1 Reductions

Recall that

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

and

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

We showed that \text{HaltingProblem} is not Turing decidable. Let us recall the proof: Assume that \text{HaltingProgram} is Turing decidable. Then there is a program (or method) \( Q \) for \text{HaltingProblem} and \( Q \) always halts. The following program, that makes a subroutine call to \( Q \), decides \( K' \).

1. Input \( e \)
2. If \( Q(\langle e, e \rangle) \) accepts, ACCEPT
3. If \( Q(\langle e, e \rangle) \) rejects, REJECT

Since we know that \( K' \) is not Turing Decidable, it follows that \text{HaltingProblem} is not Turing Decidable.

Essentially, we showed the following: “If there exists a halting program/method \( Q \) for \text{HaltingProblem}, then there exists a program \( P \) that makes subroutine calls to \( Q \), and \( P \) decides \( K' \).” We have reduced the \text{HaltingProblem} to \( K' \). Suppose \( A \) and \( B \) are two languages, informally we say that \( A \) reduces to \( B \), if the following statement holds: “If there exists a halting program/method \( Q \) for \( B \), then either exists a halting program \( P \) that makes subroutine calls to \( Q \), and \( P \) decides \( A \).”

We can use reduction as a tool to show that many languages are not decidable. The general idea is the following: Suppose you want to show that a language \( L \) is not Turing decidable. Pick a language \( A \) that is known to be undecidable. Show that \( A \) reduces to \( L \). I.e., show that if there is a halting program/method for \( L \), then there is a halting program for \( A \). Since we know that \( A \) is undecidable, it follows that there can not exist a halting program for \( L \), and thus \( L \) is undecidable.

Consider the following language.

\[ I_{ne} = \{ e \mid L(P_e) \neq \phi \}. \]

Thus \( I_{ne} \) is the set of all programs whose language is not empty. We will show that \( I_{ne} \) is undecidable by giving a reduction from \text{HaltingProblem}. Let \( Q \) be method for \( I_{ne} \). Our goal is to write a program \( P \) that makes subroutine calls to \( Q \) and decides \text{HaltingProblem}. 
We will first informally describe the idea behind the reduction. Suppose we were given \( \langle e, w \rangle \) and would like to know whether \( P_e \) halts on \( w \) or not. We are allowed to call the method \( Q \). I.e., we are allowed to build a program \( P' \) and by calling \( Q \) we will know whether \( L(P') \) is empty or not. This information should enable us to tell whether \( P_e \) halts on \( w \) or not. This means that whether \( L(P') \) is empty or not should depend on whether \( P_e \) halts on \( w \) or not. How should \( P' \) look like? Below is the code of \( P' \):

1. Input \( x \).
2. Run \( P_e \) on \( w \).
3. Accept \( x \).

What is the language accepted by \( P' \)? On every input, \( P' \) first runs \( P_e \) on \( w \). If \( P_e \) halts on \( w \), then \( P' \) accepts its input. If \( P_e \) does not halt on \( w \), then \( P' \) runs for every on every input. Thus if \( P_e \) halts on \( w \), then \( P' \) accepts every input. Thus \( L(P') \neq \emptyset \). On the other hand, if \( P_e \) does not halt on \( w \), then \( P' \) does not accept any input. Thus \( L(P') = \emptyset \). Thus knowing whether \( L(P') \) is empty or not tells us whether \( P_e \) halts on \( w \) or not.

Below is a more formal proof. Let \( Q \) be a method for \( I_{ne} \). Below is a program that decides \( \text{HaltingProblem} \) by calling the method \( Q \).

1. Input \( \langle e, w \rangle \).
2. Set \( P' \) to the following program
   (a) Input \( x \);
   (b) Run \( P_e \) on \( w \)
   (c) Accept \( x \).
3. Compute the Godel number \( \ell \) of \( P' \).
4. If \( Q(\ell) \) accepts, ACCEPT \( \langle e, w \rangle \).
5. If \( Q(\ell) \) rejects, REJECT \( \langle e, w \rangle \).

It is easy to verify that the above program halts on every input and decides \( \text{HaltingProblem} \). Since \( \text{HaltingProblem} \) is not Turing decidable, it follows that \( I_{ne} \) is not Turing Decidable.

Now consider the following language.

\[
I_{inf} = \{ e \mid L(P_e) \text{ is infinite} \}.
\]

We can show that \( I_{inf} \) is also undecidable by giving a reduction from \( \text{HaltingProblem} \) to \( I_{inf} \).

Let \( Q \) be a method for \( I_{inf} \). Our goal is to write a program that decides \( \text{HaltingProblem} \) by making subroutine calls to \( Q \). Consider the same program as above. If \( P_e \) halts on \( w \), then \( P' \) accepts \emph{every} input. This implies that \( L(P') \) is actually \( \Sigma^* \) thus is infinite. If \( P_e \) does not halt on \( w \), then \( P' \) does not accept \emph{any} input. This implies that \( L(P') \) is empty thus is not infinite. Thus knowing whether \( L(P') \) is infinite or not enables us to tell whether \( P_e \) halts on \( w \) or not. Thus \( I_{inf} \) is not Turing Decidable.
Next we consider the following language.

\[ I_{\text{Core}} = \{ e \mid P_e \text{ on } \text{on some input outputs "CORE DUMP"} \} \]

We show that \( I_{\text{Core}} \) is not decidable by giving a reduction from \textit{HaltingProblem}. Let \( Q \) be a method for \( I_{\text{core}} \). Consider the following program:

1. \( \langle e, w \rangle \).

2. Set \( P' \) to the following program:
   (a) Input: \( x \).
   (b) Run \( P_e \) on \( w \).
   (c) Output “CORE DUMP” and stop.

3. Compute the Godel number \( \ell \) of \( P' \).

4. If \( Q(\ell) \) accepts, ACCEPT \( \langle e, w \rangle \).

5. If \( Q(\ell) \) rejects, REJECT \( \langle e, w \rangle \).

It can be easily seen that if \( P_e \) halts on \( w \), then \( P' \), on any input, writes “CORE DUMP”. If \( P_e \) does not halt on \( w \), then \( P' \) never writes “CORE DUMP” on any input (except for a small subtlety: What if \( P_e \) outputs “CORE DUMP” on input \( w \)?). Thus the able program correctly decides \textit{HaltingProblem}. Thus \( I_{\text{core}} \) is not Turing Decidable.

Consider the following language:

\[ I_{\text{equal}} = \{ \langle i, j \rangle \mid L(P_i) = L(P_j) \} \]

We can show that \textit{HaltingProblem} reduces to \( I_{\text{equal}} \). Let \( Q \) be a method for \( I_{\text{equal}} \). Let \( R \) be a fixed program that accepts every input and let \( r \) be its Godel Number. Thus \( L(R) = L(P_r) = \Sigma^* \).

Consider the following program for \textit{HaltingProblem}.

1. Input \( \langle e, w \rangle \)

2. Set \( P' \) to the following program:
   (a) Input: \( x \).
   (b) Run \( P_e \) on \( w \).
   (c) accept \( x \)

3. Compute the Godel number \( \ell \) of \( P' \).

4. If \( Q(\langle \ell, r \rangle) \) accepts, ACCEPT \( \langle e, w \rangle \).

5. If \( Q(\langle \ell, r \rangle) \) rejects, REJECT \( \langle e, w \rangle \).

Suppose \( P_e \) halts on \( w \). Note that \( P' \) accepts every input. Thus \( L(P') = L(R) = \Sigma^* \). This \( L(P_\ell) = L(P_r) = \Sigma^* \). Thus \( \langle \ell, r \rangle \in I_{\text{equal}} \). Suppose that \( P_e \) does not halt on \( w \). Note that \( P' \) does not accept any input. This \( L(P') = L(P_r) = \emptyset \). Thus \( \langle \ell, r \rangle \notin I_{\text{equal}} \). Thus knowing whether \( \langle \ell, r \rangle \) belong to \( I_{\text{equal}} \) or not enables us to decide whether \( P_e \) halts on \( w \) or not. This shows that \( I_{\text{equal}} \) is not Turing Decidable.