Theory of Computation, Homework 3  Sample Solution

3.8
b.) The following machine $M$ will do:

$M = \text{"On input string $\Psi$:

1. Scan the tape and mark the first 1 which has not been marked. If no unmarked 1 is found, go to step 5. Otherwise, move the head back to the front of the tape.
2. Scan the tape and mark the first 0 which has not been marked. If no unmarked 0 is found, reject.
3. Move on to mark the next unmarked 0. If no such unmarked 0 is found, reject.
4. Move the head back to the front of the tape and go to step 1.
5. Move the head back to the front of the tape. Scan the tape to see if there is any unmarked 0 found. If yes, reject. Otherwise, accept."

c.) This machine is identical to 3.8b) if we switch "reject" with "accept".

$M = \text{"On input string $\Psi$:

1. Scan the tape and mark the first 1 which has not been marked. If no unmarked 1 is found, go to step 5. Otherwise, move the head back to the front of the tape.
2. Scan the tape and mark the first 0 which has not been marked. If no unmarked 0 is found, accept.
3. Move on to mark the next unmarked 0. If no such unmarked 0 is found, accept.
4. Move the head back to the front of the tape and go to step 1.
5. Move the head back to the front of the tape. Scan the tape to see if there is any unmarked 0 found. If yes, accept. Otherwise, reject."}
3.12
Let $\mathcal{M}_L$ be a Turing machine with left reset and $\mathcal{M}$ be an ordinary Turing machine. If $\mathcal{M}_L$ can simulate all the operations that $\mathcal{M}$ can perform, $\mathcal{M}_L$ recognizes the class of Turing-recognizable language. Obviously, $\mathcal{M}_L$ can simulate $\mathcal{M}$ without problems. How $\mathcal{M}_L$ simulates $\delta(q, a) = (r, b, R)$ of $\mathcal{M}$ is described as follows.

1. overwrite $a$ with a marked $b$
2. reset to the left-hand end
3. shift the whole tape one cell to the right but keep the mark in the same position
4. reset to the left-hand end
5. scan right to find the marked symbol; the next move will treat the marked symbol as a normal symbol.

We assume the following during the shifting for Step 3:

a. We use states to remember the symbol to be shifted right.
b. The first symbol of the tape after the shift is a symbol not used by the original $M$.
c. If the current symbol is $c$ and the next symbol is the marked $b$, after the shifting, $c$ becomes marked but $b$ is not marked.
d. The shifting stops when we see a blank symbol.
3.15  
b.)  
Let $L_1$ and $L_2$ be two decidable languages and $M_1$ and $M_2$ be the corresponding TMs. The aim is to construct a TM $M_L$ based on $M_1$ and $M_2$ such that the concatenation $L = L_1L_2$ is also decidable. Since a given input concatenation $w$ of strings $x \in L_1$ and $y \in L_2$ has finite possible partitions, a nondeterministic TM is chosen to simplify the description.

NTM $M_L$ = 

1. Nondeterministically split $w$ into $x$ and $y$
2. run $M_1$ on $x$ and $M_2$ on $y$
3. accept if both $M_1$ accepts $x$ and $M_2$ accepts $y$; reject otherwise

Obviously, the NTM $M_L$ accepts an input $w$ if there exists a split of $w$ such that $M_1$ accepts $x$ and $M_2$ accepts $y$. Besides, $M_L$ eventually halts because $M_1$ and $M_2$ are both deciders. Therefore, $L = L_1L_2$ is decidable since there exists an NTM $M_L$ which decides $L$.

c.)  
Let $L$ be a decidable language and $M$ be the corresponding TM. The aim is to construct a TM $M_L$ based on $M$ such that $L^*$ is also decidable. Since a given input $w$ has finite possible combinations of strings $x_1x_2\cdots x_n$ where $x_i \in L$ and $i = 1, 2, \ldots, n$, a nondeterministic TM is chosen to simplify the description.

NTM $M_L$ = 

1. accept if $w = \varepsilon$
2. if $w \neq \varepsilon$, nondeterministically split $w$ into $x_1x_2\cdots x_n$, where $x_i$ is not empty.
3. run $M$ on $x_i$ for all $i$.
4. accept if $M$ accepts all $x_i$, $i = 1, 2, \ldots, n$; reject otherwise

Obviously, the NTM $M_L$ accepts an input $w$ if $w = \varepsilon$ or there exists a combination of $w$ such that $M$ accepts $x_1x_2\cdots x_n$. Besides, $M_L$ eventually halts because $M$ is a decider. Therefore, $L^*$ is decidable since there exists an NTM $M_L$ which decides $L^*$. 
d.) Let $L$ be a decidable language and $M$ be the corresponding TM. The aim is to construct a TM $M_L$ based on $M$ such that $L'$, the complement of $L$, is also decidable. The resulting TM $M_L$ is described as follows.

$$TM\; M_L = \text{"On input } w \text{"}$$
1. run $M$ on $w$
2. accept if $M$ rejects $w$
3. reject if $M$ accepts $w$

Obviously, $M_L$ accepts an input $w$ iff $M$ rejects $w$. Besides, $M_L$ eventually halts because $M$ is a decider. Therefore, $L'$ is decidable since there exists a TM $M_L$ which decides $L'$.

e.) Let $L_1$ and $L_2$ be two decidable languages and $M_1$ and $M_2$ be the corresponding TMs. The aim is to construct a TM $M_L$ based on $M_1$ and $M_2$ such that the intersection $L = L_1 \cap L_2$ is also decidable. The resulting TM $M_L$ is described as follows.

$$M_L = \text{"On input } w \text{"}$$
4. run $M_1$ and $M_2$ on $w$
1. accept if both $M_1$ and $M_2$ accepts $w$; reject otherwise

Obviously, the NTM $M_L$ accepts an input $w$ iff $w$ is accepted by $M_1$ and $M_2$. Besides, $M_L$ eventually halts because $M_1$ and $M_2$ are both deciders. Therefore, $L = L_1 \cap L_2$ is decidable since there exists a TM $M_L$ which decides $L$. 
b.) Let $L_1$ and $L_2$ be two Turing-recognizable languages and $M_1$ and $M_2$ be the corresponding TMs. The aim is to construct a TM $M_L$ based on $M_1$ and $M_2$ such that the concatenation $L = L_1 | L_2$ is also Turing-recognizable. Since a given input concatenation $w$ of strings $x \in L_1$ and $y \in L_2$ has finite possible partitions, a nondeterministic TM is chosen to simplify the description.

\[NTM \ M_L = \begin{array}{c}
\text{On input } w \\
1. \text{Nondeterministically split } w \text{ into } x \text{ and } y \\
2. \text{run } M_1 \text{ on input } x \\
3. \text{reject if } M_1 \text{ halts and rejects} \\
4. \text{run } M_2 \text{ on input } y \\
5. \text{accept if } M_2 \text{ accepts } y; \text{ reject if } M_2 \text{ halts and rejects}
\end{array}\]

Obviously, the NTM $M_L$ recognizes an input $w$ iff there exists a partition of $w$ such that $M_1$ accepts $x$ and $M_2$ accepts $y$. However, $M_L$ may loop forever on some input because $M_1$ and $M_2$ are not deciders. Therefore, $L = L_1 | L_2$ is Turing-recognizable since there exists an NTM $M_L$ which recognizes $L$.

c.) Let $L$ be a Turing-recognizable language and $M$ be the corresponding TM. The aim is to construct a TM $M_L$ based on $M$ such that $L^y$ is also Turing-recognizable. Since a given input $w$ has finite possible combinations of strings $x_1 x_2 \ldots x_n$ where $x_i \in L$, $i = 1, 2, \ldots, n$, a nondeterministic TM is chosen to simplify the description.

\[NTM \ M_L = \begin{array}{c}
\text{On input } w \\
1. \text{accept if } w = \varepsilon \\
2. \text{if } w \neq \varepsilon \text{ nondeterministically split } w \text{ into } x_1 x_2 \ldots x_n, \text{ where } x_i \text{ is not empty.} \\
3. \text{run } M \text{ on } x_i \text{ for all } i. \\
4. \text{accept if } M \text{ accepts all } x_i, \text{ } i = 1, 2, \ldots, n; \\
5. \text{reject if } M \text{ halts and rejects for any } x_i.
\end{array}\]

Obviously, the NTM $M_L$ recognizes an input $w$ iff $w = \varepsilon$ or there exists a combination of $w$ such that $M$ accepts $x_1 x_2 \ldots x_n$. However, $M_L$ may loop forever on some input because $M$ is not a decider. Therefore, $L^y$ is Turing-recognizable since there exists an NTM $M_L$ which recognizes $L^y$. 
d.)
Let \( L_1 \) and \( L_2 \) be two Turing-recognizable languages and \( M_1 \) and \( M_2 \) be the corresponding TMs. The aim is to construct a TM \( M_L \) based on \( M_1 \) and \( M_2 \) such that the intersection \( L = L_1 \cap L_2 \) is also Turing-recognizable. The resulting TM \( M_L \) is described as follows.

\[
M_L = \text{"On input } w \text{"
1. run } M_1 \text{ on } w
2. rejects if } M_1 \text{ halts and rejects
3. run } M_2 \text{ on } w
4. accept if } M_2 \text{ accepts } w; \text{ reject if } M_2 \text{ halts and rejects"
}

Obviously, the TM \( M_L \) recognizes an input \( w \) iff \( w \) is accepted by \( M_1 \) and then \( M_2 \). However, \( M_L \) may loop forever on some input because \( M_1 \) and \( M_2 \) both are not deciders. Therefore, \( L = L_1 \cap L_2 \) is Turing-recognizable since there exists a TM \( M_L \) which recognizes \( L \).

4.4
Since \( A_{\mathbf{CFC}} \) is just a special case of \( A_{\mathbf{CFG}} \), it is possible to adapt TM S for \( A_{\mathbf{CFC}} \) as follows.

TM S = "On input \( \{G, \varepsilon\} \), where \( G \) is a CFG and \( \varepsilon \) is an empty string:
1. Convert \( G \) to an equivalent grammar in Chomsky normal form.
2. If \( S \to \varepsilon \) is a production rule in Chomsky normal form, accept; if not, reject."

4.10
To decide \( \text{INFINITE}_{PDA} \) is to determine if there exist strings generated by \( M \) with lengths at least the pumping length \( p \). Let \( G \) be a CFG for \( L(M) \) and design a TM \( M_G \) that decides \( L(M) \). The construction of TM \( M_G \) is as follows, deciding \( \text{INFINITE}_{PDA} \):

TM \( M_G = \text{"On input: } <M> \text{"
1. convert } G \text{ to Chomsky normal form
2. calculate the pumping length } p = 2|V| \text{ from } G, \text{ where } |V| \text{ is the number of variables in } G
\text{ (in Chomsky normal form, each rule has 0 or two variables at the right, so the pumping length is } 2|V| \text{ that is computable)
3. accept if } G \text{ produces a string of length at least } p \text{ because the string can be pumped to generate infinitely other strings
4. otherwise, reject""}

For step 3, it is decidable since it is possible to construct a DFA \( D \) such that the regular language recognized by \( D \) is the set of strings of lengths at least \( p \). Let \( L \) be \( L(D) \cap L(M) \) which is a CFL from Problem 2.18a and \( G_L \) be the CFG of \( L \). Then TM \( R \) in Theorem 4.8 can decide \( E_G \). If TM \( R \) accepts, \( G \) produces no strings of lengths at least \( p \). If not, \( G \) produces a string of length at least \( p \).
4.12 If \( L(R) \subseteq L(S) \), \( L(R) = L(S) \cap L(R) \). Therefore, we can first construct two equivalent DFA \( D_R \) and \( D_S \) recognizing \( L(R) \) and \( L(S) \cap L(R) \) and then run TM \( F \) in theorem 4.5 to decide if the two DFA are equivalent.

**TM \( M_{RS} \):**

1. construct the equivalent NFA \( N_R \) and \( N_S \) for \( R \) and \( S \)
2. construct the equivalent DFA \( D_R \) and \( D_S \) for the \( N_R \) and \( N_S \)
3. construct a DFA \( D_{SR} \) accepting \( L(S) \cap L(R) \)
4. run TM \( F \) to decide if \( D_{SR} \) and \( D_R \) are equivalent
5. accept if TM \( F \) accepts; reject if TM \( F \) rejects

4.24 Let \( L_{PAL} = \{ x \mid x \text{ is a palindrome} \} \) be a CFL containing all the palindromes, \( L(M) \) be the regular language accepted by \( M \) and \( L = L_{PAL} \cap L(M) \). The goal is that \( PAL_{DFA} \) is decidable if the emptiness of \( L \) is also decidable. Since \( L = L_{PAL} \cap L(M) \) is a CFL from Problem 2.18, its emptiness can be decided by TM \( R \) in theorem 4.8. Let \( G_L \) be the CFG of \( L \).

**TM \( PAL_{DFA} \):**

1. Let \( L_{PAL} = \{ x \mid x \text{ is a palindrome} \} \) be a CFL containing all the palindromes
2. Let \( L(M) \) be the regular language accepted by \( M \)
3. derive the CFG for \( L = L_{PAL} \cap L(M) \)
4. run TM \( R \) to decide \( E_G \)
5. accept if TM \( R \) accepts; reject if TM \( R \) rejects.
Let us prove it by contradiction. Suppose that every decider is in $A$. Since $A$ is Turing-recognizable, $A$ is also enumerable. Let $M_i$ be the $i^{th}$ decider in $A$. We may construct the following decider $M_D$ as follows:

$M_D =$ On input $w$,
(1) decide the order number of $w$, i.e., $w = x_i$, the index of $w$ be $i$;
(2) accept $x_i$ if $M_i$ rejects; rejects if $M_i$ accepts.

Apparently, $M_D$ is a decider as (1)-(2) halts. $M_D$ is different from any $M_i$ in $A$, which is a contradiction to the assumption that every decider is in $A$.

To show that $M_D$ is different from any $M_i$ in $A$, let $S = \{x_1, x_2, x_3, \ldots\}$ be the list of all the strings in an canonical order of string (length than dictionary order). Then $M_D$ can be derived by applying the diagonalization method as illustrated by the following table.

The table below demonstrates an example $D$.

<table>
<thead>
<tr>
<th></th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_1$</td>
<td>reject</td>
<td>accept</td>
<td>accept</td>
<td>...</td>
</tr>
<tr>
<td>$M_2$</td>
<td>accept</td>
<td>accept</td>
<td>accept</td>
<td>...</td>
</tr>
<tr>
<td>$M_3$</td>
<td>reject</td>
<td>accept</td>
<td>reject</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$M_D$</td>
<td>accept</td>
<td>reject</td>
<td>accept</td>
<td>...</td>
</tr>
</tbody>
</table>

Obviously, $D$ is different from any language decided by $M_i$ whose description appears in $A$. 