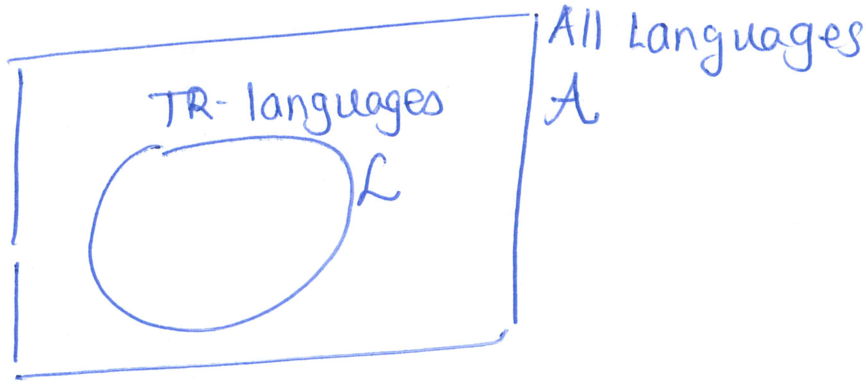
Undecidable Problems

We now study and exhibit undecidability of some problems. We first observe that there exists a language that is not Turing-recognizable and this follows easily from countability argument. Fix alphabet Σ.



Let $\mathcal{A}$ be the class of all languages

$\mathcal{L}$ 〃 Turing-recognizable languages (over Σ).

Fact $\mathcal{A}$ is uncountable.

$\mathcal{L}$ is countable.

Note. It then follows that there is a language in $\mathcal{A} \setminus \mathcal{L}$, i.e. a language that is not TR.

Claim A is uncountable.

Proof Σ^* is countable (e.g. one can consider ordering strings in increasing order of length), so let an ordering of Σ^* be

$$\Sigma^* = \{w_1, w_2, w_3, w_4, \ldots\}.$$

For any language $L \subseteq \Sigma^*$, $L \in A$, let $r(L)$ be the real number in $[0,1]$ defined as

$$r(L) = 0.b_1 b_2 b_3 b_4 \ldots \qquad b_i \in \{0,1\} \ \forall\, i \geq 1$$

where

$$b_i = \begin{cases} 1 & \text{if } w_i \in L \\ 0 & \text{if } w_i \notin L \end{cases} \qquad \forall\, i \geq 1.$$

This gives a 1-to-1 correspondence between the class of all languages A and the set of all reals in $[0,1]$. Since the latter set is uncountable, so is A. ▨

<u>Claim</u> $\mathcal{L}$ is countable.

<u>Proof</u> We note that every language $L \in \mathcal{L}$ is T.R. and is accepted by some Turing m/c, say M_L. Let $\langle M_L \rangle$ be an encoding of the m/c M_L, possibly over a different alphabet Γ. We emphasize that $\langle M_L \rangle \in \Gamma^*$ is a finite length string. This gives a 1-to-1 mapping of $\mathcal{L}$ <u>into</u> Γ^*.

Since Γ^* is countable, so is $\mathcal{L}$. $\blacksquare$

<u>Effectively Enumerable Languages</u>

Fix some alphabet Σ. Since Σ^* is countable and any language L is a subset of Σ^*, clearly every language L is countable. I.e. for any language L, there <u>exists</u> an ordering of all strings in L, say

$$L = \{ x_1, x_2, x_3, x_4, \dots \}.$$

However it does not mean that such an ordering can be "obtained" in a "constructive", "algorithmic", "effective" manner.

<u>Def</u> A <u>Turing m/c with output</u> has (in addition to an input tape) a write-only, one-way output tape. The output tape is blank initially. The m/c can write symbols on the output tape, left to right, but the m/c cannot go back and change these output symbols.

<u>Def</u> A language $L \subseteq \Sigma^*$ is called <u>effectively enumerable</u> if there is a TM with output that (on say empty input)

- runs forever and

- prints the following on the output tape

$$x_1 \# x_2 \# x_3 \# x_4 \# \cdots\cdots$$

- Here
  - $\forall i \geq 1, \quad x_i \in L$
  - $\# \notin \Sigma$ is a new separator symbol
  - Every $x \in L$ occurs as $x = x_j$ for some $j \geq 1$.

The definition above captures the notion that there is a "constructive", "algorithmic", "effective" ordering or enumeration of (all) strings in L (and only those that are in L). The enumeration is effective in the sense that it can be carried out by a TM. As we show next, a language is effectively enumerable <u>iff</u> it is Turing-recognizable! For this reason, TR languages are also referred to as effectively enumerable or recursively enumerable languages.

<u>Fact</u> L be a language.

L is effectively enumerable $\Longleftrightarrow$ L is Turing recognizable.

<u>Proof of $\Rightarrow$</u> Suppose L is effectively enumerable and let M be a TM with output that enumerates it. It is easy to construct a TM $\tilde{M}$ that recognizes L.

On input x, $\tilde{M}$ simply runs the enumerator M until the enumerator outputs a string x_j that equals x. If so, $\tilde{M}$ accepts. Otherwise $\tilde{M}$ runs forever. Clearly,

$x \in L \Rightarrow x$ occurs as $x = x_j$ for some $j \geq 1$ in the output of the enumerator M.

$\Rightarrow \tilde{M}$ (eventually) accepts x.

$x \notin L \Rightarrow x$ never occurs (as $x = x_j$) in the output of M.

$\Rightarrow \tilde{M}$ runs forever.

Thus $\tilde{M}$ recognizes L. $\blacksquare$

Proof of $\Leftarrow$: Suppose L is Turing-recognizable and $L \subseteq \Sigma^*$. We design an enumerator M for L given a TM $\tilde{M}$ that recognizes L. Let

$$\Sigma^* = \{w_1, w_2, w_3, w_4, \dots\}$$

be some effective ordering, say simply in increasing order of length.

The enumerator M works in phases. In the k^{th} phase M simulates $\tilde{M}$ on inputs $\{w_1, w_2, \dots, w_k\}$ for k steps each. If $\tilde{M}$ accepts any of these inputs, M

writes them on its output tape separated by '#' symbol.

This procedure is carried out for $k = 1, 2, 3, \ldots$.

To show that M indeed enumerates L we observe that:

① M outputs only those strings $x \in \Sigma^*$ that $\tilde{M}$ accepts, i.e. only those strings that are in the language L.

② If $x \in L$ is any string, then $\tilde{M}$ accepts x, say in k_1 steps. Moreover, $x = w_{k_2}$ for some index k_2. Let

$$k = \max\{k_1, k_2\}.$$

Then in k^{th} phase of the enumerator M, it does simulate $\tilde{M}$ on

$$x = w_{k_2} \quad (\because k \geq k_2,\ x \in \{w_1, w_2, \ldots, w_k\})$$

for k_1 steps $\quad (\because k \geq k_1)$

and when $\tilde{M}$ accepts, outputs x.

Thus every string $x \in L$ is eventually output by the enumerator.

This proves that L is effectively enumerable. ▨

We are now ready to exhibit a language that is Turing-recognizable but not decidable. We note again that:

<u>Fact</u> Σ^* is effectively enumerable. Let $\Sigma^* = \{w_1, w_2, w_3, \ldots\}$ denote its effective ordering.

We note an easy but very important fact that the set (or language) of all valid / correct Turing m/c descriptions is effectively enumerable.

<u>Fact</u> $L_{TM} = \{\langle M \rangle \mid M \text{ is a TM}\}$ is effectively enumerable.

__Proof__ The enumerator for L_{TM} simply goes through all strings $x \in \Sigma^*$, say in increasing order of length, and outputs x __iff__ $x = \langle M \rangle$ is a valid encoding of some TM M. Note that a TM can check whether a string $x \in \Sigma^*$ is a valid encoding of a TM. ∎

__Notation__ Henceforth

$$\langle M_1 \rangle, \langle M_2 \rangle, \langle M_3 \rangle, \langle M_4 \rangle, \ldots$$

will denote an effective enumeration of L_{TM}, i.e. of __all__ Turing m/c descriptions.

Now consider an infinite 2-dimensional matrix shown on the next page.

Its rows are indexed by all TM descriptions $\langle M_i \rangle$, $i \geq 1$

columns are indexed by all strings in Σ^*, w_j, $j \geq 1$.

All TM descriptions $\downarrow$ \ $\Sigma^* \rightarrow$	w_1	w_2	w_3	w_4	$\cdots w_j$
$\langle M_1 \rangle$	YES	NO	YES	YES	
$\langle M_2 \rangle$	NO	NO	NO	NO	
$\langle M_3 \rangle$			YES	NO	
$\vdots$					
$\langle M_i \rangle$					
$\vdots$					

The entries of this matrix are in $\{YES, NO\}$ defined as

$$\text{Entry}(\langle M_i \rangle, w_j) = \begin{cases} \text{YES} & \text{if } M_i \text{ accepts } w_j \\ \text{NO} & \text{otherwise.} \end{cases}$$

The "diagonal language" L_{Diag} is now defined as

<u>Def</u> $$L_{Diag} = \left\{ w_i \;\middle|\; \begin{array}{l} i \geq 1, \\ M_i \text{ accepts } w_i \end{array} \right\}.$$

I.e. the diagonal of the matrix above specifies whether inputs $w_1, w_2, w_3, \cdots$ are in the language L_{Diag} or not (YES means in, NO means out).