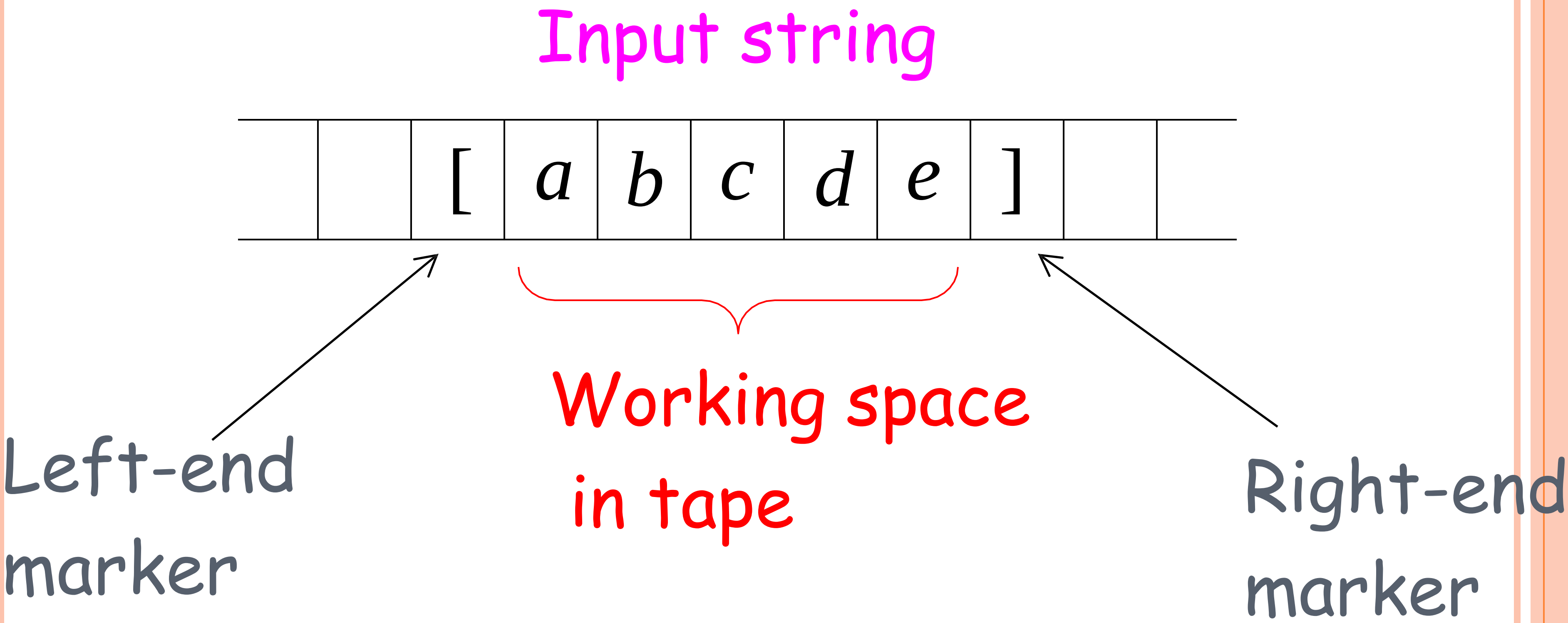
Linear Bounded Automata LBAs

Linear Bounded Automata (LBAs)
are the same as Turing Machines
with one difference:

The input string tape space
is the only tape space allowed to use

Linear Bounded Automaton (LBA)



All computation is done between end markers

We define LBA's as NonDeterministic

Open Problem:

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

Example languages accepted by LBAs:

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

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

Conclusion:

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

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

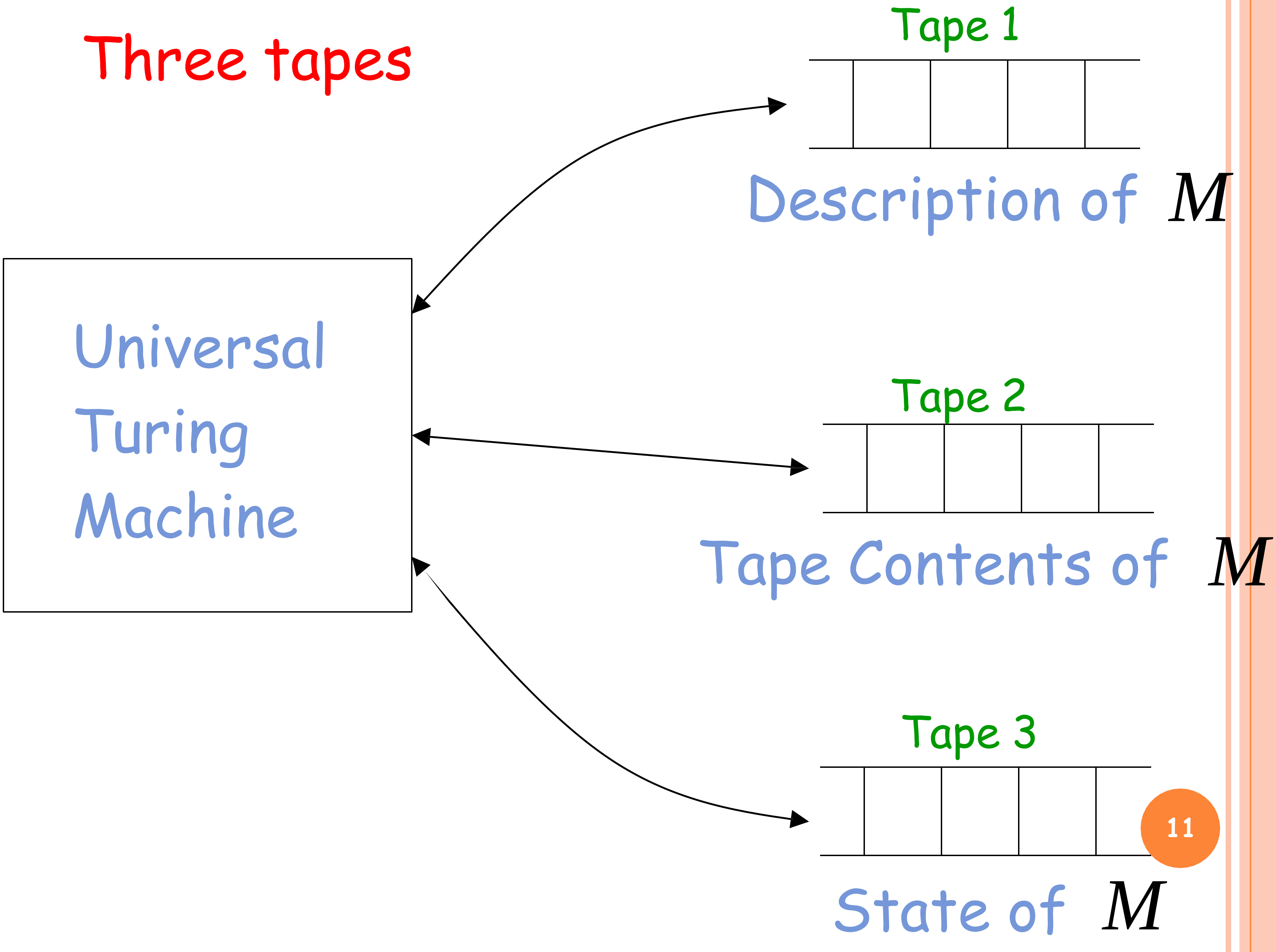
simulates any other Turing Machine M

Input of Universal Turing Machine:

Description of transitions of M

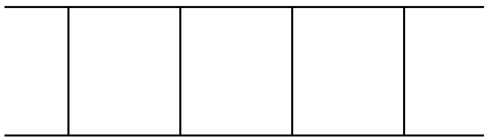
Initial tape contents of M

Three tapes



Tape 1
Description of M
Universal Turing Machine
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
	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$
Encoding:	1	11	111	1111

State Encoding

States:	q_1	q_2	q_3	q_4
	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$
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:

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

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)

001001001011111, (Turing Machine 2)

1110100111100010101, ……

…… }

Countable Sets

Infinite sets are either:

Countable

or

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:

$$
\begin{array}{ccccc}
0, & 2, & 4, & 6, & \ldots \\
\downarrow & \downarrow & \downarrow & \downarrow & \\
1, & 2, & 3, & 4, & \ldots
\end{array}
$$

Positive integers: $1, \ 2, \ 3, \ 4, \ \ldots$

$2n$ corresponds to $n+1$

Example: The set of rational numbers is countable

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

Naïve Proof

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

Correspondence: $\downarrow \quad \downarrow \quad \downarrow$

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

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

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

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

$$\frac{4}{1}$$

$$\frac{1}{1} \longrightarrow \frac{1}{2} \qquad \frac{1}{3} \qquad \frac{1}{4}$$

$$\frac{2}{1} \qquad \frac{2}{2} \qquad \frac{2}{3}$$

$$\frac{3}{1} \qquad \frac{3}{2}$$

$$\frac{4}{1}$$

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

$$
\begin{array}{cccc}
\frac{1}{1} & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} \\[2em]
\frac{2}{1} & \frac{2}{2} & \frac{2}{3} & \\[2em]
\frac{3}{1} & \frac{3}{2} & & \\[2em]
\frac{4}{1} & & &
\end{array}
$$

$$\begin{array}{cccc} \dfrac{1}{1} & \dfrac{1}{2} & \dfrac{1}{3} & \dfrac{1}{4} \\ \\ \dfrac{2}{1} & \dfrac{2}{2} & \dfrac{2}{3} & \\ \\ \dfrac{3}{1} & \dfrac{3}{2} & & \\ \\ \dfrac{4}{1} & & & \end{array}$$

$$
\begin{array}{cccc}
\frac{1}{1} & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} \\
\frac{2}{1} & \frac{2}{2} & \frac{2}{3} & \\
\frac{3}{1} & \frac{3}{2} & & \\
\frac{4}{1} & & &
\end{array}
$$

$$\begin{array}{lcccccc}
\text{Rational Numbers:} & \frac{1}{1}, & \frac{1}{2}, & \frac{2}{1}, & \frac{1}{3}, & \frac{2}{2}, & \cdots \\
\text{Correspondence:} & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \\
\text{Positive Integers:} & 1, & 2, & 3, & 4, & 5, & \ldots
\end{array}$$

We proved:

the set of rational numbers is countable
by describing an enumeration procedure

Definition

Let S be a set of strings

An **enumeration procedure** for S is a Turing Machine that generates all strings of S one by one

and

Each string is generated in finite time

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

Enumeration Machine for S

output

(on tape)

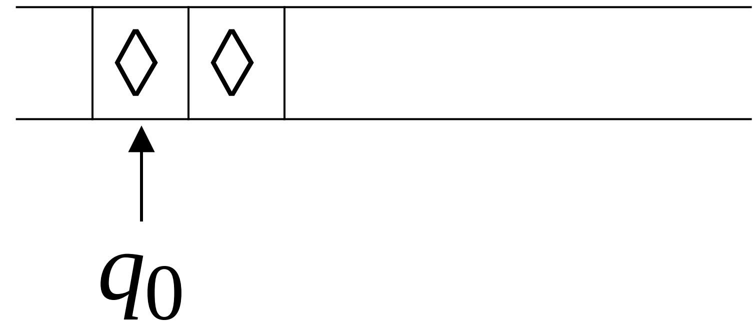
$s_1, s_2, s_3, \ldots$

Finite time: $t_1, t_2, t_3, \ldots$

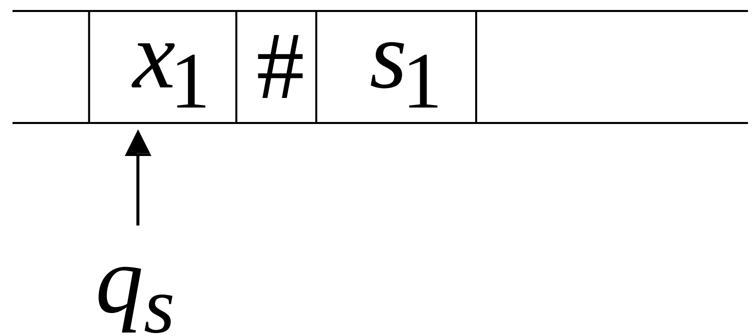
Enumeration Machine

Configuration

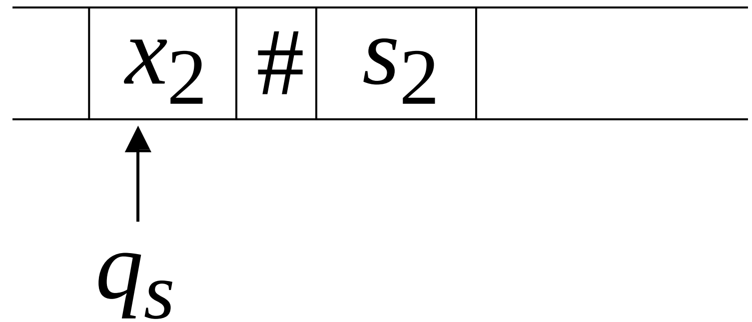
Time 0



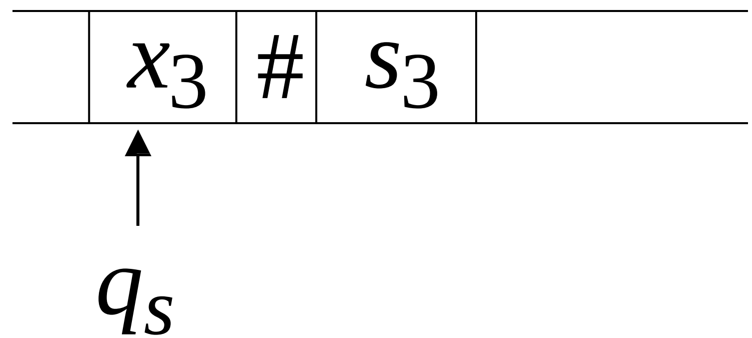
Time t_1



Time t_2



Time t_3



Observation:

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

Example:

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

Proof:

We will describe the enumeration procedure

Naive procedure:

Produce the strings in lexicographic order:

a

aa

aaa

$aaaa$

Doesn't work:

strings starting with b
will never be produced

Better procedure: **Proper Order**

1. Produce all strings of length 1
2. Produce all strings of length 2
3. Produce all strings of length 3
4. Produce all strings of length 4

....

Produce strings in **Proper Order:**

a
b
c } length 1

$a\,a$
$a\,b$
$a\,c$
$b\,a$
$b\,b$
$b\,c$
$c\,a$
$c\,b$
$c\,c$ } length 2

aaa
aab
aac } length 3

Theorem: The set of all Turing Machines is countable

Proof: Any Turing Machine can be encoded with a binary 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 binary string of 0's and 1's in proper order

2. Check if the string describes a Turing Machine

   if YES: print string on output tape

   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

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

Elements 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 binary string of 0's and 1's

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

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

Then: we can enumerate the elements of the powerset

Powerset element	Encoding				
t_1	1	0	0	0	0
t_2	1	1	0	0	0
t_3	1	1	0	1	0
t_4	1	1	0	0	1

Take the powerset element whose bits are the complements in the diagonal

t_1	1	0	0	0	0
t_2	1	1	0	0	0
t_3	1	1	0	1	0
t_4	1	1	0	0	1

New element: 0011…

(birary complement of diagonal)

The new element must be some t_i of the powerset

However, that's impossible:

from definition of t_i

the i-th bit of t_i must be the complement of itself

Contradiction!!!

Since we have a contradiction:

The powerset 2^S of S is uncountable

An Application: Languages

Example Alphabet : $\{a, b\}$

The set of all Strings:

$$S = \{a, b\}^* = \{\lambda, a, b, aa, ab, ba, bb, aaa, aab, \ldots\}$$

infinite and countable

Example Alphabet : $\{a,b\}$

The set of all Strings:

$$S = \{a,b\}^* = \{\lambda, a, b, aa, ab, ba, bb, aaa, aab, \ldots\}$$

infinite and countable

A language is a subset of S :

$$L = \{aa, ab, aab\}$$

Example Alphabet : $\{a, b\}$

The set of all Strings:

$$S = \{a, b\}^* = \{\lambda, a, b, aa, ab, ba, bb, aaa, aab, \ldots\}$$

infinite and countable

The powerset of S contains all languages:

$$2^S = \{\{\lambda\}, \{a\}, \{a, b\}\{aa, ab, aab\}, \ldots\}$$

$L_1 \quad L_2 \quad L_3 \quad L_4$

uncountable

Languages: uncountable

$$\begin{array}{cccc} L_1 & L_2 & L_3 & L_k \\ \downarrow & \downarrow & \downarrow & \downarrow \\ M_1 & M_2 & M_3 & ? \end{array}$$

Turing machines: countable

There are infinitely many more languages than Turing Machines

Conclusion:

There are some languages not accepted by Turing Machines

These languages cannot be described by algorithms