1 Variants of Turing machines

INFINITE TAPE IN BOTH DIRECTIONS.

The same definition of the machine as before except for the notion of the leftmost square, which is absent. Sometimes, the squares are numbered by integers $0, \pm 1, \pm 2, \pm 3, \ldots$; the input $w$ is positioned in the squares numbered $0, 1, 2, 3, \ldots, |w| - 1$. 

![Diagram of Turing machine with control and tape]
Theorem 1 For every 2-way infinite tape TM $M$, there is a standard TM $M'$ such that $L(M) = L(M')$.

• $uv$ is the current tape contents;

• $u$ is the part (possible empty) of the string from the leftmost symbol till (and including) the scanned square;

• if $a$ is the symbol in the scanned square, then $v$ is the part (possible empty) of the current string to the right from $a$ until the the rightmost non-blank symbol.

• $q \in K$ is the current state of $M$;
**MORE THAN ONE TAPE.**

**Definition 1** A $k$-tape Turing machine, $(k \geq 1)$ is given by a $k + 1$-dimensional transition function

$$\delta : Q \times \Gamma^k \rightarrow Q \times \Gamma^k \times \{L, R\}^k.$$ 

The outcome of the computation of a $k$-string machine $M$ on input $x$ is accept (resp. reject) if $M$ reaches state $q_a$ (resp. $q_r$).

In spite of the appearance of being more powerful computational tool, multi-tape TM are computationally equivalent to standard TM’s with one tape.
**Theorem 2** Every multiple tape TM has an equivalent single tape Turing machine.

**Theorem 3** A language $L$ is Turing-recognizable iff there is a multitape TM which recognizes it.
Start

One move
Two-dimensional tape

<table>
<thead>
<tr>
<th>a</th>
<th>4</th>
<th>c</th>
<th>u</th>
<th>u</th>
<th>u</th>
<th>u</th>
<th>u</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>u</td>
<td>u</td>
<td>u</td>
<td>u</td>
</tr>
<tr>
<td>r</td>
<td>b</td>
<td>5</td>
<td>s</td>
<td>u</td>
<td>u</td>
<td>u</td>
<td>u</td>
</tr>
<tr>
<td>s</td>
<td>b</td>
<td>a</td>
<td>t</td>
<td>c</td>
<td>u</td>
<td>u</td>
<td>u</td>
</tr>
<tr>
<td>a</td>
<td>b</td>
<td>t</td>
<td>b</td>
<td>u</td>
<td>u</td>
<td>u</td>
<td>u</td>
</tr>
<tr>
<td>c</td>
<td>b</td>
<td>c</td>
<td>u</td>
<td>u</td>
<td>u</td>
<td>u</td>
<td>u</td>
</tr>
<tr>
<td>u</td>
<td>u</td>
<td>u</td>
<td>u</td>
<td>u</td>
<td>u</td>
<td>u</td>
<td>u</td>
</tr>
</tbody>
</table>

The control section is labeled as 'control'.

There is also a diagram showing a triangular section with parallel lines, indicating a path or guide for movement on the tape.
ENUMERATORS.

**Enumerator** is a Turing machine with an attached printer. The Turing machine can use that printer as an output device to print strings. Every time the Turing machine wants to add a string to the list, it sends the string to the printer.

An enumerator starts with a blank input tape. If the enumerator doesn’t halt, it may print an infinite list of strings. The language enumerated by enumerator $E$ is the collection of all strings that it eventually prints out. $E$ may generate the strings of the language in any order, possibly with repetitions.
An **enumerator** $E$ is a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, p)$ where

1. $Q$ is a finite set of **states**;
2. $q_0 \in Q$ is the initial (start) state; $p$ is the **printing** state;
3. $\Sigma$ is finite set of **symbols** called the **alphabet** of $M$; $\sqcup \notin \Sigma$;
4. $\Gamma \supseteq \Sigma \cup \{\sqcup\}$ is finite set of **symbols** called the tape **alphabet**
5. $\delta$ is the **transition** function defined on $Q \times \Sigma$,
   \[
   \delta : Q \times \Sigma \rightarrow Q \times \Sigma \times \{L, R\}
   \]

Enumerator starts with the empty tape, and computes according to the $\delta$-function, periodically passing state $p$. At every visit of $p$, the contents of the print tape is printed out, after which the computation is continued.

The set of all strings that can appear on the print tape is defined as $L^{\text{print}}(E)$. 
**Theorem 4** A language is Turing-recognizable if and only if some enumerator enumerates it.

**Proof. (IF)** an enumerator $E$ enumerates a language $L$, $\exists$ a TM $M$ which recognizes $L$.

Given input $w$, $M$ works as follows:

1. Run $E$. Every time that $E$ outputs a string, compare it with $w$
2. If $w$ ever appears in the output of $E$, accept.

**(ONLY IF)** Show that if TM $M$ recognizes a language $L$, there is an enumerator for $L$.

Let $s_1, s_2, s_3, \ldots$ be a list of all possible strings in $\Sigma^*$. One can readily construct a special purpose enumerator $E_{spec}$ which generates such a sequence. An enumerator $E$ for $L(M)$ works as follows:

Repeat the following for $i = 1, 2, \ldots$

- Run $M$ for $i$ steps on each input $s_1, s_2, \ldots, s_i$,
- If any computation is accepted, print out the corresponding $s_j$. 


2 Universal Turing Machines

A TM is **universal** if it can accept a description of another TM as an input and perform the operation of that TM.

To construct a universal TM, we need to design one encoding scheme which uses a fixed alphabet to serve all TM’s. Let $Q_\infty = \{q_1, q_2, \ldots\}$; $\Gamma_\infty = \{a_1, a_2, \ldots\}$. Without loss of generality, we can assume that for every TM, its state set is a finite subset of $Q_\infty \cup \{q_a\} \cup \{q_r\}$ and its tape alphabet is a finite subset of $\Gamma_\infty$.

<table>
<thead>
<tr>
<th>$\sigma$</th>
<th>$q_i$</th>
<th>$q_a$</th>
<th>$q_r$</th>
<th>$L$</th>
<th>$R$</th>
<th>$a_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda(\sigma)$</td>
<td>$I^{i+2}$</td>
<td>$I$</td>
<td>$I^2$</td>
<td>$I$</td>
<td>$II$</td>
<td>$I^{i+2}$</td>
</tr>
</tbody>
</table>

$\lambda$-function
Encoding TM’s over two-letter alphabet \{c, I\}

Representing the state set and the tape alphabet of
\[ M = \{Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r\} : \]
\[ Q = \{q_{(i_1)}, q_{(i_2)}, \ldots, q_{(i_k)}, q_a, q_r\}; q_0 = q_{(i_m)}(m \leq k); \]
\[ \Gamma = \{a_{(j_1)}, a_{(j_2)}, \ldots, a_{(j_l)}\} \]
\[ q_i \text{ (resp. } a_j) \text{ is encoded as } \lambda(q_i) \text{ (resp. } \lambda(a_j)) \]

Below, instead of \(q_{(i_k)}\) we write \(q_{i_k}\) for any index \(i_k\).

**Encoding the transition function.** For every transaction
\[ \delta(q_{i_p}, a_{j_r}) = (q', b, L/R), \text{ where } q' \in Q, b \in \Gamma, \text{ we define} \]
\[ w_1 = \lambda(q_{i_p}); w_2 = \lambda(a_{j_r}); w_3 = \lambda(q'); w_4 = \lambda(b); w_5 = \lambda(L/R). \]
The encoding of the transition \(\delta(q_{i_p}, a_{j_r}) = (q', b, L/R)\) is given by
\[ \rho((q_{i_p}, a_{j_r})(q', b, L/R)) \equiv S_{pr} = c w_1 c w_2 c w_3 c w_4 c w_5 c. \]

**Encoding inputs.** If
\[ w = b_1 b_2 \ldots b_n \ (b_i \in \Gamma_{\infty}) \]
then
\[ \rho(w) = c \lambda(b_1)c \lambda(b_2)c \ldots c \lambda(b_n)c. \]

**Encoding the machine.** As above \(M\) has \(k\) states and uses \(l\) letters for the tape alphabet, some of which are used for the inputs. Then
\[ \rho(M) = c \lambda(q_0)c S_{11} \cdots S_{1l} S_{21} \cdots S_{2l} \cdots S_{k1} \cdots S_{kl} c. \]
The input $\langle M, w \rangle$ is encoded by a string

$$W = \rho(M)\rho(w).$$

Note that the shortest prefix of $W$ which ends with substring $cc$ is an encoding of $q_0$. The only substring $ccc$ of $W$ separates the encoding of $M$ from the encoding of input $w$. These observations are used to prove

**Proposition 1**

There is a Turing-machine which recognizes the set of encodings of Turing machines.

**Proposition 2**

Every TM is encoded by a unique string; if $\rho(M_1) = \rho(M_2)$, then $M_1$ and $M_2$ are identical; there is an algorithm which, given $\rho(M)$, reconstructs $M$. 
Definition 2

A universal Turing machine $U = (Q^U, \Sigma^U, \delta^U, q_0^U, q_a^U, q_r^U)$ simulates $M$ iff

$M$ accepts (resp. rejects) a string $w \in \Sigma^*$ iff $U$ accepts (resp. rejects) a string consisting of the encoding of $M$ followed by the encoding of $w$;

after $U$ halts, the tape of $U$ is the encoding of the contents of the tape of $M$ when $M$ halts.
Description of a universal Turing machine.

$U$ is a 3-tape machine which uses its tapes as follows:

1\textsuperscript{st} tape contains the encoding of the tape of $M$;
2\textsuperscript{nd} tape contains the encoding of $M$ itself;
3\textsuperscript{rd} tape contains the encoding of the state of $M$
   at the current point of the simulated computation.

Initial configuration
\begin{align*}
e & \quad q_0^U \quad \rho(M) \rho(w) \\
e & \quad q_0^U \quad e \\
e & \quad q_0^U \quad e
\end{align*}

Initialization:
\begin{align*}
e & \quad q_0^U \quad \rho(M) \rho(w) & e & \quad q^U \quad \rho(w) & e & \quad q^U \quad \rho(w) \\
e & \quad q_0^U \quad e & \quad \rho(M) & e & \quad q^U \quad e & \quad \rho(M) \\
e & \quad q_0^U \quad e & \quad \rho(M) & e & \quad q^U \quad \rho(q_0)
\end{align*}

$U$ moves $\rho(M)$ onto the 2\textsuperscript{nd} tape; shifts $\rho(w)$ to the left; extracts
the encoding of the initial state and copies it into the 3\textsuperscript{rd} tape.
Simulation of steps of $M$

1. finds, on the $2^{nd}$ tape, the block $ccI^i cI^j cI^k cI^l cc$ such that $I^i$ is the string of $I$’s identical to the string of $I$’s on the $3^{rd}$ tape;

$I^j$ is the strings of $I$’s identical to the string of $I$’s on the $1^{st}$ tape which ends at the current head position of the $1^{st}$ tape;

2. changes the $1^{st}$ tape respectively (depends on $q'$ in $\delta(p, a) = (q', b, R/L)$); puts $I^k$ on the $3^{rd}$ tape and checks if it now has $\lambda(q_a)$ or $\lambda(q_r)$;

if $q$ is not a halting state, $U$ continues the simulation; if the state is halting, $U$ moves the head on the $1^{st}$ tape to the right and accepts (resp. rejects).