10 Reducibility

10.1 What is reducibility?

Reading: Sipser 5 (pp. 187-188)

We say that problem A reduces (or is reducible) to problem B, if we can use a solution to B to solve A (i.e., if B is decidable/solvable, so is A).

We may use reducibility to prove undecidability as follows: Assume we wish to prove problem B to be undecidable and we know a problem A that has already been proved undecidable. We use contradiction. Assume B is decidable. Then there exists a TM $M_B$ that decides B. If we can use $M_B$ as a sub-routine to construct a TM $M_A$ that decides A, we have a contradiction. The construction of TM $M_A$ using TM $M_B$ establishes that A is reducible to B.

10.2 Another proof that $A_{TM}$ is not decidable

Recall $A_{TM} = \{ < M, w > \mid w \in L(M) \}$. Thus $\overline{A_{TM}} = \{ < M, w > \mid w \notin L(M) \}$. Recall $A_D = \{ w \mid w \notin L(M) \}$.

Proof: Assume that $A_{TM}$ is decidable. Then $\overline{A_{TM}}$ must be decidable. Let $\overline{M}$ be the TM that decides $\overline{A_{TM}}$. We will construct a TM $M_D$ that would decide $A_D$, an undecidable language. $M_D$ works as follows: For input $w_i$ (the ith binary string in the lexicographic sequence of all binary strings), it first makes a string $w_i11w_i$ and then feed it to $\overline{M}$. We notice that $w_i11w_i \in L(M)$ iff $w_i11w_i \notin \overline{A_{TM}}$ iff $w_i11w_i \notin A_{TM}$ iff $w_i \notin L(M_i)$ (recall that $M_i$ is the TM with code $w_i$) iff $w_i \in L(M_D)$. So $M_D$ accepts $w_i$ iff $\overline{M}$ accepts $w_i11w_i$.

We just proved that $A_D$ is reducible to $\overline{A_{TM}}$.

10.3 The halting problem

Reading: Sipser 5.1 (pp. 188-189)

Let $HALT_{TM} = \{ < M, w > \mid M$ is a TM and $M$ halts on string $w \}$. $HALT_{TM}$ is Turing-recognizable since it can be recognized by TM $U$.

$HALT_{TM}$ is not Turing-decidable.

Proof: We will reduce $A_{TM}$ to $HALT_{TM}$. Assume TM $R$ decides $HALT_{TM}$. We construct TM $S$ that decides $A_{TM}$ as follows: On input $< M, w >$ where $M$ is a TM and $w$ is a string, $S$ first run TM $R$ on $< M, w >$, if $R$ rejects, rejects. If $R$ accepts, simulate $M$ on $w$ until it halts. If $M$ accepts, accept; if $M$ rejects, reject.

10.4 Undecidable problems about Turing machines

Reading: Sipser 5.1 (pp. 189-192)

- The following problems about Turing machines are not decidable:
  - Whether $L(M) = \emptyset$ for any TM $M$. (See proofs below.)
  - Whether $L(M_1) = L(M_2)$ for any two TMs $M_1$ and $M_2$.
  - Whether $L(M)$ is finite for any TM $M$.
  - Whether $\epsilon \in L(M)$ for any TM $M$.
  - Whether $L(M) = \Sigma^*$ for any TM $M$.
- $E_{TM} = \{ < M > \mid M$ is a TM and $L(M) = \emptyset \}$ is undecidable.
  Proof: Reduce $A_{TM}$ to $E_{TM}$. Assume that $E_{TM}$ is decidable. Let $R$ be the TM that decides $E_{TM}$. We use $R$ to construct TM $S$ that decides $A_{TM}$ as follows: On input $< M, w >$,
  - Construct TM $M_1$ which on input $x$, rejects if $x \neq w$ and simulates $M$ on $w$ if $x = w$.
  - Run $R$ on $< M_1 >$.
  - If $R$ accepts, reject and if $R$ rejects, accept.
- $NE_{TM} = \{ < M > \mid M$ is a TM and $L(M) \neq \emptyset \}$ is Turing-recognizable but not decidable.
  Proof: To prove that $NE_{TM}$ is Turing-recognizable, we design a TM $M_{NE}$ to recognize $NE_{TM}$. On input $< M >$,
– $M_{NE}$ systematically generates strings $w$: $\varepsilon, 0, 1, 00, 01, \ldots$ and use the universal TM $U$ to test whether $M$ accepts $w$. (What if $M$ never halts on $w$? Run $M$ on $w_1, \ldots, w_i$ for $i$ steps for $i = 1, \ldots$)

– If $M$ accepts some $w$, then $M_{NE}$ accepts its own input $M$.

We next prove that $NE_{TM}$ is not decidable. Assume that there is a TM $M_{NE}$ that decides $NE_{TM}$, i.e., TM $M_{NE}$ determines whether $L(M) \neq \emptyset$ for any TM $M$. We will use $M_{NE}$ to construct a TM $M_\mu$ that would decides the undecidable $A_{TM}$. On input $<M, w>$,

– $M_\mu$ constructs a new TM $M'$, which rejects if its input is not $w$ and mimics $M$ if its input is $w$.

– $M'$ is then fed to $M_{NE}$.

– $M_{NE}$ accepts its input $M'$ iff $L(M') \neq \emptyset$ iff $M$ accepts $w$.

• $E_{TM}$ is not Turing-recognizable.

• Rice’s Theorem: Every nontrivial property of the Turing-recognizable languages is undecidable.

### 10.5 Other undecidable problems

*Reading: Sipser 5.2 (pp. 199-205)*

• Post’s correspondence problem is undecidable.

  We formulate the Post’s Correspondence Problem as a puzzle.

  **Post’s Correspondence Problem (PCP)**

  **INSTANCE:** $P = \{ \frac{a}{b_1}, \frac{a}{b_2}, \ldots, \frac{a}{b_k} \}$, where $t_1, t_2, \ldots, t_k$ and $b_1, b_2, \ldots, b_k$ are strings over alphabet $\Sigma$. ($P$ can be regarded as a collection of dominos, each containing two strings, with one stacked on top of the other.)

  **QUESTION:** Does $P$ contain a match, i.e., $i_1, i_2, \ldots, i_l \in \{1, 2, \ldots, k\}$ with $l \geq 1$ such that $t_{i_1}t_{i_2}\ldots t_{i_l} = b_{i_1}b_{i_2}\ldots b_{i_l}$?

  Equivalently, defined as a language, we have $L_{PCP} = \{ <P> | \text{$P$ is an instance of PCP with a match} \}$.

  For example, for $P_1 = \{ \frac{b}{ca}, \frac{a}{db}, \frac{abc}{e} \}$, sequence 2, 1, 3, 2, 4 indicates a match. For $P_2 = \{ \frac{abc}{da}, \frac{ca}{db}, \frac{acc}{eb} \}$, there is no match.

• Any nontrivial property that involves what a program does is undecidable. For example, whether a program prints a certain message, whether it terminates, or whether it calls a certain function.

• It is undecidable whether a CFG is ambiguous.

• Let $G_1$ and $G_2$ be CFG’s and let $R$ be a regular expression. It is undecidable whether

  – $L(G_1) \cap L(G_2) = \emptyset$.

  – $L(G_1) = L(G_2)$.

  – $L(G_1) = L(R)$.

  – $L(G_1) = \Sigma^*$.

  – $L(G_1) \subseteq L(G_2)$.

  – $L(R) \subseteq L(G_1)$.