Recall a few definitions from the previous lecture. A TM is a **decider** if it halts (i.e. accepts or rejects) on all inputs. A language $L$ is **Turing-recognizable** if there exists a TM $M$ such that $L = L(M) = \{w \mid M$ accepts $w\}$. $L$ is **decidable** if there exists a decider $D$ such that $L = L(D)$.

**Topics Covered**

1. Turing Machine Example
2. Multi-tape TMs
3. Nondeterministic TMs

### 1 Turing Machine Example

Let $L = \{#x_1#x_2# \ldots #x_\ell \mid \ell \geq 0 \text{ and } x_i \in \{0, 1\}^* \text{ for all } i \text{ and } x_i \neq x_j \text{ for all } i \neq j\}$. In other words, $w$ is a sequence of distinct binary strings, separated by the # symbol. This is Example 3.12 in Sipser’s book, though we consider the algorithm in greater detail. Here is an informal description of a TM $M$ that recognizes $L$:

On input $w$:

1. If read $\_\_$, accept.
   
   If read $a \neq #$, reject.
   
   Otherwise, overwrite $#$ with $\hat{#}$.

2. Move right until encounter the next $#$. 
   
   Overwrite $#$ with $\hat{#}$.

3. Zig-zag between the two “marked” strings.
   
   If they are equal, reject.

4. Remove the dot over the rightmost $\hat{#}$ and place a dot over the next $#$ to the right.
If no other # is found, instead move the dot from the leftmost # to the next # to the right. That is, move the second dot one # over. If there is no # over which to write a dot, accept.

5. Go to 3.

Note that this TM is a decider. It looks at every pair of strings and rejects if two strings match. Otherwise, it accepts. The TM is designed such that it will never enter an infinite loop. In particular, \( L \) is decidable.

## 2 Multi-tape TMs

In a multi-tape TM, we have a “control” set of states and some number of tapes; suppose we have \( k \) tapes. Every tape has a head pointing somewhere on the corresponding tape. As in a standard TM, there is a finite set \( Q \) of states, an input alphabet \( \Sigma \), and a tape alphabet \( \Gamma \). The only difference is the transition function \( \delta \). Transitions in a multi-tape TM are of the form \( \delta(q, a_1, \ldots, a_k, q', a_1 \rightarrow b_1, \ldots, a_k \rightarrow b_k, M_1, \ldots, M_k) \) where \( M_i \) represents moving left, right, or staying put. (Even if you only consider left and right transitions, it’s still an equivalent TM.)

**Theorem** For any multi-tape TM \( M \), there exists an equivalent standard TM \( S \).

**Proof** To encode the contents of \( k \) tapes, we use an extra symbol # and store all \( k \) tapes on one tape in the following form:

\[
# < \text{non-blank part of tape 1} > # < \text{non-blank part of tape 2} > # \ldots #
\]

We can encode the head locations by marking them with a dot; for example, one encoding might be #011#0110100#\ldots#. The equivalent standard TM \( S \) works as follows:

<table>
<thead>
<tr>
<th>On input ( w = w_1 \ldots w_n ) over alphabet ( \Sigma ):</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Put the tape in the correct format.</td>
</tr>
<tr>
<td>2. Simulate a single move of ( M ).</td>
</tr>
<tr>
<td>(a) Scan the tape from the beginning to the ((k+1))st # to determine the locations of the tape heads.</td>
</tr>
<tr>
<td>(b) Scan again to execute the correct transition.</td>
</tr>
</tbody>
</table>
3. If at any point one of the heads moves onto #, then pause and move everything
one cell to the right to make room for a new symbol. Then resume previous
computation.

4. If transition to \( q_{\text{accept}} \), accept. If transition to \( q_{\text{reject}} \), reject.

5. Go to 2.

3 Nondeterministic TMs

- A nondeterministic Turing machine (NTM) has the same components as a
  standard TM, but a different transition function. The transition function of an
  NTM takes the form \( \delta(q,a) = \{(q',a',L),(q'',a'',R),(q''',a''',R)\} \). In general,
  \( \delta : (Q - q_{\text{accept}} - q_{\text{reject}}) \times \Gamma \rightarrow P(Q \times \Gamma \times \{L,R\}) \).

- An NTM \( N \) is a \textbf{decider} if for all input \( w \), there exists some time limit \( t \) such that
  no computation history of \( N \) on input \( w \) can be longer than \( t \).

Theorem For every NTM \( N \), there exists an equivalent TM \( D \).

Proof It is sufficient to construct an equivalent multi-tape TM \( D \). In particular, \( D \) is a
4-tape Turing machine, with the following tapes:

1. An input tape, saving the original input \( w \)
2. A work tape
3. A nondeterminism tape
4. A counter

The high-level idea for the construction of \( D \) is to let \( b \) be the maximum value for \( |\delta(q,a)| \)
in \( N \)’s transition function. That is, \( b \) is the maximum number of possible transitions from
\((q,a)\). \( D \) fills its nondeterminism tape with the set of choices it’s going to make. Initially,
this is the empty string.

\( D \)’s work tape then simulates \( N \) by doing exactly what \( N \) would do. If it needs to choose
between two transitions, it does so based on its nondeterminism tape. This nondeterminism
tape starts as the empty string, and increases in lexicographic order until \( D \) reaches an
accept or reject state. If \( D \) doesn’t accept, then it tries again with the next string of
nondeterministic choices.
An algorithm for $D$ is:

On input $w$:

1. Initialize the tapes.
2. Simulate $N$, using the nondeterminism tape to guide nondeterministic choices.
   (a) If $N$ accepts, accept.
   (b) If not, increment the nondeterminism tape and re-initialize the work tape.
       Start step 2 again.
3. Use the counter tape to guarantee that after $t$ nondeterministic choices, if all
   have rejected, then we have explored every possibility. Since they all reject, we
   can safely conclude that $D$ does not accept the input string with these nondeter-
   ministic choices.

Observe that the nondeterminism tape initially stores the empty string, then increments
each time that $D$ starts processing $w$ over again. Given the maximum number of possible
transitions $b$ from a single (state, input), the nondeterminism tape takes on the values:
$\varepsilon, 1, 2, \ldots, b, 11, 12, \ldots, 1b, \ldots, bb, 111, \ldots$, and so on. We increment this tape in lexicog-
graphic order of the tape alphabet.