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 $\Sigma$.

Let $A$ be the class of all languages $L$ Turing-recognizable languages (over $\Sigma$).

Fact $A$ is uncountable.

$L$ is countable.

Note. It then follows that there is a language in $A \setminus L$, i.e. a language that is not TR.
Claim: \( \Sigma \) is uncountable.

Proof: \( \Sigma^* \) is countable (e.g., one can consider ordering strings in increasing order of length), so let an ordering of \( \Sigma^* \) be 
\[
\Sigma^* = \{ w_1, w_2, w_3, w_4, \ldots \}.
\]
For any language \( L \subseteq \Sigma^* \), \( L \in \mathcal{A} \), let \( r(L) \) be the real number in \([0,1]\) defined as 
\[
r(L) = 0.b_1b_2b_3b_4 \ldots \quad b_i \in \{0,1\} \quad \forall \ i \geq 1
\]
where 
\[
b_i = \begin{cases} 
1 & \text{if } w_i \in L \\
0 & \text{if } w_i \notin L 
\end{cases} \quad \forall \ i \geq 1.
\]
This gives a 1-to-1 correspondence between the class of all languages \( \mathcal{A} \) and the set of all reals in \([0,1]\). Since the latter set is uncountable, so is \( \mathcal{A} \).
Claim $L$ is countable.

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

Effectively Enumerable Languages

Fix some alphabet $\Sigma$. Since $\Sigma^*$ is countable and any language $L$ is a subset of $\Sigma^*$, clearly every language $L$ is countable. I.e. for any language $L$, there exists an ordering of all strings
in \( L \), say
\[
L = \{ x_1, x_2, x_3, x_4, \ldots \}
\]
However it does not mean that such an ordering can be "obtained" in a "constructive", "algorithmic", "effective" manner.

**Def** A Turing \( m/c \) with output 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.

**Def** A language \( L \subseteq \Sigma^* \) is called effectively enumerable if there is a TM with output that (on say empty input) runs forever and
- prints the following on the output tape
  \[ x_1 \# x_2 \# x_3 \# x_4 \# \ldots \]

- Here \( \forall i \geq 1, \ x_i \in L \)

- \( \# \notin \Sigma \) 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 iff it is Turing-recognizable!

For this reason, TR languages are also referred to as effectively enumerable or
recursively enumerable languages.

**Fact**  \( L \) be a language.

\( L \) is effectively enumerable \( \iff \) \( L \) is Turing recognizable.

**Proof of \( \Rightarrow \)** 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 \( x \).
$x \notin L \Rightarrow x \text{ never occurs (as } x = x_j) \text{ in the output of } M.$

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

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

Proof of $\subseteq$: 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

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

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 $\{w_1, w_2, \ldots, w_k\}$ 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 (\because $k \geq k_1$)
and when \( \hat{M} \) 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:

**Fact** \( \Sigma^* \) 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.

**Fact** \( L_{TM} = \{\langle M \rangle \mid M \text{ is a TM} \} \) 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. \[\square\]

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 $\Sigma^*$ $\langle W_j \rangle$, $j \geq 1$. 
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" \(L_{\text{Diag}}\) is now defined as:

\[
L_{\text{Diag}} = \left\{ W_i \mid i \geq 1, \text{ } M_i \text{ accepts } W_i \right\}
\]

I.e., the diagonal of the matrix above specifies whether inputs \(W_1, W_2, W_3, \ldots\) are in the language \(L_{\text{Diag}}\) or not (YES means in, NO means out).