§ 4.3
New Topics for Today
- Multiple-Tape Turing Machines
- Random Access Memory (RAM) Turing Machines

REVIEW:
Our basic model of Turing Machines is a one-tape TM such as

In general, a one-tape TM can be represented as a 5-tuple
\[ M = (Q, \Sigma, \delta, s, H) \]
- \( Q \) – Finite Set of States
- \( \Sigma \) – Tape alphabet, containing two special symbols \( \rhd, \rhd, \) in addition to other symbols
- \( s \) – start state
- \( H \subseteq Q \) – a set of halting states

Transition function \( \delta: (Q-H) \times (\Sigma \cup \{ \leftarrow, \rightarrow \}) \)

Note: \( Q-H \) is the set of non-halting states
MULTIPLE TAPE TM'S
We can generate one-tape TM's by adding more tapes. The most common way
to do so is to have the following device, denoted by $M_k$:

All $k$ R/W heads can move left or right independently.
The transition function for $M_k$ looks like
$\delta: (Q-H) \times (\Sigma^k \rightarrow Q \times \{\leftarrow, \rightarrow\})^k$

Let $M$ be a $k$-tape TM, we can define its configurations as follows:
$(q, u_1a_1v_1, u_2a_2v_2, \ldots, u_ka_kv_k)$,
where $u_ia_iv_i$ is the entire context (excluding the blank symbols on the right) on tape $i$, $u_i, v_i \in \Sigma^*$, $a_i \in \Sigma$. We can similarly define the transformation from one configuration to the
next, depending on $\delta$, written as $(q, u_1a_1v_1, \ldots, u_ka_kv_k) \rightarrow^* (q, u_1'a_1'v_1', \ldots, u_k'a_k'v_k')$

Let $L(M) = \{x \mid (s, u_1x, \ldots) \rightarrow^* (h, u_1a_1v_1, \ldots, u_ka_kv_1k)\}$

Note: $(\ldots u_k a_k v_{1k})$ above can be viewed as output

The advantage of using $k$-tape TM's is that it's often easier to design a $k$-tape TM
for a given computation task. We'll look at an example later when we simulate a
RAM TM with a $K$-tape TM.

Q: is a $k$-tape ($k>1$) more powerful then 1-tape TM?
I.E. is there any language (or function) that is computable by a $k$-tape TM ($k>1$),
but not compatible by a 1-tape TM?

A: NO
The reason is that we can simulate any given $K$-tape TM using a 1-tape TM

Idea: Let $M_k$ be a given $K$-tape TM. Construct a 1-tape TM $M$ s.t. $L(M) = L(M_k)$
There are 2 methods to construct such an $M$.  

FINITE
CONTROL
AKA
(TRANSITION
FUCTIONS)
Method 1 (not in textbook)

Divide the tape of $M$ into $K$ segments each segment is separated by a special symbol $\#$ not $\in \Sigma_k$ of $M_k$

Each segment is used to store the control of the tape of $M_k$
I.E., segment $i$ is for the $i$th tape of $M_k$

(Note that $x$ is the input portion excluding blank cells.)

To capture the head location, we introduce a new system $a$ for each $a \in \Sigma_k$ which is used to mark the head location, hence, each segment has exactly one such symbol.
To simulate one move of $M_K$, $M$ starts from the leftmost segment and sweeps through all segments one by one until the last # is read, and during this process $M$ will do one of the following depending on $\delta$ of $M_K$

For each $a_i$ it reads, change to $a_i$, $b_i$ if $\delta(a_i \ldots ) = (b_i \ldots )$

<table>
<thead>
<tr>
<th>$a_i$</th>
<th>$a_i$</th>
<th>$b_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

if $M_K$ moves the ith head to the right (left), $M$ will change

<table>
<thead>
<tr>
<th>$a_i$</th>
<th>$a_i$</th>
<th>$a_i$</th>
<th>$b_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

when $M_K$ on the ith tape moves to the right to a blank symbol, we need to insert a new space at the end of the ith segment.

$M$ can do so by shifting each symbol on the right of the ith # (including itself) to the right.

**Random Access Memory Turing Machines (RAM TM)**
The tapes TM uses does not have random access.
Random access, the memory (tape) is indexed and a particular location (index can be reached in one step. To be covered in more detail the following class.