Lesson 11: Reducibility

Theme: Reductions as a tool to prove undecidability.

1 Reductions

Consider a function \( F : \Sigma^* \rightarrow \Sigma^* \). A TM \( M \) that computes \( F \) is a TM that accepts every word \( w \in \Sigma^* \) and when it halts, the content of its tape is \( F(w) \). That is, on every word \( w \), \( M \) accepts \( w \) with the accepting run:

\[
q_0 \ w \Downarrow \cdots \Downarrow \ q_{\text{acc}} \ F(w)
\]

A function is computable, if there is a TM that computes it.

Definition 11.1 A language \( L_1 \) is mapping reducible to another language \( L_2 \), denoted by \( L_1 \leq_m L_2 \), if there is a computable function \( F \) such that for every \( w \in \Sigma^* \):

\[
w \in L_1 \text{ if and only if } F(w) \in L_2
\]

The function \( f \) is called mapping reduction.

Sometimes we omit the word “mapping” and call it simply “reducible” or “reduction,” instead of “mapping reducible” or “mapping reduction.” Intuitively \( L_1 \leq_m L_2 \) means that \( L_2 \) is “computationally more general,” or “more general” than \( L_1 \) and that a TM for deciding \( L_2 \) can be used to decide \( L_1 \).

Definition 11.2 A language \( L_1 \) is Turing reducible to another language \( L_2 \), denoted by \( L_1 \leq_T L_2 \), if by assuming that \( L_2 \) is decidable by a TM \( M_2 \), there is a TM \( M_1 \) that decides \( L_1 \) using \( M_2 \) as a “subroutine.”

Moreover, we also assume that \( M_2 \) decides \( L_2 \) in one step. We call \( M_1 \) a TM with oracle access to \( L_2 \).

Obviously, if \( L_1 \leq_m L_2 \), then \( L_1 \leq_T L_2 \). Also, if \( L_1 \leq_T L_2 \) and \( L_1 \) is undecidable, so is \( L_2 \).

2 Some variants of Halting problem

The following languages are all undecidable.

- \( L_0 := \{ [M] \mid L(M) = \emptyset \} \).
  That is, \( [M] \in L_0 \) if and only if \( M \) does not accept any word.

- \( L_1 := \{ [M] \mid L(M) = \{0,1\}^* \} \).
  That is, \( [M] \in L_1 \) if and only if \( M \) accepts every word.

- \( L_2 := \{ [M] \mid M \text{ accepts the empty word } \epsilon \} \).
  That is, \( [M] \in L_2 \) if and only if \( M \) accepts the empty word \( \epsilon \).

- \( L_3 := \{ [M] \mid M \text{ accepts the word } 1101 \} \).

- \( L_4 := \{ [M] \mid L(M) = \{a^n b^n \mid n \geq 0 \} \} \).

- \( L_5 := \{ [M] \mid L(M) \text{ is a regular language} \} \).
3 Post correspondence problem (PCP)

Post Correspondence Problem, denoted by PCP, is defined as follows.

<table>
<thead>
<tr>
<th>PCP</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> Pairs of strings ((u_1, v_1), \ldots, (u_m, v_m)).</td>
</tr>
<tr>
<td><strong>Task:</strong> Output True, if there is a sequence (i_1, \ldots, i_k) such that (u_{i_1} u_{i_2} \cdots u_{i_k} = v_{i_1} v_{i_2} \cdots v_{i_k}). Otherwise, output False.</td>
</tr>
</tbody>
</table>

**Theorem 11.3** PCP is undecidable.

4 Some undecidable problems concerning CFL

4.1 The non-emptiness problem for the intersection of two CGL’s

The CFL-Intersection is defined as follows.

<table>
<thead>
<tr>
<th>CFL-Intersection</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> Two CFG’s (G_1 = \langle \Sigma, V_1, R, S \rangle) and (G_2 = \langle \Sigma, V_2, R_2, S_2 \rangle).</td>
</tr>
<tr>
<td><strong>Task:</strong> Output True, if (L(G_1) \cap L(G_2) \neq \emptyset). Otherwise, output False.</td>
</tr>
</tbody>
</table>

**Theorem 11.4** The problem CFL-Intersection is undecidable.

4.2 The CFL universality problem

The problem CFL-Universality is defined as follows.

<table>
<thead>
<tr>
<th>CFL-Universality</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> A CFG (G = \langle \Sigma, V, R, S \rangle).</td>
</tr>
<tr>
<td><strong>Task:</strong> Output True, if (L(G) = \Sigma^*). Otherwise, output False.</td>
</tr>
</tbody>
</table>

**Theorem 11.5** The problem CFL-Universality is undecidable.

4.3 The CFL subset problem

This problem, denoted by CFL-Subset, is defined as follows.

<table>
<thead>
<tr>
<th>CFL-Subset</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> Two CFG’s (G_1) and (G_2).</td>
</tr>
<tr>
<td><strong>Task:</strong> Output True, if (L(G_1) \subseteq L(G_2)). Otherwise, output False.</td>
</tr>
</tbody>
</table>

**Theorem 11.6** The problem CFL-Subset is undecidable.