Variants of Turing Machines
Multitape TMs

A multitape Turing machine is a Turing machine with additional tapes with each tape is accessible individually, with the input on the first tape, and with the others blank at the beginning.

For a $k$-tape Turing machine, the transition $\delta$ is a mapping from $Q \times \Gamma^k$ to $Q \times \Gamma^k \times \{L, R\}^k$. 
**Nondeterministic TMs**

A **nondeterministic Turing machine** is one in which the transition is mapping to the power set of $Q \times \Gamma \times \{L, R\}$.

A nondeterministic Turing machine **accepts** an input if it enters an accepting state **for some computation path**.
Equivalence Between Single-tape TMs and Multitape TMs

Theorem. Every multitape Turing machine has an equivalent single-tape Turing machine.
Equivalence Between Single-tape TMs and Multitape TMs

Theorem. Every multitape Turing machine has an equivalent single-tape Turing machine.

Proof From a $k$-tape TM $M$ build a single-tape simulator $S$. 
Equivalence Between Single-tape TMs and Multitape TMs

Theorem. Every multitape Turing machine has an equivalent single-tape Turing machine.

Proof From a $k$-tape TM $M$ build a single-tape simulator $S$.

The main idea is to use the tape available to represent

- the contents of the tape squares that the head has ever visited for each tape, including the entire squares that initially hold the input,
- and the current head position for each tape.

Create such a representation for each tape and connect them with a delimiter in between, at the beginning, and at the end.
Tape Encoding

For each $a \in \Gamma$, let $\tilde{a}$ be a new symbol to signify that a head is located on the symbol.

- The input tape $w_1 \cdots w_n \sqcup \cdots$ with the head scanning the first symbol (this occurs at the beginning)

  $\tilde{w}_1 w_2 \cdots w_n$. 
**Tape Encoding**

- If a tape holds $a_1 \cdots a_s \sqcup \cdots$ and the farthest position the head has traveled is $t > r$.
  - If the head position is $r < s$, then its representation is:

    $$a_1 \cdots a_{r-1} \tilde{r}_s a_r \cdots a_s \sqcup \cdots \sqcup \sqcup_{t-s}.$$

- If the head position is $r > s$, then its representation is:

    $$a_1 \cdots a_s \sqcup \cdots \sqcup \tilde{r}_{r-s-1} \sqcup \sqcup_{t-r}.$$

The encoding never decreases in length.
Delimiter

Use a new symbol \# as a delimiter.

On input $w = w_1 \cdots w_n$, the initial form of encoding is:

$$\#\tilde{w}_1w_2\cdots w_n\#\tilde{\#}\#\tilde{\#}\# \cdots \#\tilde{\#}$$
$S$’s Action

Memorize $M$’s state using a state.

1. Construct the initial form.
2. Repeat the following:
   (a) If $M$ has accepted or rejected, accept or reject accordingly.
   (b) Otherwise, scan the tape in a direction and record, using the state, the symbols being scanned by the heads of $M$.
   (c) **Determine the next move of $M$.**
   (d) **Modify the encoding accordingly.** Insert symbols if necessary.
   (e) **Change the state accordingly.**
Identification of Symbols Scanned

In the case when the tape is scanned from right to left, use the following states.

1. \((p_{\text{scan}}, q, a_1, \ldots, a_k), a_1, \ldots, a_k \in \Gamma\): This means that the current state is \(q\) and the symbols being scanned are \(a_1, \ldots, a_k\).

2. \((p_{\text{scan}}, q, ?, \ldots, ?, a_{r+1}, \ldots, a_k), a_{r+1}, \ldots, a_k \in \Gamma\): This means that for the first \(r\) tapes the symbols being scanned are yet to be identified but for the others the symbols have been identified to be \(a_{r+1}, \ldots, a_k\).
Identification of Symbols Scanned

In the case when the tape is scanned from right to left, use the following states.

1. \((p_{\text{scan}}, q, a_1, \ldots, a_k), a_1, \ldots, a_k \in \Gamma\): This means that the current state is \(q\) and the symbols being scanned are \((p_{\text{scan}}, q, ?, \ldots, ?, a_{r+1}, \ldots, a_k), a_{r+1}, \ldots, a_k \in \Gamma\): This means that for the first \(r\) tapes the symbols being scanned are yet to be identified but for the others the symbols have been identified to be \(a_{r+1}, \ldots, a_k\).

Start scanning from the end in state \((p_{\text{scan}}, q, ?, \ldots, ?)\).

Each time a symbol of the form \(\hat{X}\) is encountered, replace the rightmost ? with that \(X\).

When all ?’s are gone, \(S\) knows the action of \(M\).

If forward motion is used, the symbols are from left to right.
Tape Modification

This step consists of

- rewriting the symbol on the current head position and
- rewriting the symbols around the current head position to move the head to the left or to the right.
Tape Modification Rules

- If the current contents are $\cdots \tilde{a}b \cdots$, $a \neq \#$, $b$ is to be replaced by $b'$, and the head moves to the left, then replace the two symbols by $\tilde{a}b'$.

- If the current contents are $\cdots \#\tilde{b} \cdots$, $b$ is to be replaced by $b'$, and the head moves to the left, then replace the two symbols by $\#\tilde{b}'$.

- If the current contents are $\cdots \tilde{b}a \cdots$, $a \neq \#$, $b$ is to be replaced by $b'$, and the head moves to the right, then replace the two symbols by $b'\tilde{a}$.
Tape Modification Rules (cont’d)

• If the current contents are \( \cdots \tilde{b}\# \cdots \), \( b \) is to be replaced by \( b' \), and the head moves to the right, then replace the \( \tilde{b}\# \) by \( b'\tilde{\#} \).

This triggers insertion:

• Use a state to remember the symbol to be inserted.

• Start by memorizing the very first insertion, \( \tilde{\#} \), and then move to the right.

• While scanning to the right, swap the symbol to be inserted and the symbol stored in the tape cell.

• Keep scanning until the \( \# \) after the very last symbol of the encoding.
Insertion

Change $Y$ to $Y'D$
Insertion

For the sequence 

# X Y' Z U # V A B C # finished

Must Insert D Here
Insertion

Must Insert Z Here
Insertion

Must Insert U Here
Insertion

# X Y' D Z U V A B C # □ □ □ □ □

Must Insert # Here
**Insertion**

```
# X Y' D Z U # A B C # ___________
```

*Must Insert V Here*
Insertion

[# X Y' D Z U # V B C #]

Must Insert A Here
Insertion

Must Insert B Here
Insertion

Must Insert C Here
Insertion

Must Insert # Here

# X Y' D Z U # V A B C □ □ □ □ □
Insertion

\[ \# \ X \ Y' \ D \ Z \ U \ # \ V \ A \ B \ C \ # \ \uparrow \uparrow \uparrow \uparrow \]

Done
Equivalence Between NTMs and TMs
Equivalence Between NTMs and TMs

**Theorem.** Every nondeterministic Turing machine has an equivalent deterministic Turing machine.
Theorem. Every nondeterministic Turing machine has an equivalent deterministic Turing machine.

Proof We may assume that $N$ is a single-tape machine — we can use the same proof as before.
Theorem. Every nondeterministic Turing machine has an equivalent deterministic Turing machine.

Proof We may assume that $\mathcal{N}$ is a single-tape machine — we can use the same proof as before.

We will construct a three-tape simulator $\mathcal{D}$ of $\mathcal{N}$. 
Construction

Let $C$ be a constant such that each transition has at most $C$ possible values. Let $\Theta = \{a_1, a_2, \ldots, a_C\}$.

Use a word $p \in \Theta^*$ to encode a nondeterministic path, where for all $i \geq 1$ and $j, 1 \leq j \leq b$, if the $i$-th symbol of $p$ is $a_j$, then it specifies at step $i$, $M$ must choose the $j$-th possibility from all possible moves available at that point (if such one exists).

The word $p$ over $\Theta$ is a valid computation path of $N$ on input $w$ if $N$ on $w$ halts according to the choices written on $p$. 
Three-tape Simulation

Use Tape 1 to store the input, Tape 2 to simulate the tape of $N$, and Tape 3 to keep an encoding of a computation path.

Define the lexicographic order of paths:

$u_1, \ldots, u_s < v_1, \ldots, v_t \in \Theta^*$ if and only if either

- $s < t$ or
- $s = t$ and there exists some $k$, $1 \leq k \leq s$, such that $u_1 = v_1, \ldots, u_{k-1} = v_{k-1}$, and $u_k < v_k$.

Here $u_k < v_k$ is evaluated according to a fixed ordering of letters in $\Theta$. 
An Algorithm for $N$

On input $w$, write the word $\#a_1$ on Tape 3, then repeat:

1. **Copy the input onto Tape 2.**
2. **Try to simulate $N$ on $w$ using the word in Tape 3 as the path.** If successful and if $N$ has accepted, then accept and halt.
3. **Modify the path to the next smallest path by incrementing it.**
4. **Erase Tape 2.**
Some Additional Results

**Corollary.** A language is Turing-recognizable if and only if it is recognized by a multitape TM.

**Corollary.** A language is Turing-recognizable if and only if it is recognized by an NTM.
Enumerators

An **enumerator** of a language $A$ is a TM with a special output tape such that the machines write on the output tape all the members of $A$ with a special symbol $\# \not= $ as a delimiter.
Enumerators

An **enumerator** of a language $A$ is a TM with a special **output tape** such that the machines write on the output tape all the members of $A$ with a special symbol $\#$ as a delimiter.

**Theorem.** A language is Turing-recognizable if and only if it has an enumerator.
Enumerators

An **enumerator** of a language $A$ is a TM with a special **output tape** such that the machines write on the output tape all the members of $A$ with a special symbol $\#$ as a delimiter.

**Theorem.** A language is Turing-recognizable if and only if it has an enumerator.

**Proof** The “if” part: Simulate the enumerator, and accept when the input word is produced by the enumerator.

The “only if” part: Simulate a recognizer $R$. For $i = 1, 2, \ldots$, for each $w$ of lexicographic order of at most $i$, simulate $M$ on $w$ for $i$ steps and outputs $w$ if $M$ accepts $w$ in $i$ steps.
Description of Objects

We assume that there is a systematic way of describing computing devices as well as their inputs. For example, a Turing machine $M$ can be described by putting down in symbols states, symbols, and transition. We fix such an encoding system. We will use $\langle M \rangle$ to represent the encoding of $M$. 
Description of Multiple Objects

To encode multiple objects in a sequence, we simply concatenate the encodings of the objects in order with a special delimiter in between.
Description of Multiple Objects

To encode multiple objects in a sequence, we simply concatenate the encodings of the objects in order with a special delimiter in between.

**Special Requirement** For Turing machines $M$ and $N$, $⟨M⟩⟨N⟩$ is a representation of a Turing machine that executes $M$’s program first and when $M$ accepts immediately jumps into $N$’s program.

This concept will be used in Chapter 6.