CS5371 Theory of Computation

Homework 3 (Suggested Solution)

1. **Ans.** The language $\{0^n 1^n 2^n \mid n \geq 1\}$ is not context-free, so that it cannot be recognized by a 1-PDA. However, we can easily design a 2-PDA to recognize this language as follows (Let S_1 and S_2 be the two stacks in the 2-PDA):

   1. For each 0 it reads, push a 0 in S_1 and push a 0 in S_2
   2. For each 1 it reads, pop a 0 from S_1
   3. For each 2 it reads, pop a 0 from S_2
   4. If at any step, we discover the input string is not in correct order (e.g., a 0 is read after 1 is read), we reject the input string
   5. If S_1 and S_2 become just empty at the end, we accept the input string

   Thus, we have found a language that can be recognized by some 2-PDA but not by any 1-PDA. On the other hand, if a language can be recognized by a 1-PDA, it must be recognized by some 2-PDA. Therefore, 2-PDAs are more powerful than 1-PDAs.

2. **Ans.** If a language L is decidable, there exists a decider D that decides L. Then, we can construct an enumerator E that enumerates the strings of L in the desired ordering (shorter string first, then lexicographical order) as follows:

   E = "On any input,
   1. Ignore the input
   2. For $k = 1, 2, \ldots$
      i. Run D on the kth string in Σ^*, according to the desired ordering
      ii. If D accepts, print the string"

   Conversely, if some enumerator E enumerates the strings of L in the desired ordering, then either L is a finite set so that it is decidable, or if L is an infinite set, we can construct a TM D based on E as follows:

   D = "On input w,
   1. Run E
      i. For every string s printed by E, if $s = w$, accept w
      ii. Else if $s < w$ in the desired ordering, continue
      iii. Else if $s > w$, reject w"

   Since at most a finite number of strings of L are smaller than w in the desired ordering, so after a finite number of strings are printed by E, we can decide if w is in L or not. So, D runs in finite steps and is thus a decider.

3. Let $S = \{\langle M \rangle \mid M$ is a DFA that accepts w whenever it accepts the reverse of $w\}$.

   (a) **Ans.** An example of a DFA in S: A DFA that accepts all strings.

   (b) **Ans.** To show S is decidable, we construct a decider D for S as follows (Let C be a TM that decides EQ_{DFA}):

$D =$ "On input $\langle M \rangle$,
1. Construct an NFA M' such that $L(M') = \{w^R \mid w \in L(M)\}$
2. Convert M' into an equivalent DFA M''
3. Use C to compare $L(M'')$ and $L(M)$
4. If $L(M'') = L(M)$, accept. Else, reject.

In the above TM, Step 1 can be done by converting M into M' in finite steps. The idea is to (i) reverse the directions of all transition arrows in M, (ii) create a new state q' in M', and connects q' to each original final states of M with ε-transitions, and (iii) make the original start state of M a final state of M'. It is easy to check that $L(M') = \{w^R \mid w \in L(M)\}$.

Also, both Step 2 and Step 3 can be done in finite steps, as we learnt from the lectures (See Notes 4 pages 13–15, and Notes 12 pages 16–17). So, D runs in finite steps and is thus a decider.

4. **Ans.** Let $PAL_{DFA} = \{\langle M \rangle \mid M \text{ is a DFA that accepts some palindrome}\}$. To show PAL_{DFA} is decidable, we construct a decider D for PAL_{DFA} as follows (Let K be a TM that decides E_{CFG}):

$D =$ "On input $\langle M \rangle$,
1. Construct a PDA P such that $L(P) = \{w \mid w \text{ is a palindrome}\}$
2. Construct a PDA P' such that $L(P') = L(P) \cap L(M)$
3. Convert P' into an equivalent CFG G
4. Use K to check if $L(G)$ is empty.
5. If $L(G)$ is empty, reject. Else, accept.

In the above TM, Step 1 can be done in finite steps. Step 2 is based on Prob 2.18 and can be done in finite steps. Step 3 is the conversion of PDA into an equivalent CFG, which can be done in finite steps (See Notes 8, pages 28–34). Step 4 is done in finite steps, because the decider K can check whether the language of a CFG is empty (For the existence of K, see Notes 12, pages 20–21). In summary, D runs in finite steps for any input, and is thus a decider.

5. **Ans.** Let C be the language

$$C_{CFG} = \{\langle G, k \rangle \mid G \text{ is a CFG and } L(G) \text{ contains exactly } k \text{ strings where } k \geq 0 \text{ or } k = \infty\}$$

In this problem, we are given a decider D that decides if the language of a CFG is infinite. Then, we can show that C is decidable, by finding a corresponding decider F as follows:

$F =$ "On input $\langle G, k \rangle$,
1. Use D to check if $L(G)$ is an infinite set.
2. There are four cases:
   i. If yes, and $k = \infty$, accept.
   ii. If yes, but $k \neq \infty$, reject.
   iii. If no, but $k = \infty$, reject.
   iv. If no, and $k \neq \infty$, continue.
3. Compute the pumping length p for the grammar G.

4. Set *count* to be 0.
5. For $x = 1, 2, \ldots, p$
   i. For all string s with length $= x$,
      Check if s can be generated by G; if so, increment *count* by 1.
6. If *count* $= k$, accept. Else, reject.

In the above TM F, Steps 1–2 correctly answer the case where $L(G)$ is an infinite set, or $k = \infty$. So, after Step 2, we only deal with a grammar G whose language is a finite set, and our task is to check whether the language size is exactly k. To do so, the loop in Step 5 counts all string that can be generated by G, whose length is at most the pumping length p. Because we know that $L(G)$ is finite, we are sure that no strings of $L(G)$ can be longer than p. In other words, the value *count* correctly computes the exact number of strings in $L(G)$. So, Step 6 can check correctly answer the case when $L(G)$ is finite, and k is finite.

Finally, it is easy to check that each step runs in finite number of steps. Thus, F is a decider, so that C is a decidable language.