Sample Solution Keys to Homework 9

5.17; 5.18; 5.22; 5.23; 5.32; 5.34; 5.35.

Problem 5.17

Let (B, T) be an instance of PCP, where B = [b_1, b_2, …, b_n] and T = [t_1, t_2, …, t_n]. If the alphabet contains a single symbol, say 1, there are only two cases where (S, T) has no solutions:

Case 1: $|b_i| > |b_i|$ for $1 \le i \le n$

Case 2: $|s_i| > |t_i|$ for $1 \le i \le n$

For other cases, either we have $|b_i| = |t_i|$ for some i and this is a trivial solution, or there exist i and j such that $|b_i| = a > |t_i| = b$ and $|b_j| = c < |t_j| = d$, a solution can be found by choosing x copies of (b_i, t_i) and y copies of (b_j, t_j), where x = c – d and y = a – b.

Problem 5.18

We encode every symbol of Σ by a binary string of the same length, then the original PCP has a solution iff the binary PCP has a solution.

Problem 5.22

If A is Turing recognizable, then there exists a Turing machine M such that L(M) = A. The mapping reduction f is defined by f(w) = <M,w>, then $w \in A$ iff $\langle M,w\rangle \in A_{TM}$.

If $A \le_m A_{TM}$, then A is Turing recognizable by Theorem 5.28 because A_{TM} is Turing recognizable.

Problem 5.23

Show that A is decidable iff $A \le_m 0^*1^*$.

($\Rightarrow$) If $A \le_m 0^*1^*$, then A is decidable because 0^*1^* is a decidable language by Theorem 5.22.

($\Leftarrow$) If A is decidable, then there exists some TM R that decides A. That is, R would receive an input w and accept if w is in A, reject if w is not in A. To show $A \le_m 0^*1^*$, we design a TM Q that does the following: On input w, run R on w. If R accepts, outputs 01; otherwise, outputs 10. It is easy to check that:

$$w \in A \Leftrightarrow \text{output of } Q \in 0^*1^*$$

Thus, we obtain a mapping reduction of A to 0^*1^*.

Problem 5.32

(a) If $OVERLAP_{CFG}$ is decidable and S is its decider, then we may use S to solve PCP. For any instance (B, T) of PCP, where B = [b_1, b_2, …, b_n] and T = [t_1, t_2, …, t_n], we define two CFG G and H as follows:

G = ({b}, Σ', B, { $B \to s_iBi \mid 1 \le i \le n$ } $\cup$ { $B \to s_i\$i$}) and H = ({T}, Σ', T, { $T \to t_iTi \mid 1 \le i \le n$ } $\cup$ { $T \to t_i\$i$}), where $\Sigma' = \Sigma \cup$ { \$, 1, 2, …, n }. Then $L(G) \cap L(H) \neq \varnothing$ iff PCP has a solution, because the strings in $L(G) \cap L(H)$ are the solutions of PCP.

(b) If *PREFIX-FREE*$_{CFG}$ is decidable and S is its decider, we may use S to solve PCP. Let G = ({ S, T, B}, Σ, S, {S $\rightarrow$T|B} $\cup$P) be the CFG defined in Problem 5.21, replacing S $\rightarrow$T|B by S $\rightarrow$ T#|B##, we obtain G' = ({S, T, B}, $\Sigma\cup${#}, S, {S $\rightarrow$T#|B##} $\cup$P), then G' is prefix-free iff PCP has no solution. That is, PCP has a solution iff there exists w such that T $\Rightarrow$* w and B $\Rightarrow$* w, iff in G', we have S $\Rightarrow$T#$\Rightarrow$* w# and S $\Rightarrow$B##$\Rightarrow$* w##, that is, S is not prefix-free because w# is a prefix of w##.

Problem 5.34

Let X = { <M, w> | M is a single-tape TM that never modifies the portion of the tape that contains the input w }

We will show that $A_{TM} \leq_m$ X. The reduction f is computed as follows:

f = "On input <M, w>, where M is a TM

1. Construct M_0 as follows:
   M_0 = "On input x
   1. Move the tape head past x and write the character $ and w
   2. Simulate on w in the space after $;
   3. If M accepts w, move to the left of $ and write anything over x"
2. Output <M_0, w>"

Claim: <M, w>$\in A_{TM}$ iff <M_0, w > $\in$ X.

Proof: If M accepts w, then at line 3 of M_0, M_0 modifies the input string. If M rejects or loops on w, then M0 will reject or loop but never change the input string.

Since A_{TM} is undecidable and $A_{TM} \leq_m$ X, X cannot be decidable.

Problem 5.35

Say that a variable A in CFG G is ***necessary*** if it appears in every derivation of some string $w \in G$. Let $NECESSARY_{CFG} = \{\langle G, A\rangle \mid A \text{ is a necessary variable in } G\}$.

a. Show that $NECESSARY_{CFG}$ is Turing-recognizable.

Let M be a TM that behaves as follows:

On input $\langle G, A\rangle$:

1. Construct CFG $G\backslash A$ by removing the nonterminal A and any production that mentions it from G
2. Enumerating strings *w* generated by G. For each such string, simulate a decider for CFG A to test whether *w* is generated by $G\backslash A$. If *w* is not generated by $G\backslash A$ then *accept*, otherwise continue the search.
3. On other input, *reject*.

The language $L(G\backslash A)$ consists of all and only those string $w \in L(G)$ that have derivations that do not use A. If $w \in L(G)$ and *w* is not in $L(G\backslash A)$, then A is necessary for G to generate $w \in L(G)$. Eventually,

M will identify such a w and will accept. On the other hand, if A is not necessary for G, then $L(G) = L(G\backslash A)$ and M will loop. Thus, M recognizes $NECESSARY_{CFG}$.

Note that it is not possible to prove this part by enumerating all strings $w \in L(G)$ and checking all derivations of each such string to see whether or not A is used, because in general a string $w \in L(G)$ can have infinitely many derivations. So the checking procedure might never terminate.

b. Show that $NECESSARY_{CFG}$ is undecidable.

We will show $ALL_{CFG} \leq_m \overline{NECESSARY_{CFG}}$.
The reduction f is computed as follows:
On input $\langle G \rangle$:

1. Construct G_0 by adding to G a new nonterminal A, together with productions: $S \to A, A \to \epsilon$, and $A \to aA$ for each $a \in \Sigma$.
2. Output $\langle G_0, A \rangle$.

Note that the grammar G_0 constructed by f is always such that $L(G_0) = \Sigma^*$. Thus, if $L(G) = \Sigma^*$, then A is not necessary for G_0, because every string $w \in \Sigma^*$ can already be derived from G, hence also from G_0 by a derivation not using A. Also, if $L(G) \neq \Sigma^*$, then A is necessary for G_0. Because if $w \notin L(G)$ the w can only be derived from G_0 by a derivation that uses A. To summarize, if $\langle G \rangle \in ALL_{CFG}$, then $\langle G_0, A \rangle \notin NECESSARY_{CFG}$, and if $\langle G \rangle \notin ALL_{CFG}$, then $\langle G_0, A \rangle \in NECESSARY_{CFG}$. Hence, f is a reduction of ALL_{CFG} to $\overline{NECESSARY_{CFG}}$, as claimed.