Lesson 9: Reducibility

**Theme:** Reductions as a tool to prove undecidability.

1 Reductions

Consider a function $F : \Sigma^* \to \Sigma^*$. A TM $M$ that computes $F$ is a 2-tape TM that accepts every word $w \in \Sigma^*$ and when it halts, the content of its second tape is $F(w)$. There is no restriction on the content of the first tape. That is, on every word $w$, $M$ accepts $w$ with the accepting run:

$$(q_0, \cdot w, \cdot) \vdash \cdots \vdash (q_{\text{acc}}, u, \cdot F(w))$$

for some string $u$ (which denotes the content of the first tape). A function is computable, if there is a TM that computes it.

**Definition 9.1** 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 9.2** 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 $\epsilon$.
- $L_3 := \{[M] \mid M \text{ accepts the word } 1101\}$.
- $L_4 := \{[M] \mid L(M) = \{a^n b^n \mid n \geq 0\}\}$.
- $L_5 := \{[M] \mid L(M) \text{ is a regular language}\}$.
Proof that $L_0$ is undecidable (via mapping reduction). We are going to show that $\text{HALT} \leq_m \overline{L_0}$, where $\overline{L_0}$ is the complement of $L_0$. The reduction is as follows.

**INPUT:** $[\mathcal{M}]w$.

- Construct a TM $\mathcal{K}_{\mathcal{M},w}$ that works as follows.
  
  **INPUT:** $u \in \Sigma^*$.
  
  - Run $\mathcal{M}$ on $w$.
  - If $\mathcal{M}$ accepts $w$, ACCEPT.
  - If $\mathcal{M}$ rejects $w$, REJECT.

  (Note: ACCEPT and REJECT above are inside $\mathcal{K}_{\mathcal{M},w}$, thus, they are supposed to mean $\mathcal{K}_{\mathcal{M},w}$ accepts and rejects its input string $u$, respectively.)

- Output $[\mathcal{K}_{\mathcal{M},w}]$.

The language accepted by $\mathcal{K}_{\mathcal{M},w}$ is as follows.

$$L(\mathcal{K}_{\mathcal{M},w}) := \begin{cases} \Sigma^*, & \text{if } \mathcal{M} \text{ accepts } w \\ \emptyset, & \text{if } \mathcal{M} \text{ does not accept } w \end{cases}$$

Thus, $\mathcal{M}$ accepts $w$ if and only if $L(\mathcal{K}_{\mathcal{M},w}) \neq \emptyset$. By definition of $L_0$ and $\text{HALT}$, $[\mathcal{M}]w \in \text{HALT}$ if and only if $[\mathcal{K}_{\mathcal{M},w}] \notin L_0$. Since $\text{HALT}$ is undecidable, $\overline{L_0}$ is undecidable, and therefore, $L_0$ is undecidable.

Proof that $L_0$ is undecidable (via Turing reduction). Suppose to the contrary that $L_0$ is decidable. Let $\mathcal{M}_0$ is a TM that decides $L_0$. Then, the following algorithm, denoted by $A$, decides the language $\text{HALT}$.

**INPUT:** $[\mathcal{M}]w$.

- Construct a TM $\mathcal{K}_{\mathcal{M},w}$ that works as follows.

  **INPUT:** $u \in \Sigma^*$.
  
  - Run $\mathcal{M}$ on $w$.
  - If $\mathcal{M}$ accepts $w$, ACCEPT. (Note: ACCEPT here is for $\mathcal{K}_{\mathcal{M},w}$ to accept $u$.)
  - If $\mathcal{M}$ rejects $w$, REJECT. (Note: REJECT here is for $\mathcal{K}_{\mathcal{M},w}$ to reject $u$.)

- Run $\mathcal{M}_0$ on $[\mathcal{K}_{\mathcal{M},w}]$.
  
  - If $\mathcal{M}_0$ accepts $[\mathcal{K}_{\mathcal{M},w}]$, REJECT. (Note: REJECT here is for $A$ to reject $[\mathcal{M}]w$.)
  - If $\mathcal{M}_0$ rejects $[\mathcal{K}_{\mathcal{M},w}]$, ACCEPT. (Note: ACCEPT here is for $A$ to accept $[\mathcal{M}]w$.)

Note that the language $L(\mathcal{K}_{\mathcal{M},w})$ is the same as above. That is,

$$L(\mathcal{K}_{\mathcal{M},w}) := \begin{cases} \Sigma^*, & \text{if } \mathcal{M} \text{ accepts } w \\ \emptyset, & \text{if } \mathcal{M} \text{ does not accept } w \end{cases}$$

Thus, $[\mathcal{M}]w \in \text{HALT}$ if and only if $[\mathcal{K}_{\mathcal{M},w}] \notin L_0$. Since $\mathcal{M}_0$ is supposed to decide $L_0$, our algorithm $A$ above decides $\text{HALT}$, which contradicts the fact that $\text{HALT}$ is undecidable. Therefore, there is no such Turing machine $\mathcal{M}_0$ that decides $L_0$, which means $L_0$ is undecidable.
Proof that $L_4$ is undecidable (via mapping reduction). We are going to show that $\text{HALT} \leq_m L_4$. The reduction is as follows.

**INPUT:** $[M]w$.

- Construct a TM $K_{M,w}$ that works as follows.

  **INPUT:** $u \in \Sigma^*$.
  - Run $M$ on $w$.
  - If $M$ accepts $w$, check if $u$ is of the form $a^n b^n$, for some $n \geq 1$.
    * If $u$ is of the form $a^n b^n$, ACCEPT.
    * If $u$ is not of the form $a^n b^n$, REJECT.
  - If $M$ rejects $w$, REJECT.

(Note: ACCEPT and REJECT above are inside $K_{M,w}$, thus, they are supposed to mean $K_{M,w}$ accepts and rejects its input string $u$, respectively.)

- Output $[K_{M,w}]$.

The language accepted by $K_{M,w}$ is as follows.

$$L(K_{M,w}) := \begin{cases} \{a^n b^n | n \geq 1\}, & \text{if } M \text{ accepts } w \\ \emptyset, & \text{if } M \text{ does not accept } w \end{cases}$$

Thus, $M$ accepts $w$ if and only if $L(K_{M,w}) = \{a^n b^n | n \geq 1\}$. By definition of $\text{HALT}$ and $L_4$, $Mw \in \text{HALT}$ if and only if $[K_{M,w}] \in L_4$. Since $\text{HALT}$ is undecidable, $L_4$ is undecidable too.

Proof that $L_4$ is undecidable (via Turing reduction). Suppose to the contrary that $L_4$ is decidable. Let $M_4$ be a TM that decides $L_4$. Then, the following algorithm, denoted by $A$, decides the language $\text{HALT}$.

**INPUT:** $[M]w$.

- Construct a TM $K_{M,w}$ that works as follows.

  **INPUT:** $u \in \Sigma^*$.
  - Run $M$ on $w$.
  - If $M$ accepts $w$, check if $u$ is of the form $a^n b^n$, for some $n \geq 1$.
    * If $u$ is of the form $a^n b^n$, ACCEPT. (Here ACCEPT is for $K_{M,w}$ to accept $u$.)
    * If $u$ is not of the form $a^n b^n$, REJECT. (Here REJECT is for $K_{M,w}$ to reject $u$.)
  - If $M$ rejects $w$, REJECT. (Here REJECT is for $K_{M,w}$ to reject $u$.)

- Run $M_4$ on $[K_{M,w}]$.
- If $M_4$ accepts $[K_{M,w}]$, ACCEPT. (Here ACCEPT is for $A$ to accept $[M]w$.)
- If $M_4$ rejects $[K_{M,w}]$, REJECT. (Here REJECT is for $A$ to reject $[M]w$.)

The reasoning is the same as above. $M$ accepts $w$ if and only if $L(K_{M,w}) = \{a^n b^n | n \geq 1\}$. Thus, $[M]w \in \text{HALT}$ if and only if $[K_{M,w}] \in L_4$. Since $M_4$ is supposed to decide $L_4$, our algorithm $A$ above decides $\text{HALT}$, which contradicts the fact that $\text{HALT}$ is undecidable. Therefore, there is no such Turing machine $M_4$ that decides $L_4$, which means $L_4$ is undecidable.
3 Some undecidable problems concerning CFL

We consider the following three problems defined below.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Input</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CFL-Intersection</strong></td>
<td>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>
<td>Output True, if $L(G_1) \cap L(G_2) \neq \emptyset$. Otherwise, output False.</td>
</tr>
<tr>
<td><strong>CFL-Universality</strong></td>
<td>A CFG $\mathcal{G} = \langle \Sigma, V, R, S \rangle$.</td>
<td>Output True, if $L(\mathcal{G}) = \Sigma^*$. Otherwise, output False.</td>
</tr>
<tr>
<td><strong>CFL-Subset</strong></td>
<td>Two CFG’s $G_1$ and $G_2$.</td>
<td>Output True, if $L(G_1) \subseteq L(G_2)$. Otherwise, output False.</td>
</tr>
</tbody>
</table>

**Theorem 9.3** All the problems above, CFL-Intersection, CFL-Universality and CFL-Subset, are undecidable.