Lecture 8
Intro to Theory of Computation

Last time:
\[ A_{TM} = \{ \langle M, w \rangle : M \text{ is a TM that accepts } w \} \]

This \[ A_{TM} \] is Turing-recognizable

Proof: usual TM U

Then (Cantor) \( P(\mathbb{N}) \) is not countable

Proof: Assume by contradiction that \( P(\mathbb{N}) \) is countable.

\[ \Rightarrow \text{there is a listing } S_0, S_1, \ldots \]

of all subsets of \( \mathbb{N} \).

Show that this list must miss some set \( D \in \mathbb{N} \) "flipped diagonal" set \( \square \)

Then \( \Sigma^* \) is countable

Proof: dovetailing: list by length, then break ties within each length by integer value

\[ 0, 1, 00, 01, 10, 11, 000, 001, \ldots \]

Then \( \Sigma^* : M \text{ is a TM} \) is countable

If \( \Sigma^* : M \text{ is a TM} \)

same, so \( \downarrow \)

\[ \langle M \rangle : M \text{ is a TM} \in \Sigma^* \]

\[ \square \]
Claim: \( \mathcal{P}(\Sigma^*) \) is not countable

Proof:

Choose any \( a \in \Sigma \)

\[ \mathcal{P}(\Sigma^{\ast}) \supseteq \mathcal{P}(\{a^n: n \geq 0\}) \subseteq \mathcal{P}(\Sigma^*) \]

Conclusion:

There is some language \( D \) that is not Turing recognizable

Proof:

<table>
<thead>
<tr>
<th>#TMs</th>
<th>#languages</th>
</tr>
</thead>
</table>

More precise:

\( \text{A_TM is not decidable} \)

Proof: Suppose: \( \text{A_TM is decidable} \)

On input \( \langle M, w \rangle \):

- \( H \) must accept if \( M \) accepts
- \( H \) must reject if \( M \) does not accept (rejects or runs forever)

We give two proofs:

**Proof 1:** We consider a table of the behavior of \( H \)
on a variety of inputs \( \langle M, w \rangle \)
Focus on strings \( w \) that are codes of TMs \( M_1, M_2, M_3, \ldots \). Can list all TMs. Since set of TMs is countable, can list all TMs.

\[
\begin{array}{cccccc}
M_1 & M_2 & M_3 & M_4 & M_5 & \cdots \\
1 & 0 & 1 & 0 & 1 & \cdots \\
0 & 1 & 0 & 0 & 1 & \cdots \\
1 & 0 & 1 & 0 & 0 & \cdots \\
0 & 1 & 0 & 0 & 0 & \cdots \\
\end{array}
\]

\((i,j)\) entry =

\[
\begin{cases}
1 & \text{if } M_i \text{ accepts } \langle M_j \rangle \text{ \& \& } H \text{ accepts } \langle M_i, M_j \rangle \\
0 & \text{if } M_i \text{ does not accept } \langle M_j \rangle \text{ \& \& } H \text{ rejects } \langle M_i, M_j \rangle
\end{cases}
\]

Given the TM \( H \), this motivates defining a TM \( D \) as follows:

\[
D: \begin{align*}
\text{On input } \langle M \rangle: \\
\text{Let } \omega = \langle M \rangle. \\
\text{Run } H \text{ on input } \langle M, \omega \rangle: \\
\quad \text{if } H \text{ accepts } \text{ then reject} \\
\quad \text{if } H \text{ rejects } \text{ then accept}
\end{align*}
\]

For any \( i \):

Since \( D \) behaves differently from \( M_i \) on input \( \langle M_i \rangle \), \( D \neq M_i \).
However, by construction the list of TMs $M_1, M_2, \ldots$ was a complete list which is a contradiction to our assumption.

\[ \therefore \text{ ATM is not decidable.} \]

Proof 2: In this version we ignore the table and just use the definition of $D$.

Since $H$ is a decider, $D$ is a decider.

We consider the following question: Does $D$ accept $\langle D \rangle$?

Now $D$ accepts $\langle D \rangle$.

$\Rightarrow$ $H$ accepts $\langle D, \langle D \rangle \rangle$ by defn of $H$.

$\Rightarrow$ $D$ rejects $\langle D \rangle$ by defn of $D$ (contradiction).

But also $D$ rejects $\langle D \rangle$.

$\Rightarrow$ $H$ rejects $\langle D, \langle D \rangle \rangle$ by defn of $H$.

$\Rightarrow$ $D$ accepts $\langle D \rangle$ by defn of $D$ (contradiction).
Either way we get a contradiction, so the only possibility is that \( D \) doesn't exist, which implies that \( H \) doesn't exist. 

**Notes:**
- In the first proof, unlike the case of \( \mathcal{P}(\mathbb{N}) \), the table does exist.
- The issue is assuming that there is a decoder \( H \) that corresponds to it.

- \( U \) is like \( H \) except that
  - \( U \) exists
  - \( U \) can run forever (in which case \( H \) must reject)

If we replace \( H \) with \( U \) in \( D \) we don't get a contradiction because we can't "slip" running forever.

This is the essence of Turing's proof of the Undecidability of the Halting Problem, though strictly speaking a \( U \) is different.

We can use this to find an explicit language that is not Turing-recognizable.

Thus, a language \( A \) is decidable if and only if \( A \) and \( \overline{A} \) are both Turing-recognizable.
Proof (\[\rightarrow\]) \ A \ decidable \Rightarrow \ \overline{A} \ decidable

\[\downarrow\]

A \ Turing-recognizable \ \Rightarrow \ \overline{A} \ Turing-recognizable

(\[\leftarrow\]) Suppose \ A \ and \ \overline{A} \ are \ Turing-recognizable

by \ T_M, \ M_A \ and \ \overline{M}_{\overline{A}}.

Decide \ M for \ \overline{A}:

On input \ w:

1. Copy \ w \ to \ a \ second \ tape
2. Run \ M_A \ and \ \overline{M}_{\overline{A}} \ alternately \ one \ step \ at \ a \ time \ on \ each \ tape
3. One of \ M_A \ or \ \overline{M}_{\overline{A}} \ will \ halt \ and \ accept \ first
   - \ If \ M_A \ accepts \ then \ accept
   - \ If \ \overline{M}_{\overline{A}} \ accepts \ then \ reject

Cor: \ \overline{A}_{TM} \ is \ not \ Turing \ recognizable

Proof: \ Since \ \overline{A}_{TM} \ is \ T-rec, \ if \ \overline{A}_{TM} \ were \ T-rec, \ the \ \overline{A}_{TM} \ would \ be \ decidable \ which \ it \ isn't.
Def. \( A \) is co-Turing recognizable.

\[ \overline{A} \text{ is Turing recognizable} \]

We have the following picture of the space of languages over \( \Sigma \):

- \( A_{TM} \) and \( A_{DFA} \) are both Turing recognizable.
- \( A_{TM} \) is decidable.
- \( A_{DFA} \) is context-free.
- \( A_{TM} \) is Turing recognizable.
- \( \overline{A_{TM}} \) is Turing recognizable.

We can now show many other languages are undecidable.

Def. \( HALT_{TM} = \{ \langle M, w \rangle : \text{TM } M \text{ halts on input } w \} \)

This is the Halting Problem. Turing considered it, and we now show it is undecidable.

Thus, \( HALT_{TM} \) is undecidable.

Proof: Today we give one proof. Next time we give another.

Suppose \( HALT_{TM} \) were decidable with
Idea: show that if we had \( R \) then we could get a decider for \( A_{TM} \) as follows (which is impossible)

\[ \text{TM } S: \text{ On input } \langle M, w \rangle: \]
\[ \text{Run } R \text{ on input } \langle M, w \rangle: \]
\[ \text{If } R \text{ reject, then reject} \]
\[ \text{If } R \text{ accept, then run } U \text{ on input } \langle M, w \rangle \]

\( S \) would be a decider for \( A_{TM} \) which can't exist so \( R \) doesn't exist \( \Box \)

\[ E_{TM} = \{ \langle M \rangle \mid M \text{ is a TM with } L(M) = \emptyset \} \]

Thus \( E_{TM} \) is undecidable

Proof: Suppose we had a decider \( E \) for \( E_{TM} \)

Idea: we show that with \( E \) we could build a decider for \( A_{TM} \) (impossible)

\[ \text{TM } F: \text{ On input } \langle M, w \rangle \]
\[ \text{Modify } M \text{ to ensure its input and replace it with } w \text{ on its tape and then behave as } M \text{ does from there.} \]
\[ \text{Call new TM } M_w \text{ and code } \langle M, w \rangle. \]
* Run E on input \(<Mw>\)
  - If E accepts then reject
  - If E rejects then accept

Observe that \(L(Mw)\) is either \(\Sigma^*\)
  - If M accepts w
  - \(\emptyset\) (if M doesn't accept w)

\(\text{Accept w} \implies L(M_w) = \Sigma^* \implies E \text{ reject } <Mw> \implies F \text{ accept } <Mw>\)

\(M \text{ doesn't accept } w \implies L(M_w) = \emptyset \implies E \text{ accept } <Mw> \implies F \text{ accept } <M_{w}>\)

i.e. F decides ATM (contradiction)