Formal Languages

Context-Sensitive Languages

Hinrich Schütze

IMS, Uni Stuttgart, WS 2006/07

With slides borrowed from:

C. Busch, E. Rich, R. Sproat, G. Taylor and M. Volk

Linear Bounded Automata (LBAs)

are the same as Turing Machines

with one difference:

The input string tape space

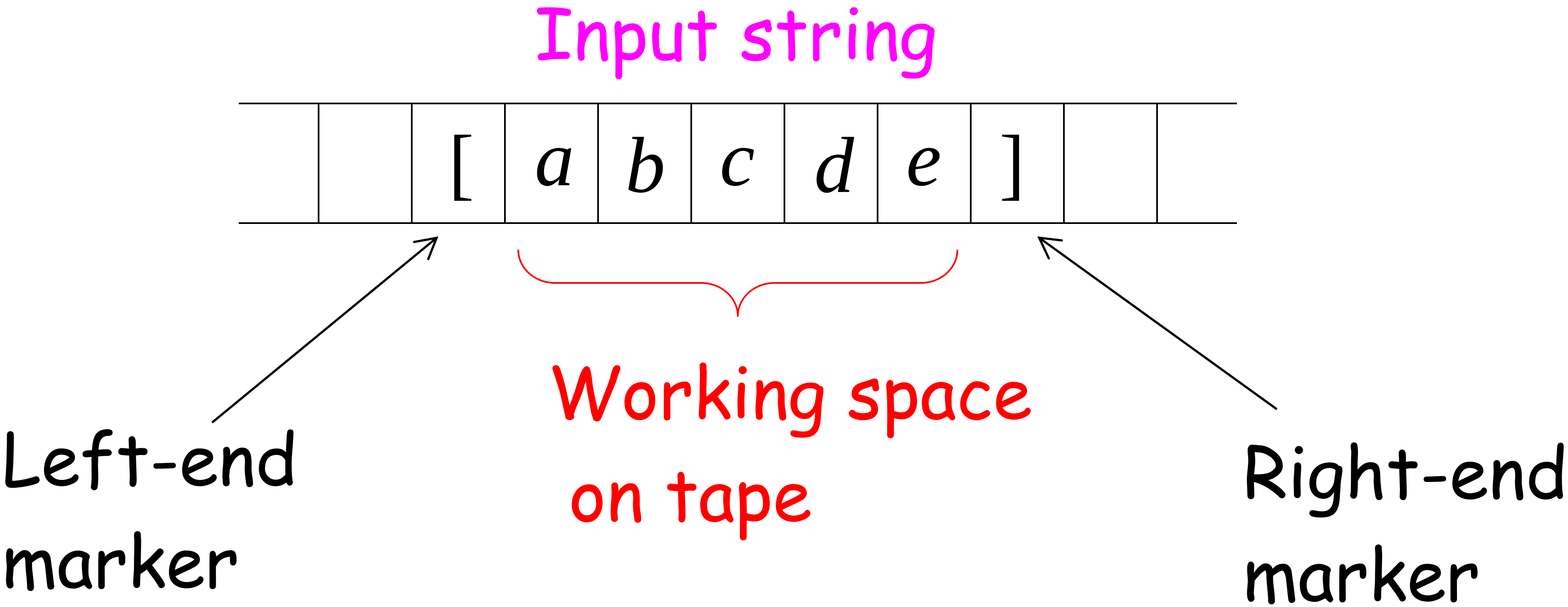
is the only tape space the machine is allowed to use

A linear bounded automaton is a nondeterministic Turing machine $M = (Q, \Sigma, \Gamma, \delta, q0, B, F)$, subject to the restriction that Σ must contain two special symbols [and], such that $\delta(qi, [)$ can contain only elements of the form $(qj, [, R)$, and $\delta(qi,])$ can contain only elements of the form $(qj,], L)$.

$$q_0 [w] \overset{*}{\vdash} [x_1 q_f x_2]$$

for some $qf \in F$, $x1, x2 \in \Gamma^*$.

Linear Bounded Automaton (LBA)



All computation is done between end markers

Example languages accepted by LBAs:

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

LBA's have more power than NPDA's

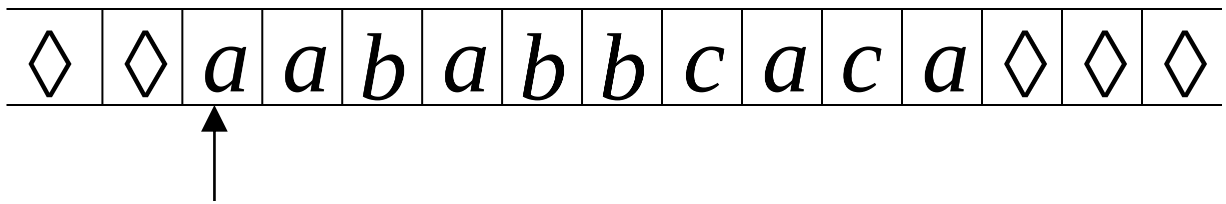
LBA's have also less power than Turing Machines

LBA for $L=\{a^n b^n \mid n>=1\}$

Variations of the Turing Machine

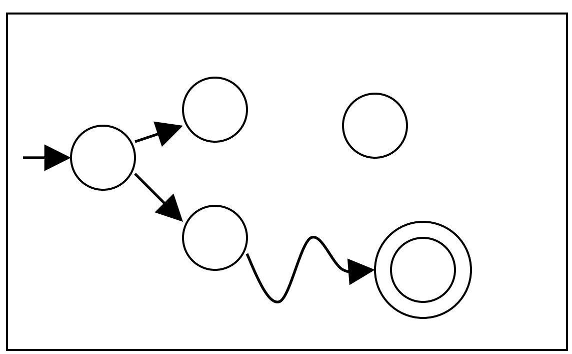
The Standard Model

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 as the Standard Model

Same Power of two classes means:

The two 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

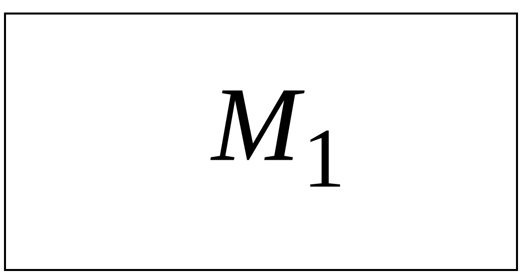
such that: $L(M_1) = L(M_2)$

And vice-versa

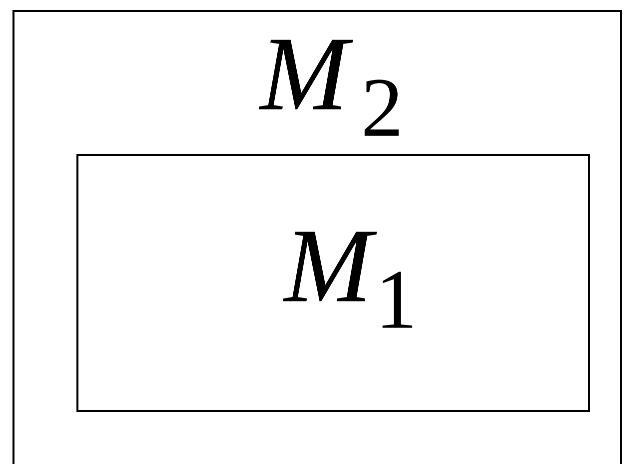
Simulation: a technique to prove same power

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

First Class
Original Machine



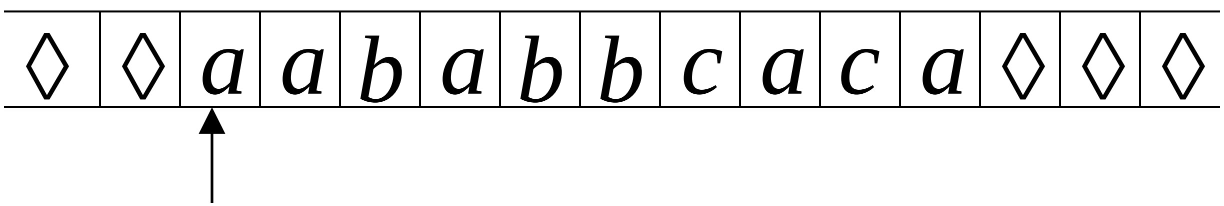
Second Class
Simulation Machine



Turing Machines with Stay-Option

The head can stay in the same position

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

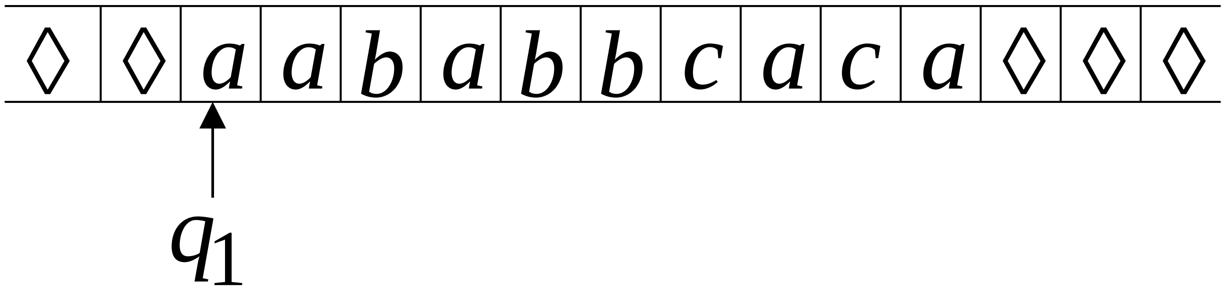


Left, Right, Stay

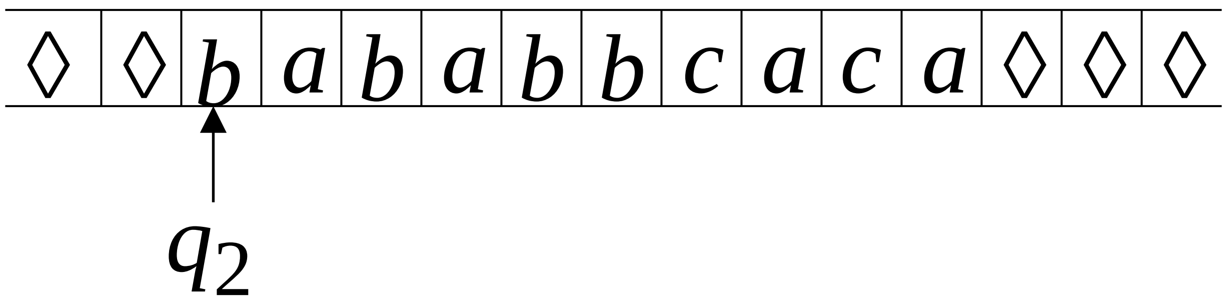
L,R,S: moves

Example:

Time 1



Time 2



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

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

Proof:

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

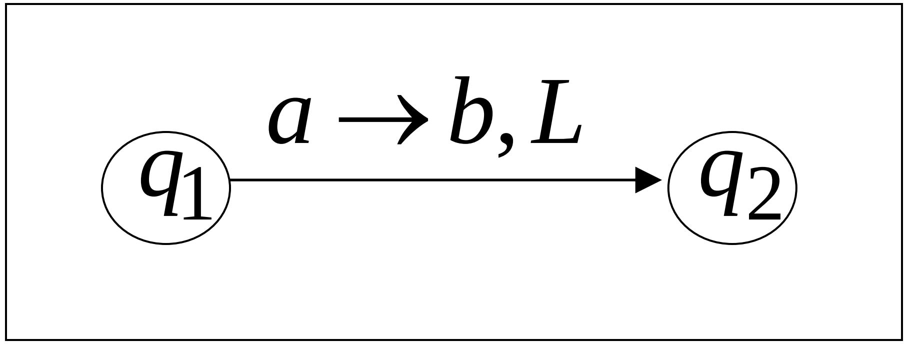
Proof: a Standard machine is also
a Stay-Option machine
(that never uses the S move)

Proof:

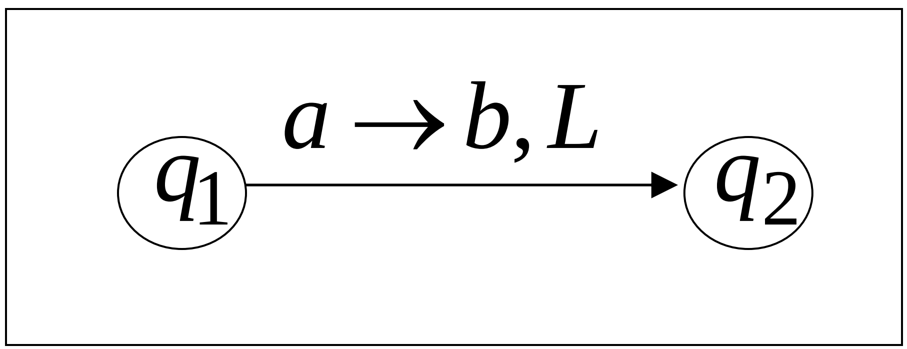
Part 2: Standard Machines
are at least as powerful as
Stay-Option machines

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

Stay-Option Machine

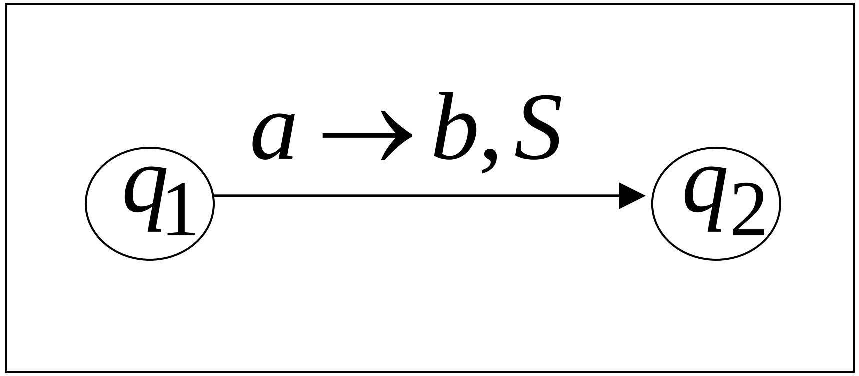


Simulation in Standard Machine

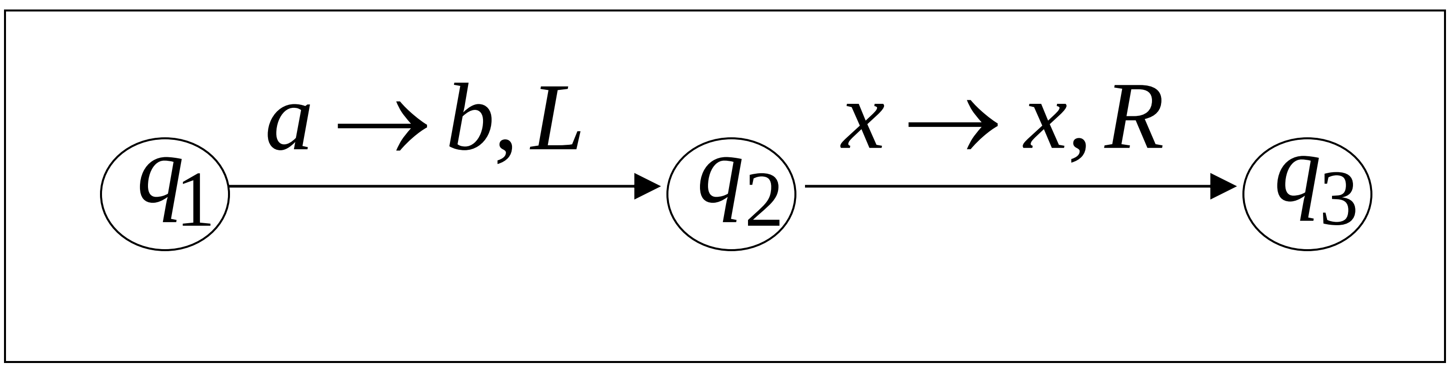


Similar for Right moves

Stay-Option Machine



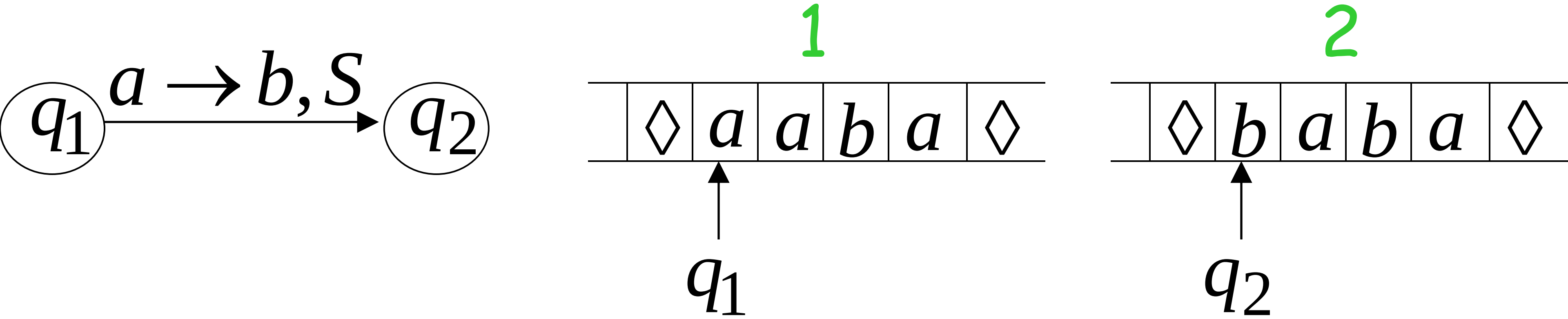
Simulation in Standard Machine



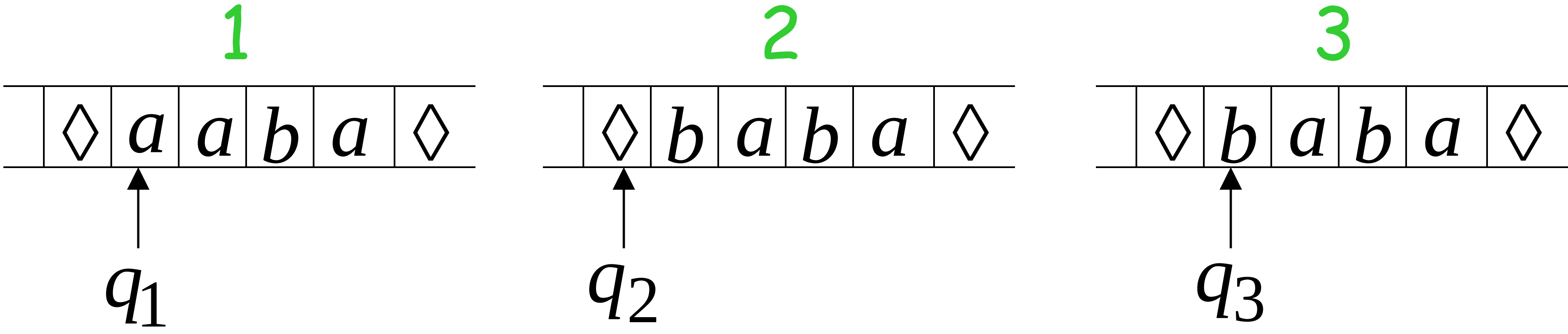
For every symbol x

Example

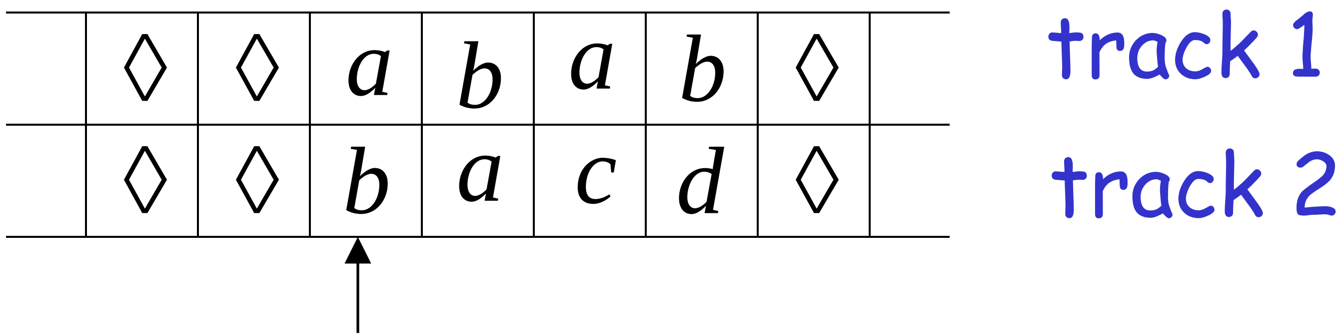
Stay-Option Machine:



Simulation in Standard Machine:

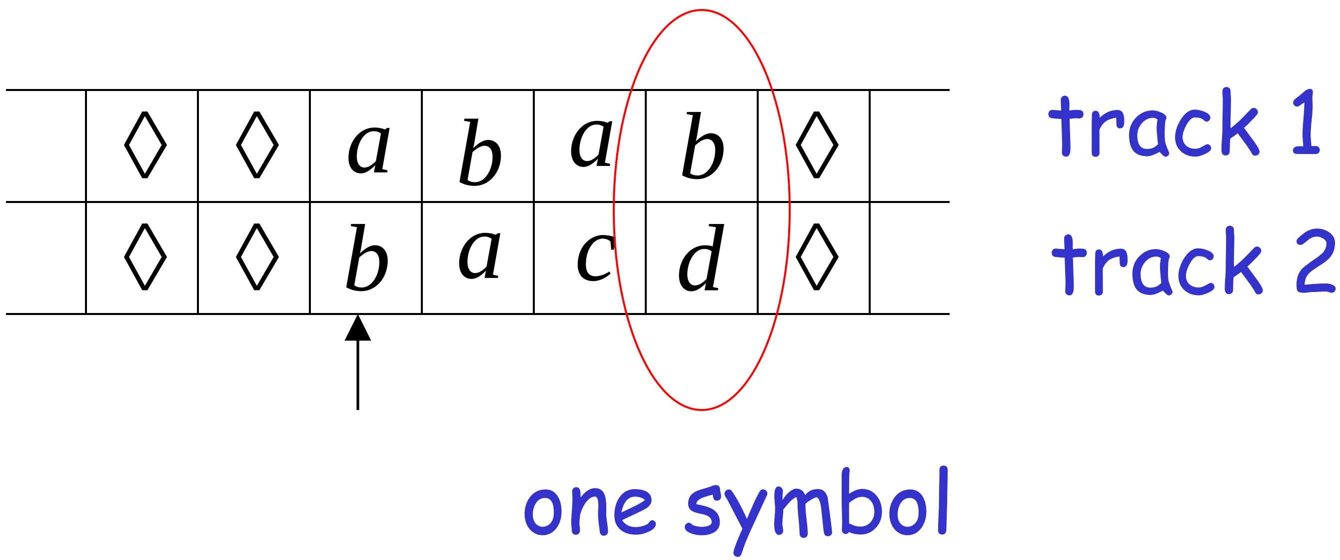


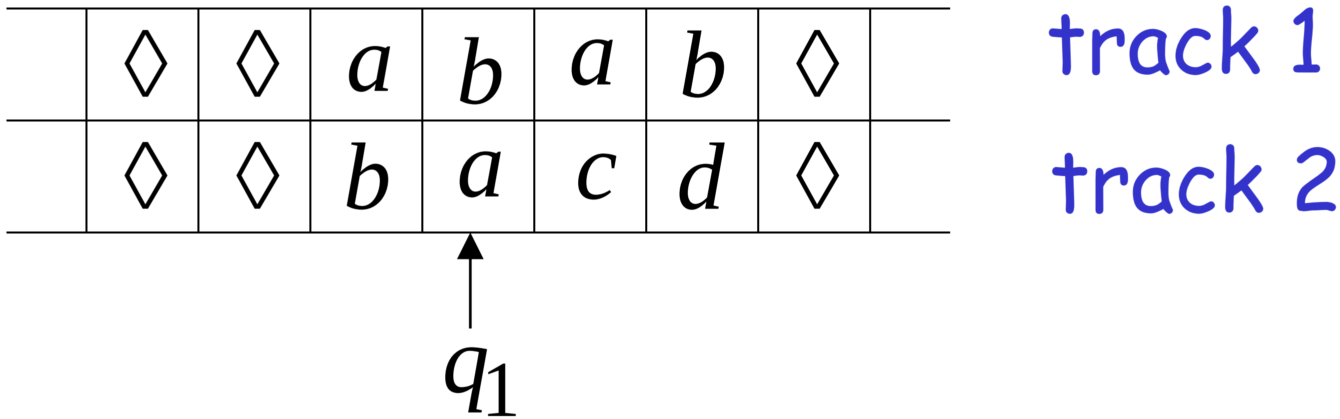
Standard Machine--Multiple Track Tape

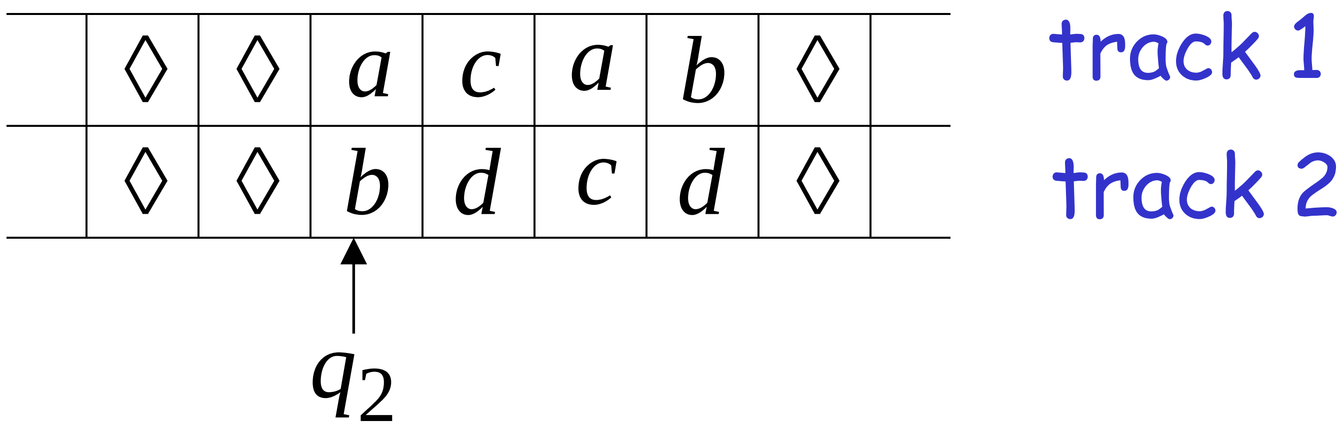


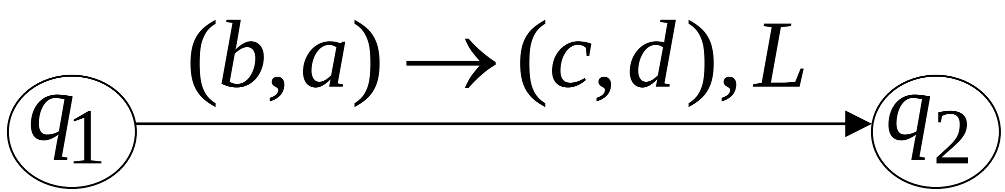
Proof of equivalence?

Standard Machine--Multiple Track Tape

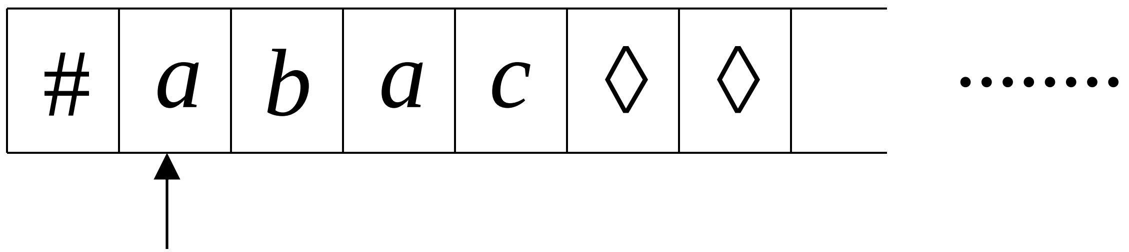


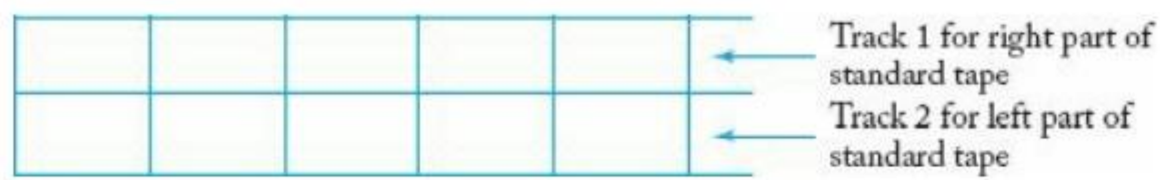

◊ ◊ a b a b ◊
◊ ◊ b a c d ◊
track 1
track 2
q_1


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


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

Semi-Infinite Tape





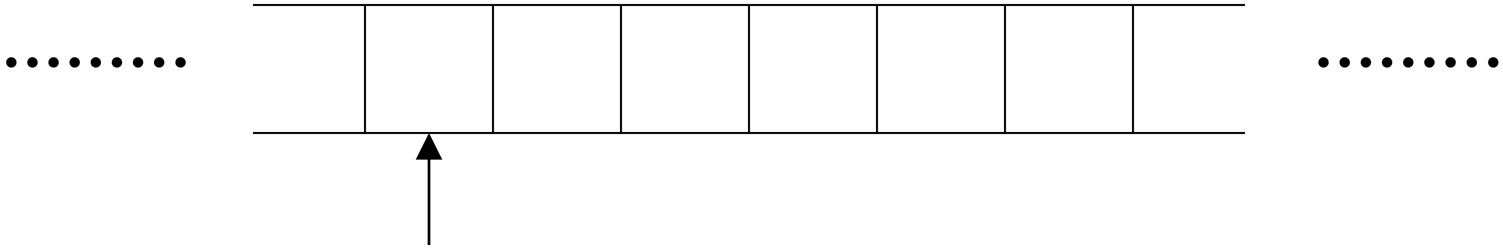
Proof of equivalence?

Standard Turing machines simulate
Semi-infinite tape machines:

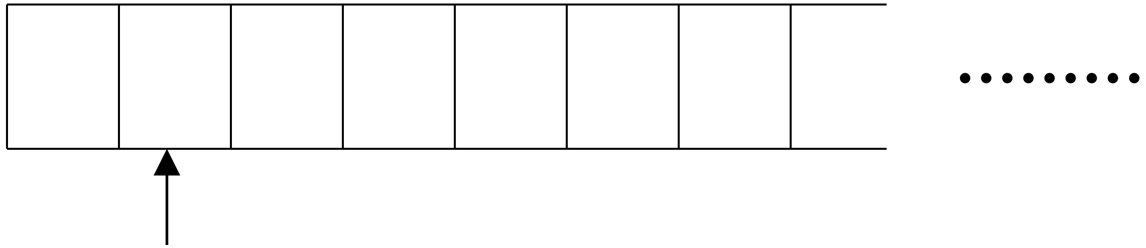
Trivial

Semi-infinite tape machines simulate Standard Turing machines:

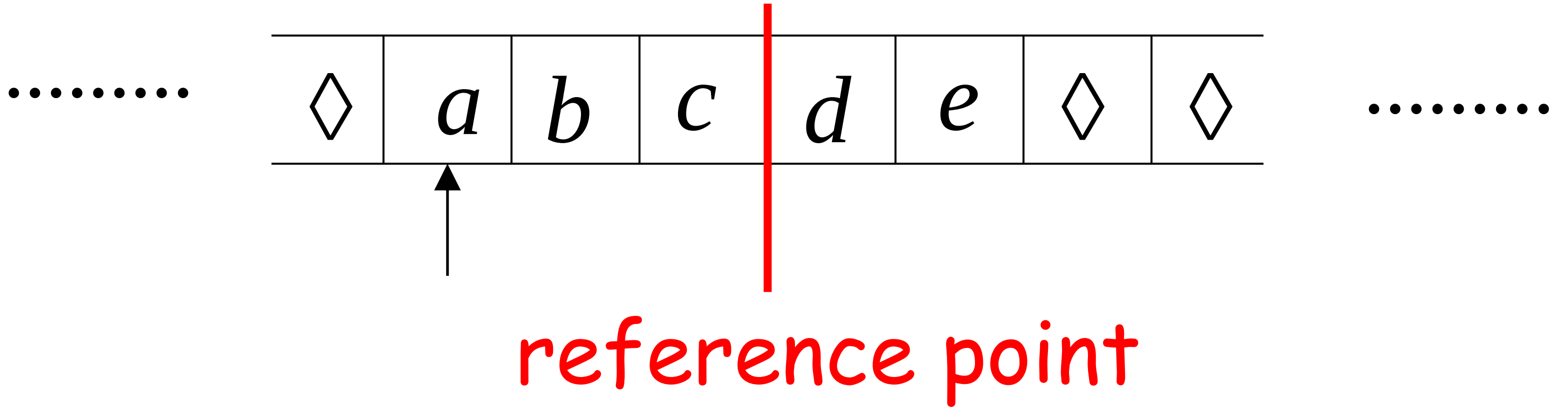
Standard machine



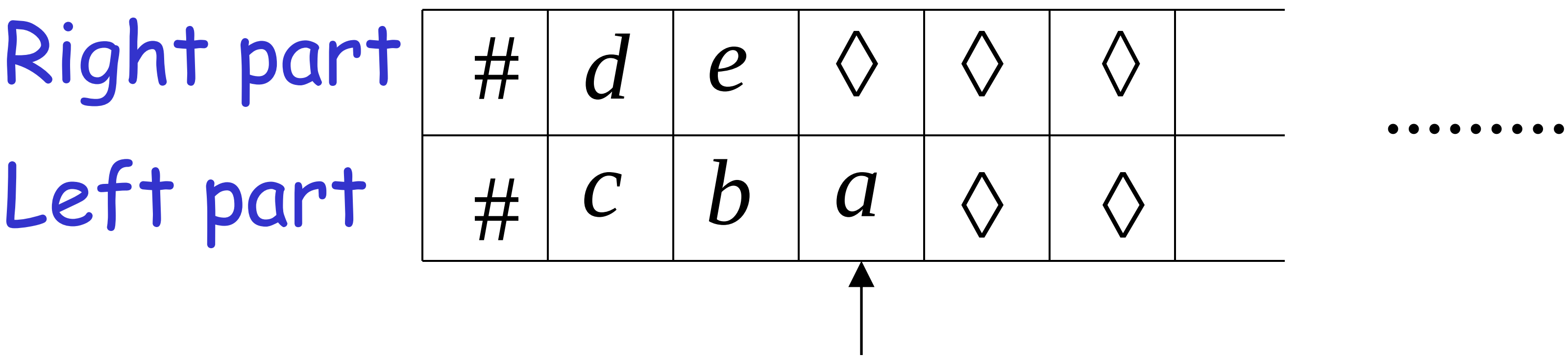
Semi-infinite tape machine



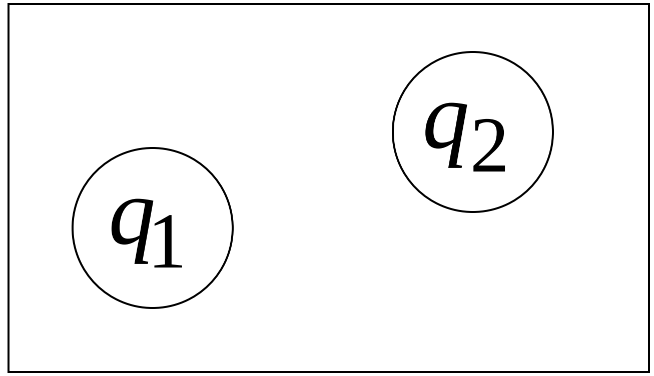
Standard machine



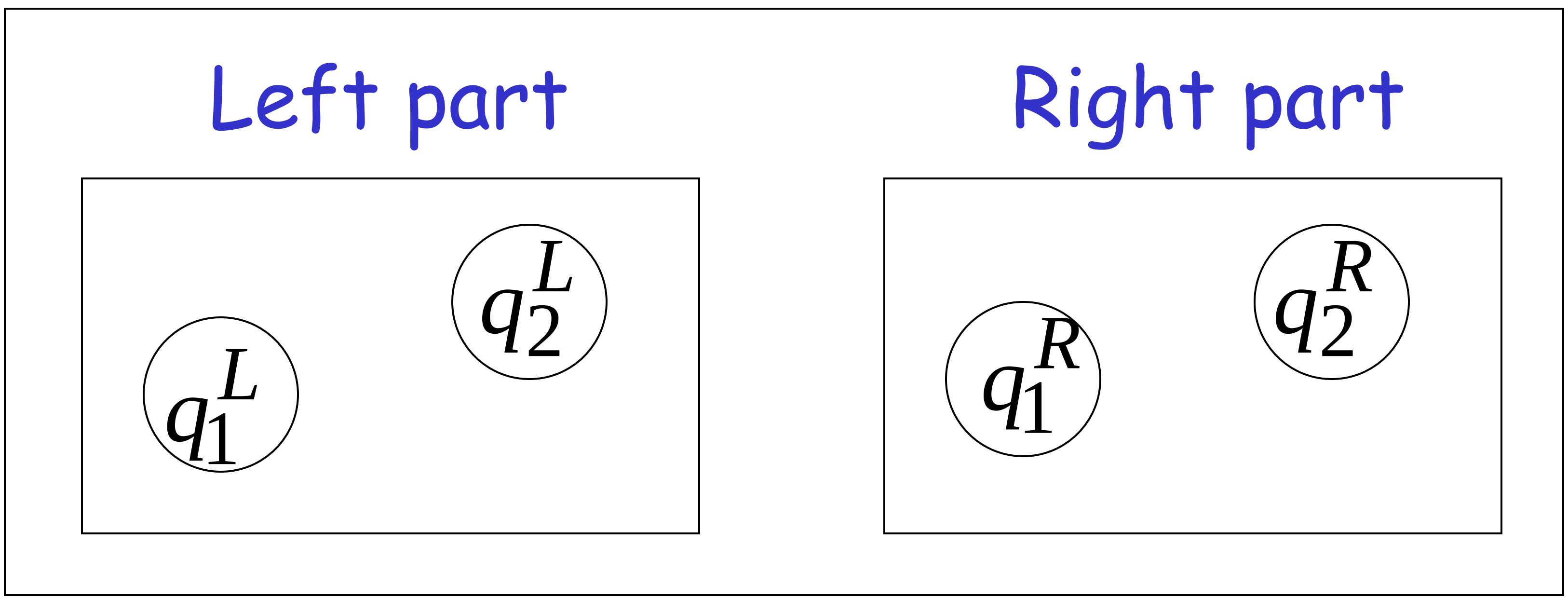
Semi-infinite tape machine with two tracks



Standard machine



Semi-infinite tape machine



Standard machine

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

Semi-infinite tape machine

Right part

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

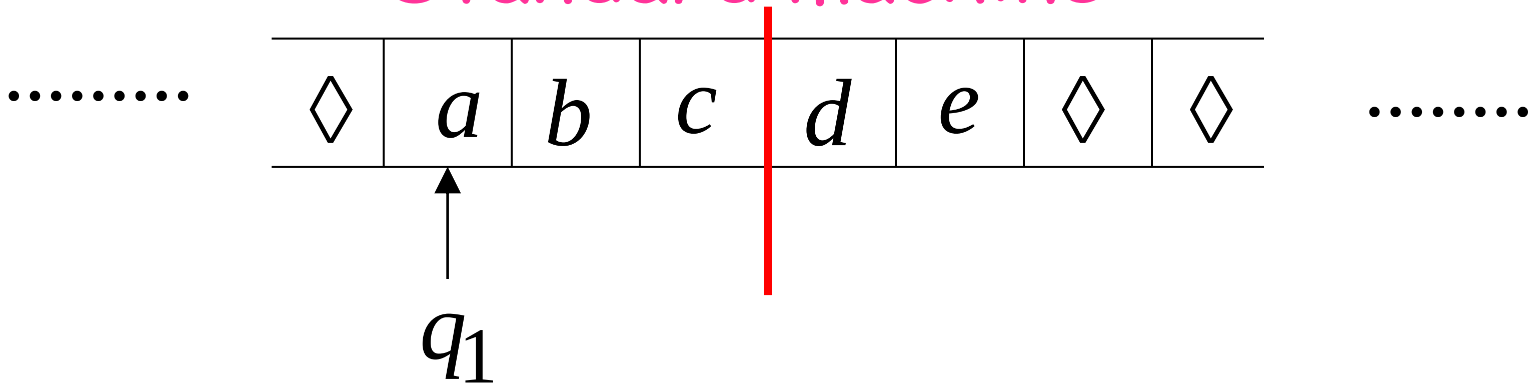
Left part

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

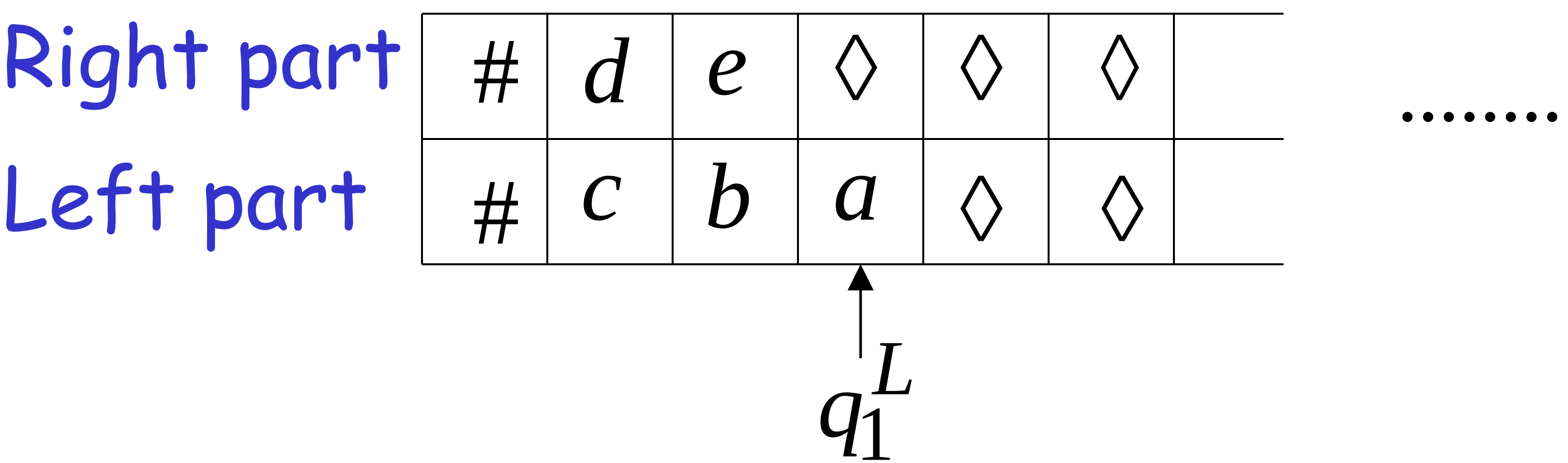
For all symbols x

Time 1

Standard machine



Semi-infinite tape machine



Time 2

Standard machine

....	◊	g	b	c	d	e	◊	◊	....

$\uparrow$ q_2

Semi-infinite tape machine

Right part	#	d	e	◊	◊	◊		....
Left part	#	c	b	g	◊	◊		

$\uparrow$ q_2^L

At the border:

Semi-infinite tape machine

Right part

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

Left part

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

Semi-infinite tape machine

Time 1

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

q_1^L

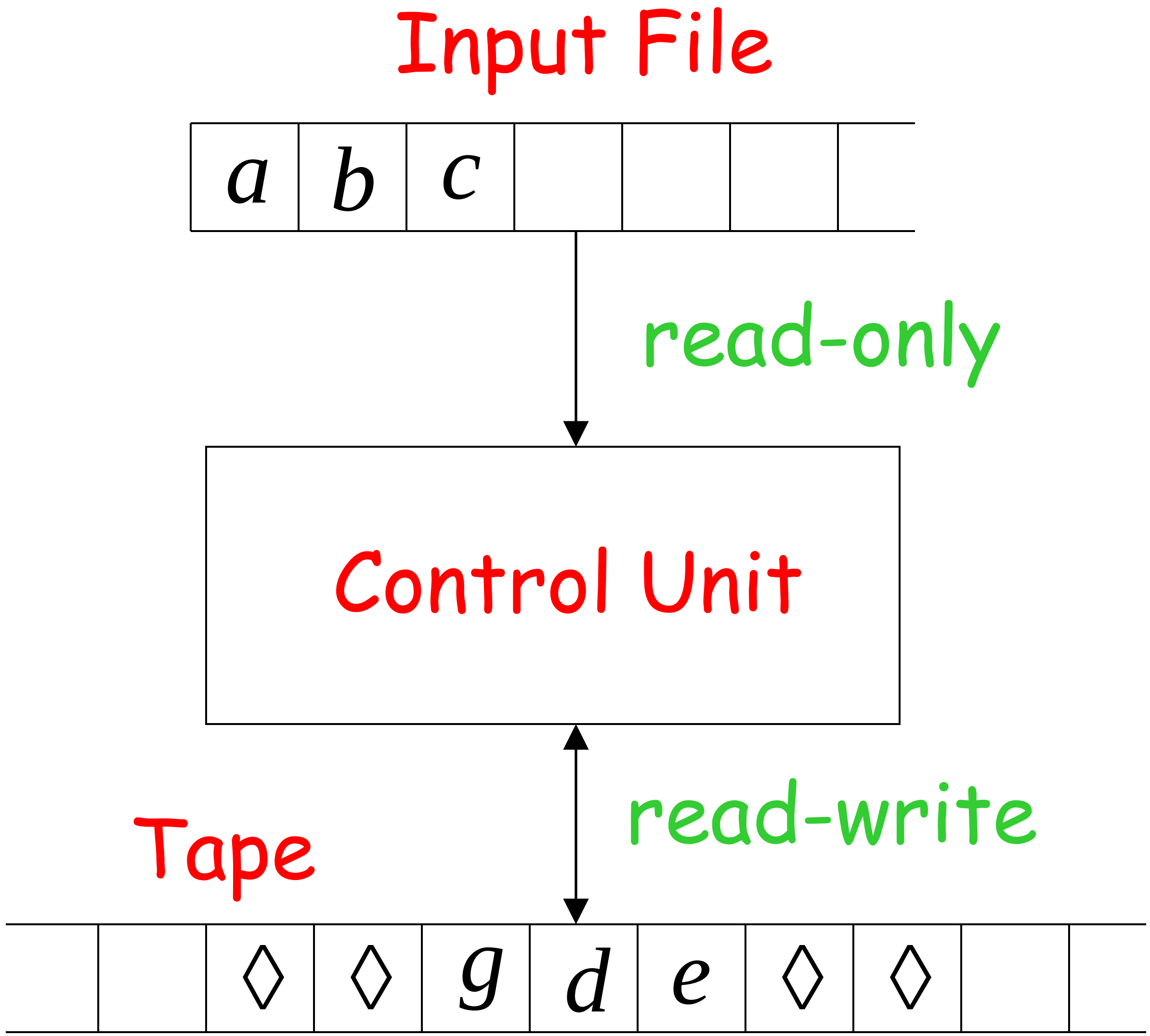
Time 2

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 as Standard Turing machines

The Off-Line Machine



Proof of equivalence?

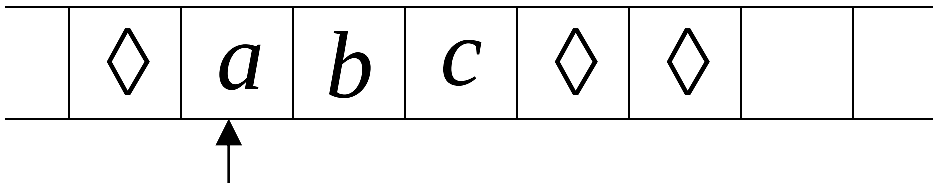
Off-line machines simulate Standard Turing Machines:

Off-line machine:

1. Copy input file to tape

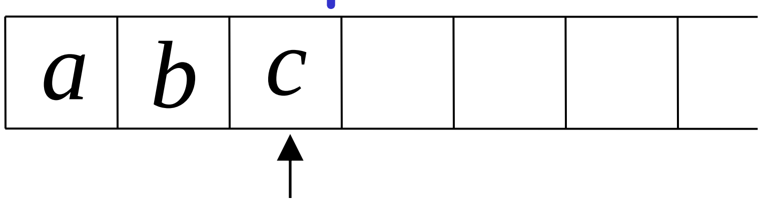
2. Continue computation as in Standard Turing machine

Standard machine

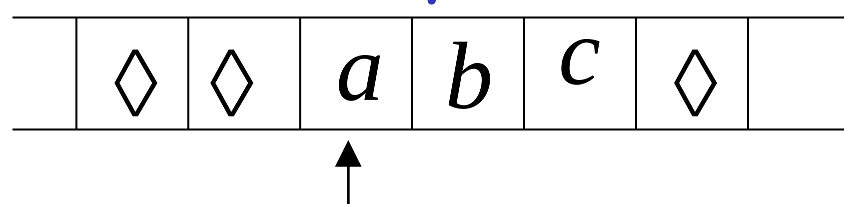


Off-line machine

Input File

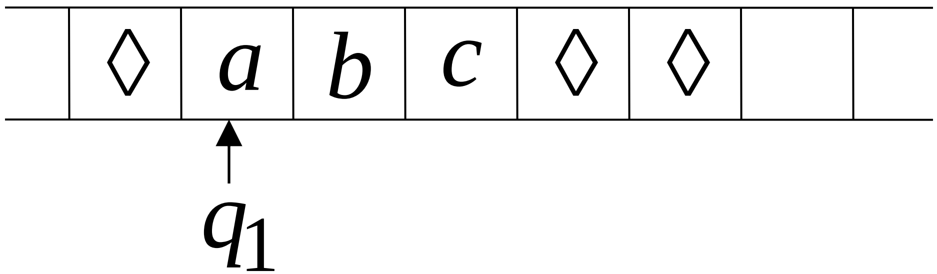


Tape



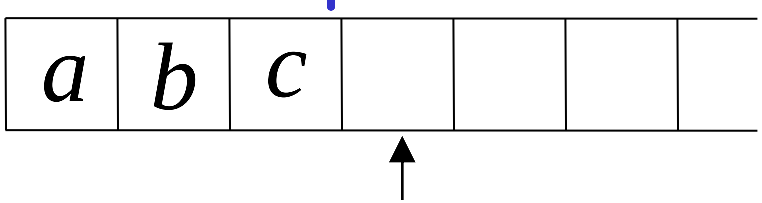
1. Copy input file to tape

Standard machine

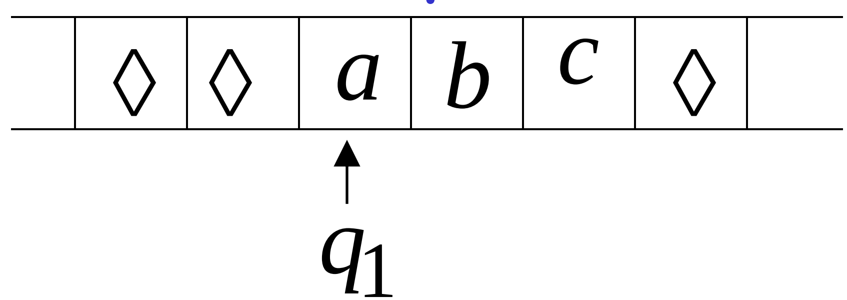


Off-line machine

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

a	b	c	d			

Tape

	◊	◊	e	f	g	◊

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

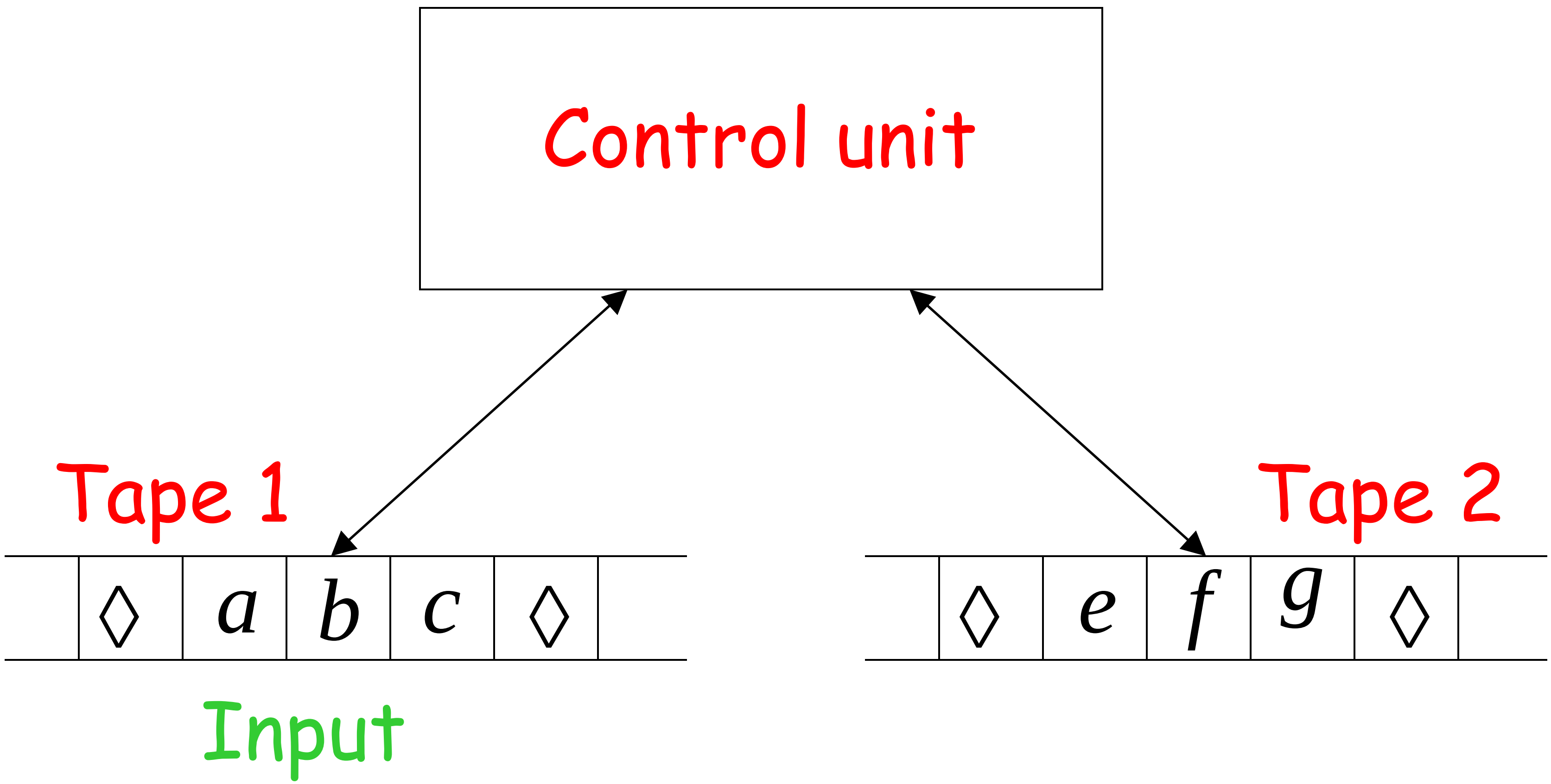
	#	a	b	c	d			Input File
	#	0	0	1	0			head position
		e	f	g				Tape
		0	1	0				head position

Repeat for each state transition:

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

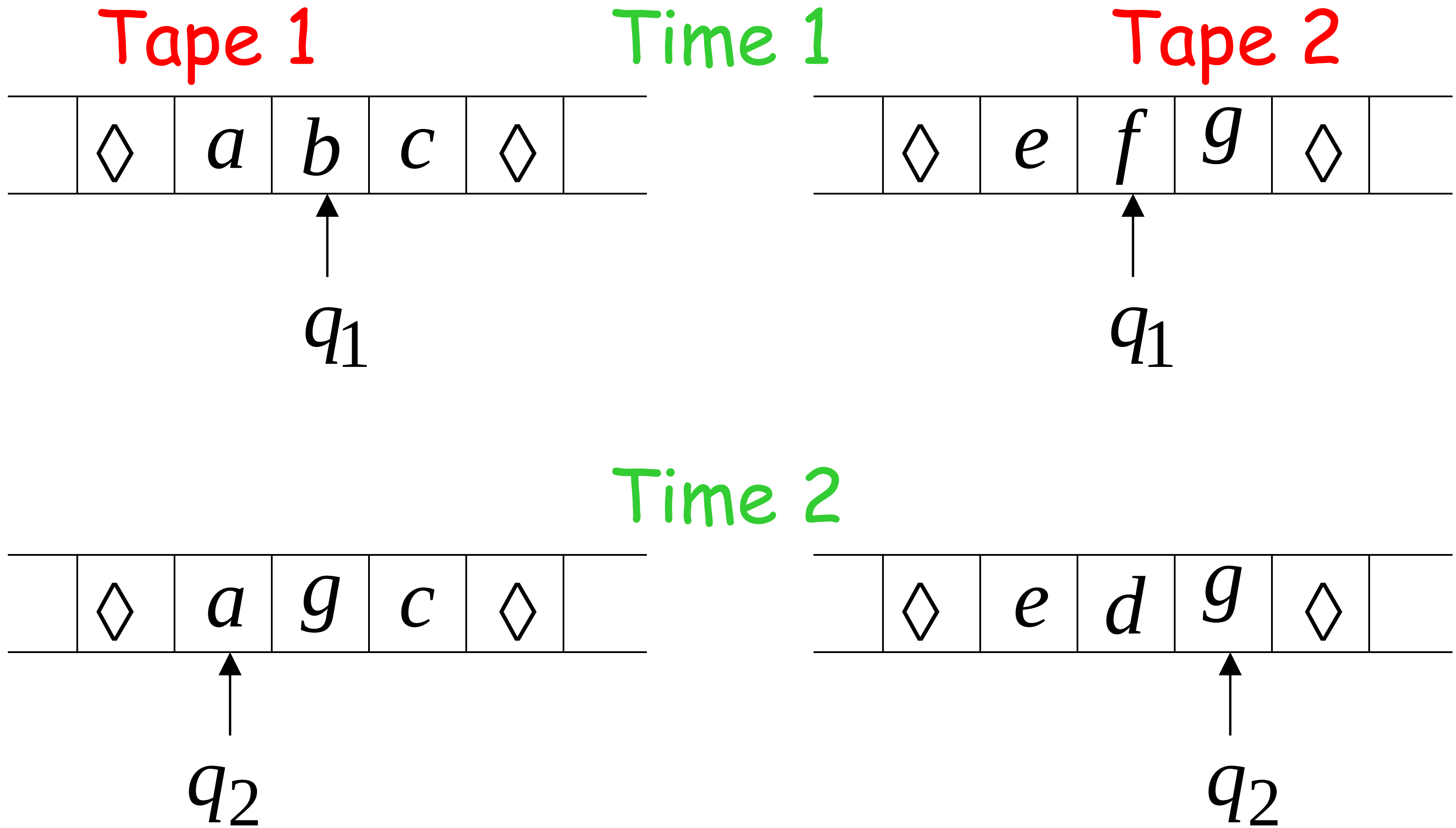
Theorem: Off-line machines have the same power as Standard machines

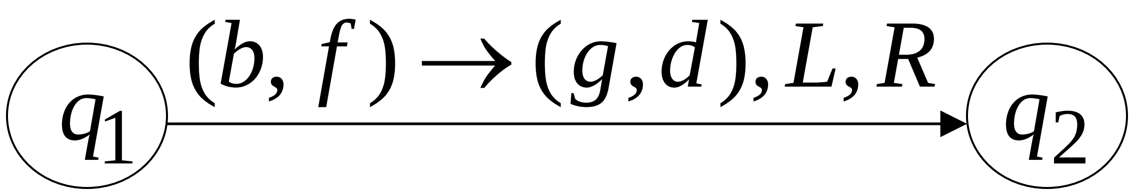
Multitape Turing Machines



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

$$\delta\,(q_0, a, e) = (q_1, x, y, L, R)$$


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


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

Proof of equivalence?

Multitape machines simulate Standard Machines:

Use just one tape

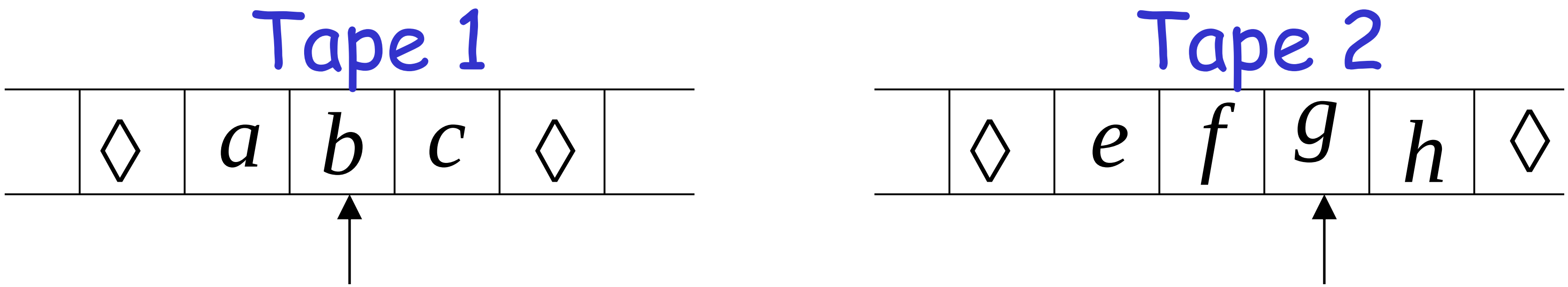
Standard machines simulate Multitape machines:

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

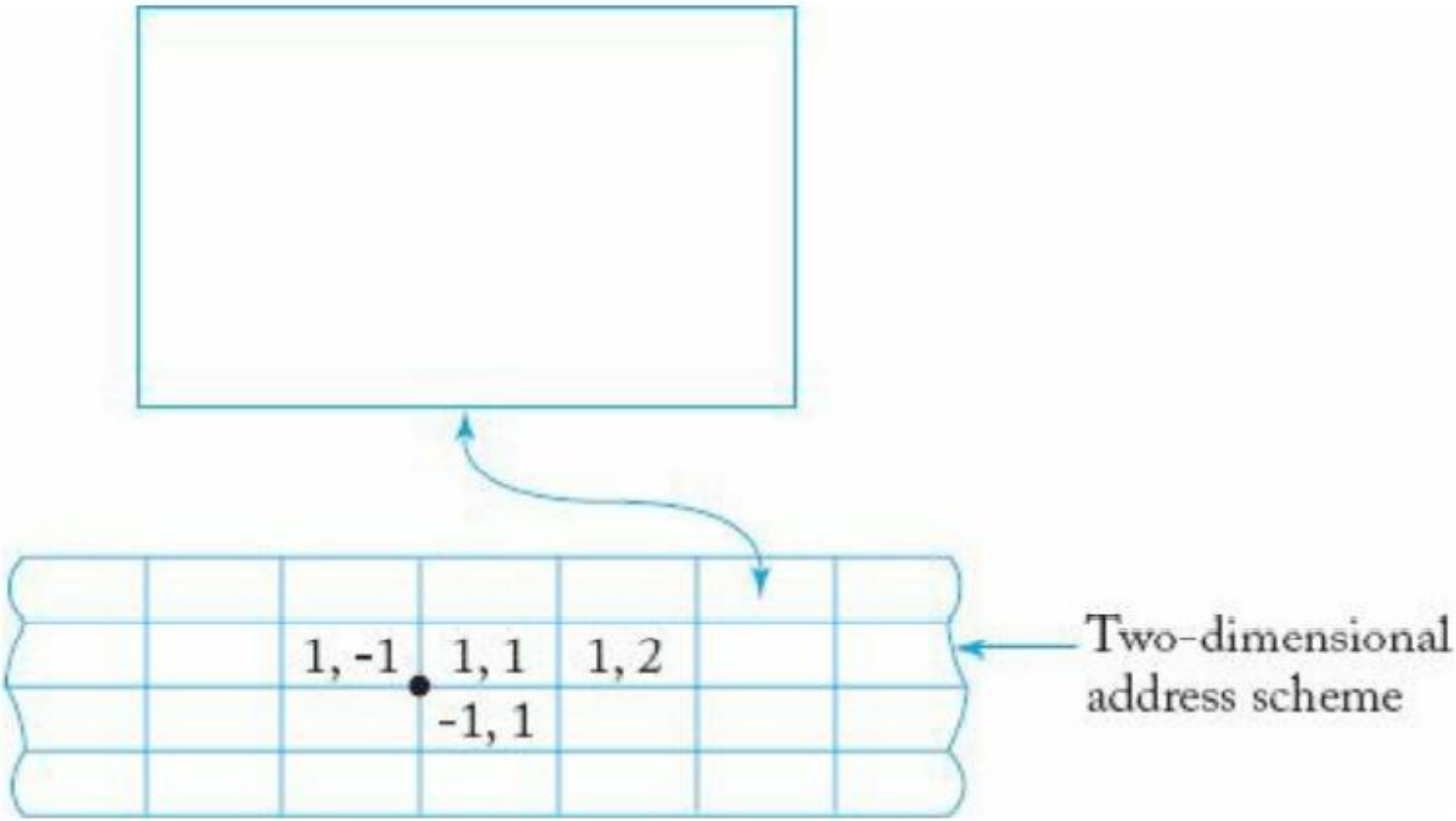
Repeat for each state transition:

- 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 as Standard Turing Machines

Multidimensional Turing Machines

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



	a				*b*					
	1	#	2	#	1	0	#	-	3	#

A limitation of Turing Machines:

Turing Machines are "hardwired"

they execute only one program

Real Computers are re-programmable

Solution: Universal Turing Machine

Attributes:

- Reprogrammable machine
- Simulates any other Turing Machine

Universal Turing Machine

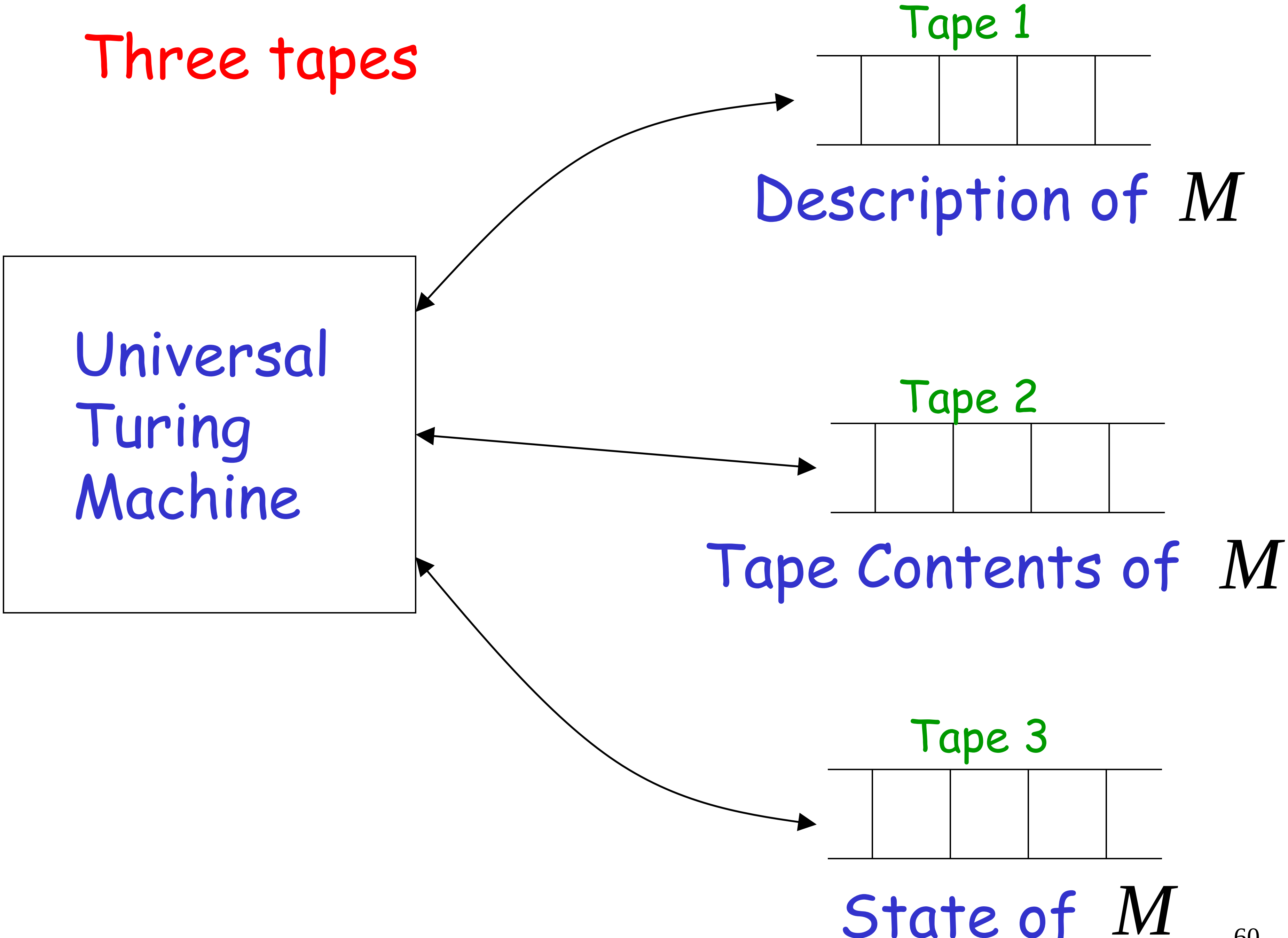
simulates any other Turing Machine M

Input of Universal Turing Machine:

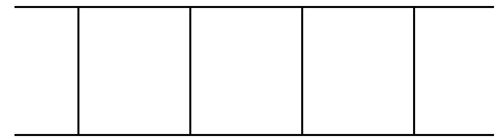
Description of transitions of M

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	$\cdots$
	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	
Encoding:	1	11	111	1111	

Head Move Encoding

Move:	L	R
	$\downarrow$	$\downarrow$
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:

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

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,

…… ……

}