SOLUTIONS: AUTOMATA AND FORMAL LANGUAGES

Instructions: You are expected to *sketch* the main ideas in your solutions, but be very brief and avoid unnecessary detail. You are permitted to invoke any result proved in the Hopcroft-Ullman book provided you include the appropriate citation.

1. (9 points) Consider the following context-free grammar G over the alphabet $\Sigma = \{a, b\}$.

$$\begin{aligned} S &\rightarrow aB \mid Ab \mid ab \\ A &\rightarrow aS \\ B &\rightarrow Sb \end{aligned}$$

(a) [3 points] Give a succinct description of $L(G)$.

(b) [4 points] Show that G is an ambiguous grammar.

(c) [2 points] What productions should you *delete* from G to render it unambiguous *without changing its language?*

Solution:

(a) $L(G) = \{a^n b^n \mid n \geq 1\}$.

(b) The following are two distinct leftmost derivations for the string *aabb*, implying that the grammar must be ambiguous.

$$S \overset{lm}{\Longrightarrow} aB \overset{lm}{\Longrightarrow} aSb \overset{lm}{\Longrightarrow} aabb$$

$$S \overset{lm}{\Longrightarrow} Ab \overset{lm}{\Longrightarrow} aSb \overset{lm}{\Longrightarrow} aabb$$

(c) Deleting the variable A or the variable B, as well as all related productions, gives an unambiguous grammar with the same language.

2. (12 points) Let $<M>$ denote any natural encoding of a Turing machine M. Consider the following decision problem:

$L_O = \{<M> \mid \exists$ odd numbers p, q such that $L(M)$ has strings of length both p and $q\}$

Prove that L_O is undecidable by using a reduction from the halting problem (which is known to be undecidable). Recall that the halting problem corresponds to the language $L_H = \{<M, w> \mid$ Turing machine M halts on input $w\}$.

Solution: We use the following reduction from L_H to L_O to obtain the proof of undecidability. Given a Turing machine M and input w, construct a Turing machine $\widehat{M}$ which behaves as follows on being given input $\widehat{w}$.

(a) $\widehat{M}$ simulates the behavior of M on input w, and

(b) $\widehat{M}$ accepts its input $\widehat{w}$ if M halts on w.

To show that the above reduction works, we need to prove that:

(a) There is an algorithm that computes $\widehat{M}$ given M, w.

(b) $<M, w> \in L_H$ if and only if $<\widehat{M}> \in L_O$.

The proof proceeds as follows.

(a) Given w and a description of M, it is easy to construct a Turing machine $\widehat{M}$ that simulates the computation of M on input w, using the same ideas used in the construction of the universal Turing machine (see Theorem 8.4 in the textbook).

(b) Let $\widehat{w}$ be any input string. Then,

$\widehat{M}$ accepts $\widehat{w}$
$\Leftrightarrow$ the simulation of M on w halts
$\Leftrightarrow$ $<M, w> \in L_H$.

Consequently, if $<M, w> \in L_H$ then $\widehat{M}$ accepts *all* strings and so must belong to L_O. Conversely, if $<M, w> \notin L_H$ then $\widehat{M}$ does not accept *any* string at all and so cannot belong to L_O.

3. (9 points) Prove that the following decision version of the following 3-PATH problem is NP-complete.

INSTANCE: An undirected graph $G(V, E)$.
QUESTION: Is there a collection of 3 vertex-disjoint paths in G that cover all the vertices?

Informally, the answer is YES if there exist 3 paths in the graph such that each vertex is contained in *exactly* one of these paths.

(Hint: You may assume that the Hamiltonian Path problem is NP-complete. This is the problem of deciding whether a given graph has a path containing each vertex exactly once.)

Solution: We first verify that the 3-PATH problem is in NP. A polynomial time algorithm can easily "guess" three disjoint sequences of vertex labels, and then check that the following two conditions are satisfied: each vertex appears in exactly one of these three sequences; and, each sequence corresponds to a path in the graph G.

To establish the NP-hardness of 3-PATH, we provide a polynomial time reduction from the HAMILTONIAN PATH problem, which is known to be NP-complete. An instance of the HAMILTONIAN PATH problem is some graph H and the goal is to check whether it has a path containing all the vertices. We transform this to the 3-PATH problem by producing a graph G which consists of 3 disjoint and disconnected copies of the graph H. We now need to show that G has a 3-path cover if and only if H is hamiltonian.

If H is hamiltonian, then the three hamiltonian paths in the three copies of H in G will constitute a 3-path cover for G. Conversely, suppose that G has a 3-path cover. Since G consists of 3 disconnected copies of H, the 3 paths must lie in distinct copies. Clearly, each path must be a 1-path cover, or a hamiltonian path, for the copy of H in which it lies. Thus, H must be hamiltonian.