CS 530: Theory of Computation. Based on Sipser second edition

Reducibility

In this chapter, we introduce the primary method for proving that problems are computationally unsolvable. It is called **reducibility**.

A **reduction** is a way of converting one problem into another problem in such a way that a solution to the second problem can be used to solve the first problem.

As before, we use $\langle B \rangle$ to denote an encoding of a mathematical object (such as TM or CFG) as a string.

We have already established the undecidability if A_{TM}, the problem of determining whether a Turing machine accepts a given input. Let's consider a related problem, $HALT_{TM}$, the problem of determining whether a Turing machine halts (by accepting or rejecting) on a given input. Let $HALT_{TM} = \{\langle M \rangle, w \mid M \text{ is a TM and M halts on input } w\}$.

Theorem 1 (Theorem 5.1, page 188) *$HALT_{TM}$ is undecidable.*

Let $E_{TM} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) = \emptyset\}$.

Theorem 2 (Theorem 5.2, page 189) *E_{TM} is undecidable.*

Let $REGULAR_{TM}$ be the problem of determining whether a given Turing machine has an equivalent finite automaton. This problem is the same as determining whether the Turing machine recognizes a regular language. Let $REGULAR_{TM} = \{\langle M \rangle \mid M \text{ is a TM that } L(M) \text{ is a regular language } \}$.

Theorem 3 (Theorem 5.3, page 191) *$REGULAR_{TM}$ is undecidable.*

The following theorem shows that testing the equivalence of two Turing machines is an undecidable problem. We could prove it by a reduction from A_{TM}, but we use this opportunity to give an example of an undecidability proof by reduction from E_{TM}. Let $EQ_{TM} = \{\langle M_1 \rangle, \langle M_2 \rangle \mid M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) = L(M_2)\}$.

Theorem 4 (Theorem 5.4, page 192) *EQ_{TM} is undecidable.*

Definition 5 (Definition 5.5, page 193) *Let M be a Turing machine and w an input string. An **accepting computation history** for M on w is a sequence of configurations, $C_1, C_2, ..., C_l$, where C_1 is the start configuration of M on w, C_l is an accepting configuration of M, and each C_i legally follows from C_{i-1} according to the rules of M. A **rejecting computation history** for M on w is defined similarly, except that C_l is a rejecting configuration.*

Definition 6 (Definition 5.6, page 193) *A **linear bounded automaton** is a restricted type of Turing machine wherein the tape head isn't permitted to move off the portion of the tape containing the input. If the machine tries to move its head off either end of the input, the head stays where it is, in the same way that the head will not move off the left-hand end of an ordinary Turing machine's tape.*

Let $A_{LBA} = \{\langle M \rangle\, w; \mid M \text{ is an LBA that accepts string } w\}$.

Lemma 7 (Lemma 5.8, page 194) *Let M be an LBA with q states and g symbols in the tape alphabet. There are exactly qng^n distinct configurations of M for a tape of length n.*

Theorem 8 (Theorem 5.9, page 194) *A_{LBA} is decidable.*

Theorem 5.9 shows that LBAs and TMs differ in one essential way: For LBAs the acceptance problem is decidable, but for TMs it isn't. However, certain other problems involving LBAs remain undecidable. One is the emptiness problem $E_{LBA} = \{\langle M \rangle \mid M \text{ is an LBA where } L(M) = \emptyset\}$.

Theorem 9 (Theorem 5.10, page 195) *E_{LBA} is undecidable.*

We can also use the technique of reduction via computation histories to establish the undecidable of certain problems related to context-free grammars and pushdown automata. Let $ALL_{CFG} = \{\langle G \rangle \mid G \text{ is an CFG and } L(G) = \Sigma^*\}$.

Theorem 10 (Theorem 5.13, page 197) *ALL_{CFG} is undecidable.*

We give an example of an undecidable problem concerning simple manipulations of strings. It is called the **Post correspondence problem**, or **PCP**.

We can describe this problem easily as a type of puzzle. We begin with a collection of dominos, each containing two strings, one on each side. An individual domino looks like

$$[\frac{a}{ab}]$$

and a collection of dominos looks like

$$\{[\frac{a}{ab}], [\frac{b}{ca}], [\frac{ca}{a}], [\frac{a}{ab}], [\frac{abc}{c}]\}$$

The task is to make a list of these dominos (repetitions permitted) so that the string we get by reading off the symbols on the top is the same as the string of symbols on the bottom. This list is called a **match**.

An instance of the PCP is a collection P of dominos:

$$P = \{[\frac{t_1}{b_1}], [\frac{t_2}{b_2}], ..., [\frac{t_k}{b_k}]\}.$$

and a match is a sequence $i_1, i_2, ..., i_l$, where $t_{i_1}, t_{i_2}, ...t_{i_l} = b_{i_1}, b_{i_2}, ...b_{i_l}$. The problem is to determine whether P has a match. Let $PCP = \{\langle P\rangle \mid P$ is an instance of the Post correspondence problem with a match $\}$.

Theorem 11 (Theorem 5.15, page 200) *PCP is undecidable.*

In the proof, we first modify the PCP to require that a match starts with the first domino,

$$[\frac{t_1}{b_1}].$$

Later we show how to eliminate this requirement. We call this problem the modified Post correspondence problem, MPCP. Let $MPCP = \{\langle P\rangle \mid P$ is an instance of the Post correspondence problem with a match that starts with the first domino$\}$.

Definition 12 (Definition 5.17, page 206) *A function $f : \Sigma^* \to \Sigma^*$ is a **computable function** if some Turing machine M, on every input w, halts with just $f(w)$ on its tape.*

Definition 13 (Definition 5.20, page 207) *Language A is **mapping reducible** to language B, written $A \leq_m B$, if there is a computable function $f : \Sigma^* \to \Sigma^*$, where for every w,*

$$w \in A \iff f(w) \in B.$$

*The function f is called the **reduction** of A to B.*

Theorem 14 (Theorem 5.22, page 208) *If $A \leq_m B$ and B is decidable, then A is decidable.*

Theorem 15 *(Implicit from the proof above) For any three languages A, B, C, if $A \leq_m B$ and $B \leq_m C$, then $A \leq_m C$.*

Corollary 16 (Corollary 5.23, page 208) *If $A \leq_m B$ and A is undecidable, then B is undecidable.*

Theorem 17 (Example 5.24, page 208) $A_{TM} \leq_m HALT_{TM}$.

Theorem 18 (Example 5.26, page 209) $E_{TM} \leq_m EQ_{TM}$.

Theorem 19 (Example 5.27, page 209) $A_{TM} \leq_m \overline{E_{TM}}$.

Theorem 20 (Theorem 5.28, page 209) *If $A \leq_m B$ and B is Turing-recognizable, then A is Turing-recognizable.*

Corollary 21 (Corollary 5.29, page 210) *If $A \leq_m B$ and A is not Turing-recognizable, then B is not Turing-recognizable.*

Theorem 22 (Theorem 5.30, page 210) *EQ_{TM} is neither Turing-recognizable nor co-Turing-recognizable.*