1 Undecidable languages

Consider the set of all languages. We know that this set is not countable. On the other hand consider the set of all programs, this set is countable. So, there exist languages that are not accepted by any program. More specifically, there exist languages that are not Turing Acceptable. Can we exhibit an explicit example of a language that is not Turing Acceptable?

Let us construct one such language by diagonalization. Let $P_0, P_1, \cdots$ be a enumeration of all Programs. Let us denote the language accepted by $P_i$ with $L_i$. Consider the following language $L_{new}$

\[ k \in L_{new} \iff k \notin L_k. \]

We now claim that $L_{new}$ is not Turing Acceptable. Suppose it is Turing Acceptable, then there is a program $P_\ell$ that accepts $L_{new}$. Thus $L_{new} = L_\ell$. However, by definition of $L_{new}$

\[ \ell \in L_\ell \iff \ell \in L_{new} \iff \ell \notin L_\ell. \]

This is a contradiction. Thus $L_{new}$ is not Turing Acceptable.

What is $L_{new}$? A number $k$ is in $L_{new}$ if and only if $k$ is not in $L_k$. By our notation, $L(P_k) = L_k$. Thus $k$ is in $L_{new}$ if and only of $P_k$ does not accept $k$. So $L_{new} = \overline{K}$, where

\[ \overline{K} = \{ e \mid P_e \text{ does not accept } e \}. \]

Thus $\overline{K}$ is an explicit example of a language that is not Turing Acceptable.

Recall that if a language $L$ is Turing Decidable, then $L$ is also Turing Decidable. Every Turing decidable language is Turing acceptable. Thus if $K$ is Turing Decidable, then $\overline{K}$ is also Turing Decidable. However, we have just established that $\overline{K}$ is not Turing Acceptable. Thus $K$ is not Turing Decidable. Thus $K$ is an explicit example of a language that is Turing Acceptable but not Turing Decidable.

We have showed that $K$ is not Turing Decidable by showing that $\overline{K}$ is not Turing Acceptable. Now, we give a more direct proof of this fact.

We will show that $K$ is not Turing Decidable. Assume that $K$ is Turing Decidable. There is a program $Q$ that always halts and accepts the language $K$. Consider the following program.

1. Input $e$.
2. Run $Q(e)$.
3. If \( Q(e) \) accepts, then REJECT
4. If \( Q(e) \) rejects, then ACCEPT.

Since \( Q \) is a valid program, above is a valid program. Let \( \ell \) be its Godel Number. This means that the above program is \( P_\ell \). Now consider the behavior of \( P_\ell(\ell) \).

Suppose \( \ell \in K \). This means that \( Q(\ell) \) accepts. What happens when we run \( P_\ell \) on \( \ell \)? This will run \( Q(\ell) \). Since \( Q(\ell) \) accepts, \( P_\ell \) rejects its input which is \( \ell \). Thus \( P_\ell \) does not accepts \( \ell \). Thus \( \ell \notin K \). This is a contradiction.

Suppose \( \ell \notin K \). This means that \( Q(\ell) \) rejects. What happens when we run \( P_\ell \) on \( \ell \)? This will run \( Q(\ell) \). Since \( Q(\ell) \) rejects, \( P_\ell \) accepts its input which is \( \ell \). Thus \( P_\ell \) accepts \( \ell \). Thus \( \ell \in K \). This is a contradiction.

Thus \( Q \) does not exist. So \( K \) is not Turing Decidable.

Using the result that \( K \) is not Turing decidable, we can prove undecidability of additional languages. Consider the following language:

\[
K' = \{ e \mid P_e \text{ halts on input } e \}.
\]

We will now show that \( K' \) is not Turing decidable. We will show that if \( K' \) is Turing decidable, then \( K \) is also Turing decidable. Suppose \( K' \) is Turing decidable. Thus there is a program \( Q \) that always halts and accepts \( K' \). Consider the following program \( R \):

1. Input \( e \).
2. Run \( Q(e) \).
3. If \( Q(e) \) rejects, then REJECT.
4. If \( Q(e) \) accepts, then run \( P_e \) on input \( e \).
5. If \( P_e(e) \) accepts, then ACCEPT.
6. If \( P_e(e) \) rejects, then REJECT.

We claim that the above program always halts and accepts \( K \).

Let \( e \in K \). This means that \( P_e \) accepts \( e \). In particular, \( P_e \) halts on input \( e \). So \( e \in K' \). Thus \( Q(e) \) accepts. Consider the behavior of the above program \( R \) on this input \( e \). It first runs \( Q(e) \), since \( Q(e) \) accepts, it runs \( P_e \) on \( e \). Since \( e \in K \), \( P_e \) accepts \( e \). Thus the above program accepts \( e \).

Let \( e \notin K \). There are two possibilities: \( P_e \) does not halt on \( e \) or \( P_e(e) \) rejects. Consider the first possibility. This means that \( e \notin K' \). So \( Q(e) \) rejects. When we run \( R \) on input \( e \), it runs \( Q(e) \). Since \( Q(e) \) rejects, \( R \) rejects \( e \). Now consider the second possibility. Since \( P_e(e) \) rejects, \( P_e \) halts on input \( e \). So \( Q(e) \) accepts. When we run \( R \) on \( e \), it first run \( Q(e) \). Since \( Q(e) \) accepts, it runs \( P_e(e) \). Since \( P_e(e) \) rejects, \( R \) rejects \( e \).

This shows that if \( K' \) is Turing decidable, then \( K \) is Turing decidable. Thus \( K' \) is not Turing Decidable.

Now consider the following language.

\[
HaltingProblem = \{ \langle e, w \rangle \mid P_e \text{ halts on input } w \}.
\]

We can show that if Halting Problem is Turing decidable, then \( K' \) is Turing decidable. This shows that Halting Problem is not Turing Decidable.