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)

Input string

\[
\begin{array}{c|c|c|c|c|c|c}
 & a & b & c & d & e \\
\hline
\end{array}
\]

Left-end marker

Working space in tape

Right-end marker

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

Universal Turing Machine

Description of $M$

Tape 1

Tape 2

Tape 3

State of $M$
We describe Turing machine $M$ as a string of symbols:

We encode $M$ as a string of symbols.
Alphabet Encoding

Symbols: $a$, $b$, $c$, $d$, ...

Encoding: $1$, $11$, $111$, $1111$, ...
### State Encoding

<table>
<thead>
<tr>
<th>States</th>
<th>$q_1$</th>
<th>$q_2$</th>
<th>$q_3$</th>
<th>$q_4$</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encoding</td>
<td>1</td>
<td>11</td>
<td>111</td>
<td>1111</td>
<td>1111</td>
</tr>
</tbody>
</table>

### Head Move Encoding

<table>
<thead>
<tr>
<th>Move</th>
<th>$L$</th>
<th>$R$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encoding</td>
<td>1</td>
<td>11</td>
</tr>
</tbody>
</table>
Transition Encoding

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

Encoding: 10101101101

separator
Machine Encoding

Transitions:

\[ \delta(q_1, a) = (q_2, b, L) \quad \delta(q_2, b) = (q_3, c, R) \]

Encoding:

10101101101 00 110110111101111011
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

\[ L = \{ \text{010100101}, \quad \text{(Turing Machine 1)} \] 

\[ \text{00100100101111}, \quad \text{(Turing Machine 2)} \] 

\[ 111010011110010101, \quad \ldots \]

\[ \ldots \} \]
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:

Positive integers: \[1, 2, 3, 4, \ldots\]

\[2n \text{ corresponds to } n + 1\]
Example: The set of rational numbers is countable

Rational numbers: \[ \frac{1}{2}, \frac{3}{4}, \frac{7}{8}, \ldots \]
Naïve Proof

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

\[
\begin{array}{cccc}
1 & 1 & 1 & 1 \\
\hline
1 & 2 & 3 & 4 \\
\hline
1 & 2 & 3 & \ldots \\
\hline
2 & 2 & 2 & \ldots \\
\hline
1 & 2 & 3 & \ldots \\
\hline
3 & 3 & \ldots \\
\hline
1 & 2 & \ldots \\
\hline
4 & \ldots \\
\hline
1 & \ldots \\
\end{array}
\]
\[
\begin{array}{ccc}
1 & 1 & 1 & 1 & \ldots \\
1 & 2 & 3 & 4 & \ldots \\
2 & 2 & 2 & \ldots \\
1 & 2 & 3 & \ldots \\
3 & 3 & \ldots \\
1 & 2 & \ldots \\
4 & \ldots \\
1 & \ldots \\
\end{array}
\]
\[
\begin{array}{cccccc}
1 & 1 & 1 & 1 & \ldots \\
1 & 2 & 3 & 4 & \ldots \\
2 & 2 & 2 & \ldots \\
1 & 1 & 2 & \ldots \\
3 & 3 & \ldots \\
1 & 2 & \ldots \\
4 & \ldots \\
1 & \ldots \\
\end{array}
\]
Rational Numbers:

\[
\begin{array}{cccccc}
1 & 1 & 2 & 1 & 2 \\
1' & 2' & 1' & 3' & 2' \\
\end{array}
\]

Correspondence:

Positive Integers:

1, 2, 3, 4, 5, ...
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** \( S \) \hspace{2cm} \text{output} \hspace{1cm} \text{(on tape)} \hspace{2cm} \text{Finite time:} \hspace{1cm} t_1, t_2, t_3, \ldots

\[ s_1, s_2, s_3, \ldots \in S \]
Enumeration Machine

Configuration

Time 0

Time $t_1$
Time $t_2$

<table>
<thead>
<tr>
<th>$x_2$</th>
<th>#</th>
<th>$s_2$</th>
</tr>
</thead>
</table>

$q_s$

Time $t_3$

<table>
<thead>
<tr>
<th>$x_3$</th>
<th>#</th>
<th>$s_3$</th>
</tr>
</thead>
</table>

$q_s$
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:

\[ a \]
\[ aa \]
\[ aaa \]
\[ aaaa \]
\[ \ldots \]

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:

- **Length 1**
  - $a$
  - $b$
  - $c$
  - $ab$
  - $ac$
  - $bc$

- **Length 2**
  - $aa$
  - $ab$
  - $ac$
  - $ba$
  - $ca$
  - $bb$
  - $bc$
  - $cb$

- **Length 3**
  - $aaa$
  - $aab$
  - $aac$
  - ...
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.

<table>
<thead>
<tr>
<th>Powerset element</th>
<th>Encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>{s_1}</td>
<td>1 0 0 0 0 \ldots</td>
</tr>
<tr>
<td>{s_2, s_3}</td>
<td>0 1 1 0 \ldots</td>
</tr>
<tr>
<td>{s_1, s_3, s_4}</td>
<td>1 0 1 1 \ldots</td>
</tr>
</tbody>
</table>
Let’s assume (for contradiction) that the powerset is countable.

Then: we can enumerate the elements of the powerset
<table>
<thead>
<tr>
<th>Powerset element</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>…</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t_1 )</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>( t_2 )</td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>( t_3 )</td>
<td></td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( t_4 )</td>
<td></td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>…</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The encoding is represented by the binary sequences corresponding to each power set element.
Take the powerset element whose bits are the complements in the diagonal
<table>
<thead>
<tr>
<th>$t_1$</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_2$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>...</td>
</tr>
<tr>
<td>$t_3$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>...</td>
</tr>
<tr>
<td>$t_4$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>...</td>
</tr>
</tbody>
</table>

New element: 0011...

(binary 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:

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

\( L_1, L_2, L_3, L_4, \ldots \)

uncountable
Languages: uncountable

$L_1$ $L_2$ $L_3$ $\cdots$ $L_k$ $\cdots$

$M_1$ $M_2$ $M_3$ $?$

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