1 Undecidability

Undecidability

Definition 1. A language $L$ is undecidable if $L$ is not decidable. Thus, there is no Turing machine $M$ that halts on every input and $L(M) = L$.

- This means that either $L$ is not recursively enumerable. That is there is no turing machine $M$ such that $L(M) = L$, or
- $L$ is recursively enumerable but not decidable. That is, any Turing machine $M$ such that $L(M) = L$, $M$ does not halt on some inputs.

---

Big Picture

1.1 Diagonalization

The Diagonal Language

Definition 2. Define $L_d = \{\langle M \rangle \mid \langle M \rangle \notin L(M)\}$. Thus, $L_d$ is the collection of Turing machines (programs) $M$ such that $M$ does not halt and accept when given itself as input.

A non-Recursively Enumerable Language

Diagonalization: Cantor

Proposition 3. $L_d$ is not recursively enumerable.

Proof. Recall that,
• Inputs are strings over \{0, 1\}

• Every Turing Machine can be described by a binary string and every binary string can be viewed as Turing Machine.

• In what follows, we will denote the \(i\)th binary string (in lexicographic order) as the number \(i\). Thus, we can say \(j \in L(i)\), which means that the Turing machine corresponding to \(i\)th binary string accepts the \(j\)th binary string.

• We can organize all programs and inputs as a (infinite) matrix, where the \((i,j)\)th entry is \(Y\) if and only if \(j \in L(i)\).

\[
\begin{array}{cccccccc}
\text{Inputs} & \rightarrow & 1 & 2 & 3 & 4 & 5 & 6 & 7 & \cdots \\
\text{TMs} & \downarrow & N & N & N & N & N & N & N & \cdots \\
1 & Y & N & Y & N & Y & N & Y & N & \cdots \\
2 & N & Y & N & Y & N & Y & N & Y & \cdots \\
3 & N & Y & N & Y & N & Y & N & Y & \cdots \\
4 & N & Y & N & Y & N & Y & N & Y & \cdots \\
5 & N & Y & N & Y & N & Y & N & Y & \cdots \\
6 & N & Y & N & Y & N & Y & N & Y & \cdots \\
\end{array}
\]

• Suppose \(L_d\) is recognized by a Turing machine, which is the \(j\)th binary string. i.e., \(L_d = L(j)\). But \(j \in L_d\) iff \(j \notin L(j)\)!

\[\square \]

**Acceptor for \(L_d\)?**

Consider the following program

On input \(\langle M \rangle\)

Run program \(M\) on \(\langle M \rangle\)

Output ‘‘yes’’ if \(M\) does not accept \(\langle M \rangle\)

Output ‘‘no’’ if \(M\) accepts \(\langle M \rangle\)

The above program does not recognize \(L_d\) because it may never output ‘‘yes’’ if \(M\) does not halt on \(\langle M \rangle\).

**Models for Decidable Languages**

**Question**

Is there a machine model such that

• all programs in the model halt on all inputs, and

• for each problem decidable by a TM, there is a program in the model that decides it?
Answer
There is no such model! Suppose there is a programming language in which all programs always halt. Programs in this language can be described by binary strings, and can be simulated by TMs.
Consider the Turing Machine $M_d$

On input $\langle M \rangle$
- Run program $M$ on $\langle M \rangle$
- Output ‘yes’ if $M$ does not accept $\langle M \rangle$
- Output ‘no’ if $M$ accepts $\langle M \rangle$

$M_d$ always halts and solves a problem not solved by any program in our language! Inability to halt is essential to capture all computation.

1.2 The Universal Language

Recursively Enumerable but not Decidable

- $L_d$ not recursively enumerable, and therefore not decidable. Are there languages that are recursively enumerable but not decidable?
- Yes, $A_{TM} = \{ \langle M, w \rangle | M$ is a TM and $M$ accepts $w \}$

Proposition 4. $A_{TM}$ is r.e. but not decidable.

Proof. We have already seen that $A_{TM}$ is r.e. Suppose (for contradiction) $A_{TM}$ is decidable. Then there is a TM $M$ that always halts and $L(M) = A_{TM}$. Consider a TM $D$ as follows:

On input $\langle N \rangle$
- Run $M$ on input $\langle N, \langle N \rangle \rangle$
- Output ‘yes’ if $M$ rejects $\langle N, \langle N \rangle \rangle$
- Output ‘no’ if $M$ accepts $\langle N, \langle N \rangle \rangle$

Observe that $L(D) = L_d$. But, $L_d$ is not r.e. which gives us the contradiction. □

A more complete Big Picture
Languages

Recursively Enumerable

Decidable

CFL

Regular

$L_0^n 1^n$

$L_{anbncn}$

$L_d, A_{TM}$

$A_{TM}$