Overview

⇒ Diagonalization
• Undecidability

Diagonalization Method

• To prove $A_{TM}$ is undecidable, use diagonalization method
  - Technique developed by Georg Cantor in 1873 for measuring size of infinite sets
• To see if finite sets are the same size, we simply count them
• What about infinite sets?
  e.g. set of natural numbers $\mathcal{N} = \{1, 2, 3, \ldots\}$ and set of even numbers $\mathcal{E} = \{2, 4, 6, \ldots\}$
  - We can’t count them, as we will never stop
  - Idea: Set up a mapping from one set to the other
Correspondences

**Definition:** Assume we have sets $A$ and $B$ and a function $f$ from $A$ to $B$. We say that $f$ is **one-to-one map** if it never maps two different elements to the same place—that is $f(a) \neq f(b)$ whenever $a \neq b$.

**Definition:** We say that $f$ is **onto** if it hits every element of $B$—that is, if for every $b \in B$ there is an $a \in A$ such that $f(a) = b$.

**Definition:** We say that $f : A \to B$ is a **correspondence** if $f$ is one-to-one and onto.

**Definition:** We say that $A$ and $B$ are the **same size** if there is a correspondence between $A$ and $B$.

- We can set up a correspondence between $\mathcal{N}$ and $\mathcal{E}$: $f(n) = 2n$.
  - So, same size!

Countable

**Definition:** A set $A$ is **countable** if either it is finite or it has the same size as $\mathcal{N}$.

- Is the set of positive rational numbers countable?
  - $Q = \{ \frac{m}{n} \mid m, n \in \mathcal{N} \}$
  - Can we set up a correspondence between $\mathcal{N}$ and $Q$
Set of Real Numbers is Uncountable

- Proof by contradiction
  - Assume that there is a correspondence from $\mathcal{N}$ to $\mathcal{R}$ called $f$
  
  $\begin{array}{|c|c|c|c|}
  \hline
  n & f(n) \\
  \hline
  1 & 3.14159... \\
  2 & 5.55555... \\
  3 & 0.14345... \\
  4 & 0.50000... \\
  5 & 0.25888... \\
  \vdots & \vdots \\
  \hline
  \end{array}$
  
  - Construct a number $x$ by giving its decimal representation
    + Between 0 and 1, so all its significant digits follow the decimal point
    + Objective: Ensure that $x \neq f(n)$ for any $n$
    + Construct $x$ so that $n$ digit differs from the $n$ digit of $f(n)$
    + Don’t use 0 or 9 to avoid problem that 0.1999... and 0.2000... are equal

More Definitions

**Definition:** An infinite set that does not have a correspondence with $\mathcal{N}$ is called **uncountable**.

**Definition:** A real number is one that has a decimal representation.
Some languages are not Turing-recognizable

- Set of all strings $\Sigma^*$ is countable:
  + Only finitely many strings of each length
  + Write down all strings of length 0, length 1, length 2, etc
- The set of all Turing machines is countable:
  + Each has an encoding as a string $\langle M \rangle$
  + Subset of a countable number is a countable number

How many possible languages are there?

- Let $L$ be the set of languages over $\Sigma$ ($L = P(\Sigma^*)$)
  - How many languages are in $L$? i.e. How many subsets are in $P(\Sigma^*)$
- If $|X| = n$, $P(X) = 2^n$
  - What if $|X|$ is infinite, but countable?
- Set of all infinite binary sequences $B$ is uncountable
  + Can use the diagonalization method used to show $R$ is uncountable
- Can construct a correspondence from $B$ to $L$
  - $b \in B$ indicates which strings of $\Sigma^*$ to include in a $A \in L$
- So, uncountable number of languages
Continued

• Set of Turing machines is countable, set of languages are not countable, so must be some languages that a Turing machine cannot recognize

• This is pretty powerful
  - There are some languages that we cannot build a Turing machine that will accept \( w \) iff \( w \in L \)
  - We haven’t said what any of those languages are, but we know they exist

• Another way to think about it (not in textbook)
  - There are some languages (set of strings) in which there is no relationship between them that can be captured in a finite way (by a TM)

Overview

• Diagonalization
  \[ \Rightarrow \] Undecidability
Introduction

- Showed how to use diagonalization method with real numbers
- Showed there are some languages that are not Turing-recognizable
- Showed that Halting problem is Turing-recognizable
- Next we will show that the Halting problem is not decidable
  - using diagonalization
- Then we will give a language that is not Turing recognizable

Halting Problem is Undecidable

- Proof by contradiction
  - Assume $A_{TM}$ is decidable
  - Must exist a TM $H$ that is a decider for $A_{TM}$
    + $H$ must accept if $M$ accepts $w$
    + $H$ must reject if $M$ does not accept $w$ (rejects or goes forever)
    + Because $H$ decides whether $M$ accepts
  - Construct a new TM $D$ that takes a TM $\langle M \rangle$ as input
    
    $$D(\langle M \rangle) = \begin{cases} 
    \text{accept} & \text{if } M \text{ does not accept } \langle M \rangle \\
    \text{reject} & \text{if } M \text{ accepts } \langle M \rangle 
    \end{cases}$$

  + We can implement $D$ using $H$ as a subroutine
    $D = \text{"On input } \langle M \rangle, \text{ where } M \text{ is a TM:} \text{\"}
    1. \text{Run } H \text{ on input } \langle M, \langle M \rangle \rangle \text{\"}
    2. \text{If } H \text{ accepts, reject, and if } H \text{ rejects, accept"}
Continued

• When we run D on a TM \(\langle M \rangle\)

\[
D(\langle M \rangle) = \begin{cases} 
  \text{accept} & \text{if } M \text{ does not accept } \langle M \rangle \\
  \text{reject} & \text{if } M \text{ accepts } \langle M \rangle
\end{cases}
\]

• Now let’s run D on itself!!!

\[
D(\langle D \rangle) = \begin{cases} 
  \text{accept} & \text{if } D \text{ does not accept } \langle D \rangle \\
  \text{reject} & \text{if } D \text{ accepts } \langle D \rangle
\end{cases}
\]

• No matter what \(D\) does, it is forced to the opposite, which is obviously a contradiction

• Thus neither TM \(D\) nor TM \(H\) can exist

Where is the Diagonalization?

• List all TMs as rows (countable number), all computations on TMs as inputs as columns

- Running \(M’\)’s

<table>
<thead>
<tr>
<th>(M_1)</th>
<th>(M_2)</th>
<th>(M_3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>accept</td>
<td>accept</td>
<td>accept</td>
</tr>
<tr>
<td>accept</td>
<td>accept</td>
<td>accept</td>
</tr>
<tr>
<td>accept</td>
<td>accept</td>
<td>accept</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

- Entry \(i, j\) is \(M_i\) on \(\langle M_j \rangle\)

- Running \(H\) that simulates \(M\)’s

<table>
<thead>
<tr>
<th>(M_1)</th>
<th>(M_2)</th>
<th>(M_3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>accept</td>
<td>reject</td>
<td>accept</td>
</tr>
<tr>
<td>accept</td>
<td>accept</td>
<td>accept</td>
</tr>
<tr>
<td>reject</td>
<td>reject</td>
<td>reject</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

- Entry \(i, j\) is \(H\) on \(\langle M_j, \langle M_j \rangle \rangle\)
Turing-Unrecognizable Language

**Definition:** A language is **co-Turing-recognizable** if it is the complement of a Turing-Recognizable language.

**Theorem:** A language is decidable iff it is Turing-recognizable and co-Turing recognizable.
Proof $\Rightarrow$:

- Let $A$ be decidable
  - Hence there is a deterministic TM $M_1$ that decides $A$
  - Construct $M_2$ that accepts any string that $M_1$ does not accept by having it accept when $M$ rejects, and vice-versa
    - So, $M_2$ recognizes $\overline{A}$
  - $M_1$ recognizes $A$ and $M_2$ recognizes $\overline{A}$
  - So, $A$ is Turing-recognizable and co-Turing-recognizable

- Hard to do this proof with a non-deterministic TM
  - Non-deterministic TMs can decide a language
  - But, they reject when all computation paths die
  - So, can’t easily ‘complementize’ a non-deterministic TM
  - Would need to first convert it to a deterministic TM

Proof $\Leftarrow$:

- Let $A$ be Turing recognizable and co-Turing recognizable
  - Let $M_1$ recognize $A$ and $M_2$ recognize $\overline{A}$
  - Construct a decider for $A$ as follows:
    $\overline{M}$ = “On input $w$
    1. Run both $M_1$ and $M_2$ on input $w$ in parallel.
    2. If $M_1$ accepts, accept; if $M_2$ accepts, reject.”
  - As every string $w$ is either in $A$ or $\overline{A}$, one of $M_1$ or $M_2$ must accept $w$.
    So $M$ is a decider
  - Since $M$ accepts all strings in $A$ and rejects all strings not in $A$, $M$ is a decider for $A$
  - Thus $A$ is decidable
A Turing-Unrecognizable Language

**Corollary:** \( \overline{A_{TM}} \) is not Turing-recognizable

- We know that \( A_{TM} \) is Turing-recognizable
- Assume \( \overline{A_{TM}} \) is Turing-recognizable
- Then \( A_{TM} \) is decidable
- Contradiction
- So, \( \overline{A_{TM}} \) is not Turing-recognizable
  - There is no Turing machine that can tell whether a certain TM will loop forever or reject on a certain input