Chapter 6: Uncomputability

A. Introduction

1. Recall:
   - Church's Thesis.
   - Equivalence of Turing-computable, grammatically-computable, and μ-recursive functions.
   - Definitions of Turing-decidable and Turing-acceptable.
   - Universal TM's and the encoding ρ.
2. Questions:
   - Are there interesting languages that are not Turing-decidable? Turing-acceptable?
   - If so, Church's Thesis says that deciding membership in these languages is not computable in the intuitive sense.

B. The halting problem

1. Two easy results about languages:
   - **Theorem 6.1.1.** Every Turing-decidable language is Turing-acceptable.
   - **Theorem 6.1.2.** If L is a Turing-decidable language then so is its complement $\bar{L}$.
2. Two slightly strange languages:
   - $K_0 = \{\rho(M)\rho(w) \mid \text{TM } M \text{ halts on input string } w.\}$
   - $K_1 = \{\rho(M) \mid \text{TM } M \text{ halts on input string } \rho(M).\}$
3. Undecidability of the halting problem.
   - **Lemma.** K_0 is decidable iff every Turing-acceptable language is Turing-decidable.
     Proof.
     $\Longrightarrow$ Suppose K_0 is decidable by some machine M_0. And suppose M_1 accepts language L. Then can decide whether $w \in L$ by giving $\rho(M_1)\rho(w)$ to M_0.
     $\Longleftarrow$ Suppose every Turing-acceptable language is Turing-decidable. But K_0 is Turing-acceptable (using essentially the Universal TM). Hence K_0 is Turing-decidable.
   - **Theorem.** K_0 is undecidable.
     Proof.
     Suppose K_0 were decidable.
     Then K_1 would also be decidable.
     And $\bar{K}_1$ would also be decidable, where $w \in \bar{K}_1$ iff w is not an encoding of a TM or if $w = \rho(M)$ for some machine M which does not halt on input w.
     Hence, $\bar{K}_1$ is Turing-acceptable.
     Suppose M^* accepts $\bar{K}_1$, and consider the string $\rho(M^*)$.
     We have the following contradiction:

     $$\rho(M^*) \in \bar{K}_1 \quad \text{iff} \quad M^* \text{ does not halt on input } \rho(M^*), \text{ and}$$
     $$\rho(M^*) \in \bar{K}_1 \quad \text{iff} \quad M^* \text{ does halt on input } \rho(M^*).$$

     We conclude that K_0 is not decidable.

4. Implications
   - **Theorem 6.1.3.** Not every Turing-acceptable language is Turing-decidable.
   - **Theorem 6.1.4.** The complements of some Turing-acceptable languages are not Turing-acceptable.
   - There are problems about TMs that aren't solvable by algorithms.
   - This doesn't say that you can *never* decide whether a given machine is going to halt on a given input.
   - But fixing the machine doesn't necessarily help either.
5. More results on languages.
   - **Theorem 6.2.1.** A language L is Turing-decidable iff both L and $\bar{L}$ are Turing-acceptable.
   - Definitions.
     - The **output language** of a TM M:
     - A TM M **enumerates** a language L iff for some fixed state q,
   - **Theorem 6.2.2.** A language is Turing-acceptable iff it is the output language of some TM.
   - **Theorem 6.2.3.** A language is Turing-acceptable iff it is Turing-enumerable.
   - **Theorem 6.2.4.** A language is Turing-acceptable iff it is generated by some grammar.

C. More unsolvable problems

1. Basic idea. Show a language L_2 is undecidable by *reducing* L_1 to it, where L_1 is already known to be undecidable (e.g., halting problem K_0).
2. **Theorem 6.3.1.** The following problems about TMs are unsolvable.

   (c) Does M halt on ε?
   (d) Does M halt on anything?
   (e) Does M halt on everything?
   (f) Do M_1 and M_2 halt on the same set?
   (g) Is the language accepted by M regular? Context-free? Turing-decidable?

3. **Theorem 6.4.1.** Each of the following problems about grammars is unsolvable.

   (a) Is $w \in L(G)$?
   (b) Is $w \in L(G_0)$, for some fixed G_0?
   (c) Is $L(G_1) = L(G_2)$?
   (d) Is $L(G) = \emptyset$?

4. Post's correspondence problem.
   - Definition. A **correspondence system** is a finite set P, where $P \subset \Sigma^+ \times \Sigma^+$ for some alphabet Σ. A **match** of P is any $w \in \Sigma^+$ such that

     $$w = u_1 u_2 \ldots u_n = v_1 v_2 \ldots v_n,$$

     where $(u_1, v_1), \ldots, (u_n, v_n) \in P$.
   - Theorem. Post's correspondence problem (given a correspondence system, determine if it has a match) is unsolvable.
   - Idea of proof. Show "derivability in a grammar" reduces to "modified Post correspondence problem," which reduces to the Post correspondence problem.
5. Implications for grammars.
   - **Theorem 6.4.4.** Determining whether $L(G_1) \cap L(G_2) = \emptyset$ is unsolvable.
   - **Theorem 6.4.5.** Determining whether a grammar is ambiguous is unsolvable.