1 Rice’s Theorem

Let \( C \) be a property of languages. Examples of \( C \) are: \textit{emptiness} (Is the language empty?), \( = \Sigma^* \) (Is the language equal to \( \Sigma^* \)), \textit{finiteness} (Is the language finite?). Given a property \( C \), the index set of \( C \) is

\[ I_C = \{ e \mid L(P_e) \text{ has the property } C \}. \]

For example, if the property we are interested is \textit{emptiness}, then the index set is

\[ \{ e \mid L(P_e) \text{ is empty } \}. \]

Given a property \( C \), we are interested in the question \textit{is} \( I_C \) \textit{decidable}? We can use Rice’s theorem to answer this question.

Rice’s Theorem Given a property \( C \), its index set \( I_C \) is decidable if and only if \( I_C \) either the empty set or the set of all the programs.

As an application of Rice’s theorem, consider the following set.

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

To prove that this set is undecidable we have to show i) \( I_{\text{empty}} \) is an index set ii) \( I_{\text{empty}} \) is not empty, and iii) \( I_{\text{empty}} \) is not the set of all programs. It is obvious that \( I_{\text{empty}} \) is an index set. It is the index set of the property “emptiness”. \( I_{\text{empty}} \) is not empty as there are programs whose languages are empty (example: a program that does not accept any string). It is also clear that \( I_{\text{empty}} \) is not the set of all the program as there are programs whose languages are not empty (example: a program that accepts every string). Hence by Rice’s theorem \( I_{\text{empty}} \) is undecidable.

Using Rice’s theorem we can show that the following languages are not Turing decidable.

\[ I_{\text{All}} = \{ e \mid L(P_e) = \Sigma^* \}, \]

\[ I_{\text{even}} = \{ e \mid L(P_e) \text{ is the set of all even length strings} \}, \]

\[ I_{\text{td}} = \{ e \mid L(P_e) \text{ is Turing decidable} \}. \]

Languages below are Turning Decidable as they equal \( \mathbb{N} \).

\[ I_{\text{ta}} = \{ e \mid L(P_e) \text{ is Turing acceptable} \}, \]

\[ I_{\text{count}} = \{ e \mid L(P_e) \text{ is countable} \}. \]
We will not prove Rice’s theorem. Finally, we note that Rice’s theorem applies to index sets only. For example, the following sets are not index sets

\[ L_{\text{HaltAll}} = \{ e \mid P_e \text{Halts on every input} \}, \]
\[ L_{\text{equal}} = \{ \langle e, j \rangle \mid L(P_e) = L(P_j) \}. \]

2 Uncomputable Functions

A function \( f : \mathbb{N} \to \mathbb{N} \) is computable, if there is a program \( P \) such that on input \( x \), the program \( P(x) \) outputs \( f(x) \) and accepts.

Given a language \( L \), the characteristic function of \( L \) is the following function \( \chi_L : \Sigma^* \to \{0, 1\} \):

\[ \chi_L(x) = 1 \text{ if } x \in L, \chi_L(x) = 0 \text{ if } x \notin L. \]

It is easy to show that a language \( L \) is Turing Decidable if and only if \( \chi_L \) is computable.

Let \( L \) be an infinite subset of natural numbers. We know that there is a bijection from \( \mathbb{N} \) to \( L \). In fact, there are infinitely many bijections from \( \mathbb{N} \) to \( L \) (You can actually show that the number of bijections from \( \mathbb{N} \) to \( L \) is not countable. Think about the proof). Is there a bijection from \( \mathbb{N} \) to \( L \) that is computable? Now we will give an example of language \( L \) for which there is no computable bijection from \( \mathbb{N} \) to \( L \).

Recall that

\[ L_{\text{HaltAll}} = \{ e \mid P_e \text{ halts on every input} \}. \]

Since \( L_{\text{HaltAll}} \subseteq \mathbb{N} \), it is countable. Thus there is a function \( f : \mathbb{N} \to L_{\text{HaltAll}} \) and \( f \) is a bijection. Is this function computable? We will now show that every bijection from \( \mathbb{N} \) to \( L_{\text{HaltAll}} \) is not computable.

Assume that there exists a computable function \( f : \mathbb{N} \to L_{\text{HaltAll}} \) such that \( f \) is a bijection.

Let \( Q \) be a program that computes \( f \).

Consider the following program:

1. Input \( e \).
2. Run \( Q \) on input \( e \) to compute \( f(e) \).
3. Run \( P_{f(e)} \) on input \( e \).
4. If \( P_{f(e)} \) accepts, then REJECT \( e \)
5. If \( P_{f(e)} \) rejects, then ACCEPT \( e \).

Let \( l \) be the Gödel number of the above program. That is, the above program is the same as \( P_l \). We first claim that \( l \in L_{\text{HaltAll}} \). Consider any input \( e \) to \( P_l \). Since \( f \) is a computable, \( Q \) halts on every input. Thus Step 2 of \( P_l \) always terminates. In Step 3, \( P_l \) runs \( P_{f(e)} \). Since \( f(e) \in L_{\text{HaltAll}} \), \( P_{f(e)} \) halts on every input. Thus \( P_{f(e)} \) halts on input \( e \). Thus \( P_l \) halts on every input. Thus \( l \in L_{\text{HaltAll}} \).

Since \( f \) is a bijection from \( \mathbb{N} \) to \( L_{\text{HaltAll}} \) and \( l \in L_{\text{HaltAll}} \), there is a natural number \( k \) such that \( f(k) = l \). Thus \( P_l = P_{f(k)} \).

Now consider the behavior of \( P_l \) on input \( k \). Program \( P_l \) on input \( k \) first computes \( f(k) \) and runs \( P_{f(k)}(k) \). If \( P_{f(k)} \) accepts \( k \), then \( P_l \) rejects \( k \). If \( P_{f(k)} \) rejects \( k \), then \( P_l \) accepts \( k \). However,
\( l = f(k) \). This means that if \( P_l \) accepts \( k \), then \( P_l \) rejects \( k \), and if \( P_k \) rejects \( k \), then \( P_l \) accepts \( k \). This is a contradiction. Thus \( f \) is not computable.

Now we will precisely characterize languages \( L \) for which there is a computable bijection from \( \mathbb{N} \) to \( L \). We will show that an infinite language \( L \) is Turing acceptable if and only if there is a computable bijection \( f \) from \( \mathbb{N} \) to \( L \). From this it follows that \( L_{HALTALL} \) is not Turing Acceptable.

Let \( f \) be a computable bijection from \( \mathbb{N} \) to \( L \). We will show that \( L \) is Turing acceptable. Consider the following program for \( L \).

1. Input \( e \).
2. Set \( n = 0 \).
3. If \( f(n) \) equals \( e \), then ACCEPT and halt.
4. Else, \( n = n + 1 \) and GoTo Step 3.

Suppose \( e \in L \), let \( m \) be the smallest number such that \( f(m) = e \). The number \( m \) must exist, because \( f \) is a bijection from \( \mathbb{N} \) to \( L \). For every \( \ell < m \), \( f(m) \neq e \). The above program keeps on incrementing the value of \( n \) starting from zero. This is because since \( f \) is computable, there is a program that always halts and outputs the value of \( f \) on any number, and so Step 3 can be done by a halting program. When the value of \( n \) becomes \( m \) it discovers that \( f(m) = e \) and the program accepts \( e \). This if \( e \in L \), then the above program accepts.

Note that the above program accepts a number \( e \) only when discovers a number \( m \) for which \( f(m) = e \). Since \( f \) is a function from \( \mathbb{N} \) to \( L \), if \( e \notin L \), for every \( m \), \( f(m) \neq e \). Thus if \( e \notin L \), then the above program does not accept \( e \). This shows that \( L \) is Turing acceptable.

We will skip the proof of the other direction.