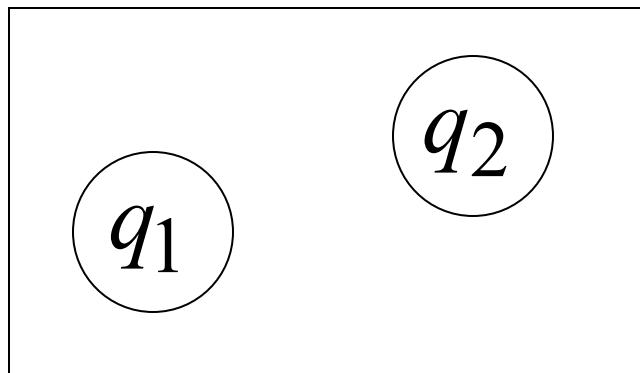
## Standard machine



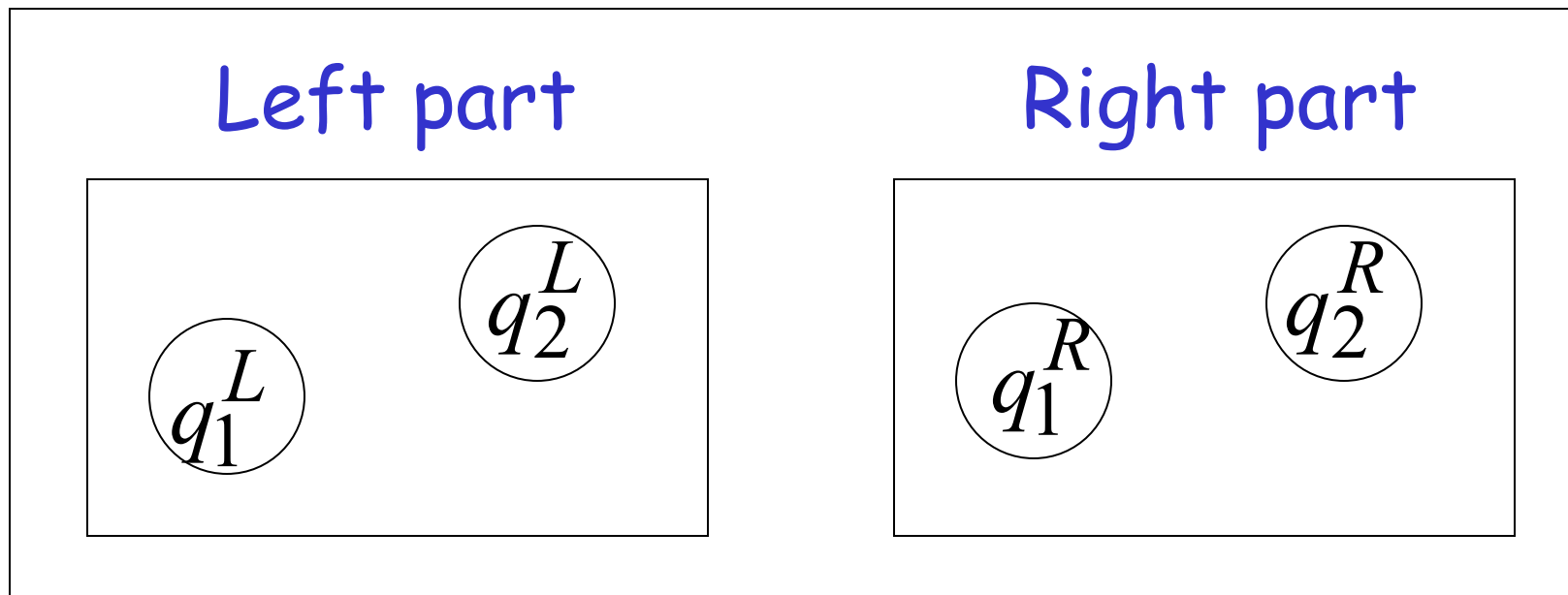
## Semi-Infinite tape machine with two tracks



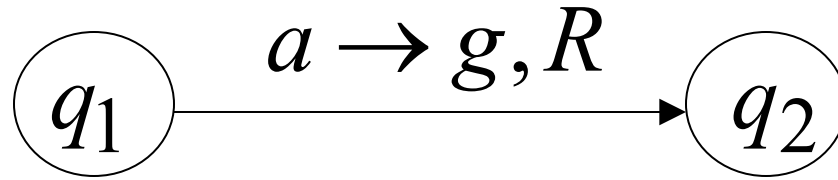
# Standard machine



# Semi-Infinite tape machine

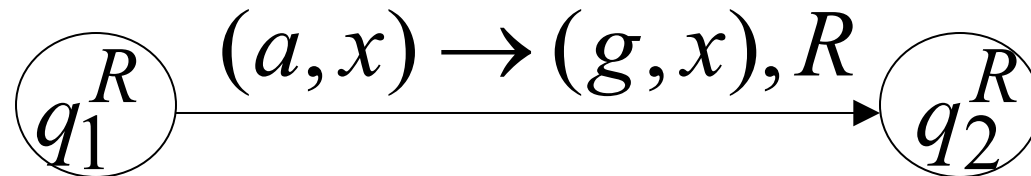


# Standard machine

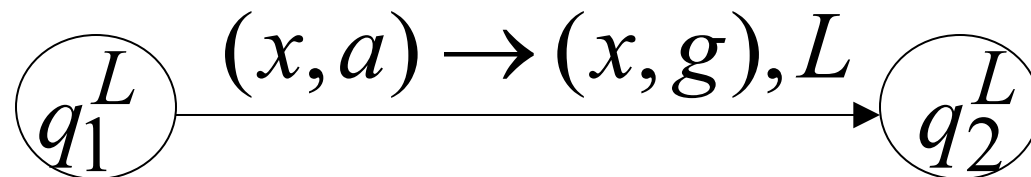


# Semi-Infinite tape machine

Right part



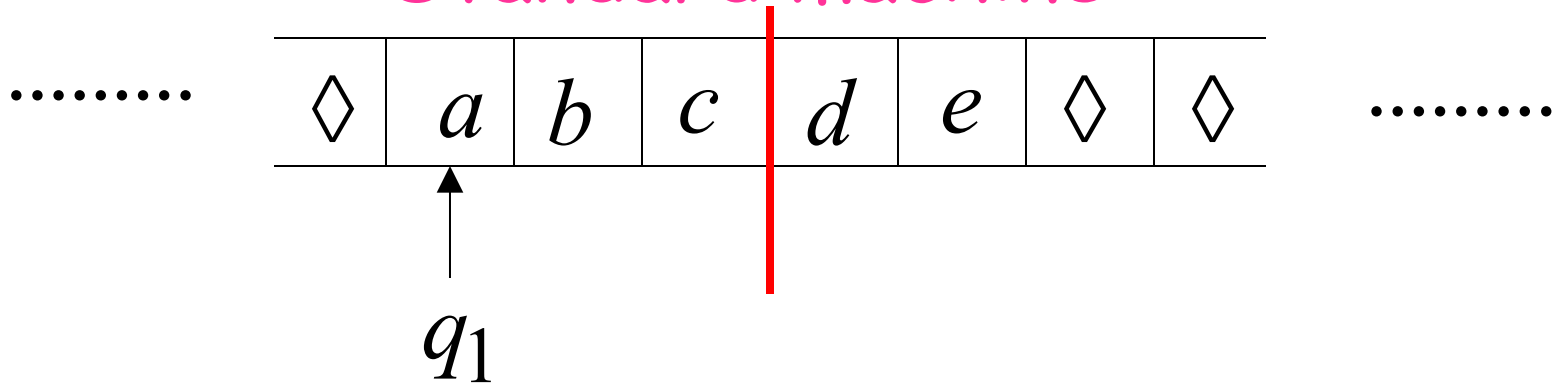
Left part



For all tape symbols  $x$

Time 1

### Standard machine



### Semi-Infinite tape machine

Right part

#	<i>d</i>	<i>e</i>	◇	◇	◇	
---	----------	----------	---	---	---	--

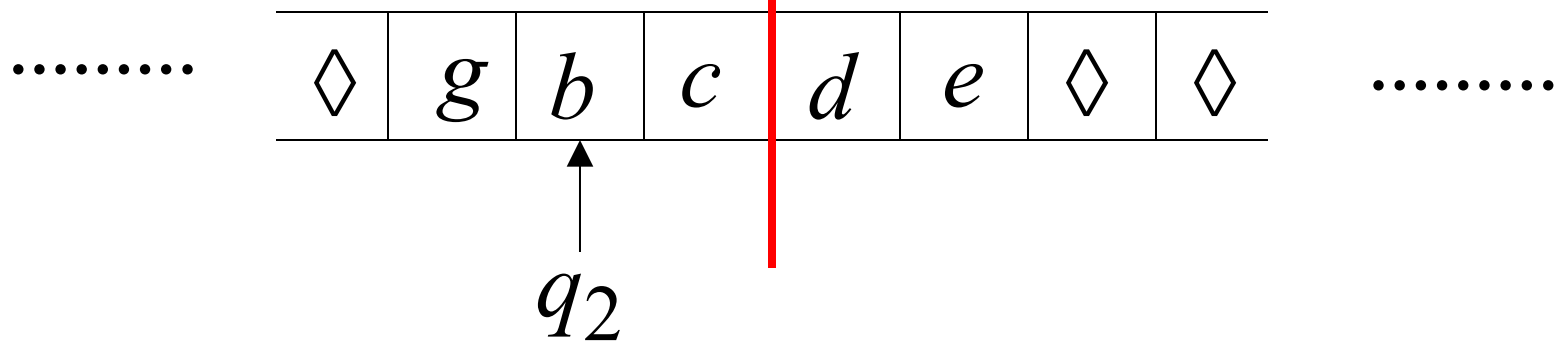
Left part

#	<i>c</i>	<i>b</i>	<i>a</i>	◇	◇	
---	----------	----------	----------	---	---	--



Time 2

### Standard machine



### Semi-Infinite tape machine

Right part

#	<i>d</i>	<i>e</i>	◇	◇	◇	
---	----------	----------	---	---	---	--

Left part

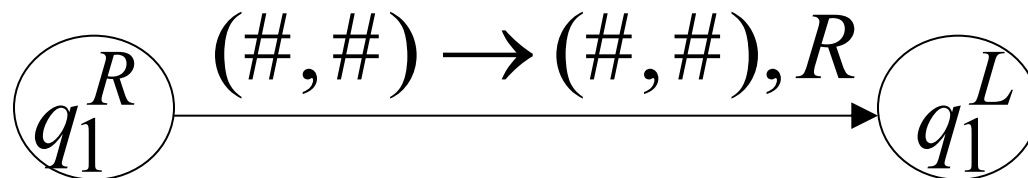
#	<i>c</i>	<i>b</i>	<i>g</i>	◇	◇	
---	----------	----------	----------	---	---	--

$q_2^L$

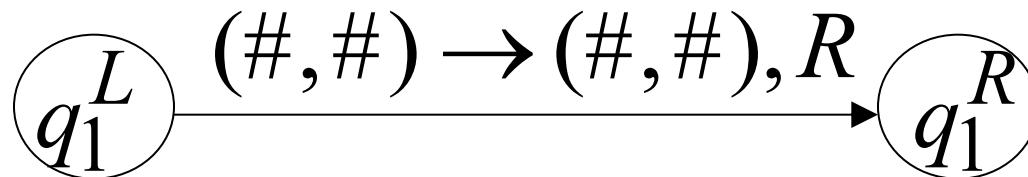
At the border:

## Semi-Infinite tape machine

Right part

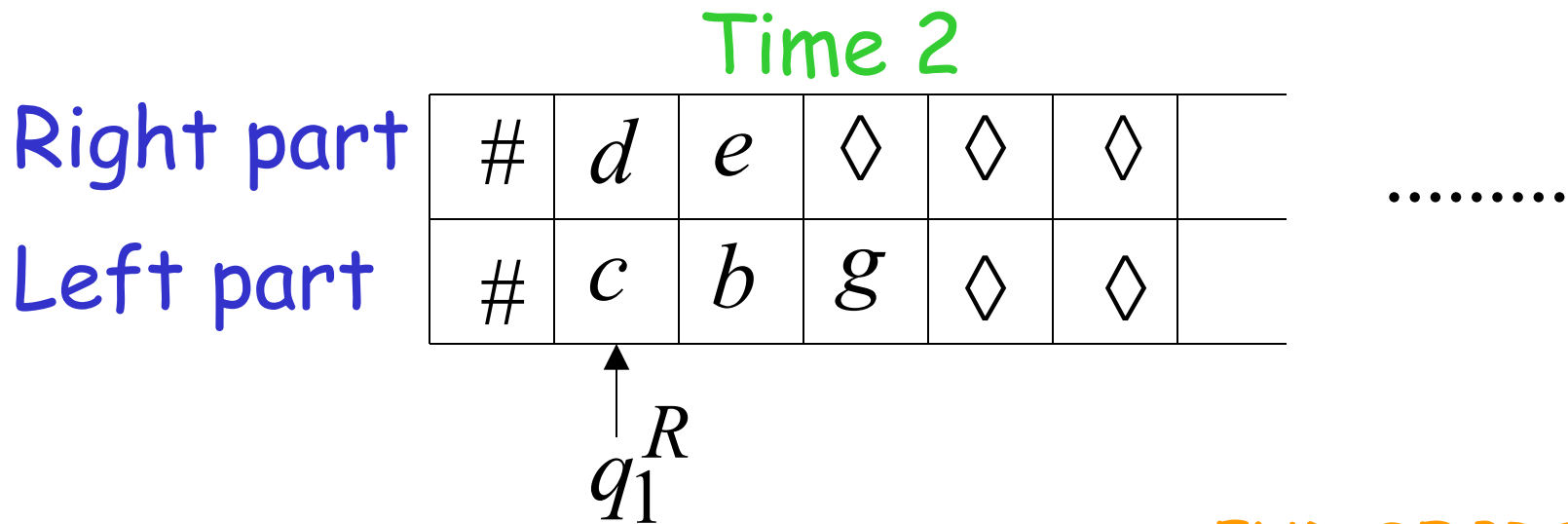
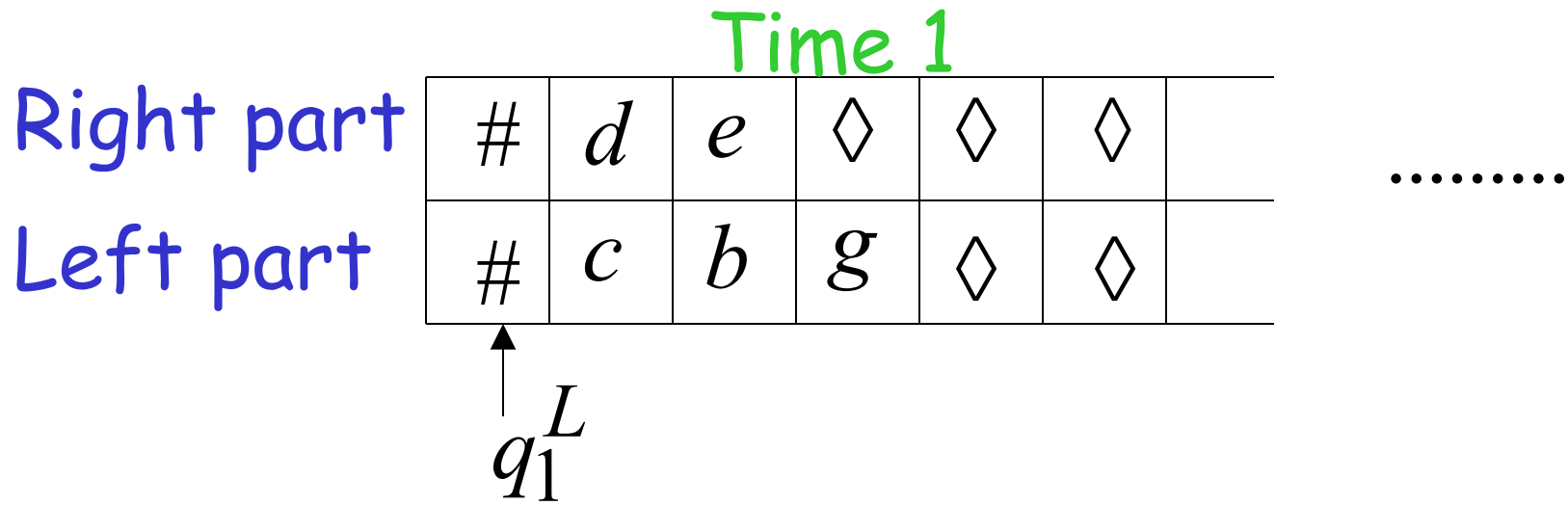


Left part





# Semi-Infinite tape machine



END OF PROOF

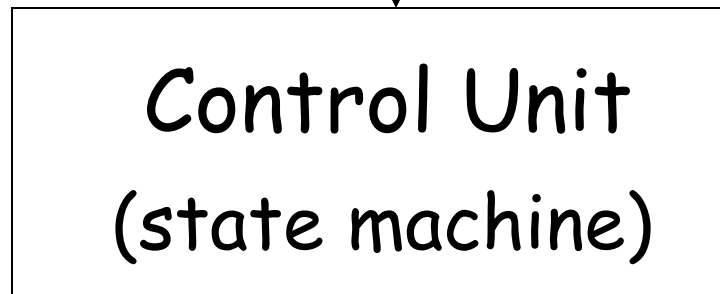
# The Off-Line Machine

Input File read-only (once)

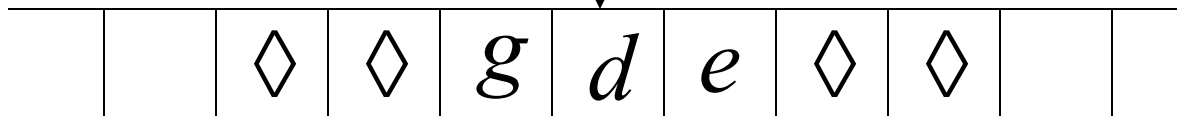


Input string

Input string  
Appears on  
input file only



Tape read-write



**Theorem:** Off-Line machines  
have the same power with  
Standard Turing machines

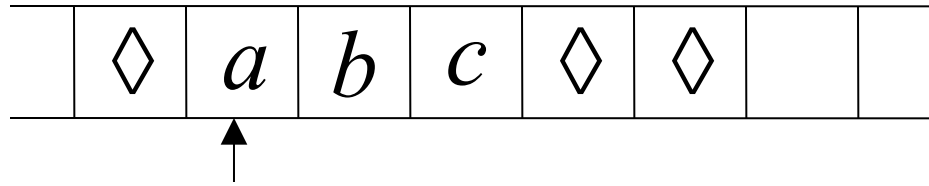
- Proof:**
1. Off-Line machines  
simulate Standard Turing machines
  2. Standard Turing machines  
simulate Off-Line machines

# 1. Off-line machines simulate Standard Turing Machines

## Off-line machine:

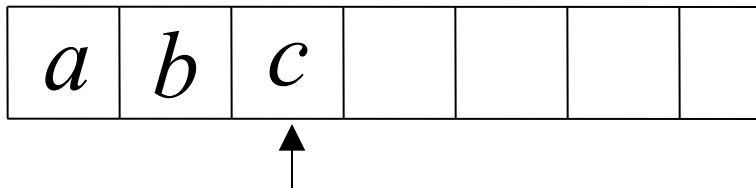
1. Copy input file to tape
2. Continue computation as in  
Standard Turing machine

# Standard machine

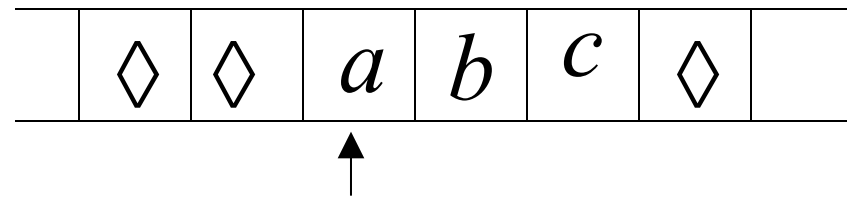


# Off-line machine

## Input File

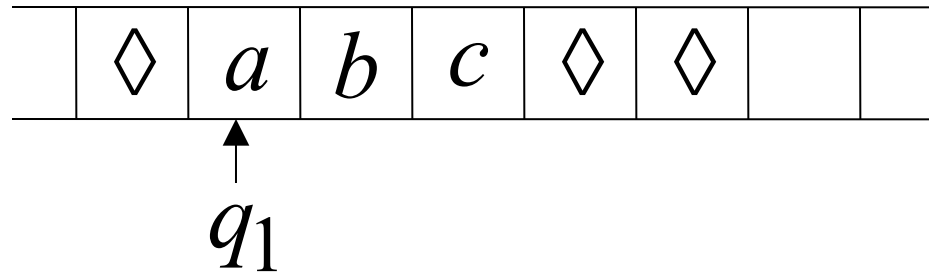


## Tape



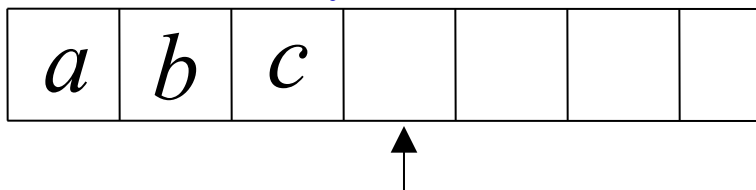
1. Copy input file to tape

## Standard machine

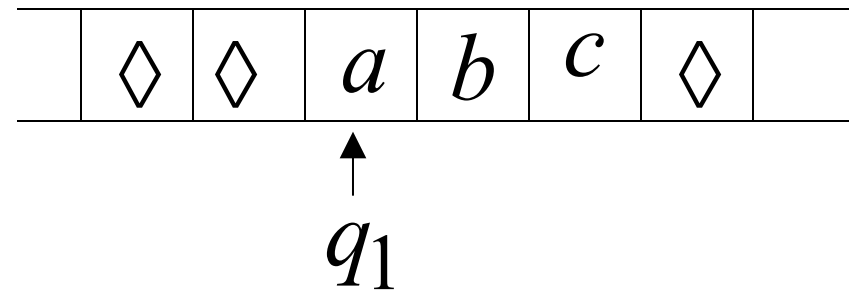


## Off-line machine

### Input File



### Tape



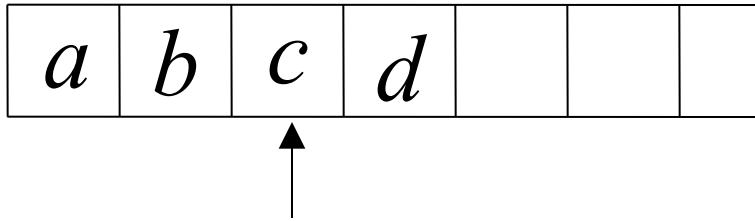
2. Do computations as in Turing machine

## 2. Standard Turing machines simulate Off-Line machines:

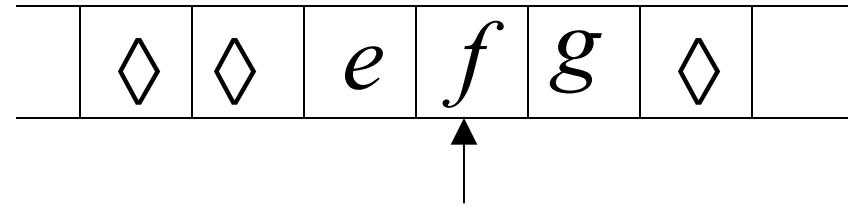
Use a Standard machine with  
a four-track tape to keep track of  
the Off-line input file and tape contents

# Off-line Machine

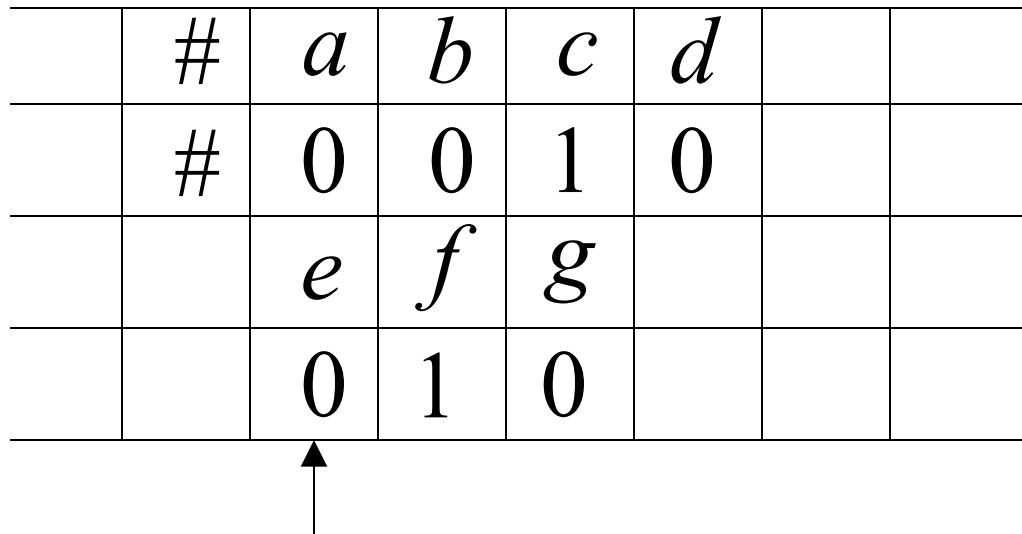
Input File



Tape



## Standard Machine -- Four track tape



Input File

head position

Tape

head position



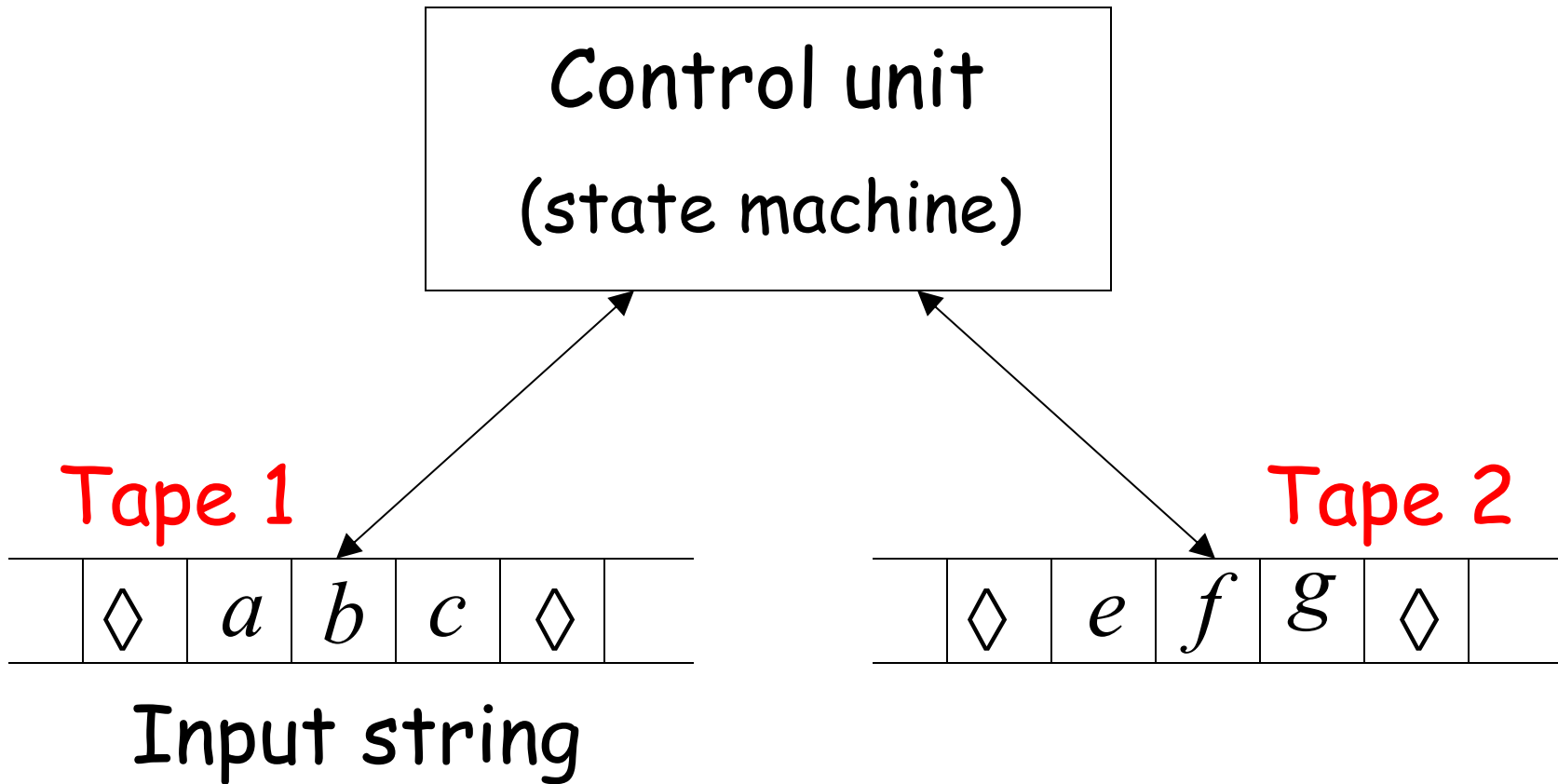
Reference point (uses special symbol # )

#	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>			Input File
#	0	0	1	0			head position
#	<i>e</i>	<i>f</i>	<i>g</i>				Tape
#	0	1	0				head position

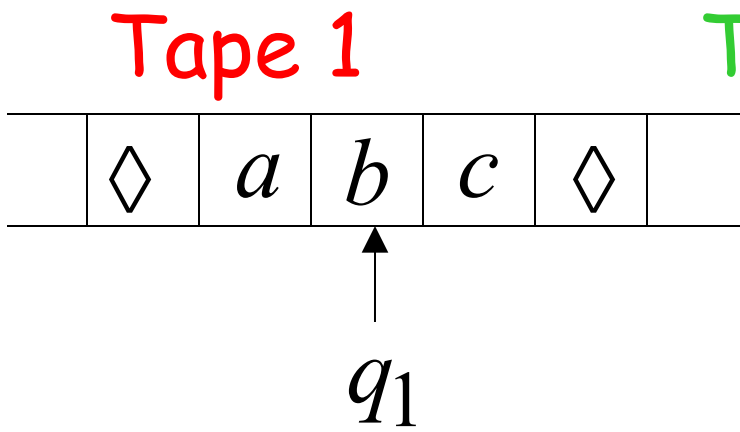
Repeat for each state transition:

1. Return to reference point
2. Find current input file symbol
3. Find current tape symbol
4. Make transition

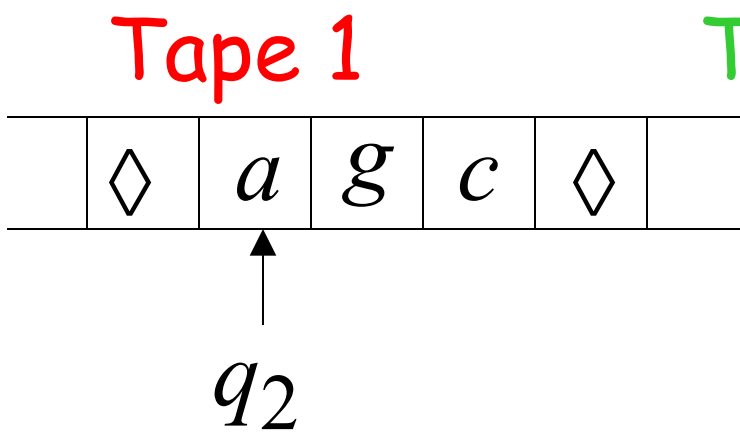
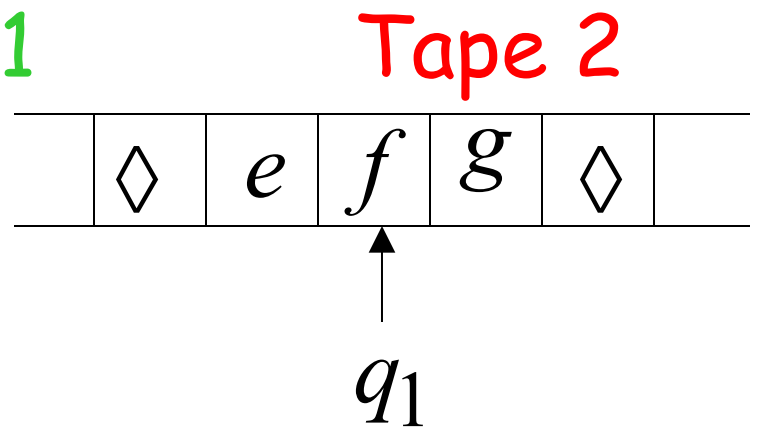
# Multi-tape Turing Machines



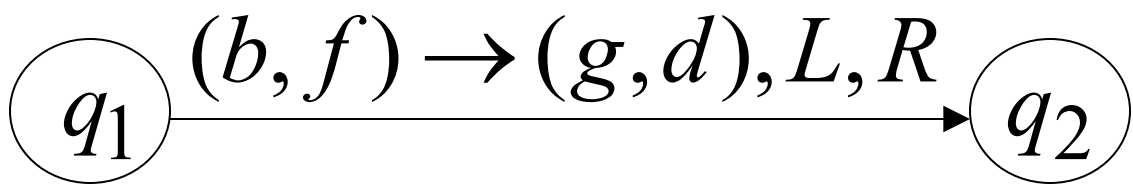
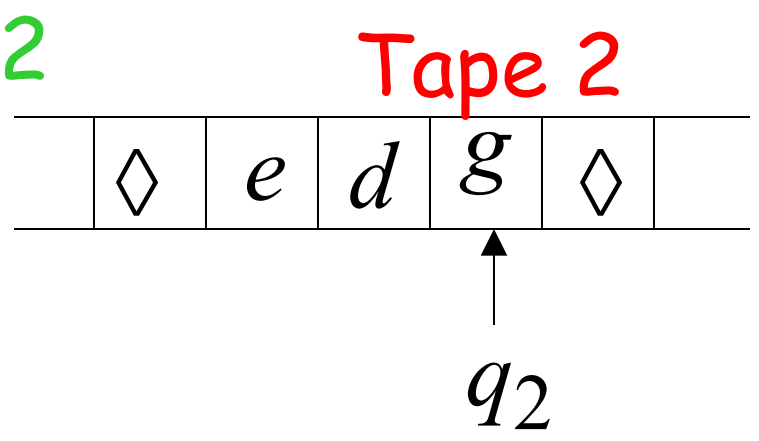
Input string appears on Tape 1



**Time 1**



**Time 2**



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

- Proof:**
1. Multi-tape machines  
simulate Standard Turing machines
  2. Standard Turing machines  
simulate Multi-tape machines

# 1. Multi-tape machines simulate Standard Turing Machines:

Trivial: Use just one tape

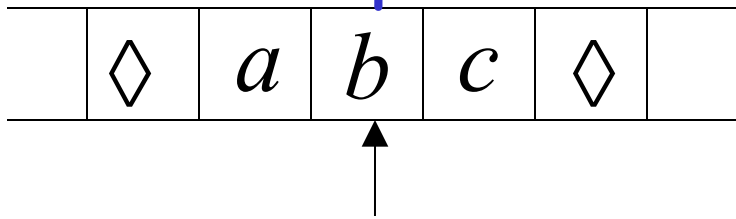
## 2. Standard Turing machines simulate Multi-tape machines:

### Standard machine:

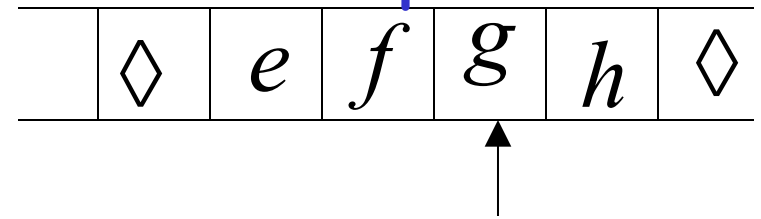
- Uses a multi-track tape to simulate the multiple tapes
- A tape of the Multi-tape machine corresponds to a pair of tracks

# Multi-tape Machine

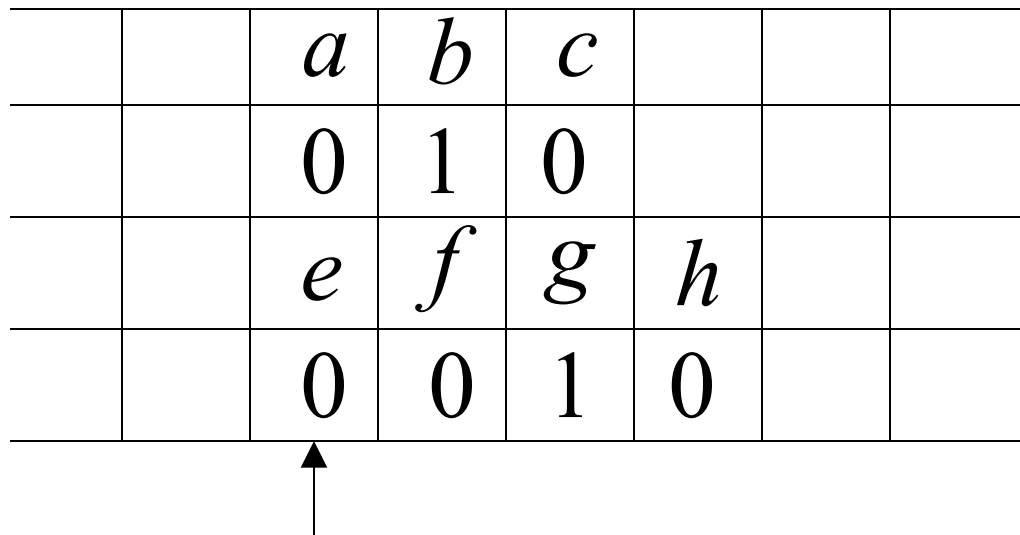
Tape 1



Tape 2



Standard machine with four track tape



Tape 1

head position

Tape 2

head position

# Reference point

#	<i>a</i>	<i>b</i>	<i>c</i>				
#	0	1	0				
#	<i>e</i>	<i>f</i>	<i>g</i>	<i>h</i>			
#	0	0	1	0			

Tape 1

head position

Tape 2

head position

Repeat for each state transition:

1. Return to reference point
2. Find current symbol in Tape 1
3. Find current symbol in Tape 2
4. Make transition

END OF PROOF



Same power doesn't imply same speed:

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

Standard Turing machine:  $O(n^2)$  time

Go back and forth  $O(n^2)$  times  
to match the a's with the b's

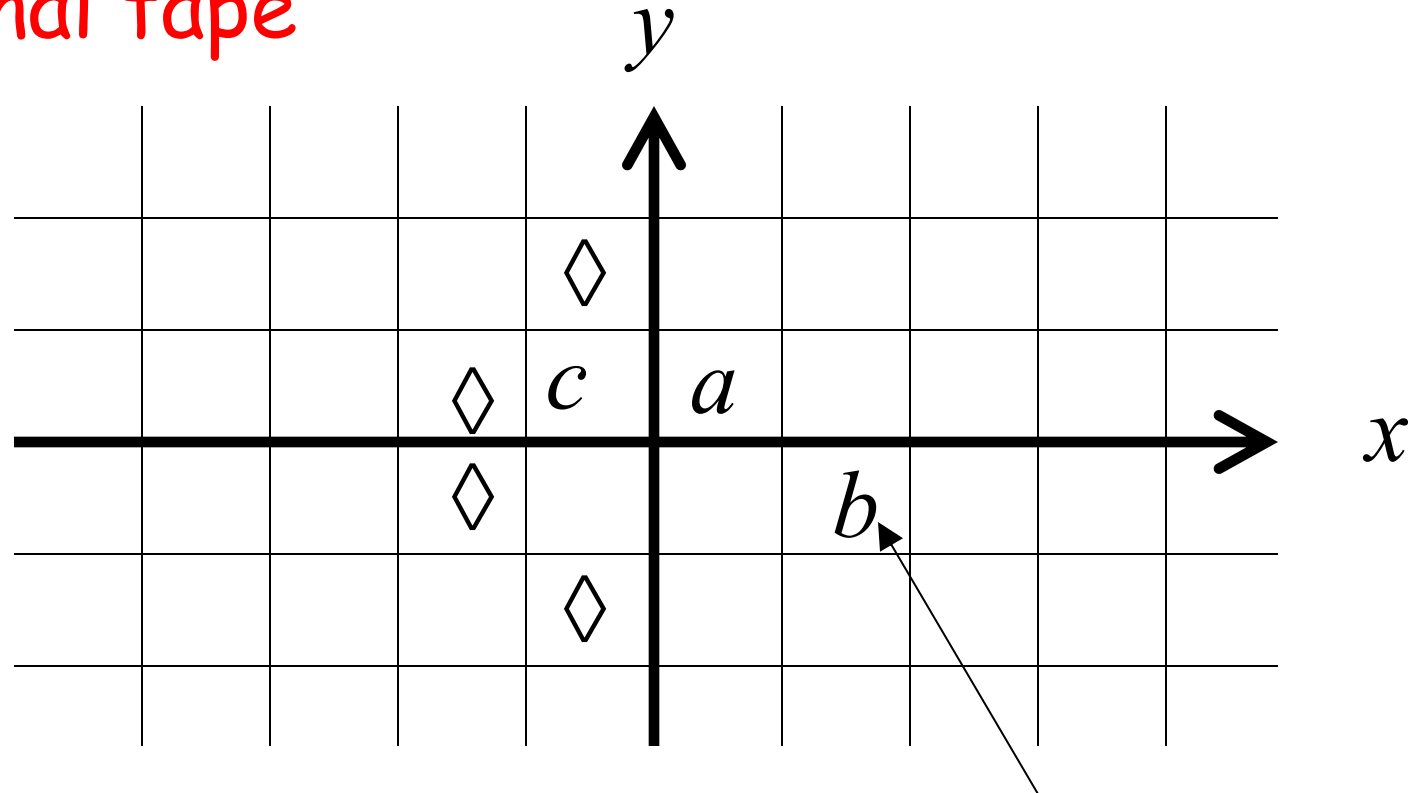
2-tape machine:  $O(n)$  time

1. Copy  $b^n$  to tape 2 ( $O(n)$  steps)

2. Compare  $a^n$  on tape 1  
and  $b^n$  tape 2 ( $O(n)$  steps)

# Multidimensional Turing Machines

2-dimensional tape



MOVES: L,R,U,D

U: up    D: down

HEAD

Position: +2, -1

**Theorem:** Multidimensional machines  
have the same power with  
Standard Turing machines

- Proof:**
1. Multidimensional machines  
simulate Standard Turing machines
  2. Standard Turing machines  
simulate Multi-Dimensional machines

# 1. Multidimensional machines simulate Standard Turing machines

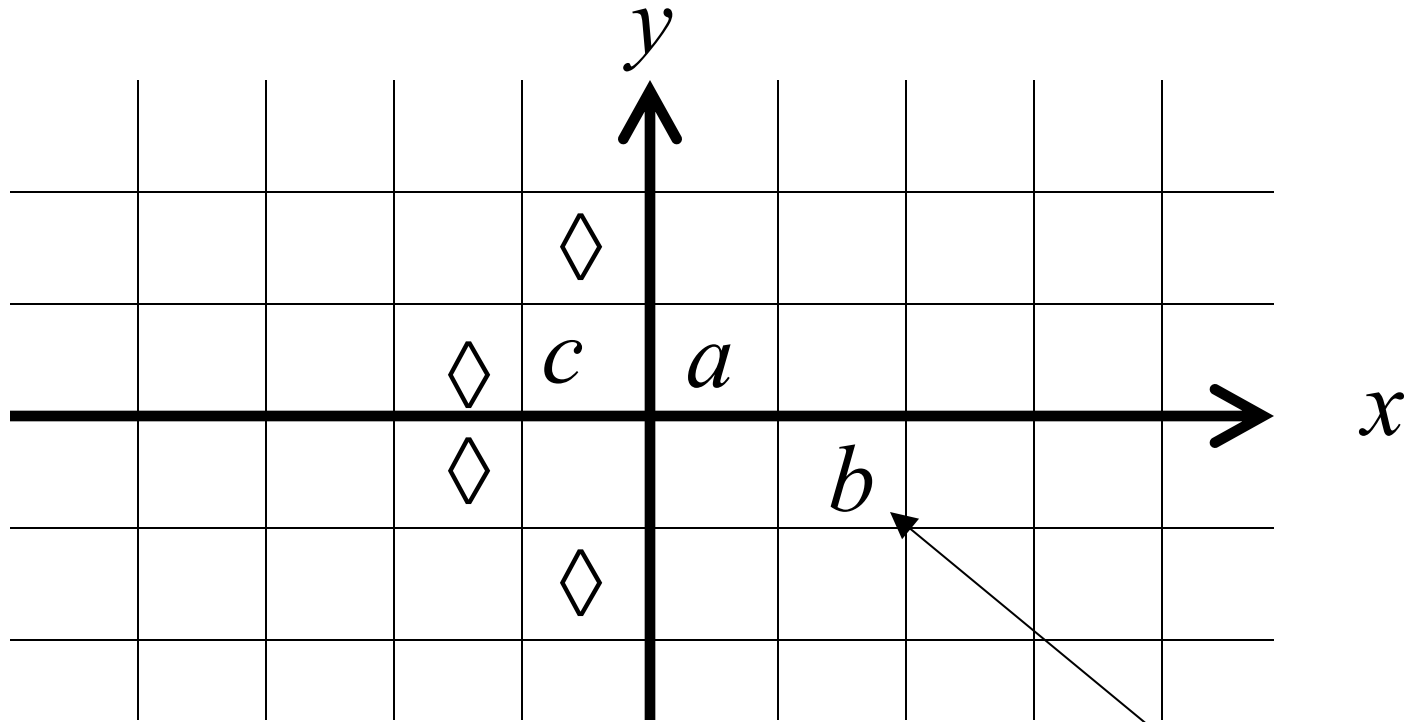
Trivial: Use one dimension

## 2. Standard Turing machines simulate Multidimensional machines

### Standard machine:

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

# 2-dimensional machine



## Standard Machine

<i>a</i>				<i>b</i>				<i>c</i>		
1	#	1	#	2	#	-	1	#	-	1

symbols  
coordinates

$q_1$

$q_1$

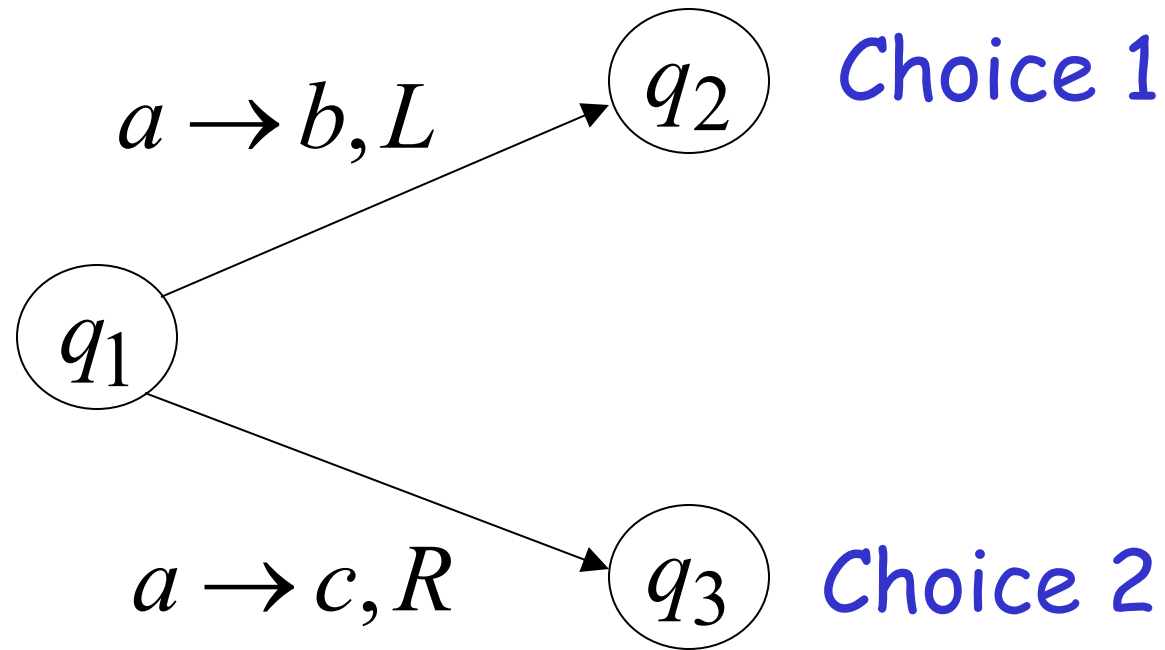
## Standard machine:

Repeat for each transition followed in the 2-dimensional machine:

1. Update current symbol
2. Compute coordinates of next position
3. Go to new position

END OF PROOF

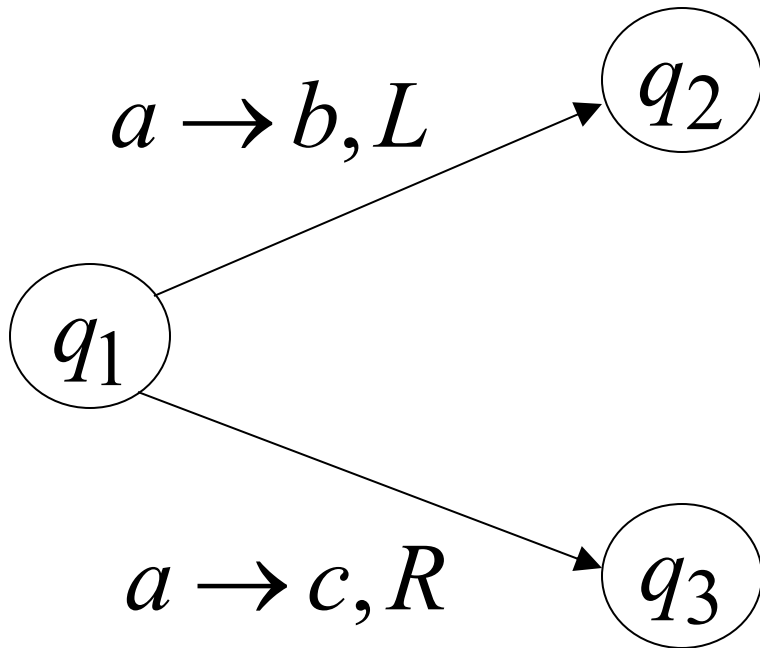
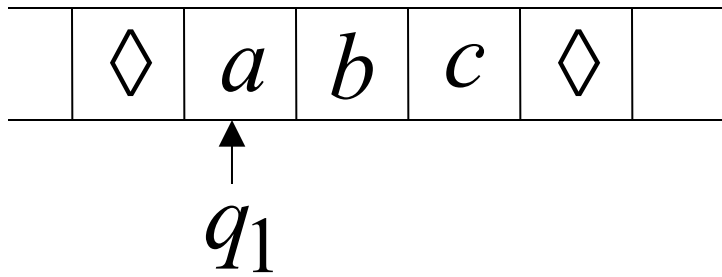
# Nondeterministic Turing Machines



**Allows Non Deterministic Choices**

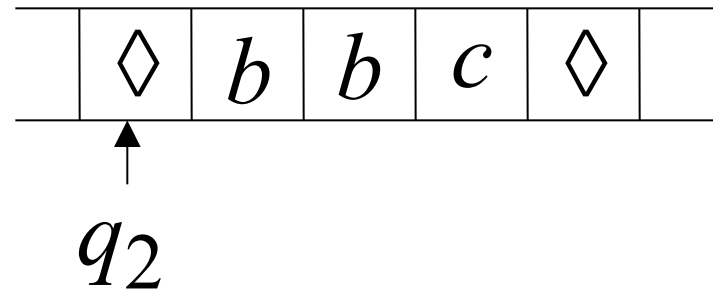


Time 0

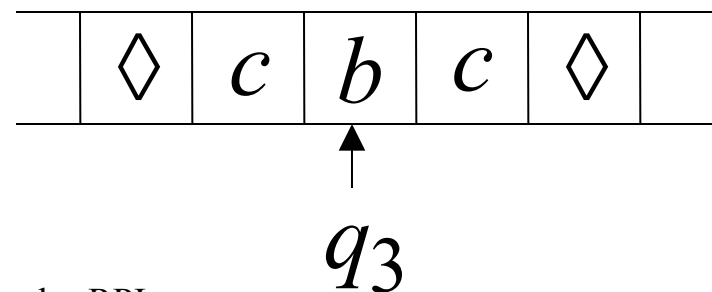


Time 1

Choice 1



Choice 2



Input string  $w$  is accepted if there is a computation:

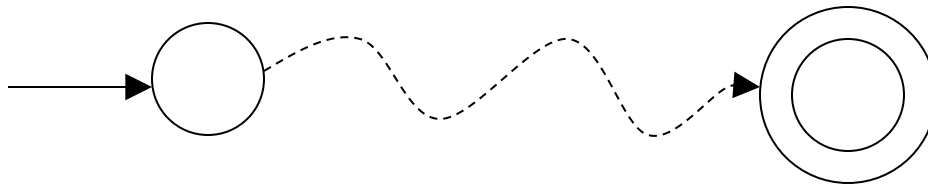
$q_0 w \quad \overset{*}{\phi} \quad x q_f y$

Initial configuration

Final Configuration

Any accept state

There is a computation:



**Theorem:** Nondeterministic machines  
have the same power with  
Standard Turing machines

- Proof:**
1. Nondeterministic machines  
simulate Standard Turing machines
  2. Standard Turing machines  
simulate Nondeterministic machines

# 1. Nondeterministic Machines simulate Standard (deterministic) Turing Machines

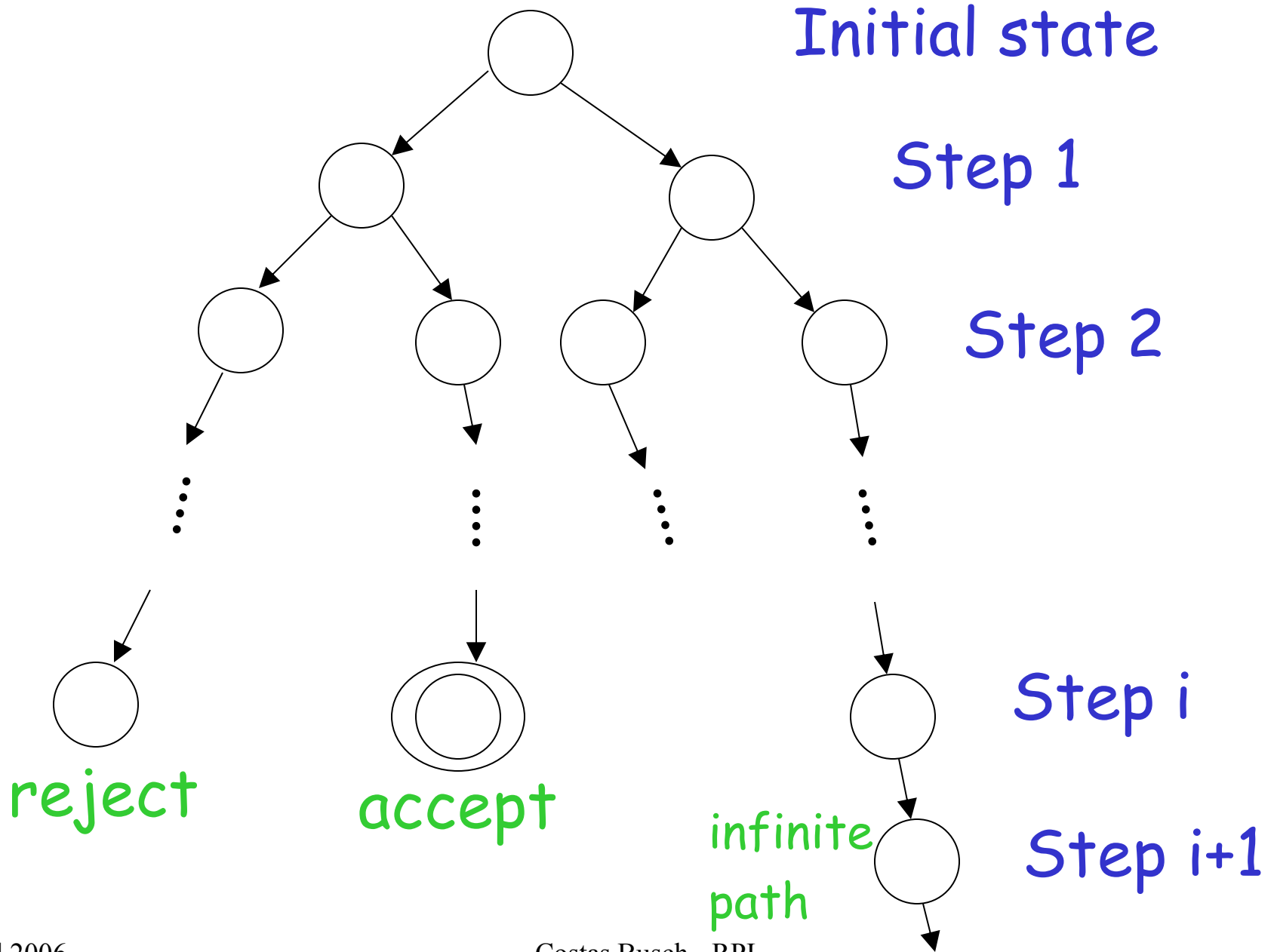
Trivial: every deterministic machine  
is also nondeterministic

## 2. Standard (deterministic) Turing machines simulate Nondeterministic machines:

### Deterministic machine:

- Uses a 2-dimensional tape  
(which is equivalent to 1-dimensional tape)
- Stores all possible computations of the non-deterministic machine on the 2-dimensional tape

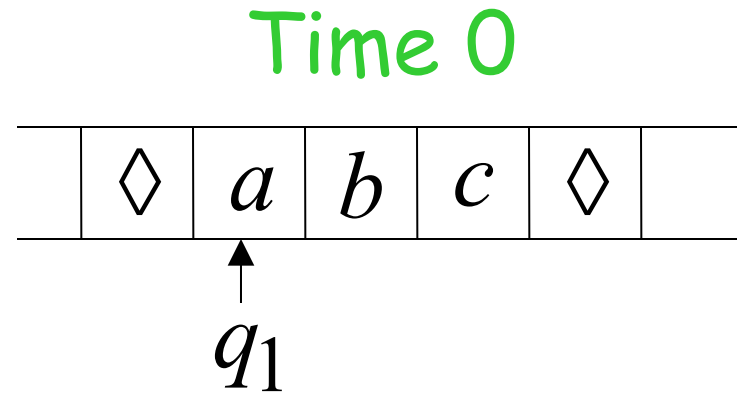
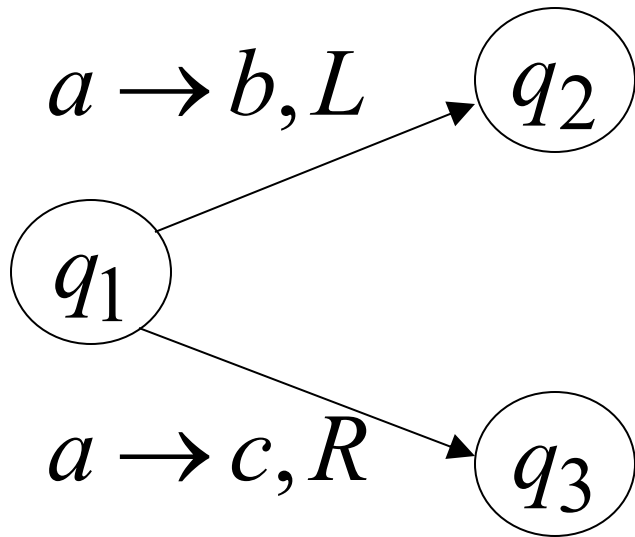
# All possible computation paths



The Deterministic Turing machine  
simulates all possible computation paths:

- simultaneously
- step-by-step
- in a breadth-first search fashion

# NonDeterministic machine



# Deterministic machine

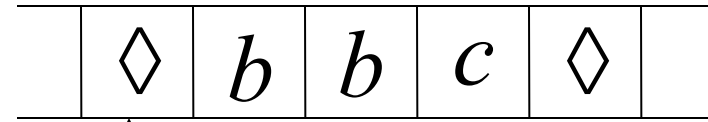
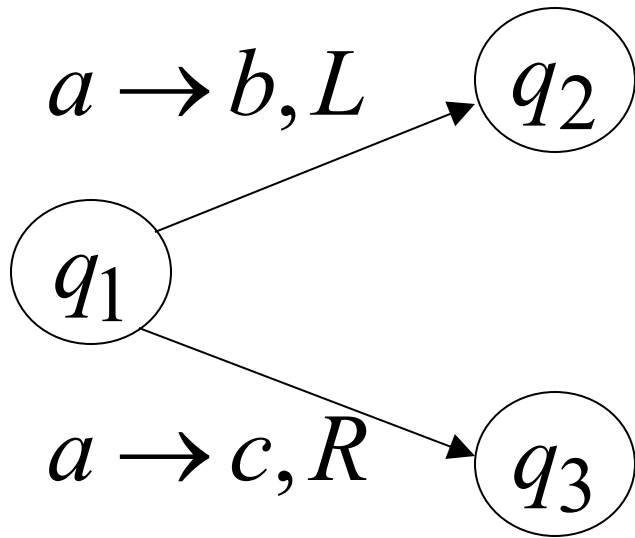
	#	#	#	#	#	#	
	#	a	b	c	#		
	#	q <sub>1</sub>			#		
	#	#	#	#	#		

current  
configuration



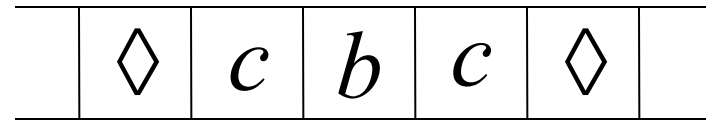
# NonDeterministic machine

Time 1



Choice 1

↑  
q2



Choice 2

↑  
q3

# Deterministic machine

	#	#	#	#	#	#
#		b	b	c	#	
#	q2				#	
#		c	b	c	#	
#			q3		#	

Computation 1

Computation 2

# Deterministic Turing machine

## Repeat

For each configuration in current step of non-deterministic machine,  
if there are two or more choices:

1. Replicate configuration
2. Change the state in the replicas

Until either the input string is accepted  
or rejected in all configurations

END OF PROOF

## Remark:

The simulation takes in the worst case exponential time compared to the shortest accepting path length of the nondeterministic machine