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

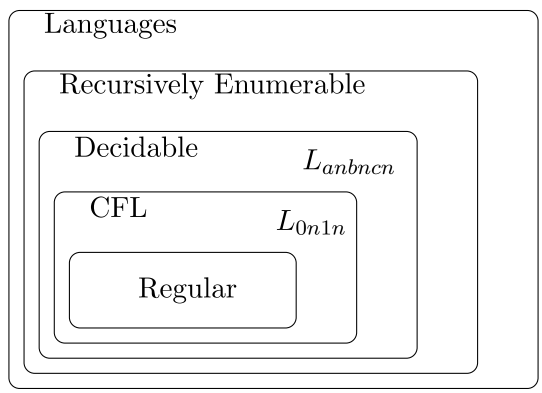


Figure 1: Relationship between classes of Languages

1.1 Diagonalization

The Diagonal Language

Definition 2. Define $L_d = \{\langle M\rangle \mid \langle M\rangle \notin \mathbf{L}(M)\}$. 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 ith binary string (in lexicographic order) as the number i. Thus, we can say $j \in \mathbf{L}(i)$, which means that the Turing machine corresponding to ith binary string accepts the jth 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 $j \in \mathbf{L}(i)$.

TMs $\downarrow$ \ Inputs $\longrightarrow$	1	2	3	4	5	6	7	$\cdots$
1	$\boxed{\text{N}}$	N	N	N	N	N	N	
2	N	$\boxed{\text{N}}$	N	N	N	N	N	
3	Y	N	$\boxed{\text{Y}}$	N	Y	Y	Y	
4	N	Y	N	$\boxed{\text{Y}}$	Y	N	N	
5	N	Y	N	Y	$\boxed{\text{Y}}$	N	N	
6	N	N	Y	N	Y	$\boxed{\text{N}}$	Y	

- Suppose L_d is recognized by a Turing machine, which is the jth binary string. i.e., $L_d = \mathbf{L}(j)$. But $j \in L_d$ iff $j \notin \mathbf{L}(j)$!

$\square$

Acceptor for L_d?

Consider the following program

```
On input ⟨M⟩
    Run program M on ⟨M⟩
    Output ``yes'' if M does not accept ⟨M⟩
    Output ``no'' if M accepts ⟨M⟩
```

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, $A_{\mathrm{TM}} = \{\langle M, w \rangle \mid M \text{ is a TM and } M \text{ accepts } w\}$

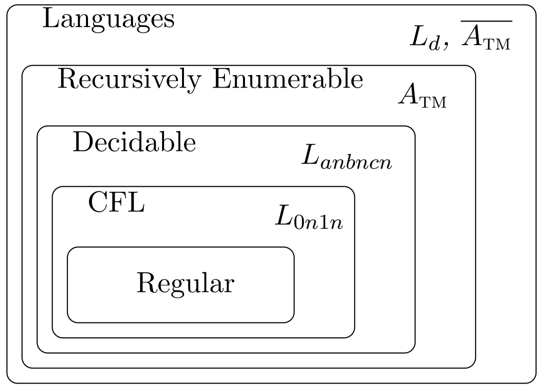
Proposition 4. *A_{TM} is r.e. but not decidable.*

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 $\mathbf{L}(M) = A_{\mathrm{TM}}$. 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 $\mathbf{L}(D) = L_d$! But, L_d is not r.e. which gives us the contradiction. □

A more complete Big Picture


Languages
L_d, $\overline{A_{\text{TM}}}$
Recursively Enumerable
A_{TM}
Decidable
L_{anbncn}
CFL
L_{0n1n}
Regular
