This lecture covers Chapter 8 of HMU: Properties of CFLs

- Turing Machine
- Extensions of Turing Machines
- Restrictions of Turing Machines

Additional Reading: Chapter 8 of HMU.
Turing Machine: Informal Definition

- An tape extending infinitely in both sides
- A reading head that can edit tape, move right or left.
- A finite control.
- A string is accepted if finite control reaches a final/accepting state
Turing Machine: Formal Definition

A Turing machine \( M = (Q, \Sigma, \Gamma, \delta, q_0, B, F) \) such that:

- \( Q \): finite set of states
- \( \Sigma \): finite set of input symbols
- \( \Gamma \): finite set of tape symbols such that \( \Sigma \subseteq \Gamma \)
- \( \delta \): transition function. \( \delta \) is a **partial function** over \( Q \times \Gamma \), where the first component is viewed as the present state, and the second is viewed as the tape symbol read. If \( \delta(q, X) \) is defined, then

  \[ \delta(q, X) = (q', Y, D) \]

- \( B \in \Gamma \setminus \Sigma \) is the blank symbol. All but a finite number of tape symbols are \( Bs \).
- \( q_0 \): the initial state of the TM.
- \( F \): the set of final/accepting states fo the TM.
- Head **always** moves to the left or right. Being stationary is not an option.
- The Turing Machine is deterministic.
Describing TMs

- Turing machines can be defined by describing $\delta$ using a transition table.
- They can also be defined using transition diagrams (with labels appropriately altered).

If $\delta(q, X) = (q', Y, D)$

A TM that accepts any binary string that does not contain 111
Instantaneous Descriptions of TMs

- An instantaneous description (or configuration) of a TM is a complete description of the system that enables one to determine the trajectory of the TM as it operates.
- The instantaneous description or configuration or ID of a TM contains 3 parts: (a) The (finite, non-trivial) portion of tape to the left of the reading head; (b) the state that the TM is presently in; and (c) the (finite, non-trivial) portion of the tape to the right of the reading head.

<table>
<thead>
<tr>
<th>State, Tape contents, Reading head location</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 ≤ i ≤ ℓ</td>
<td></td>
</tr>
</tbody>
</table>
| q  \[\begin{array}{ccccccc}
\ldots & B & B & X_1 & X_2 & X_3 & \ldots & X_i & \ldots & X_\ell & B & B & \ldots
\end{array}\] | segment to the strict left \[X_1 \ldots X_{i-1}\] state segment \[q\] segment from the head onwards \[X_i \ldots X_\ell\] |
| head                                      |    |
| i Blanks                                  |    |
| q  \[\begin{array}{ccccccc}
\ldots & B & B & X_1 & X_2 & X_3 & \ldots & X_\ell & B & \ldots & B & \ldots
\end{array}\] | segment to the strict left \[X_1 \ldots X_\ell B^{i-1}\] state segment \[q\] |
| head                                      |    |
| i Blanks                                  |    |
| q  \[\begin{array}{ccccccc}
\ldots & B & \ldots & B & X_1 & X_2 & X_3 & \ldots & X_\ell & B & B & \ldots
\end{array}\] | state segment \[q\] segment from the head onwards \[B^i X_1 \ldots X_\ell\] |
‘Moves’ of a TM

> Just as in the case of a PDA, we use $\vdash_M$ to indicate a single move of a TM $M$, and $\vdash^*_M$ to indicate zero or a finite number of moves of a TM.

<table>
<thead>
<tr>
<th>Present ID</th>
<th>Transition</th>
<th>Next ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_1 \cdots X_{i-1}qX_i \cdots X_\ell$ ($1 &lt; i &lt; \ell$)</td>
<td>$\delta(q, X_i) = (q', Y, R)$</td>
<td>$X_1 \cdots X_{i-1}Y q'X_{i+1} \cdots X_\ell$</td>
</tr>
<tr>
<td></td>
<td>$\delta(q, X_i) = (q', Y, L)$</td>
<td>$X_1 \cdots X_{i-2}q'X_{i-1}YX_{i+1} \cdots X_\ell$</td>
</tr>
<tr>
<td>$X_1 \cdots X_\ell B^{i-1}q$</td>
<td>$\delta(q, B) = (q', Y, R)$</td>
<td>$X_1 \cdots X_\ell B^{i-1}Yq'$</td>
</tr>
<tr>
<td></td>
<td>$\delta(q, B) = (q', Y, L)$</td>
<td>$\begin{cases} X_1 \cdots X_{\ell-1}q'X_\ell Y &amp; i = 1 \ X_1 \cdots X_\ell B^{i-2}q'BY &amp; i &gt; 1 \end{cases}$</td>
</tr>
<tr>
<td>$qB^i X_1 \cdots X_\ell$</td>
<td>$\delta(q, B) = (q', Y, R)$</td>
<td>$\begin{cases} Yq'X_2 \cdots X_\ell &amp; i = 0 \ Yq'B^{i-1}X_1 \cdots X_\ell &amp; i &gt; 0 \end{cases}$</td>
</tr>
<tr>
<td></td>
<td>$\delta(q, B) = (q', Y, L)$</td>
<td>$\begin{cases} q'BYX_2 \cdots X_\ell &amp; i = 0 \ q'BYB^{i-1}X_1 \cdots X_\ell &amp; i &gt; 0 \end{cases}$</td>
</tr>
</tbody>
</table>
Language accepted by a TM

- A string $w$ is in the language accepted by a TM $M$ iff $q_0 w \xrightarrow{\ast} \alpha p \beta$ for some $p \in F$.
- Another notion of acceptance that is common is to require a TM to halt (i.e., no further transitions are possible).
- It is always possible to design a TM such that the TM halts when it reaches a final state without changing the language the TM accepts.
- However, we cannot require (all) TMs to halt for all inputs.
- A language $L$ is said to be recursively enumerable if it is accepted by some TM.
- A language $L$ is said to be recursive if both $L$ and $L^c$ are recursively enumerable.
Multiple-Track TMs

- There are $k$ tracks, each having symbols written on them.
- The machine can only read symbols from each tape corresponding to one location, i.e., all symbols in a column at any one time.
- A $k$-track TM with tape alphabet $\Gamma$ has the same language-acceptance power as a TM with tape alphabet $\Gamma^k$. 

![Diagram of Multiple-track TM]

Finite Control

\[
\begin{array}{cccc}
\cdots & \cdots & \cdots & \\
\cdots & X_1 & \cdots & \\
\cdots & X_2 & \cdots & \\
\cdots & \cdots & \cdots & \\
\cdots & X_k & \cdots & \\
\cdots & \cdots & \cdots & \\
\end{array}
\]
Multi-tape TMs

There are \( k \) tapes, each having symbols written on them.

The machine can each tape independently, i.e., the symbols read from each tape need not correspond to the same location.

After a read of each tapes, each reading head can move independently to the right, left, or stay stationary.
Multi-tape TMs

Theorem 7.1.1

*Every language that is accepted by a multi-tape TM is also recursively enumerable (i.e., accepted by some ‘standard’ TM).*

Proof of Theorem 7.1.1

1. Let $L$ be accepted by a $k$-tape TM $M$. We’ll devise a $2k$-track TM $M'$ that accepts $L$.
2. Every even tape of $M'$ has the same alphabet as that of the $k$-tape TM. The $2i^{th}$ track of $M'$ contains exactly the same contents as the $i^{th}$ tape of $M$.
3. Every odd track has an alphabet $\{B, \dagger, \ddagger\}$, and contains a single $\dagger$ or $\ddagger$; the $2i - 1^{th}$ track of $M'$ contains $\dagger$ or $\ddagger$ at the location where the $i^{th}$ reading head of $M$ is located.

\[ M' \]

```
M'  Finite Control

<table>
<thead>
<tr>
<th></th>
<th>6</th>
<th>12</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>↑</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>↑</td>
<td>↑</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
```

```
M    Finite Control

<table>
<thead>
<tr>
<th></th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
```

```
M    Finite Control

<table>
<thead>
<tr>
<th></th>
<th>12</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
```

Finite Control

\[ \cdots \]

Finite Control

\[ \cdots \]
Multi-tape TMs

Proof of Theorem 7.1.1

- The state of $M'$ has 3 components: (a) the state of $M$; (b) the number of $\dagger$'s to its strict left; and (c) a vector of length $k$ with each component taking value in $\Gamma \cup \{?\}$.
- Each move of $M$ corresponds to multiple moves of $M'$. Each move of $M$ corresponds of a sweep of the tape from the location of the leftmost dagger to that of the rightmost dagger and back performing the changes to tracks that $M$ would do to its corresponding tapes.
- At the beginning of the sweep, the head of $M'$ is at a location where the leftmost $\dagger$ is and the state of $M'$ is $(q, 0, [?, \cdots, ?])$. The head moves to the right uncovering $\dagger$'s and the corresponding track symbols.
- The right sweep ends when the second component is $k$. 

![Diagram](image-url)
Multi-tape TMs

Proof of Theorem 7.1.1

- At this stage, $M'$ knows the input symbols $M$ will have read, and knows what actions to take.
- It then sweeps left making appropriate changes to the tracks (just like $M$ does to its tape) each time a † is encountered. $M'$ also moves the †s accordingly and alters it to ‡ to indicate that it has processed this track.
- The left sweep ends when the second component is zero. At this time, $M'$ would have completed moving the †s and the track contents; they’ll now match those of $M$.
- The TM then sweeps right and returns reverting each ‡ back to †.
- $M'$ then moves the state to $(q', 0, [?, \cdots, ?])$ and start the next sweep if $q'$ is not a final state.
- Note that $M'$ mimics $M$ and hence the languages accepted are identical.
Multi-tape TMs

> The running time of a TM $M$ with input $w$ is the number of moves $M$ makes before it halts. (If it does not, the running time is $\infty$).
> The time complexity $T_M : \{0, 1, \ldots\} \to \{0, 1, \ldots\}$ of a TM $M$ is defined as follows:
  > $T_M(n) :=$ maximum running time of $M$ for an input $w$ of length $n$ symbols.

Theorem 7.1.2

*The time taken for $M'$ in Theorem 7.1.2 to process $n$ moves of $M$ is $O(n^2)$.***

Outline of Proof of Theorem 7.1.2

> After $n$ moves of $M$, any two heads of $M$ can be at most $2n$ locations apart.
> Each sweep then requires $8n$ moves of $M'$.
> Each track update requires a finite number of moves. Totally, to update the tracks, $\Theta(k)$ time steps are needed.
Non-deterministic TMs

Non-deterministic TM: $\delta(q, X)$ is a set of triples representing possible moves.

**Theorem 7.1.3**

For every non-deterministic TM $N$, there is a TM $M$ such that $L(M) = L(N)$.

**Outline of Proof of Theorem 7.1.3**

(N does Breadth-First exploration of IDs of $M$)

(If $M$ does not halt at $ID_1$)

(If $M$ does not halt at $ID_1$ and $ID_2;1$)

(If $M$ does not halt at $ID_1, ID_{2,1}$ and $ID_{2,2}$)
Outline of Proof of Theorem 7.1.3

> We can devise a 2-tape TM $M$ that simulates $N$.
> $M$ first replaces the content of the first tape by $‡$ followed by the ID that $N$ is initially in, which is then followed by a special symbol $†$, which serves as ID separator. ($M$ uses the second tape as scratch tape to enable this operation).
> If the ID corresponds to a final state, $N$ halts (as would $M$).
> If not, $M$ then identifies all possible choices for the next IDs for $N$ and enters each one of them followed by $†$ at the right end of it’s first tape. (Again, $M$ uses the second tape as scratch tape to enable this operation)
> $M$ then searches for $†$ to the right of $‡$, changes the $†$ to a $‡$ (to signify that it is processing the succeeding ID), and processes that ID in the similar way.
> $M$ halts at an ID it iff $M$ would at that ID.
Restrictions of TMs
Theorem 7.2.1

Every recursively enumerable language is also accepted by a TM with semi-infinite tape.

Outline of Proof of Theorem 7.2.1

> Given a TM $M$ that accepts a language $L$, construct a two-track TM $M'$ as follows.
> The first and second tracks of $M'$ are the R and L semi-infinite parts of the tape of $M$.
> First, write a special symbol, say $\dagger$ at the leftmost part of the second track; this indicates to $M'$ that a left move is not to be attempted at this location.
> At any time, $M'$ keeps track of whether $M$ is to the right or left of its start location.
> If $M$ is to the strict right of its start location, $M'$ mimics $M$ on the first track. If $M$ is to the strict left of its start location, $M'$ mimics $M$ on second track, but with the head directions reversed. $M'$ detects the start by the $\dagger$ symbol.
> It can be formally shown that $M'$ accepts a string iff $M$ accepts it.
Multi-stack Machines

A multistack machine is a PDA with several independent stacks (i.e., one can be popping a symbol, while the other is writing a symbol).

**Theorem 7.2.2**

*Every recursively enumerable language is accepted by a two-stack PDA*

**Outline of Proof of Theorem 7.2.2**

- If TM moves right changing tape symbol $X$ to $Y$ and state from $q$ to $q'$, PDA moves from state $q$ to $q'$ popping $X$ from left stack and pushing $Y$ to the right stack.

$\dagger$ indicates the end of the stack content (to prevent PDA from halting)
Counter Machines

- A counter machine is a multi-stack machine whose stack alphabet contains two symbols: $Z_0$ (stack end marker) and $X$
- $Z_0$ is initially in the stack.
- $Z_0$ may be replaced by $X^i Z_0$ for any $i \geq 0$
- $X$ may be replaced by $X^i$ for any $i \geq 0$.
- A counter machine effectively stores a non-negative number.
Counter Machines

Theorem 7.2.3

Every recursively enumerable language is accepted by a three-counter machine

Outline of Proof of Theorem 7.2.3

- We know a two-stack PDA can simulate any TM.
- We’ll show that a 3-counter machine can simulate any PDA.
- WLOG, let the stack alphabet of \( \Gamma = \{0, 1, \ldots, r\} \).
- Suppose the first stack contains \( Y_1 \) (top), \( \ldots, Y_k \). Then the first counter stores \( Y_1 + rY_2 + \cdots + r^{k-1}Y_k \). Similarly for the second stack.
- The third counter is used to change the two stack contents.
- Popping the top symbol a stack (say A) = finding quotient when \( Y_1 + rY_2 + \cdots + r^{k-1}Y_k \) is divided by \( r \).
  - pop \( r \) X’s from stack A, and push a single X on the third stack. Repeat until all Xs are exhausted on the stack where popping is performed.
  - Now empty stack A and copy the third stack contents onto stack A.
- Change \( Y_1 \) to some \( Y_1' \) requires adding or subtracting, which is done by popping or pushing the corresponding number of Xs.
Counter Machines

Outline of Proof of Theorem 7.2.3

- pushing a symbol $Z$ onto a stack (say $A$) = compute $rC + Z$ where $C$ is the number presently stored in the stack $A$.
  - pop one $X$ from stack $A$, and push $r X$s on the third stack.
  - Finally push $Z X$s onto the third stack. Now empty stack $A$ and copy the third stack contents onto stack $A$.
- Since the above three are the only operations needed to simulate a TM on a two-stack PDA, we can stimulate a 2-stack PDA and hence a TM using a 3-counter machine.

Theorem 7.2.4

Every recursively enumerable language is accepted by a two-counter machine

Outline of Proof of Theorem 7.2.4

- The key idea: simulate three counters using one, and use the other for manipulations.
- The first counter stores $2^i3^j5^k$ where $i, j, k$ are the contents of the 3-counter machine.
- Updates to the stack involve: (a) divide by 2, 3, or 5; (b) multiply by 2, 3, or 5; or (c) identify if $i$ or $j$ or $k$ is zero (check divisibility).
- Each operation can be easily seen to be done with a spare counter.