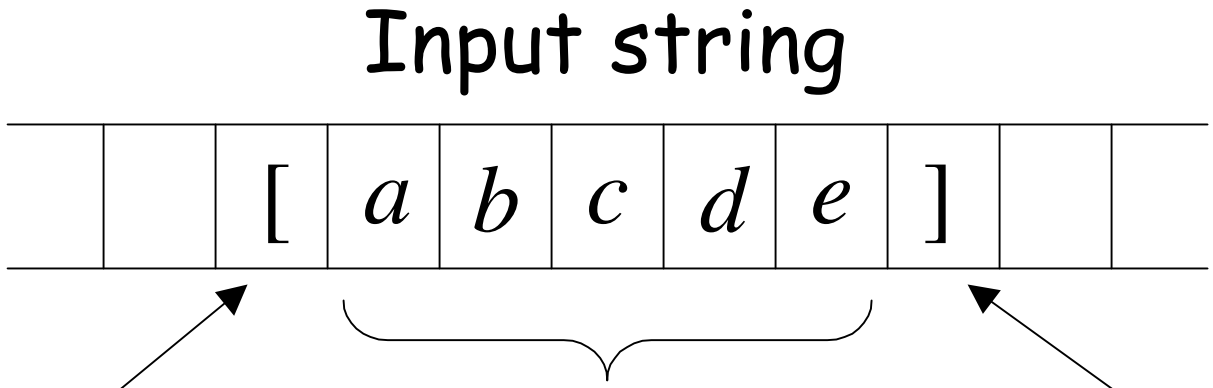
Linear Bounded Automata
LBAs

Linear Bounded Automata
are like Turing Machines

with a restriction:

The working space of the tape
is the space of the input string



Left-end marker

Working space of tape

Right-end marker

All computation is done between end markers

We define LBA's to be NonDeterministic

Open Problem:

NonDeterministic LBA's
have same power with
Deterministic LBA's ?

Example languages accepted by LBAs:

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

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

LBA's have more power than NPDA's

Later in class we will prove:

LBA's have less power
than Turing Machines

A Universal Turing Machine

Limitation of Turing Machines:

Turing Machines are "hardwired"

They execute
only one program

Real Computers are reprogrammable

Solution: Universal Turing Machine

- is a reprogrammable machine
- simulates any other Turing Machine

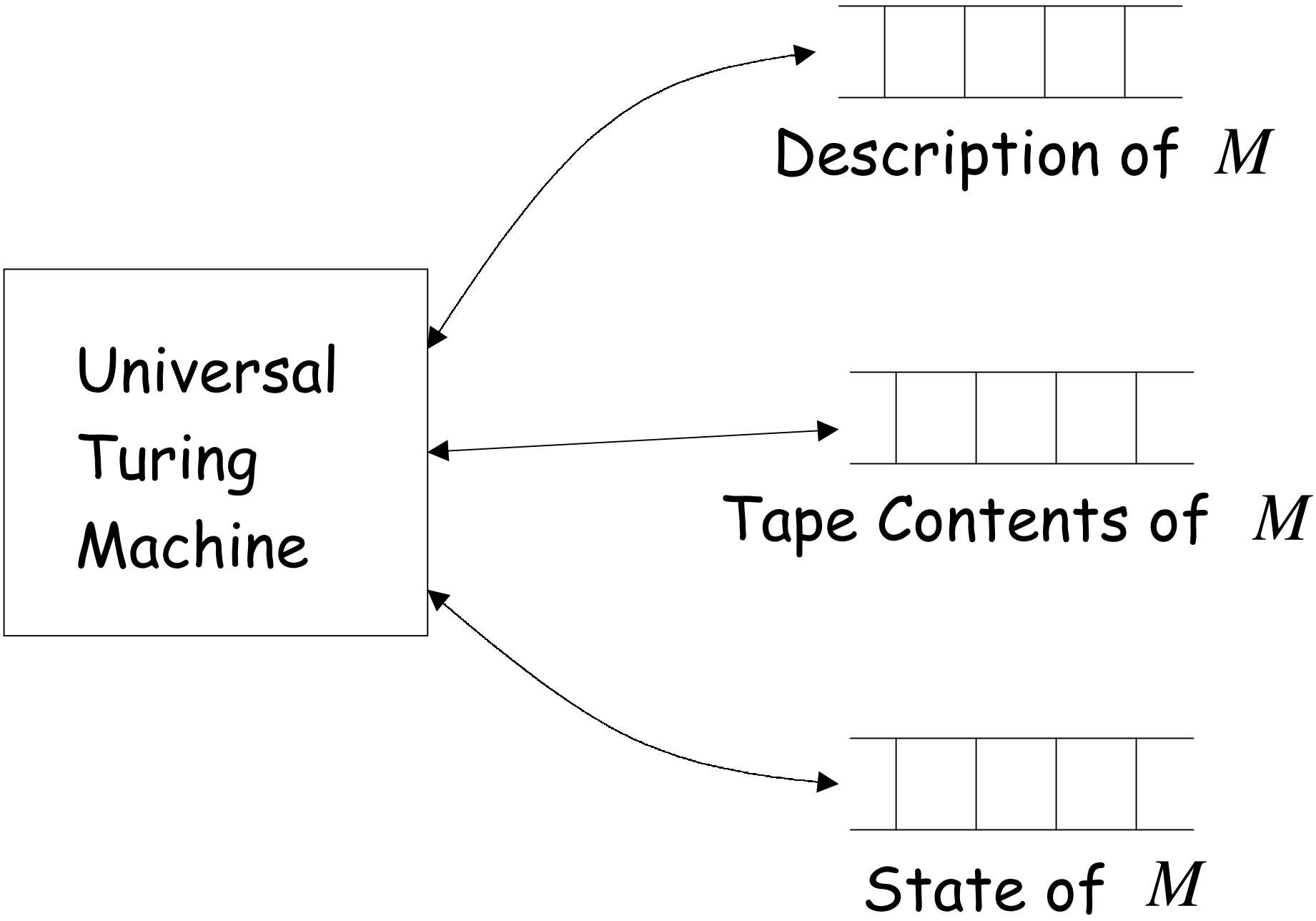
Universal Turing machine

simulates any Turing machine M

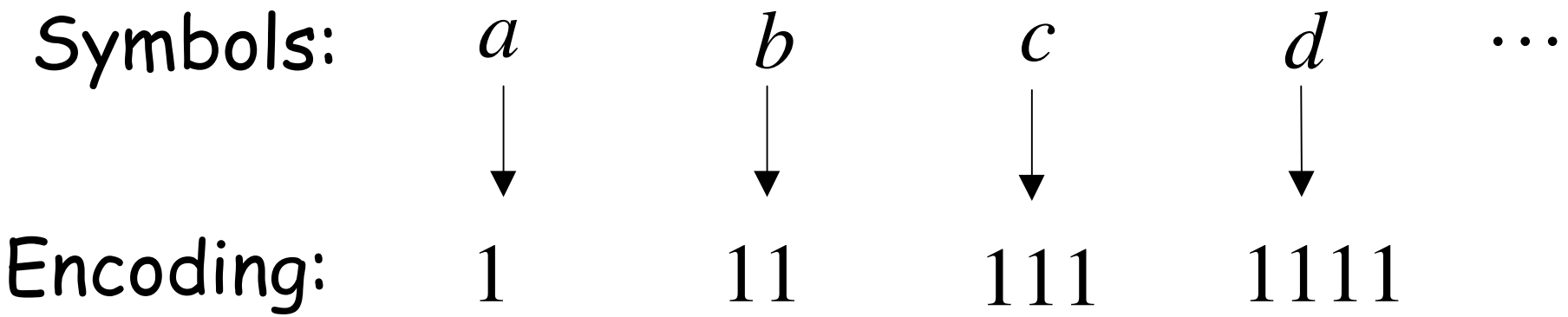
Input of Universal Machine:

Description of transitions of M

Initial tape contents of M


Universal
Turing
Machine
Description of M
Tape Contents of M
State of M

Alphabet Encoding



State Encoding

States: q_1 q_2 q_3 q_4 $\cdots$

Encoding: 1 11 111 1111

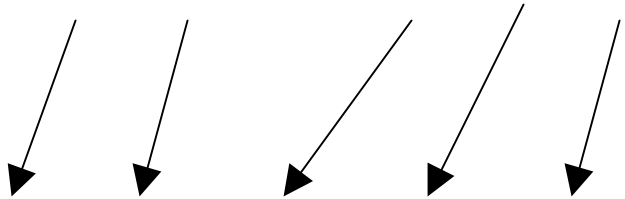
Head Move Encoding

Move: L R

Encoding: 1 11

Transition Encoding

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



Encoding: 1 0 1 0 1 1 0 1 1 0 1

separator

Machine Encoding

Transitions:

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

Encoding:

10101101101 00 1101101110111011

separator

Input of Universal Turing Machine:

encoding of the simulated machine M

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

The set of Turing machines form a language:

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

Countable Sets

Infinite sets are either:

- Countable
- Uncountable

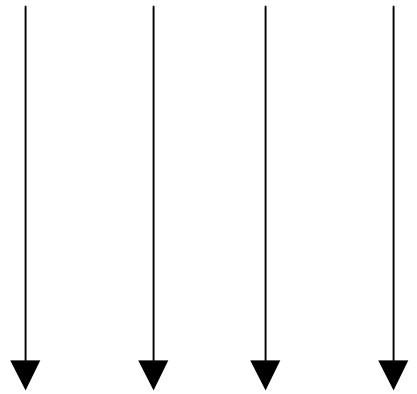
Countable set:

There is a one to one correspondence
between
elements of the set
and
positive integers

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 $n+1$

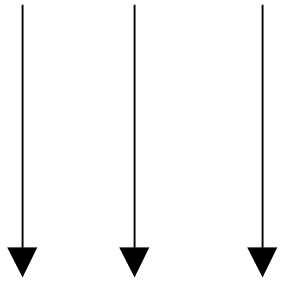
Example: The set of rational numbers is countable

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

Naive Approach

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

Correspondence:



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

Doesn't work:

we will never count numbers with nominator 2

$$\frac{2}{1}, \frac{2}{2}, \frac{2}{3}, \ldots$$

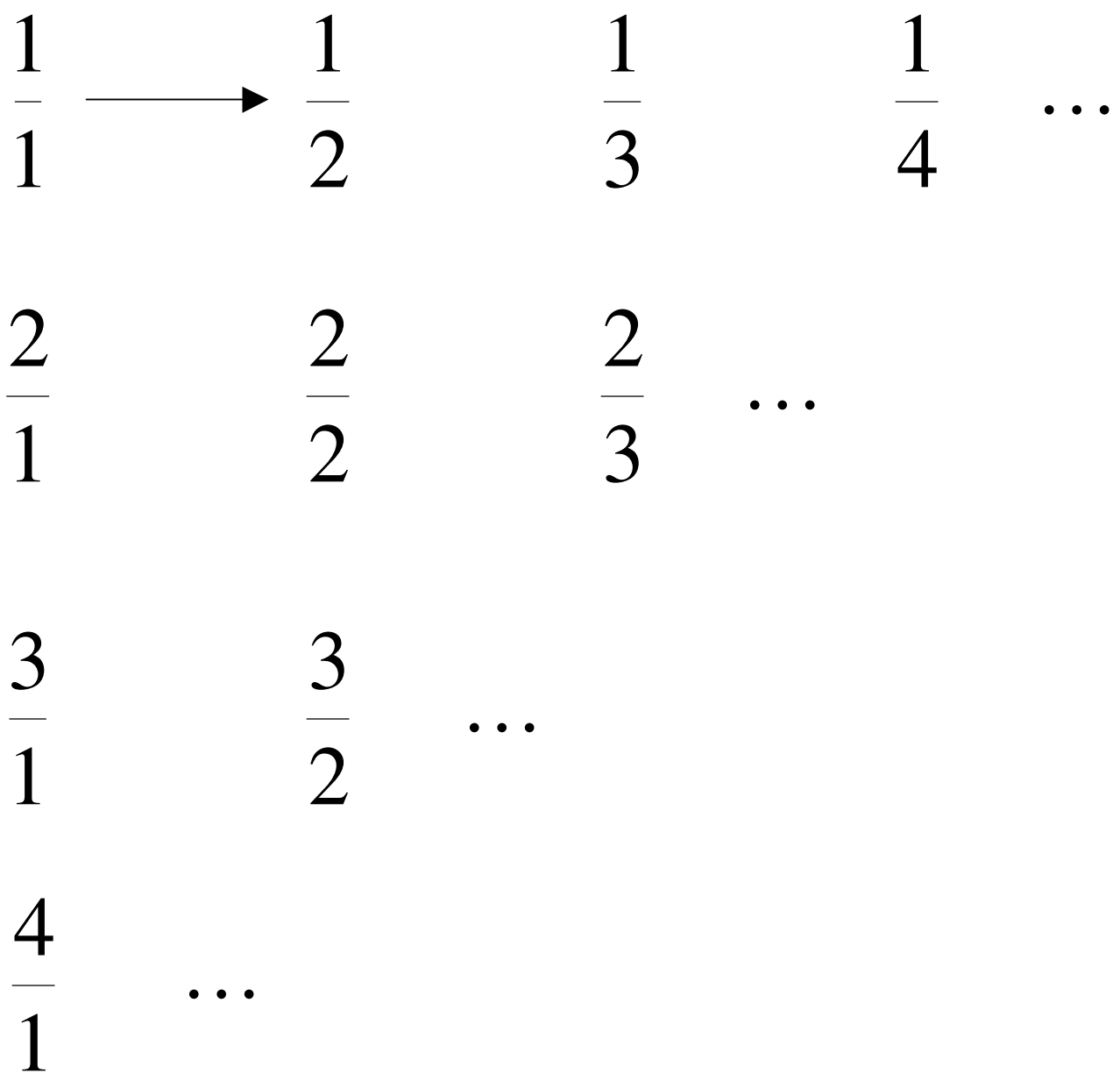
Better Approach

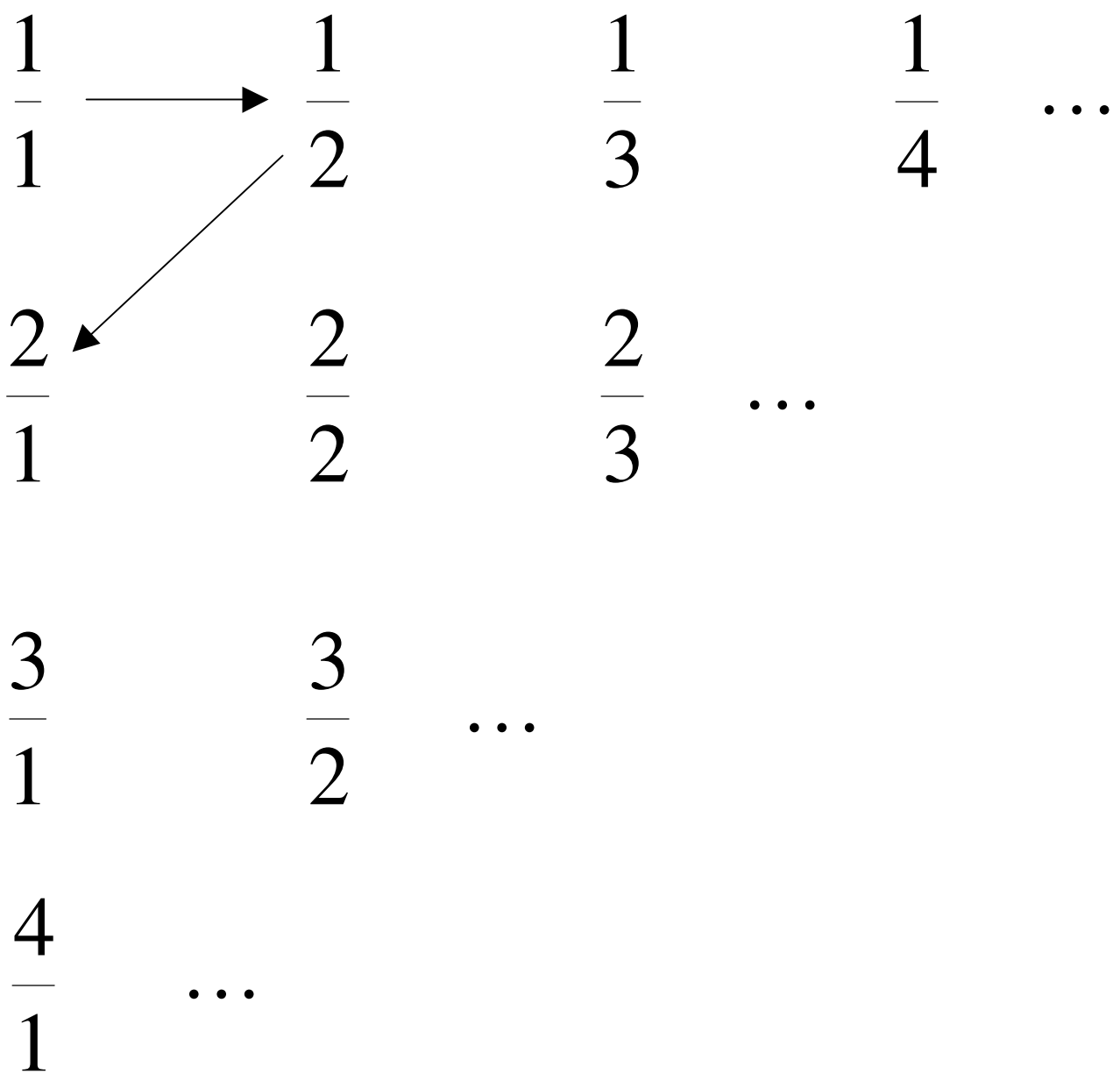
$$\frac{1}{1} \quad \frac{1}{2} \quad \frac{1}{3} \quad \frac{1}{4} \quad \cdots$$

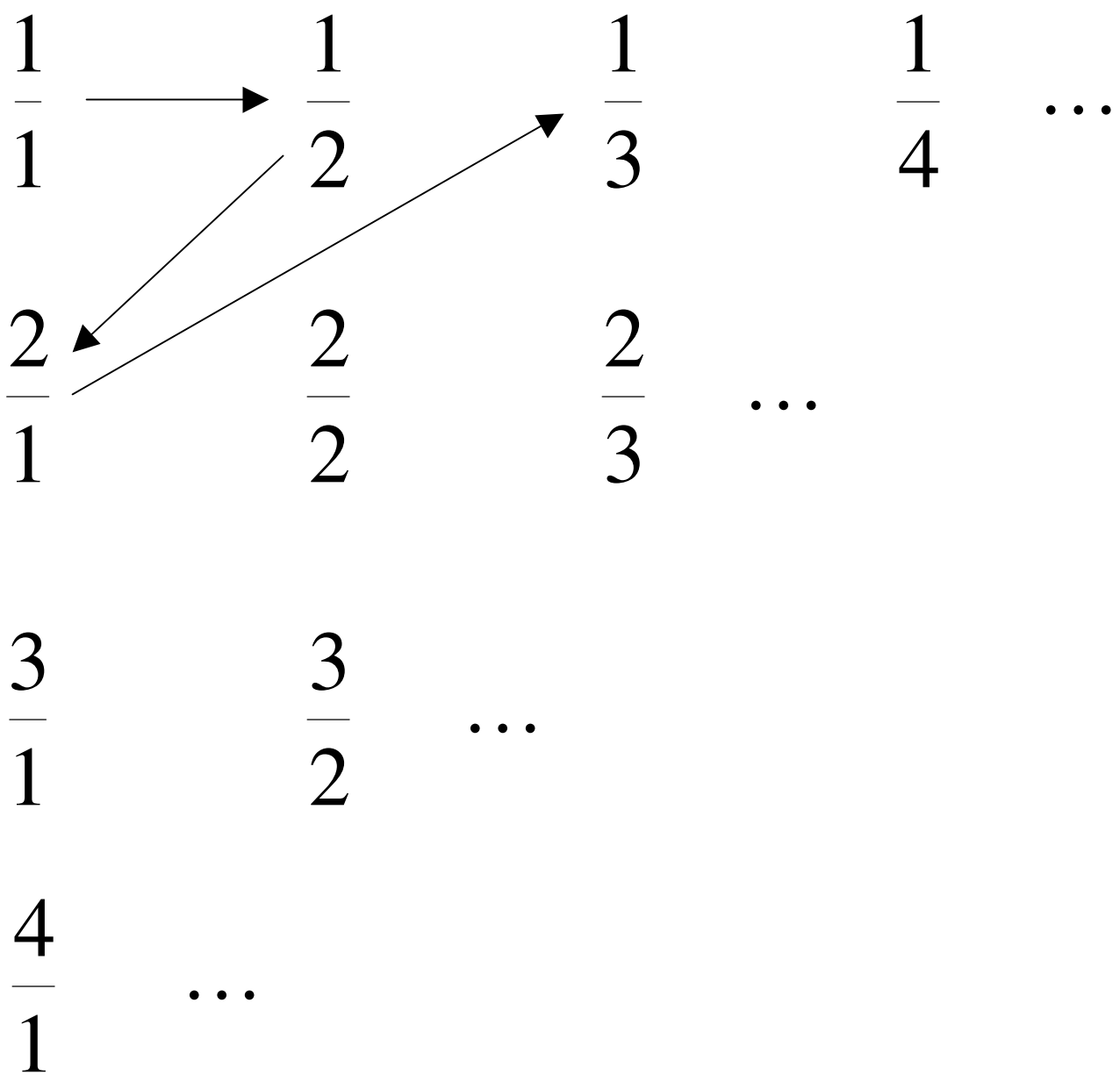
$$\frac{2}{1} \quad \frac{2}{2} \quad \frac{2}{3} \quad \cdots$$

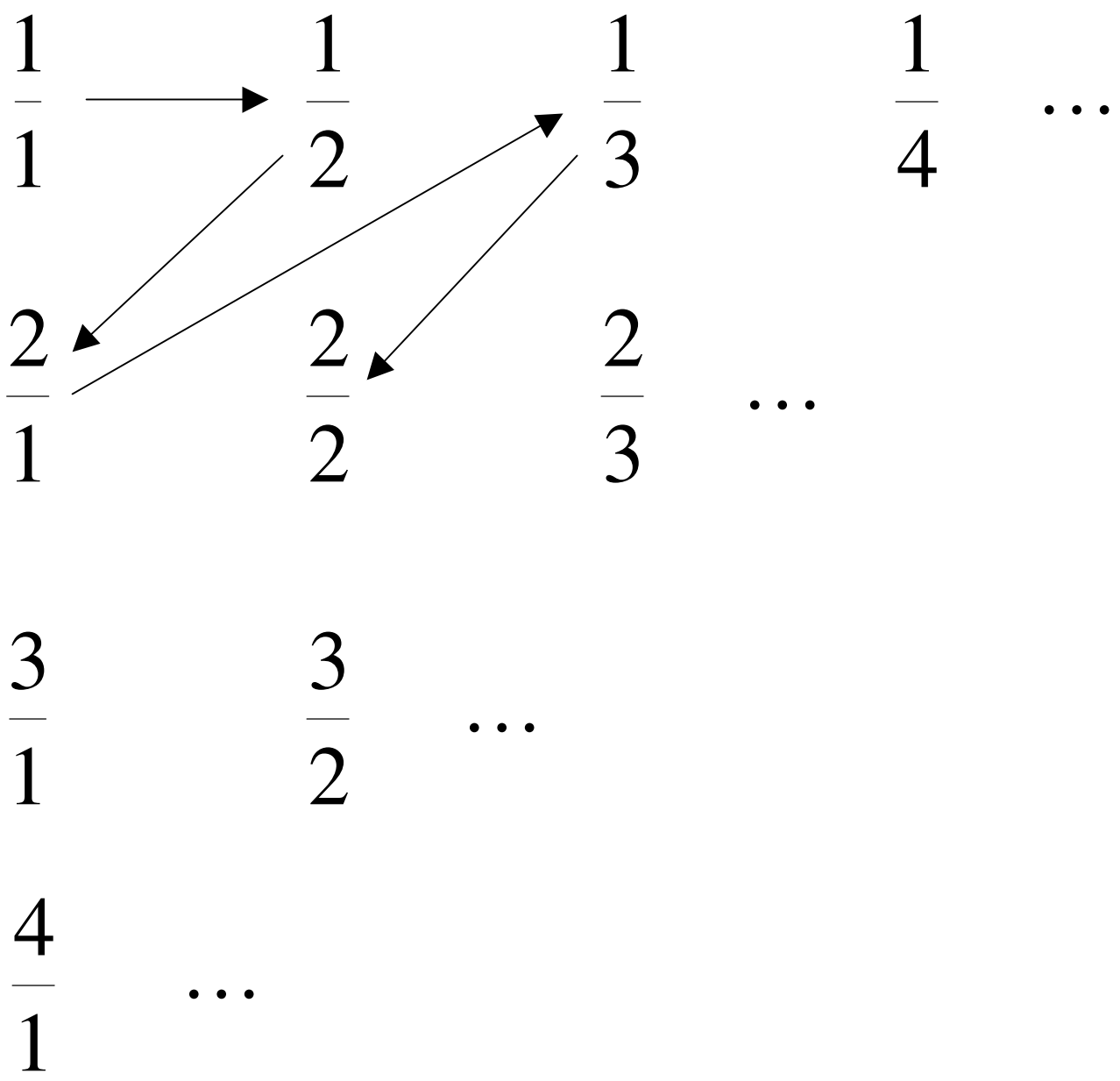
$$\frac{3}{1} \quad \frac{3}{2} \quad \cdots$$

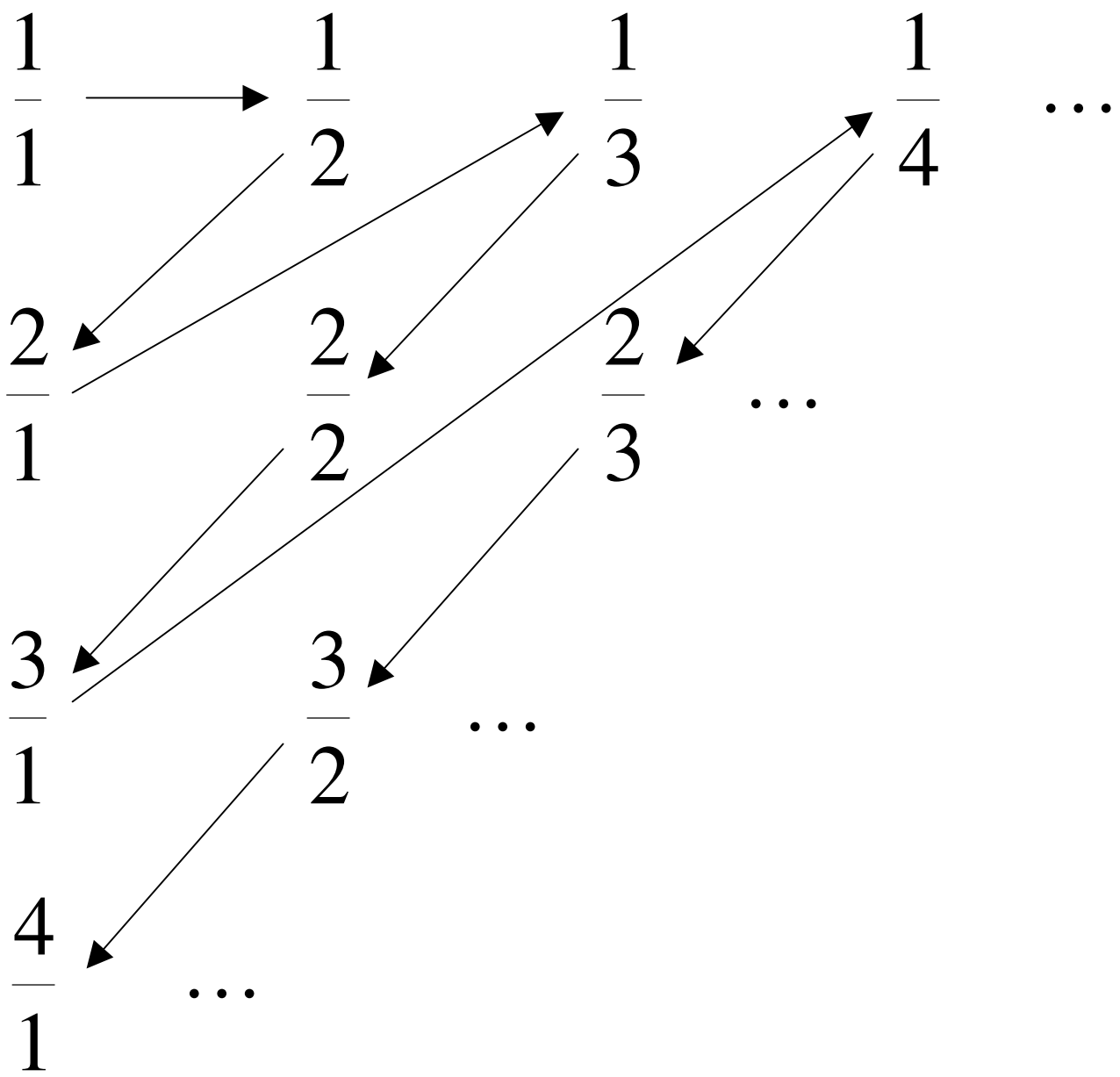
$$\frac{4}{1} \quad \cdots$$




1/1
1/2
1/3
1/4
…
2/1
2/2
2/3
…
3/1
3/2
…
4/1
…

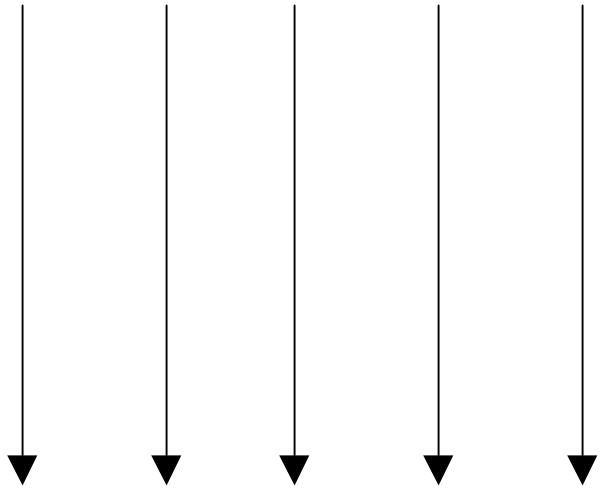

1/1
1/2
1/3
1/4
…
2/1
2/2
2/3
…
3/1
3/2
…
4/1
…


1/1
1/2
1/3
1/4
…
2/1
2/2
2/3
…
3/1
3/2
…
4/1
…


1/1
1/2
1/3
1/4
…
2/1
2/2
2/3
…
3/1
3/2
…
4/1
…

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

Correspondence:



Positive Integers: $1, \quad 2, \quad 3, \quad 4, \quad 5, \quad \ldots$

We proved:

the set of rational numbers is countable

by giving

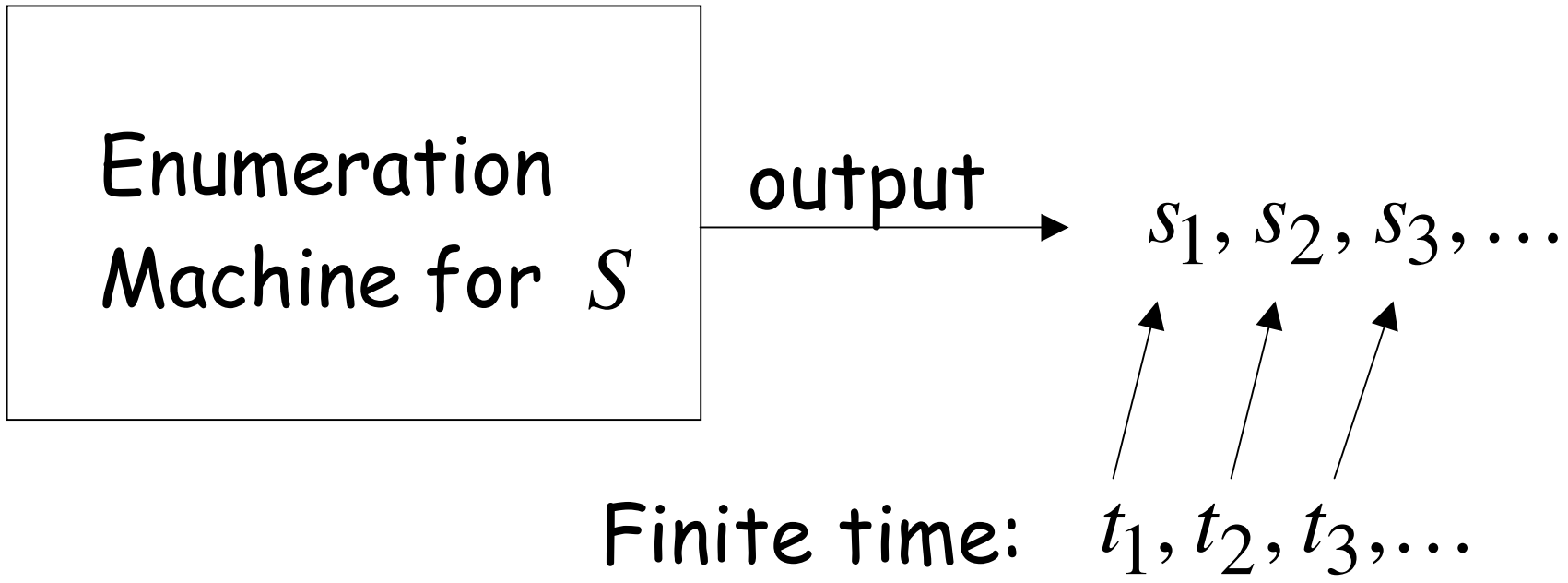
an **enumeration procedure**

Definition

Let S be a set of strings

An **enumeration procedure** for S is a Turing Machine that generates any string of S in finite number of steps

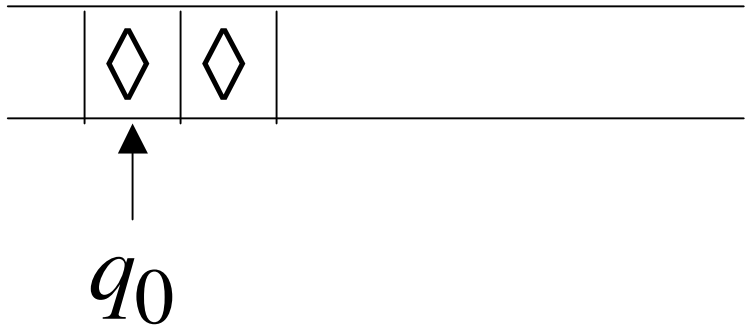
strings $s_1, s_2, s_3, \ldots \in S$



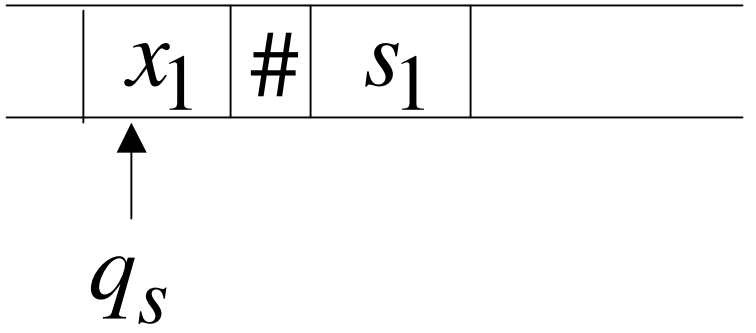
Enumeration Machine

Configuration

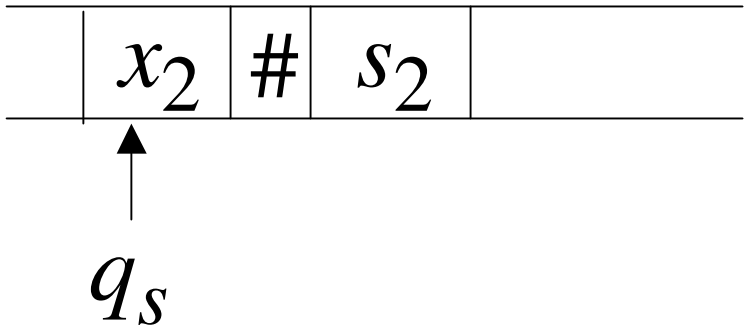
Time 0



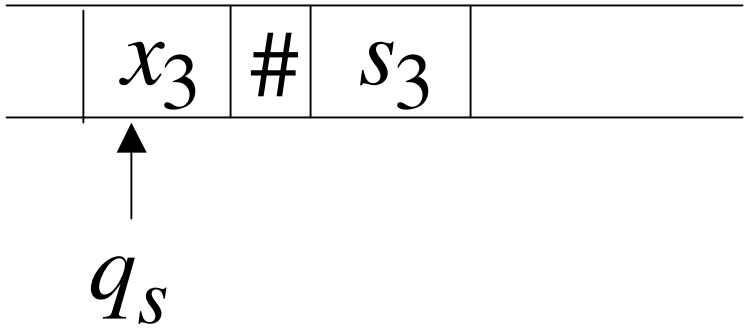
Time t_1



Time t_2



Time t_3



A set is countable if there is an enumeration procedure for it

Example:

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

We will describe the enumeration procedure

Naive procedure:

Produce the strings in lexicographic order:

a

aa

aaa

...

Doesn't work:

strings starting with b will never be produced

Better procedure: Proper Order

Produce all strings of length 1

Produce all strings of length 2

Produce all strings of length 2

.....

Produce strings: a, b, c Length 1

aa
ab
ac
ba
bb
bc
ca
cb
cc

Length 2

Proper Order

aaa
aab
aac
$\ldots\ldots$

Length 3

Theorem:

The set of all Turing Machines is countable

Theorem:

The set of all Turing Machines
is countable

Proof:

Any Turing Machine is encoded
with a 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 string of 0's and 1's in proper order

2. Check if the string defines a Turing Machine

   if YES: print string on output

   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

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

Element of S

Elements of the powerset have the form:

$$\{s_1, s_3\}$$

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

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

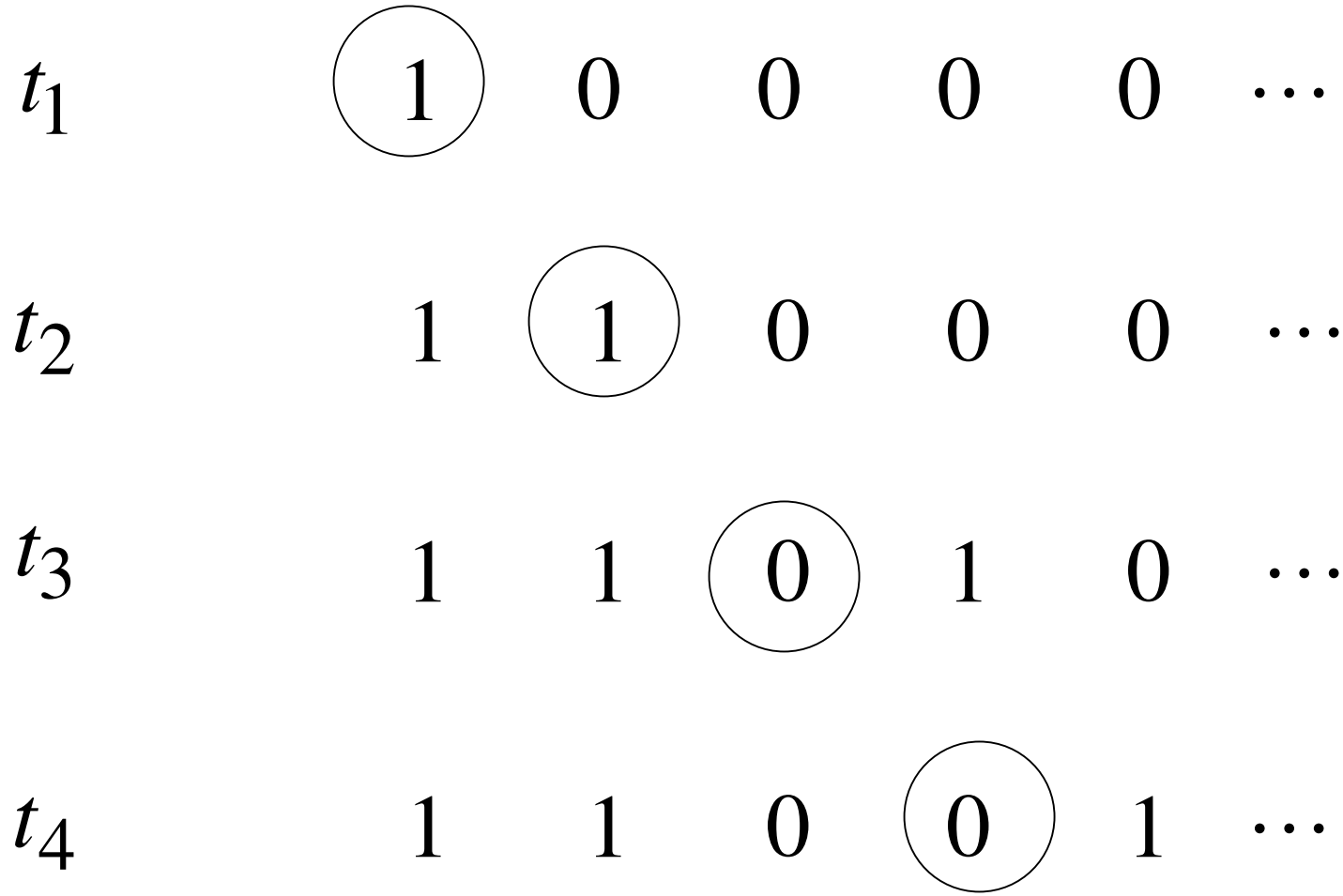
Powerset element	Encoding				
	s_1	s_2	s_3	s_4	$\cdots$
$\{s_1\}$	1	0	0	0	$\cdots$
$\{s_2, s_3\}$	0	1	1	0	$\cdots$
$\{s_1, s_3, s_4\}$	1	0	1	1	$\cdots$

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

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
if the diagonal



New element: 0011…

(Diagonal complement)

The new element must be some t_i

This is impossible:

The i-th bit must be the complement of itself

We have contradiction!

Therefore the powerset is uncountable