Turing's Thesis

Turing's thesis:

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

Turing's Thesis is also called as Church-Turing Thesis or Church thesis.

(1930)

Different versions of Turing's thesis

- 1, Every effective computation can be carried out by a Turing machine.
- 2, Every 'function which would naturally be regarded as computable' can be computed by the universal Turing machine.
- it should be noted that there is ambiguity as to what, precisely, a function which would naturally be regarded as computable means. Due to this ambiguity, this statement is not subject to rigorous proof.

- There is strong evidence for this hypothesis; many diverse models of computation have been shown to compute the same set of functions as a Turing machine, as yet there have been no counterexamples to the thesis.

- This thesis gives us insight into the ``power'' of computing machines. If a computing device can solve all the problems a Turing machine can solve, then it is as powerful as a Turing machine.

Computer Science Law:

A computation is mechanical
if and only if
it can be performed by a Turing Machine

There is no known model of computation
more powerful than Turing Machines

Definition of Algorithm:

An algorithm for function $f(w)$

is a

Turing Machine which computes $f(w)$

Algorithms are Turing Machines

When we say:

There exists an algorithm

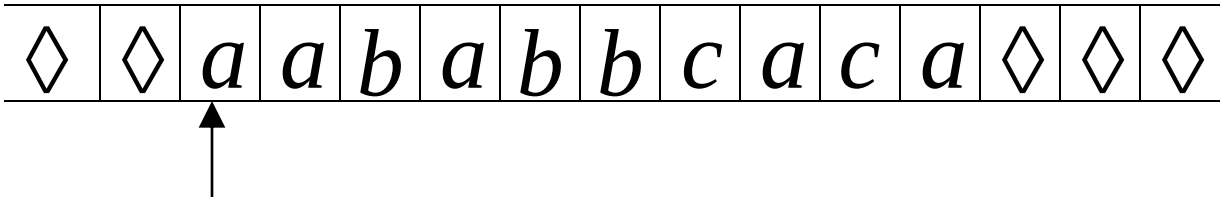
We mean:

There exists a Turing Machine
that executes the algorithm

Variations of the Turing Machine

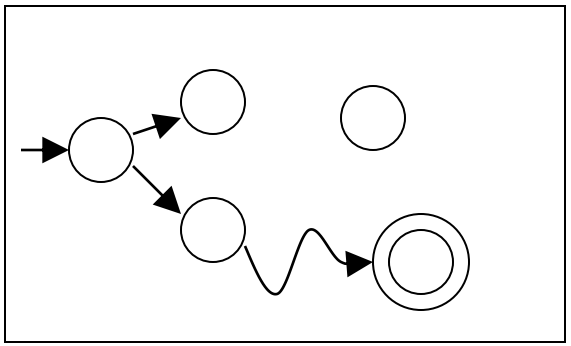
The Standard Model

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

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

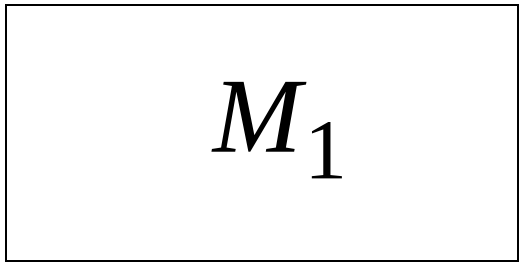
And vice-versa

Simulation: a technique to prove same power

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

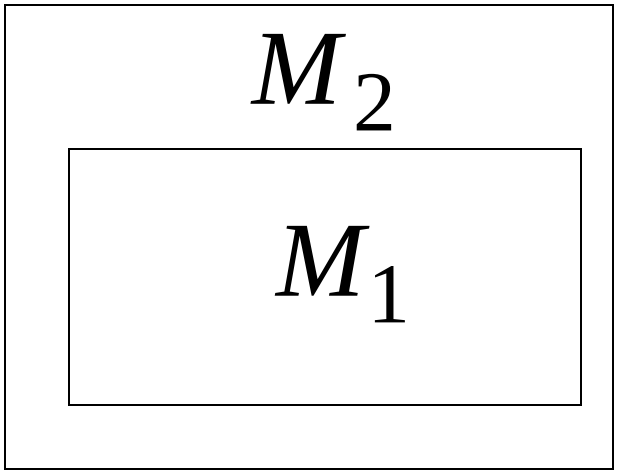
First Class

Original Machine



Second Class

Simulation Machine



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

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

$\updownarrow \qquad \updownarrow \qquad\qquad\qquad \updownarrow$

$\quad * \qquad\quad * \qquad\quad\; *$

Simulation Machine: $d'_0 \succ d'_1 \succ \cdots \succ d'_n$

Final Configuration

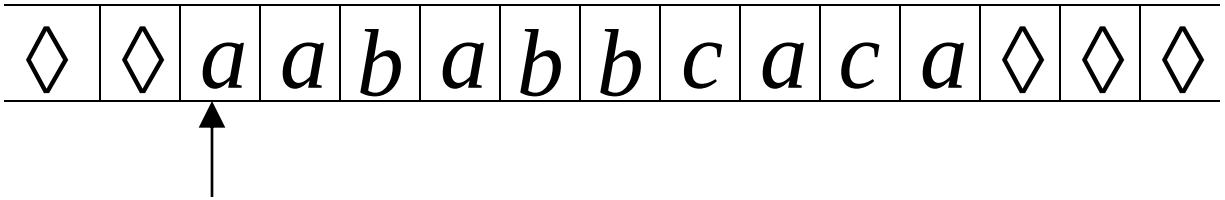
Original Machine: d_f

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

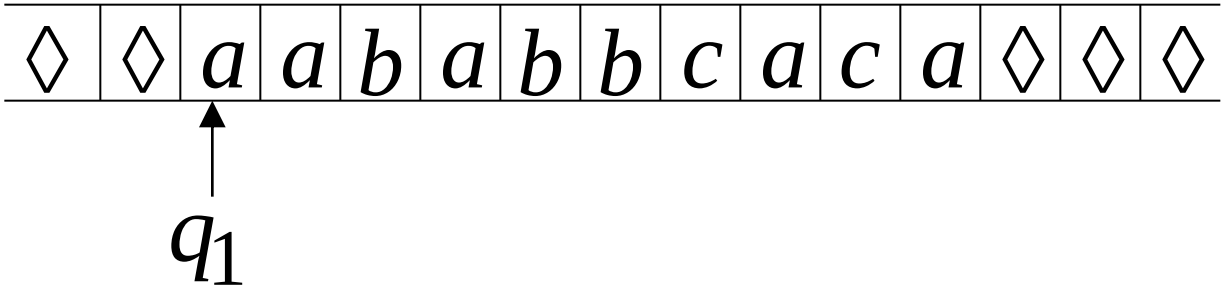


Left, Right, Stay

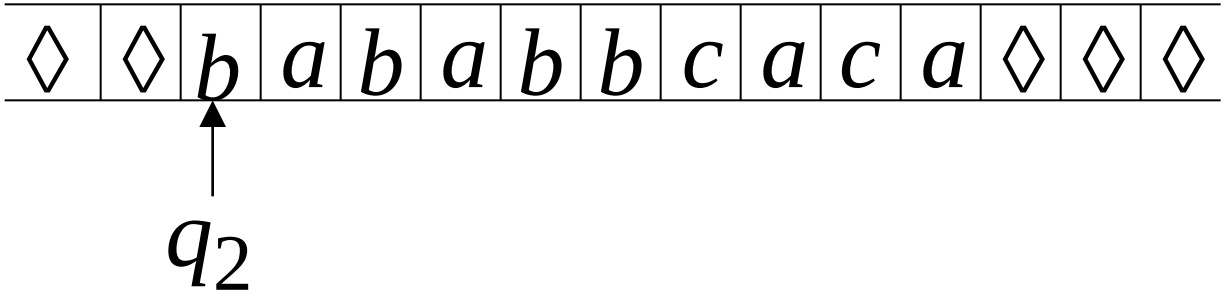
L,R,S: moves

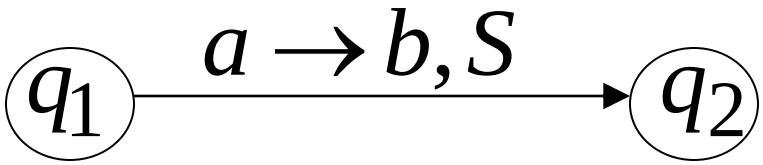
Example:

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

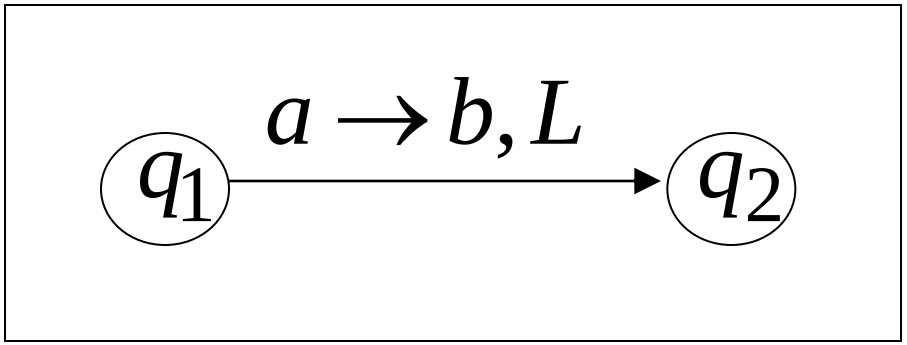
Proof: a Standard machine is also
a Stay-Option machine
(that never uses the S move)

Proof:

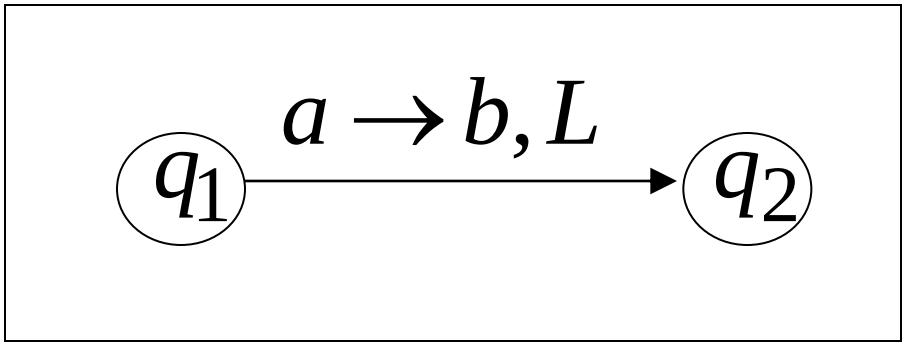
Part 2: Standard Machines
are at least as powerful as
Stay-Option machines

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

Stay-Option Machine

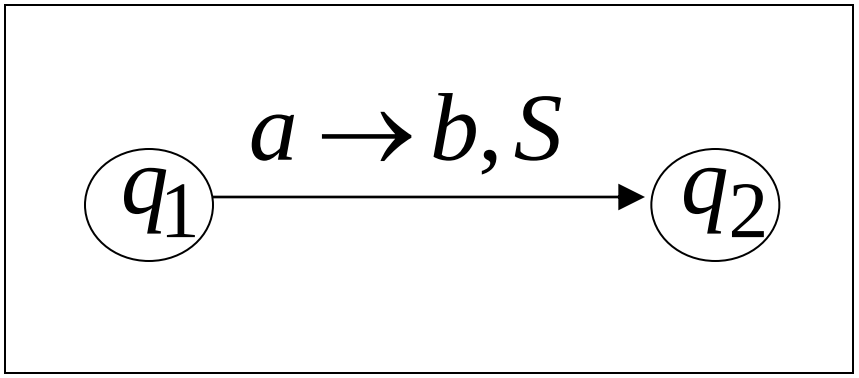


Simulation in Standard Machine

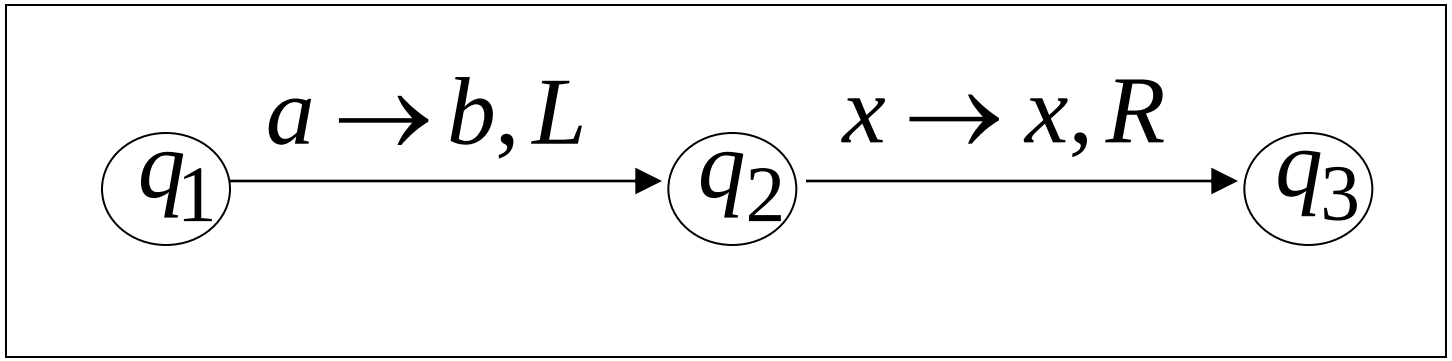


Similar for Right moves

Stay-Option Machine



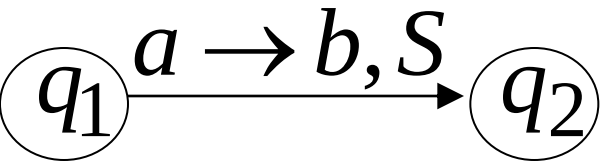
Simulation in Standard Machine

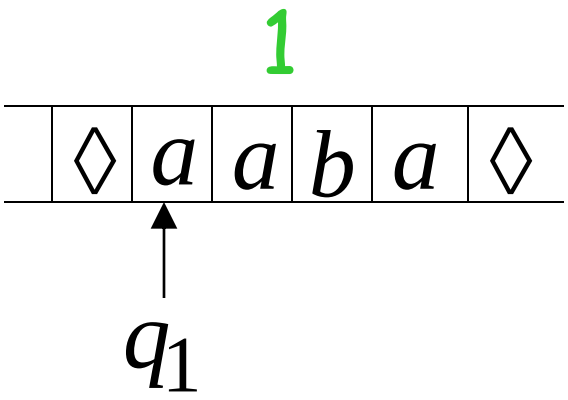


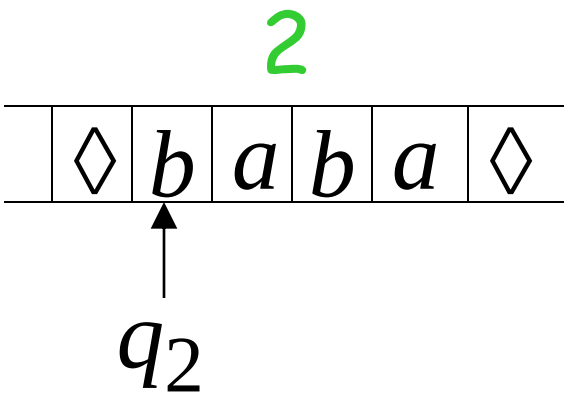
For every symbol x

Example

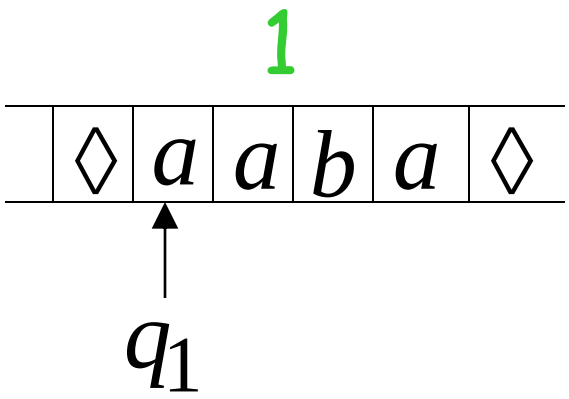
Stay-Option Machine:

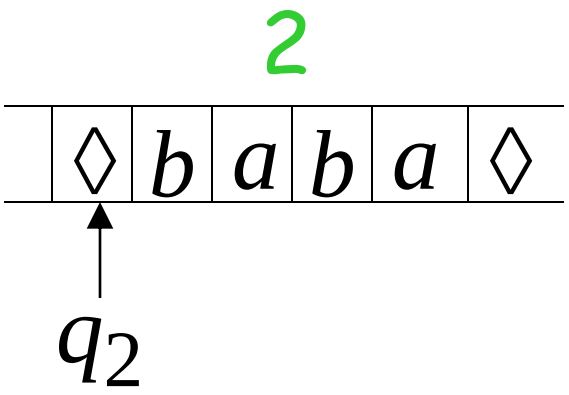


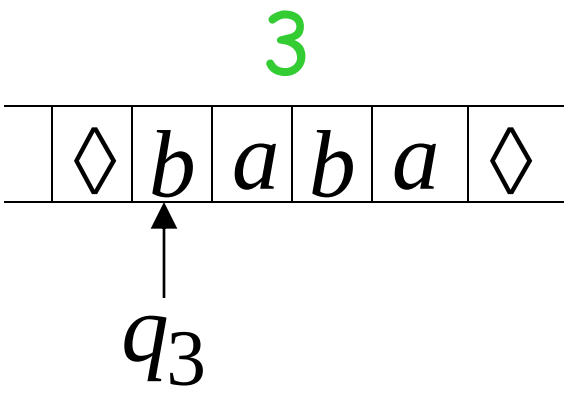




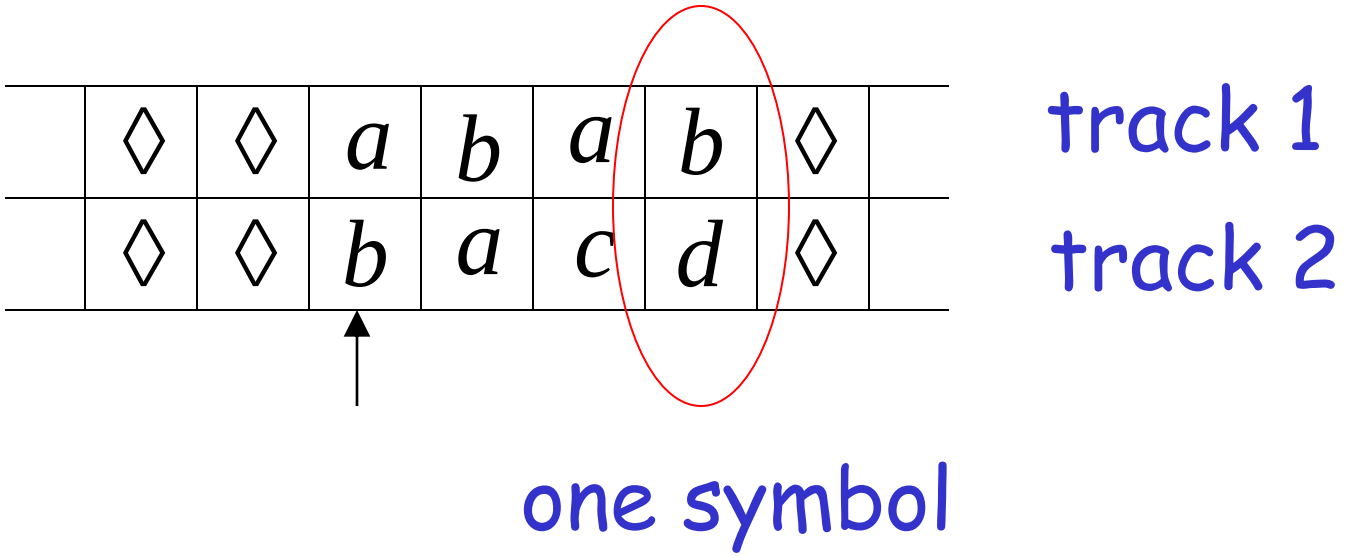
Simulation in Standard Machine:

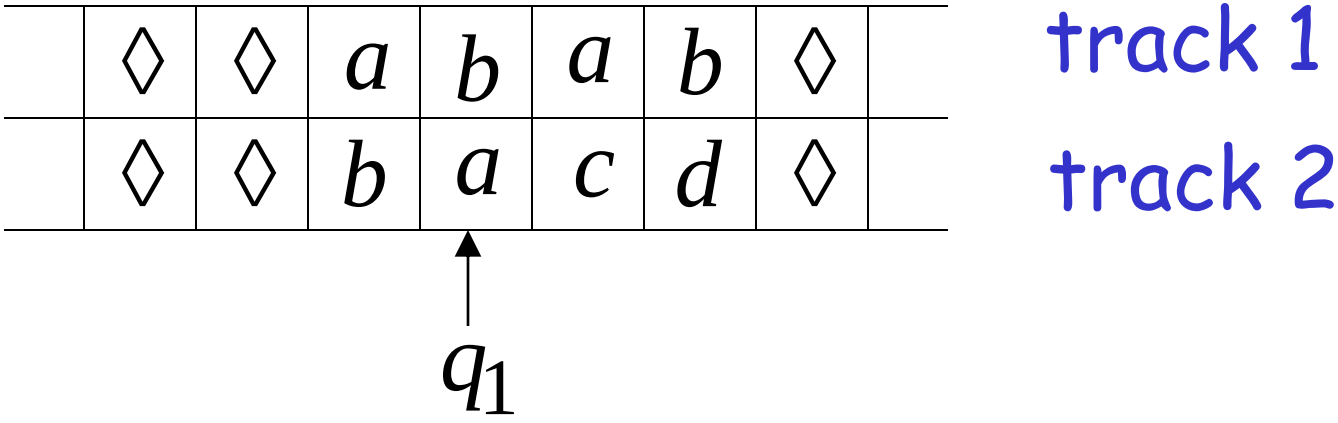


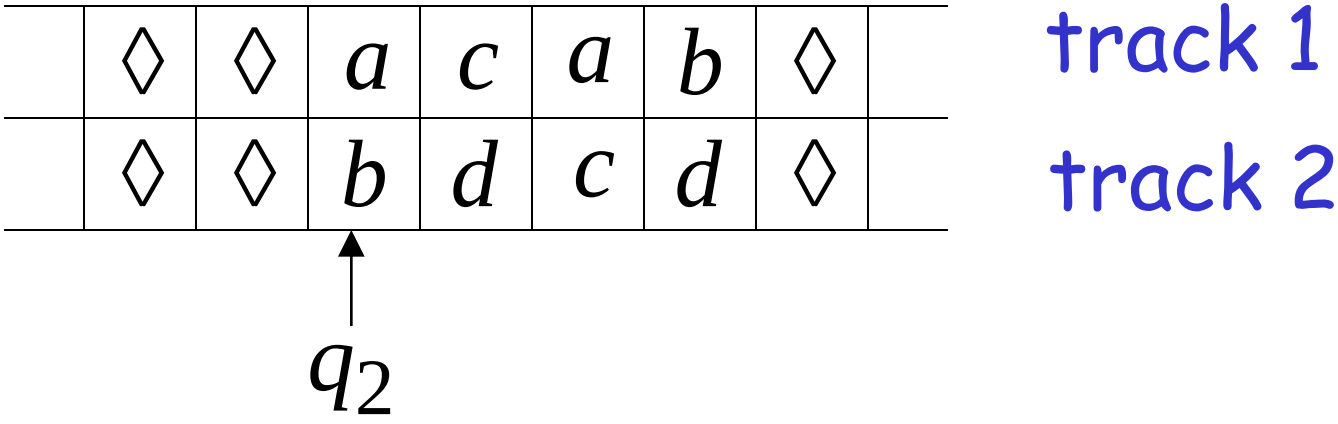


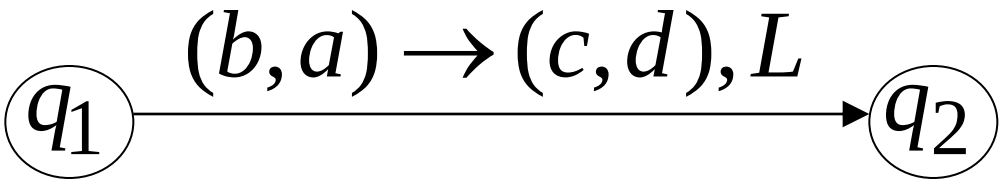


Standard Machine--Multiple Track Tape

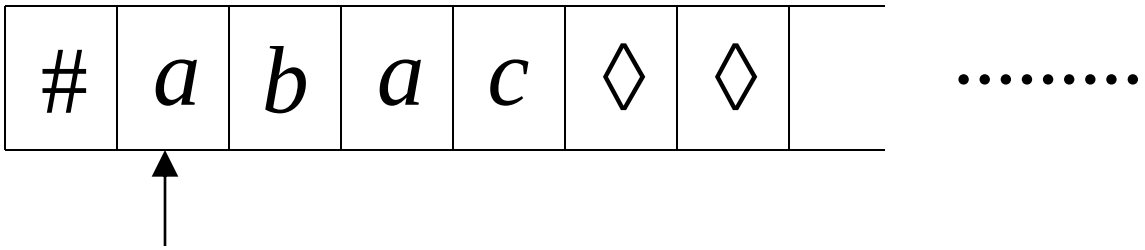



◊ ◊ a b a b ◊
◊ ◊ b a c d ◊
track 1
track 2
q_1


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


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

Semi-Infinite Tape

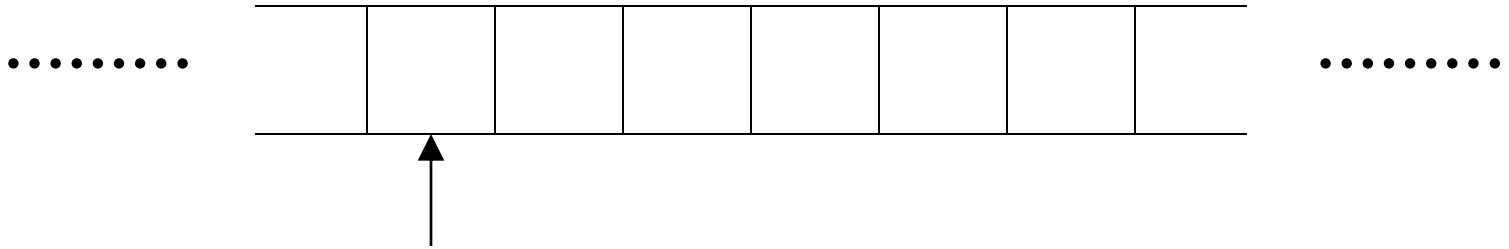


Standard Turing machines simulate Semi-infinite tape machines:

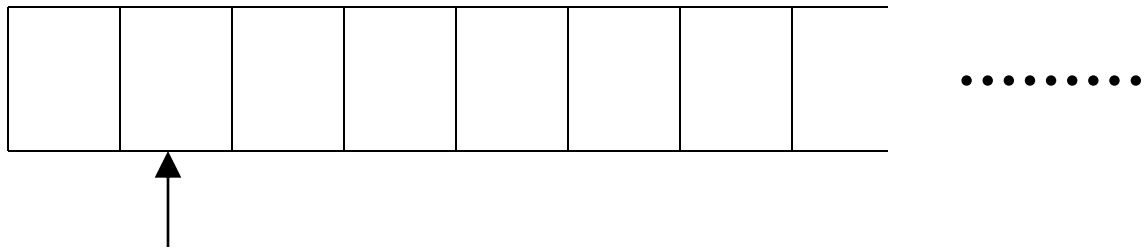
Trivial

Semi-infinite tape machines simulate Standard Turing machines:

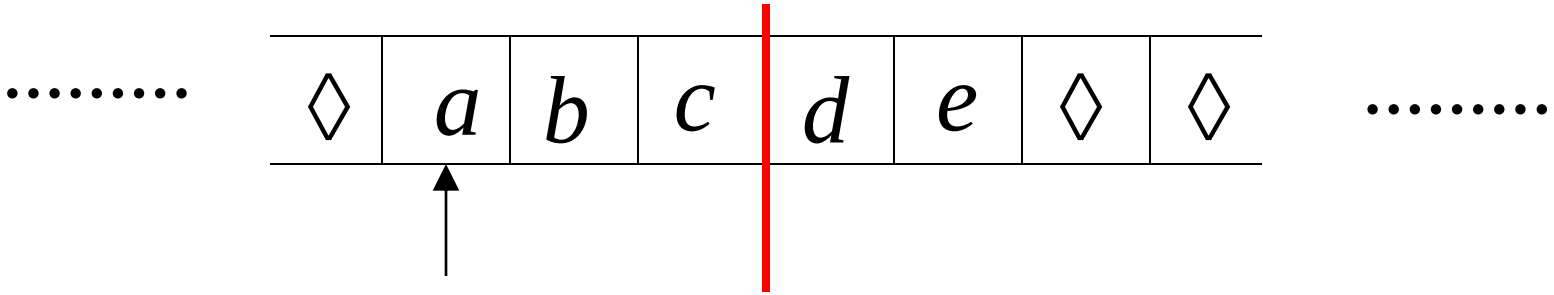
Standard machine



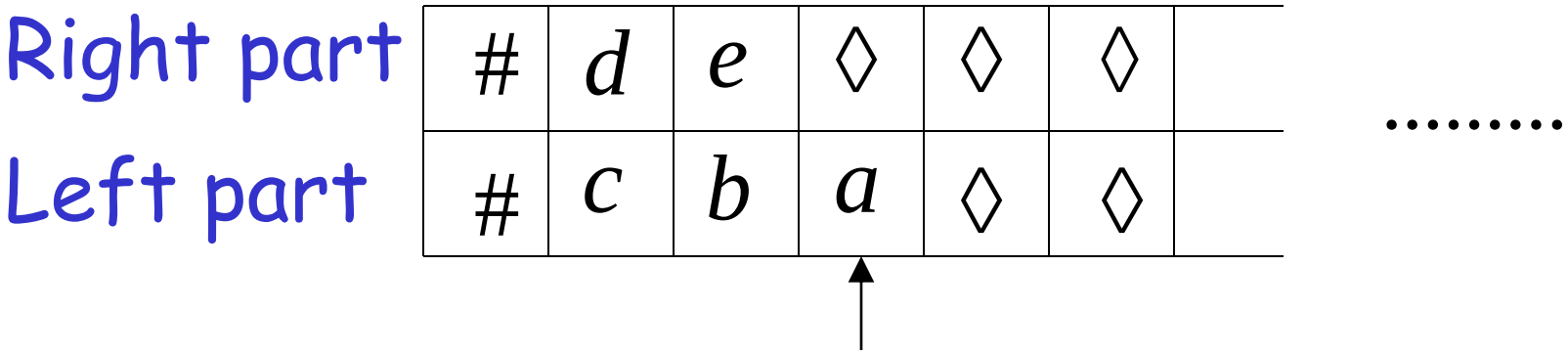
Semi-infinite tape machine



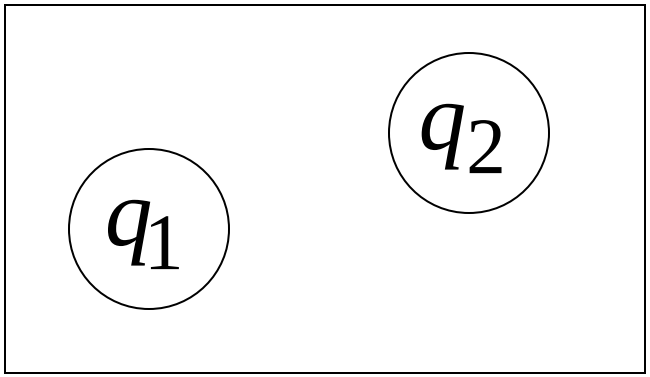
Standard machine



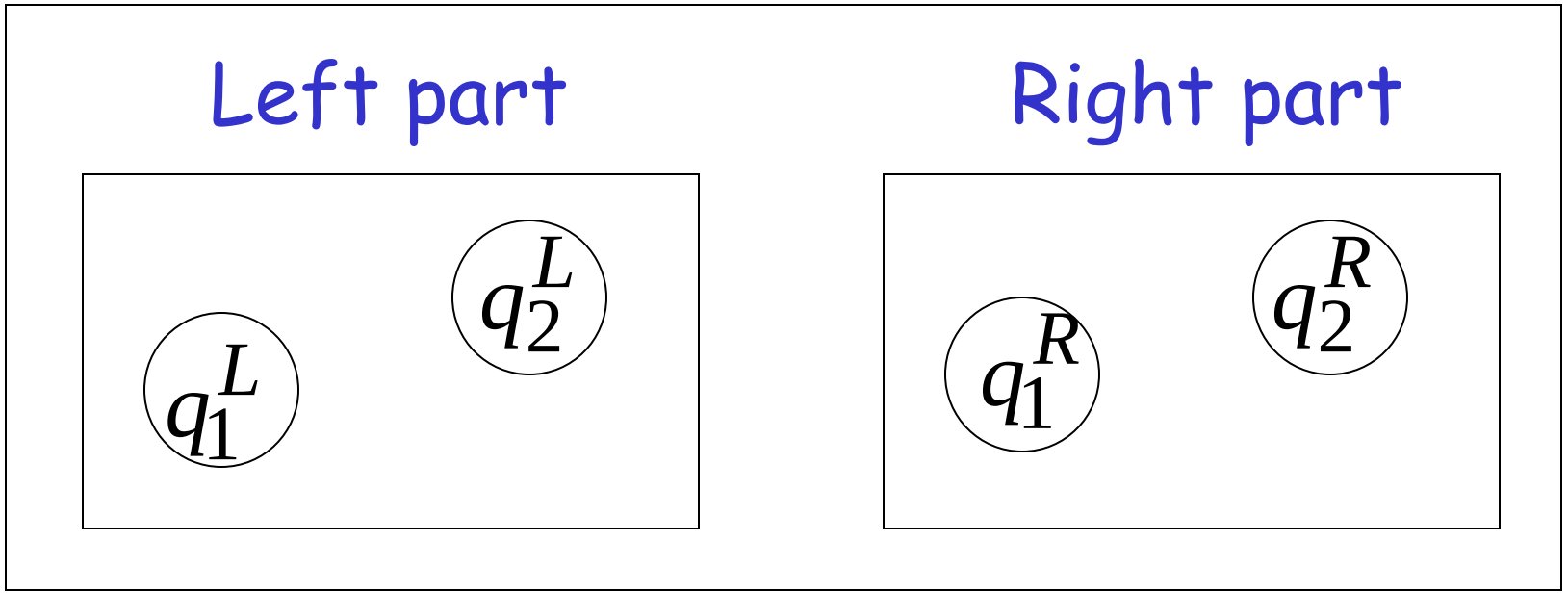
Semi-infinite tape machine with two tracks



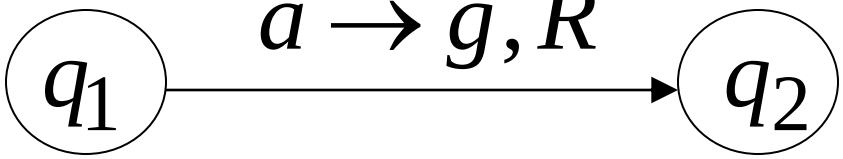
Standard machine



Semi-infinite tape machine

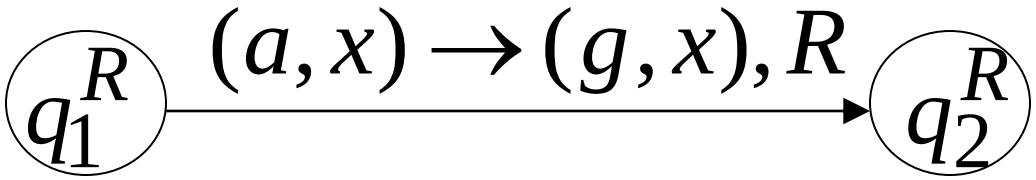


Standard machine

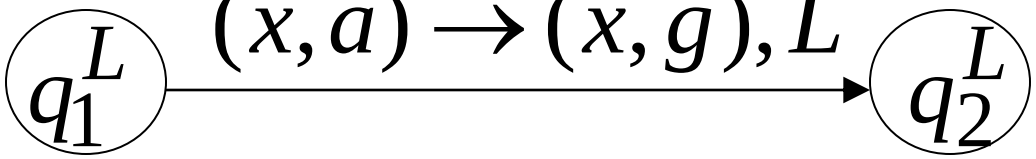


Semi-infinite tape machine

Right part



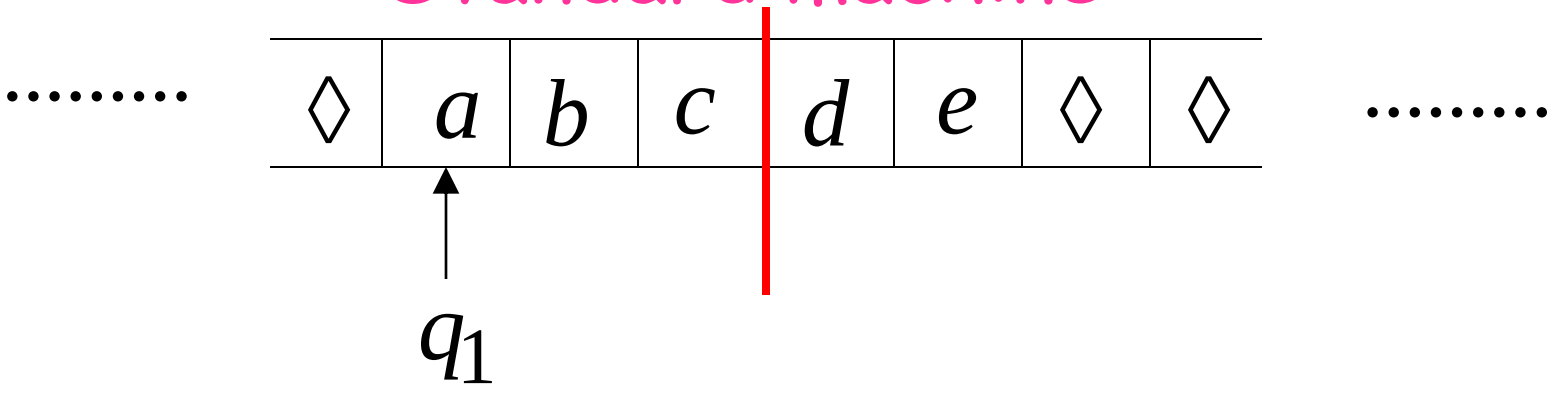
Left part



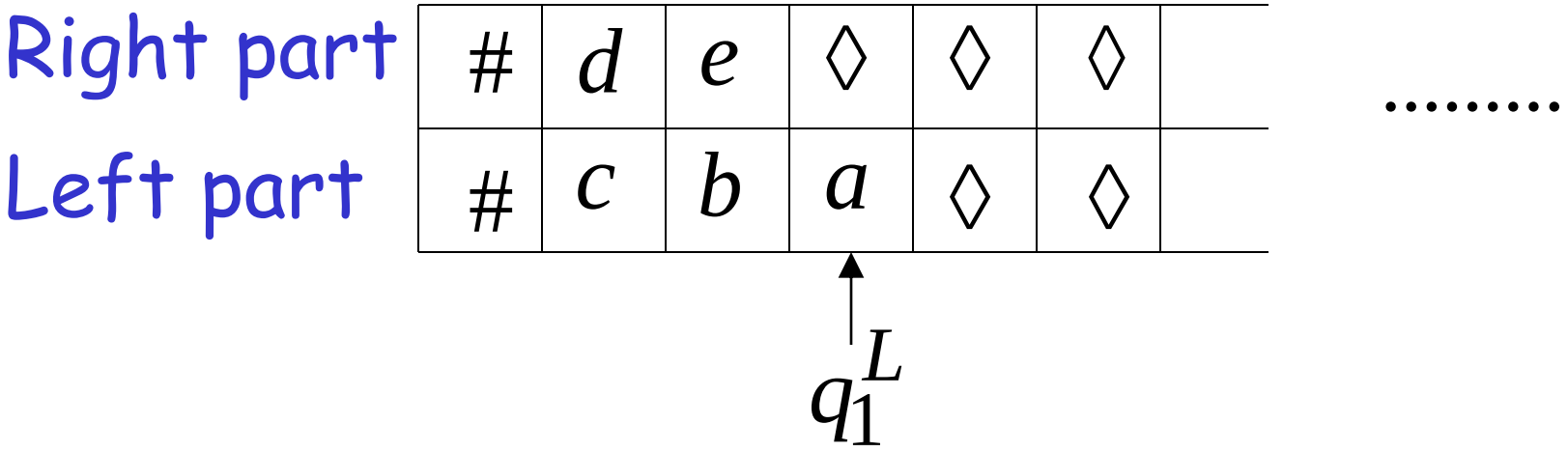
For all symbols x

Time 1

Standard machine

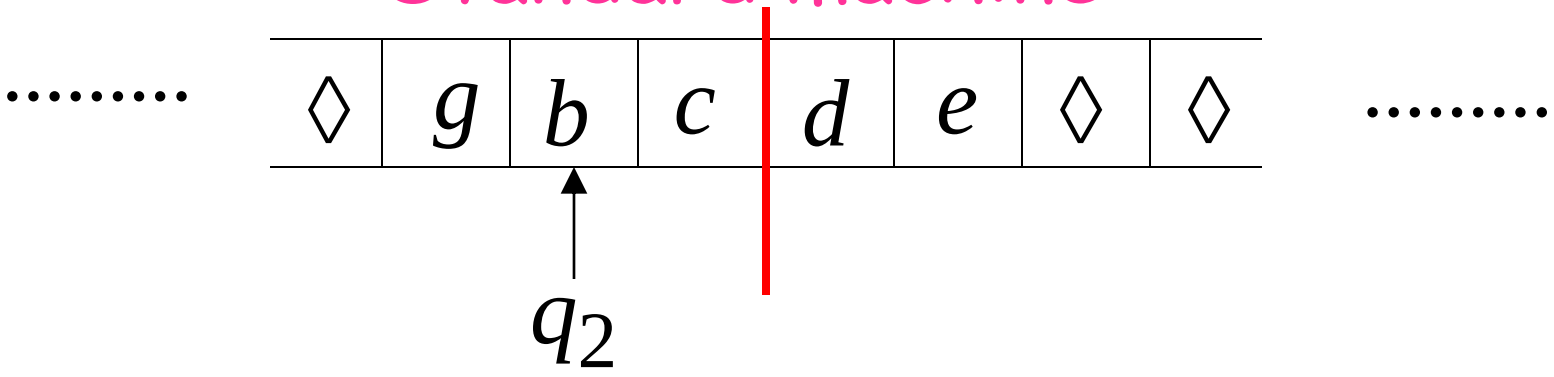


Semi-infinite tape machine

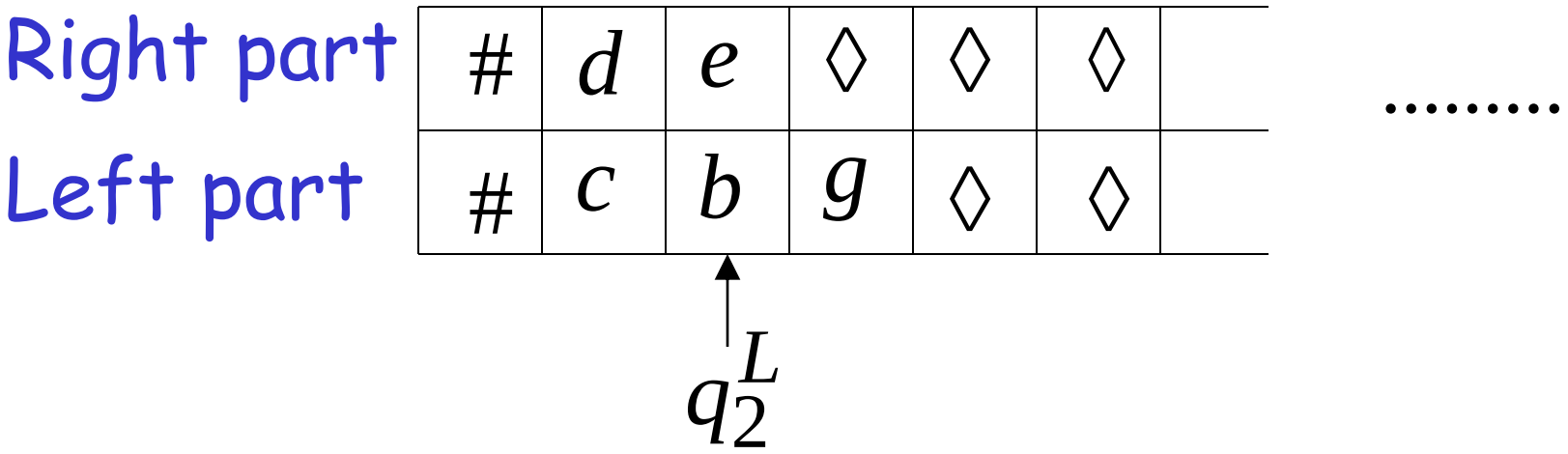


Time 2

Standard machine



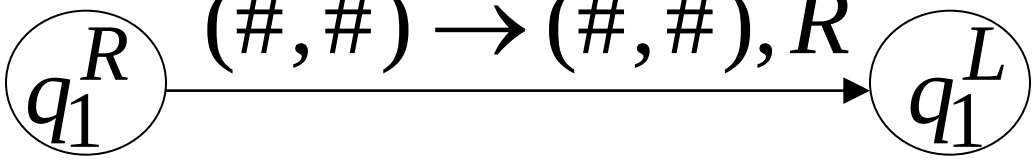
Semi-infinite tape machine



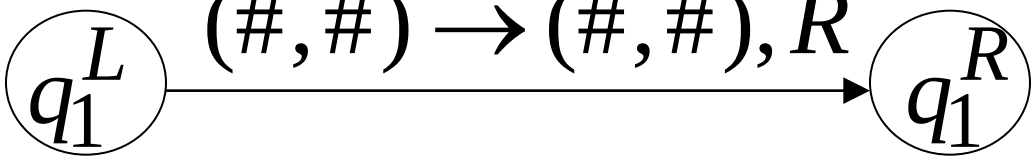
At the border:

Semi-infinite tape machine

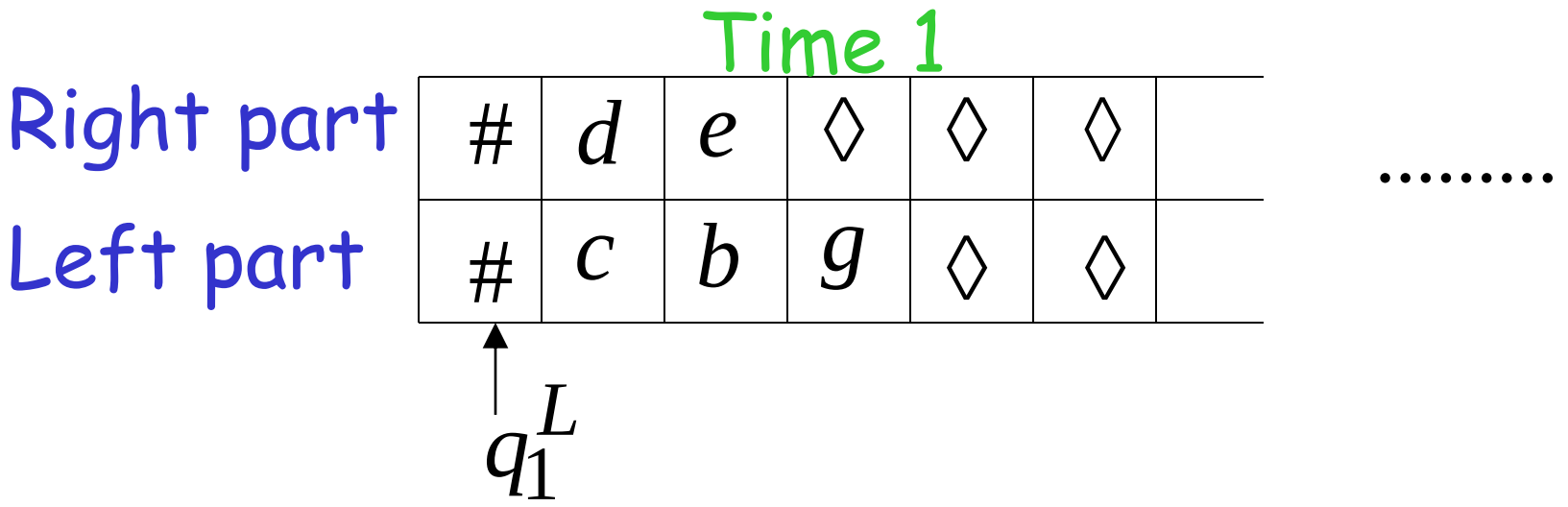
Right part

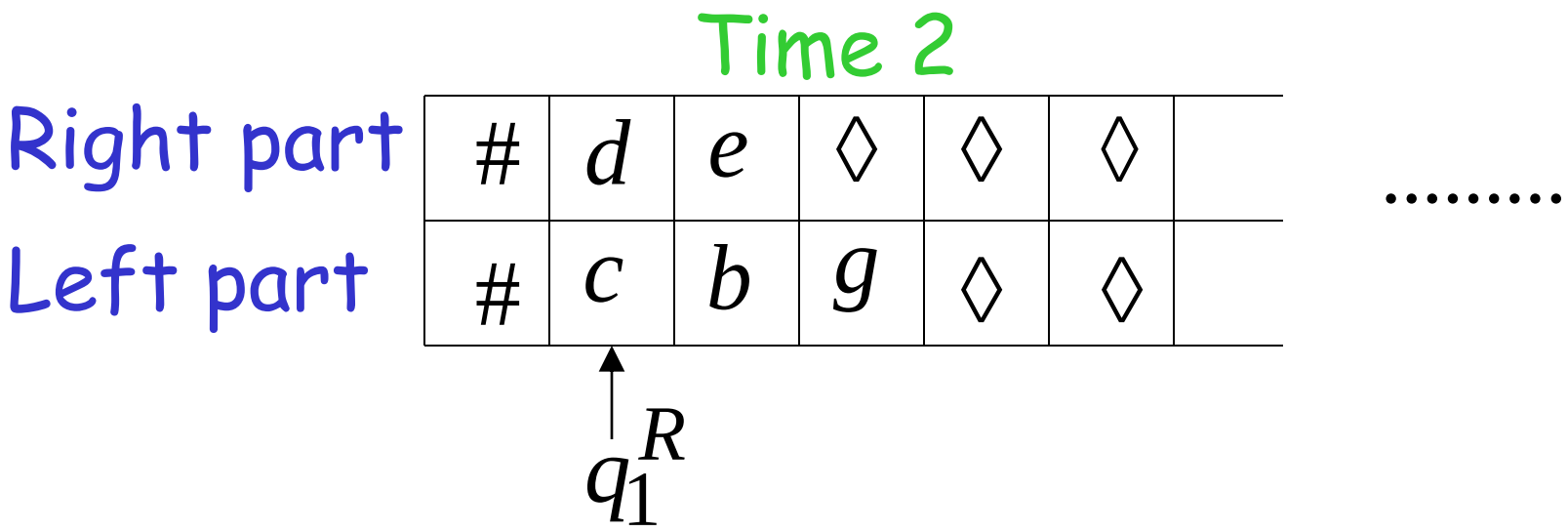


Left part



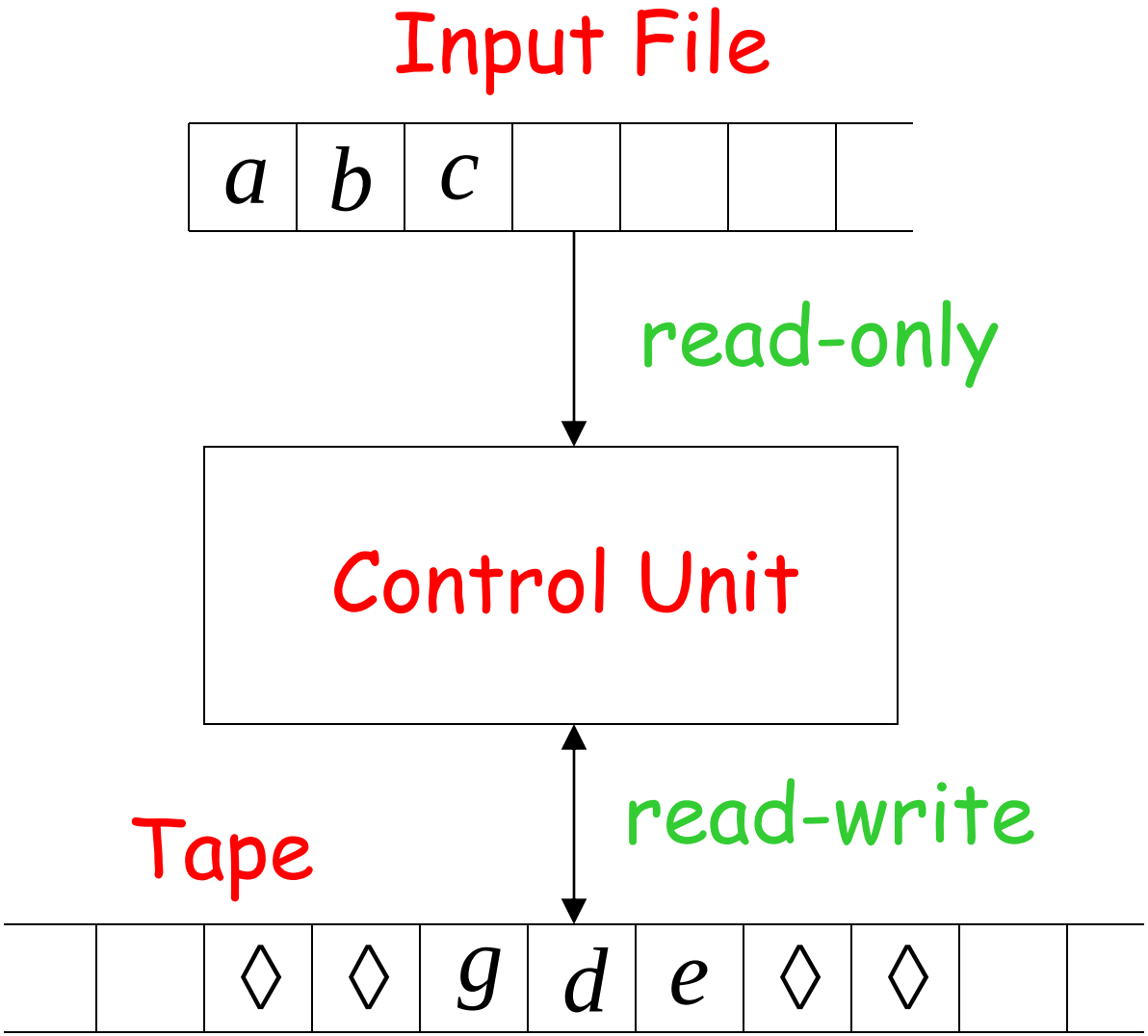
Semi-infinite tape machine





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

The Off-Line Machine



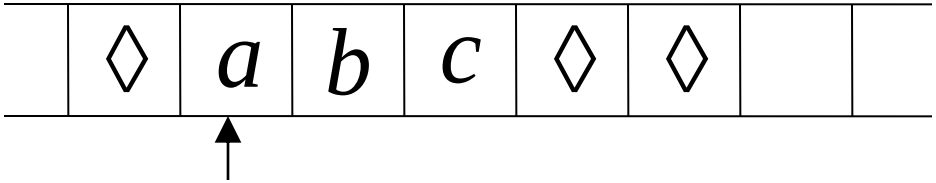
Off-line machines simulate Standard Turing Machines:

Off-line machine:

1. Copy input file to tape

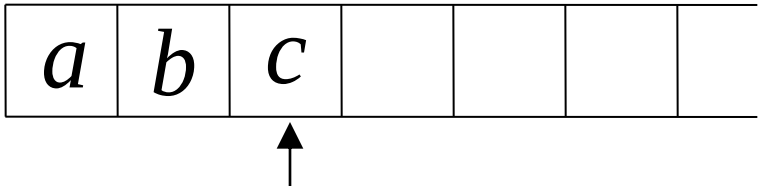
2. Continue computation as in Standard Turing machine

Standard machine

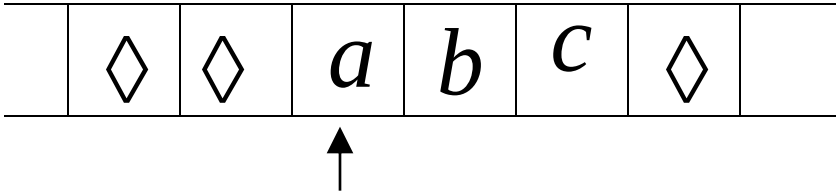


Off-line machine

Input File

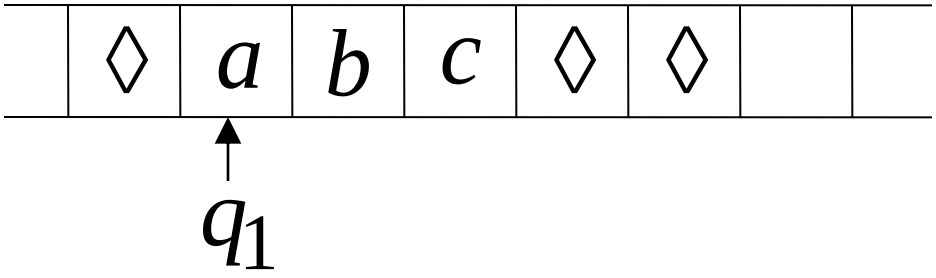


Tape



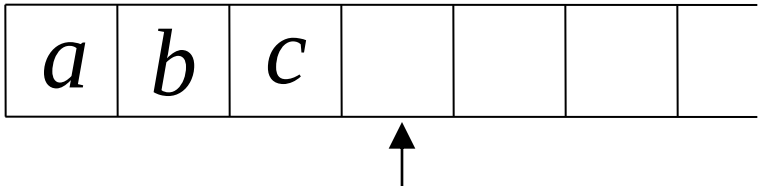
1. Copy input file to tape

Standard machine

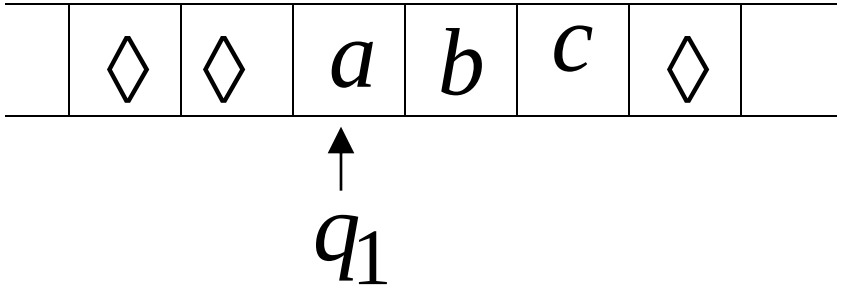


Off-line machine

Input File



Tape



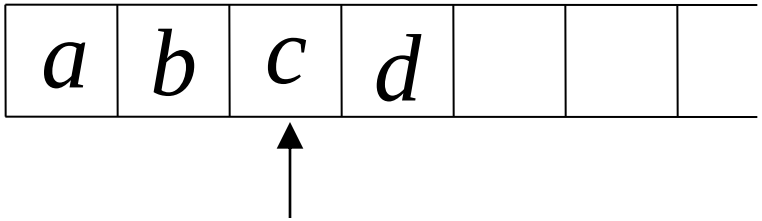
2. Do computations as in Turing machine

Standard Turing machines simulate Off-line machines:

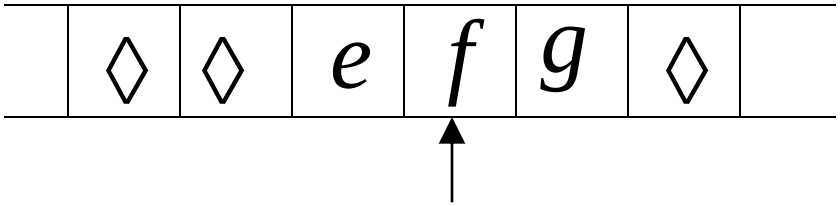
Use a Standard machine with four track tape to keep track of
the Off-line input file and tape contents

Off-line Machine

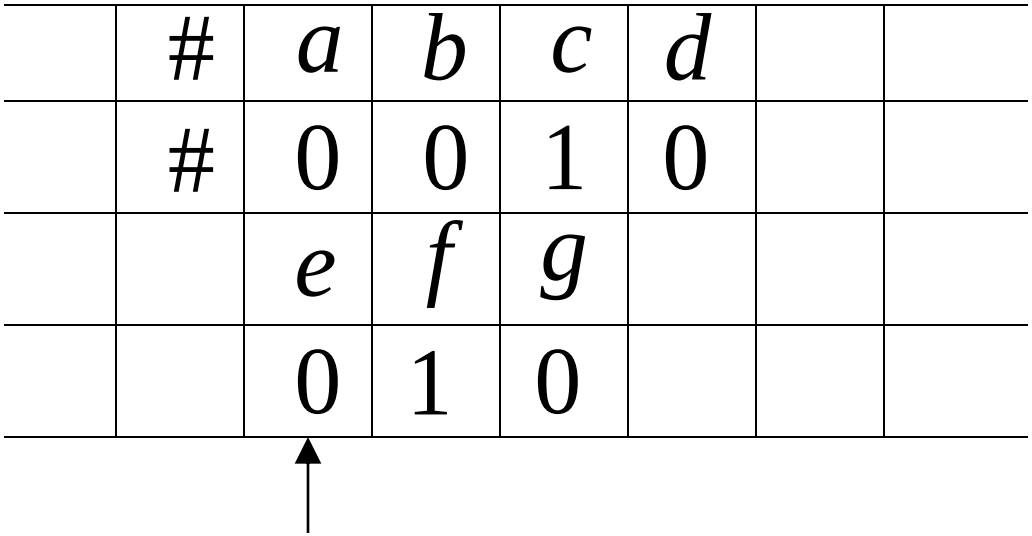
Input File



Tape



Four track tape -- Standard Machine



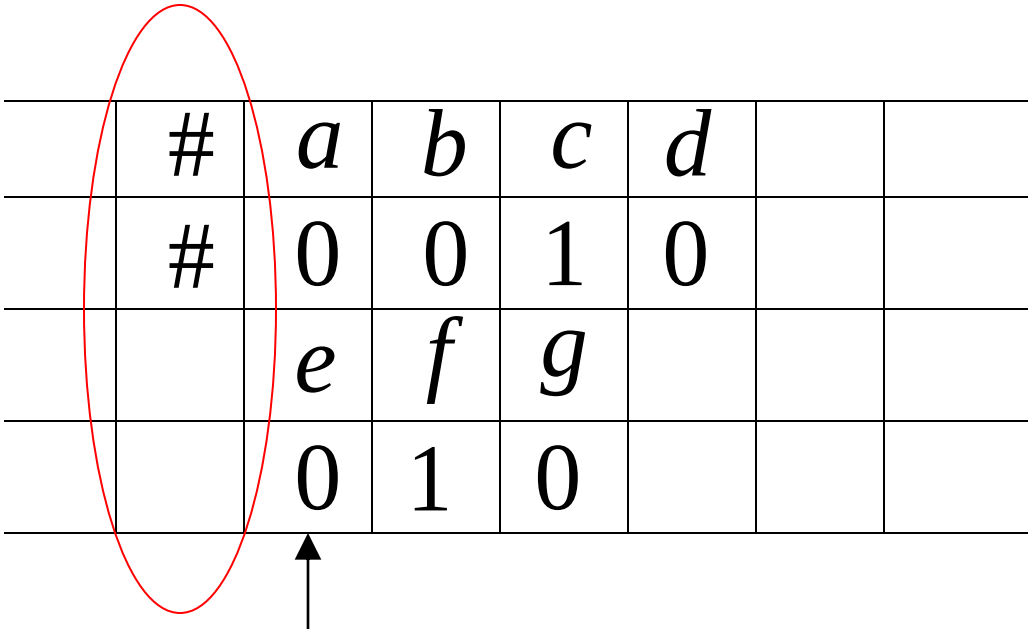
Input File

head position

Tape

head position

Reference point



Input File

head position

Tape

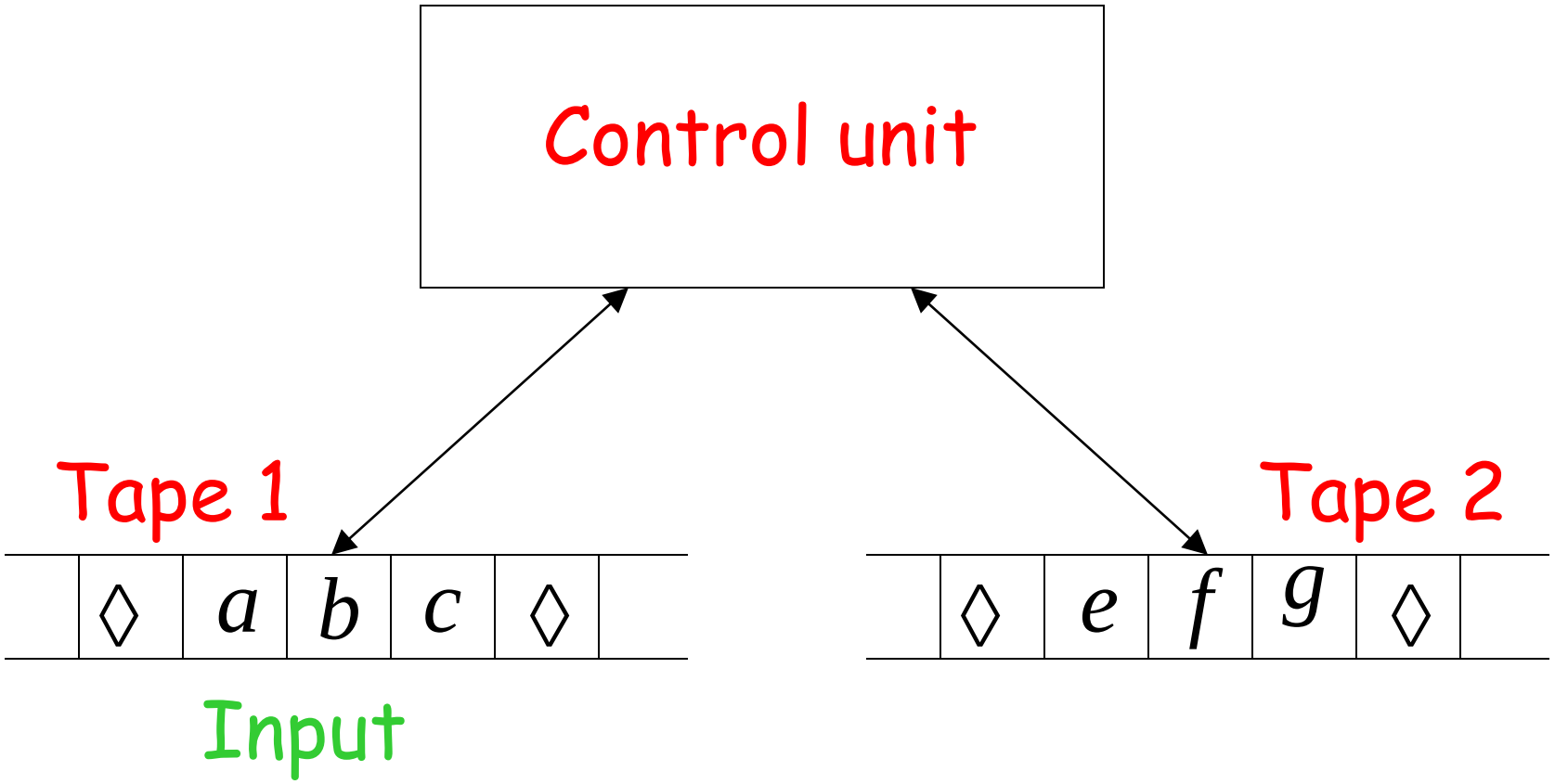
head position

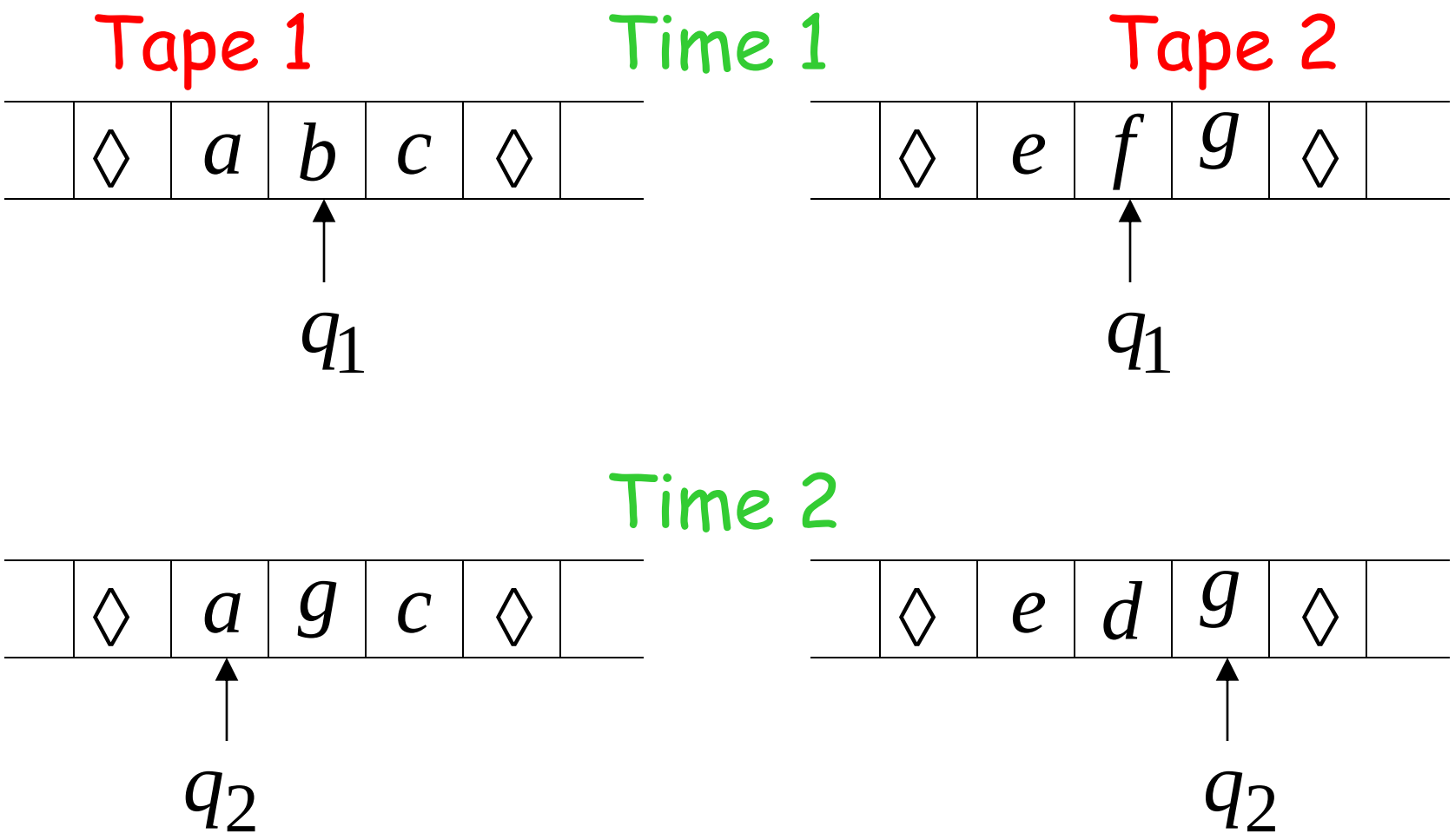
Repeat for each state transition:

- 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




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
$(b, f) \rightarrow (g, d), L, R$
q_1
q_2

Multitape machines simulate Standard Machines:

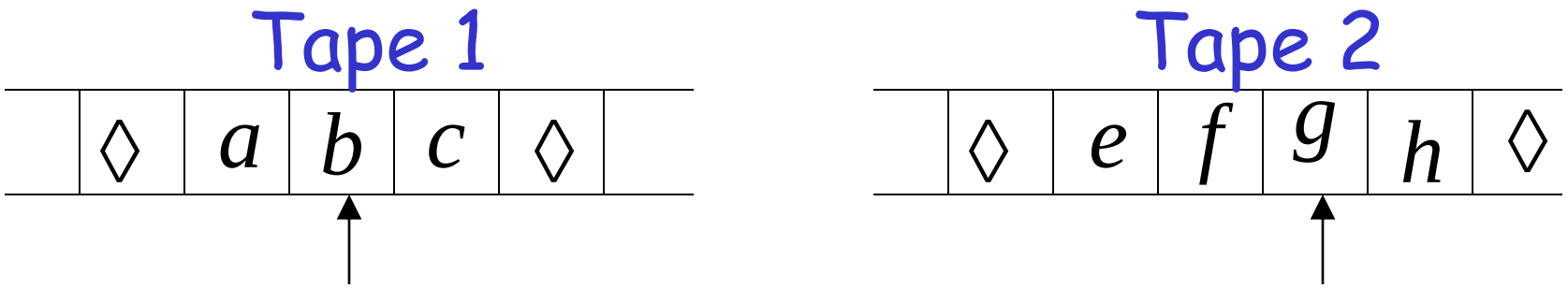
Use just one tape

Standard machines simulate Multitape machines:

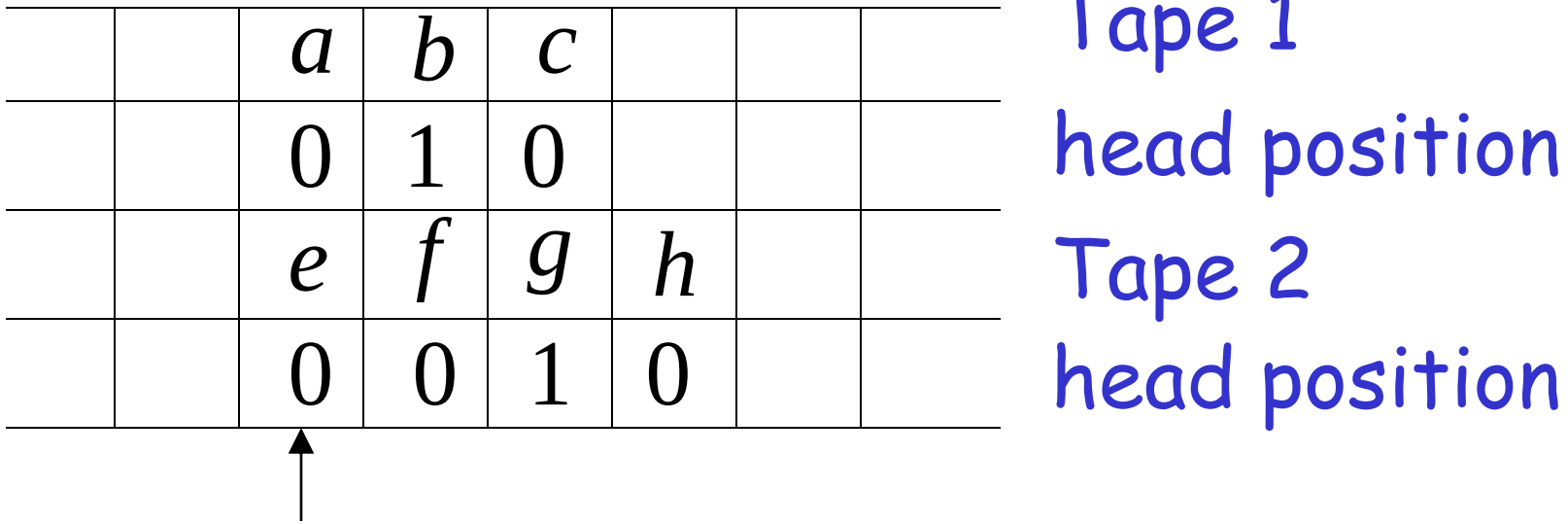
Standard machine:

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

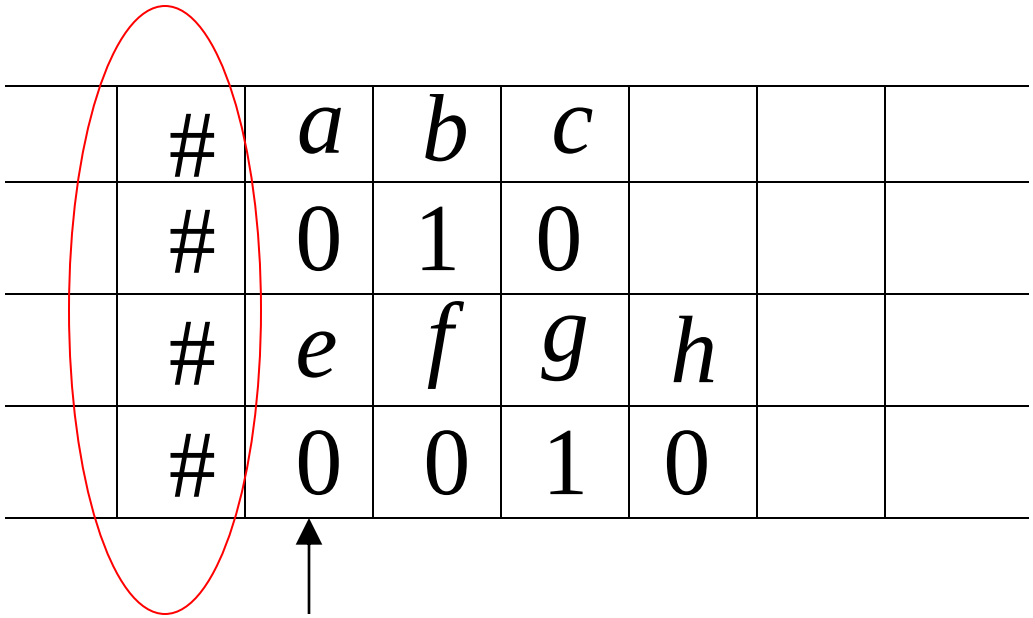
Multitape Machine



Standard machine with four track tape



Reference point



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:

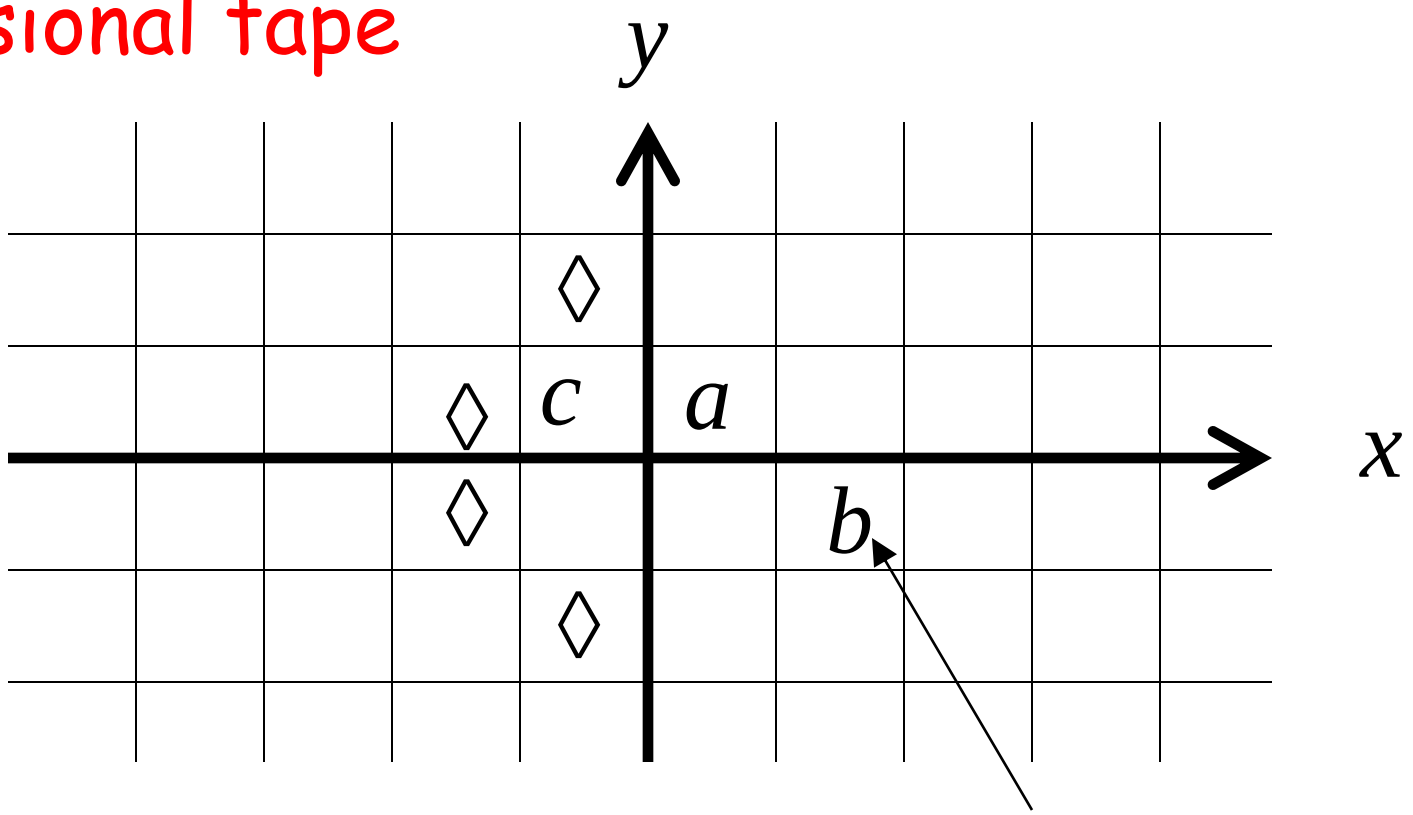
Copy b^n to tape 2 (n steps)

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

Multidimensional machines simulate Standard machines:

Use one dimension

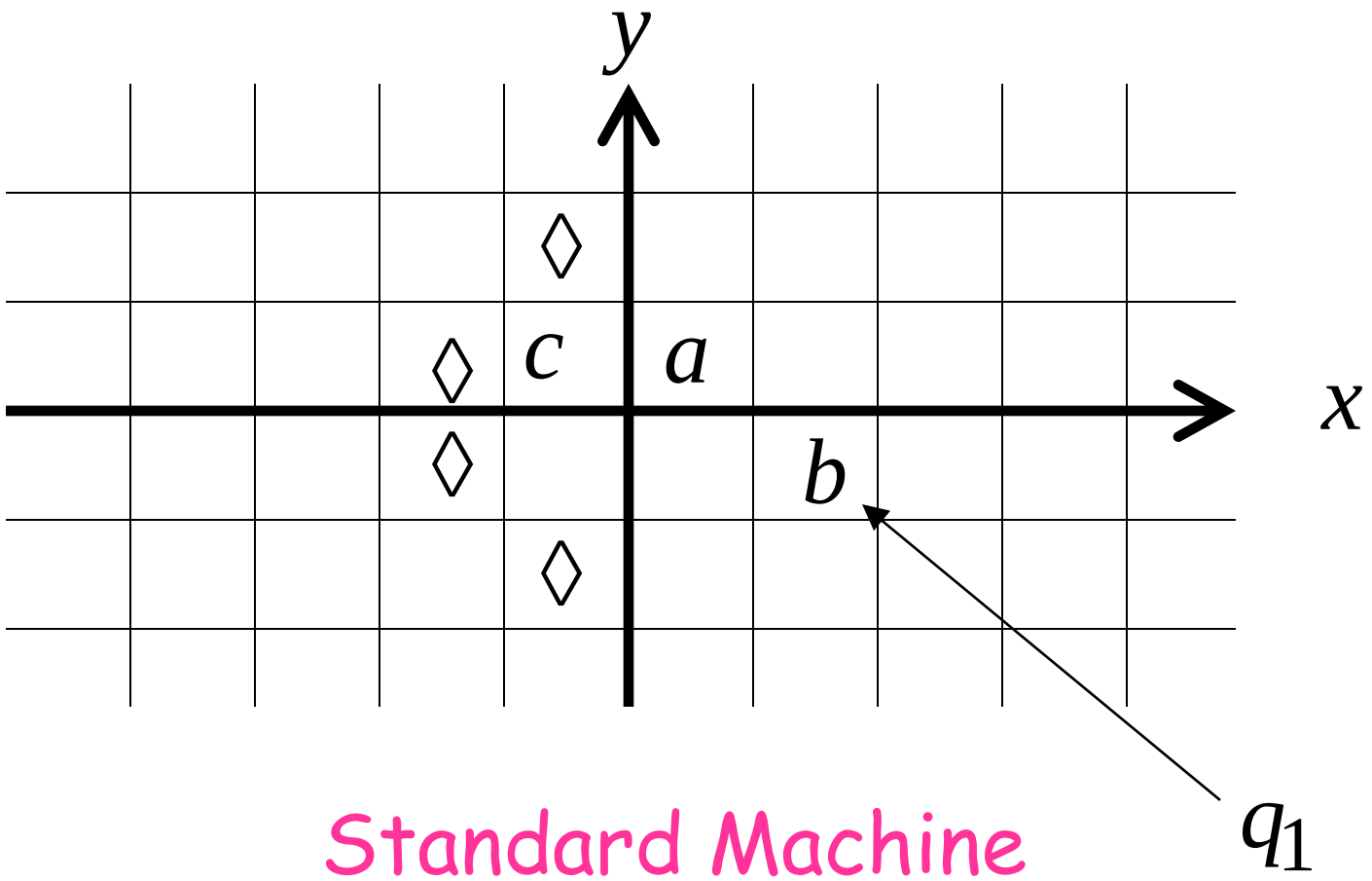
Standard machines simulate Multidimensional machines:

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	
	1	#	1	#	2	#	−	1	#	−	1

q_1

symbols

coordinates

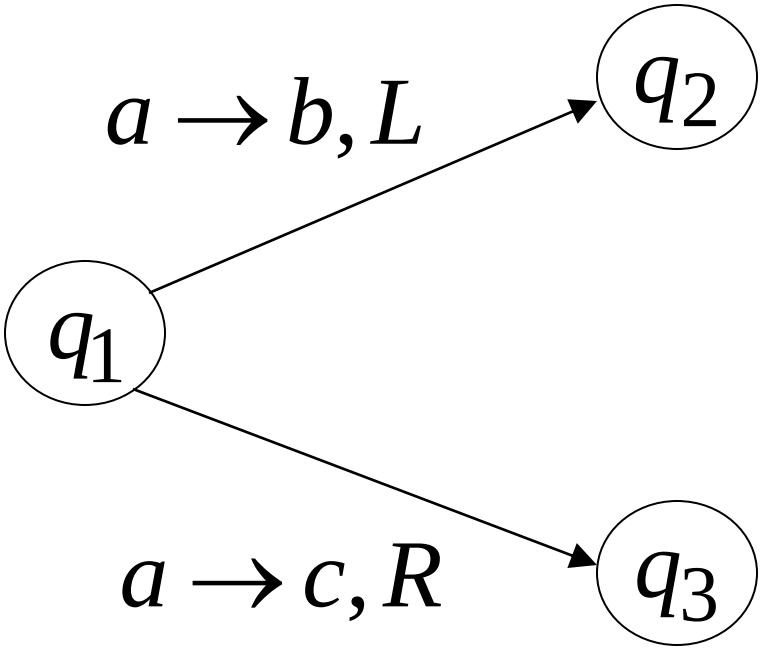
Standard machine:

Repeat for each transition

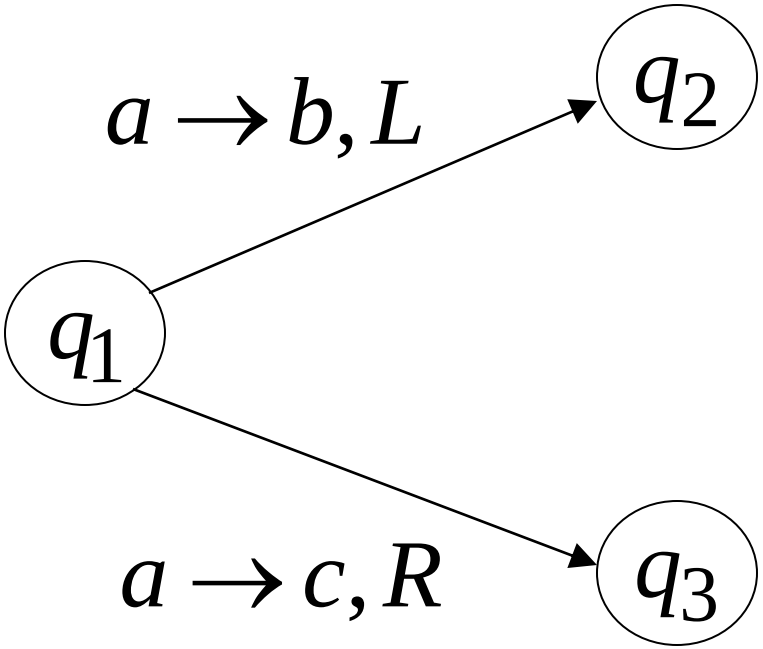
- Update current symbol
- Compute coordinates of next position
- Go to new position

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

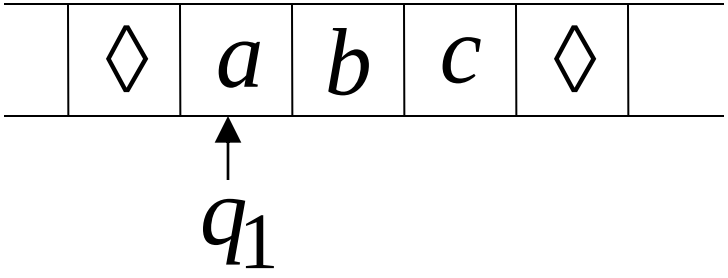
NonDeterministic Turing Machines



Non Deterministic Choice

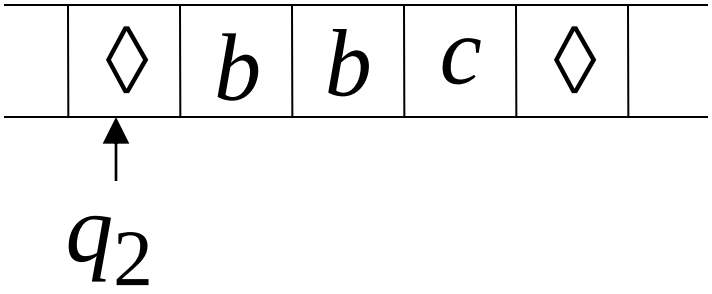


Time 0

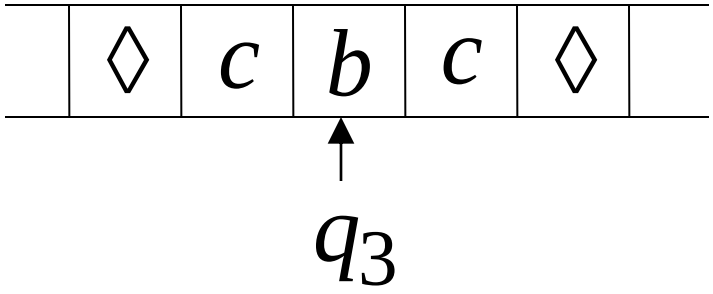


Time 1

Choice 1



Choice 2



Input string w is accepted if this is a possible computation

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

Initial configuration

Final state

Final Configuration

NonDeterministic Machines simulate Standard (deterministic) Machines:

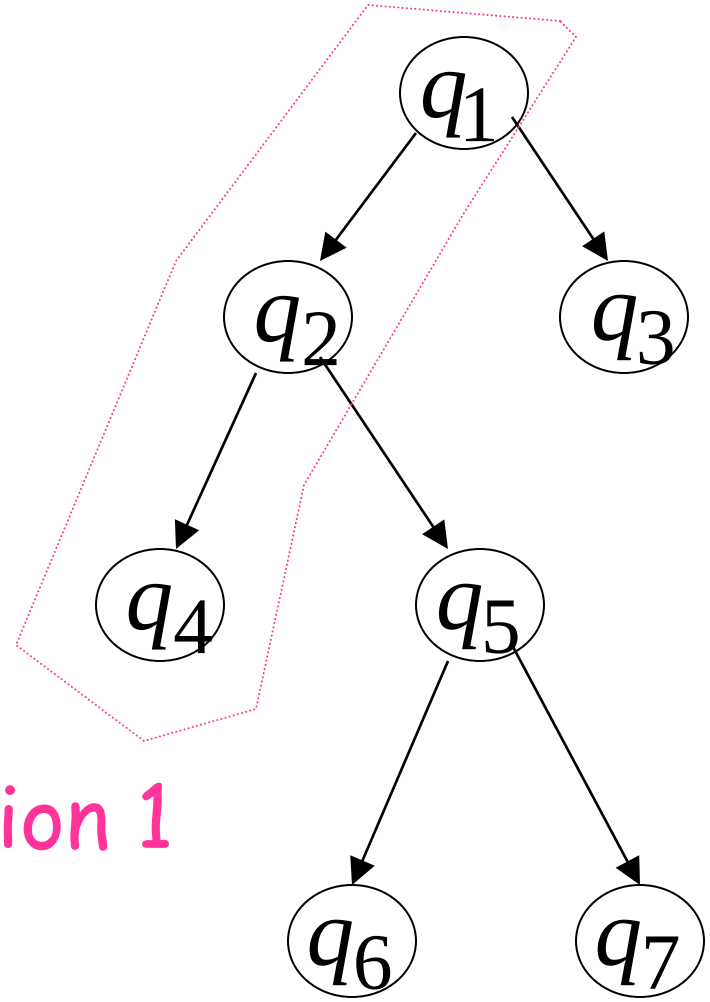
Every deterministic machine
is also a nondeterministic machine

Deterministic machines simulate NonDeterministic machines:

Deterministic machine:

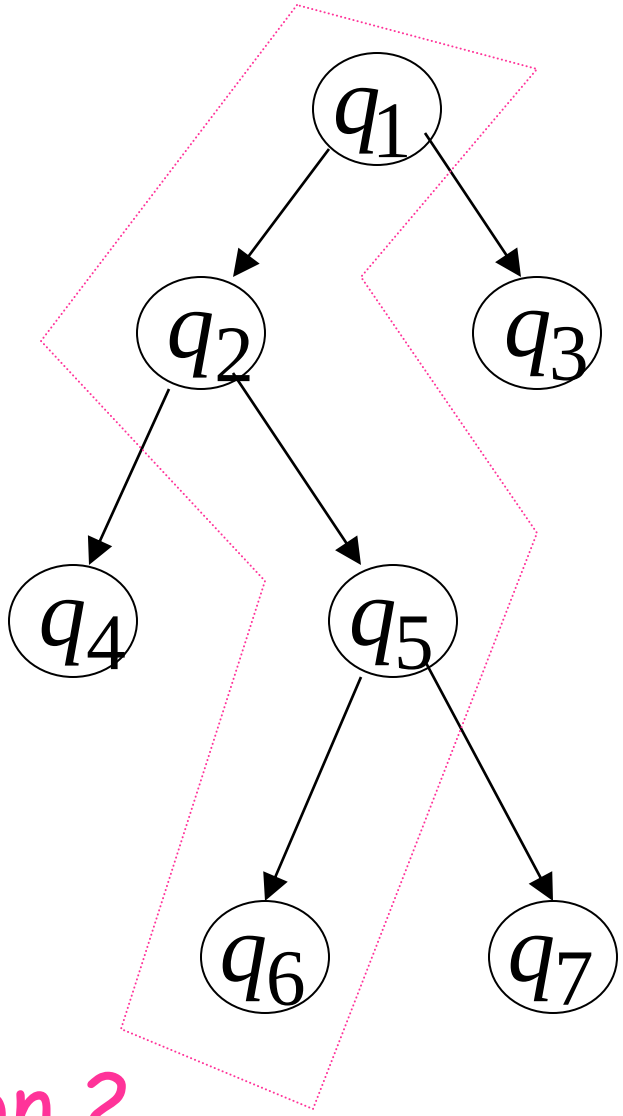
Keeps track of all possible computations

Non-Deterministic Choices



Computation 1

Non-Deterministic Choices



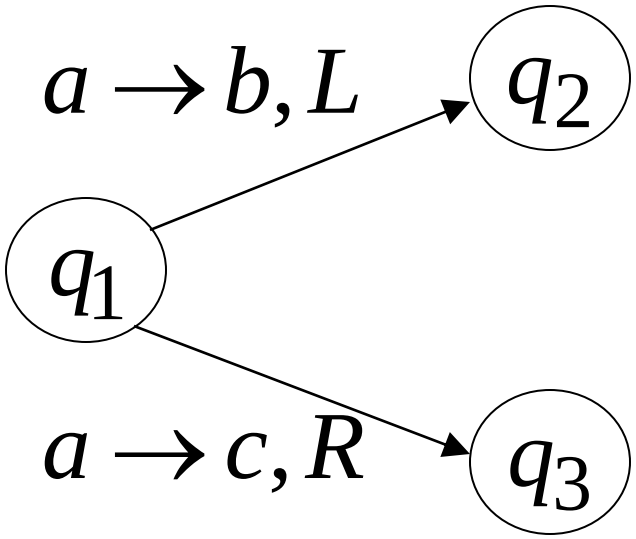
Computation 2

Simulation

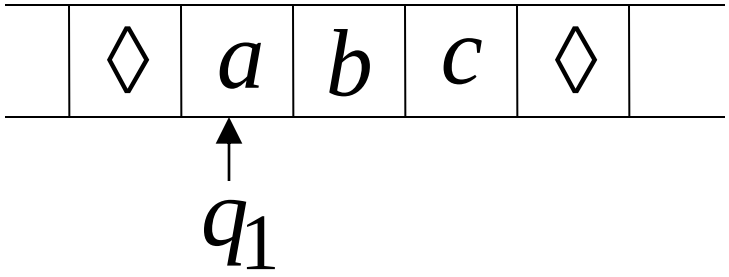
Deterministic machine:

- Keeps track of all possible computations
- Stores computations in a two-dimensional tape

NonDeterministic machine



Time 0

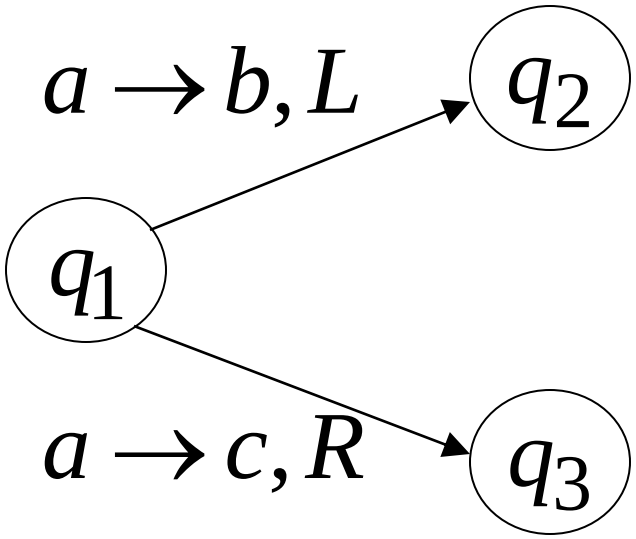


Deterministic machine

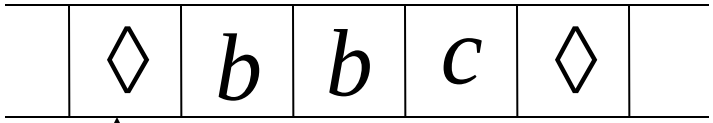
	#	#	#	#	#	#	
	#	a	b	c	#		
	#	q_1			#		
	#	#	#	#	#		

Computation 1

NonDeterministic machine

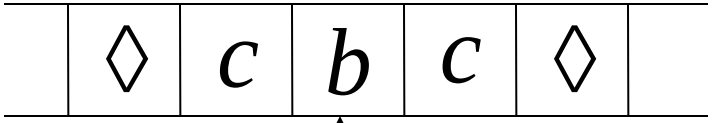


Time 1



Choice 1

q_2



Choice 2

q_3

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 replica

Theorem: NonDeterministic Machines have the same power with Deterministic machines

Remark:

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