1 Exercise 6.12 (c)

For the sake of contradiction, suppose that $L \cup F$ is a CFL. Let $F_1 = F - L$. Since $F$ is finite, then $F_1$ is finite as well, so it is also regular, which means that $F'_1$ - the complement of $F_1$ - is also regular. Observe that, $L = (L \cup F) \cap F'_1$ is therefore an intersection of a CFL and a regular language, and has to be a CFL itself, contradiction.
Exercise 7.4 (b)
Exercise 7.7
4 Exercise 7.17 (b)

5 Exercise 7.25
6 Exercise 7.29

Nondeterministically select a position at which to partition the input word \( x \) into to words: \( x = uv \), and put a \( \Delta \) in between (shifting \( v \) by one), obtaining \( \Delta u \Delta v \). Next, simulate \( T_2 \). If \( T_2 \) accepts, erase all of the cells it modified, and go to the beginning of the tape, obtaining \( \Delta u \) (if \( T_2 \) halts without accepting or crashes, then reject). Finally, simulate \( T_1 \), and accept iff \( T_1 \) accepts.

7 Exercise 6.16(b) [Extra Credit]

7.1 PDA Solution

You can solve this problem by taking a PDA \( M \) for \( L \) and converting it into a PDA \( M' \) for prefixes(\( L \)). To do so, make two copies of \( M \) and smoosh them together to become \( M' \), so then \( M' \) has twice as many states as \( M \). In the first copy of \( M \), change all the final states into non-final states. In the second copy of \( M \), change all the transitions into \( \Lambda \) transitions (so \( \sigma, \gamma/\alpha \) becomes \( \Lambda, \gamma/\alpha \)). Finally, draw a transition labeled \( \Lambda, \Lambda/\Lambda \) from each state in the first copy to its corresponding state in the second copy. The start state of \( M' \) should be the start state of the first copy.

7.2 CFG Solution

Let \( G \) be a context-free grammar accepting a language \( L \). I will make a grammar accepting prefixes(\( L \)).

Without loss of generality, we may assume that \( G \) is in Chomsky normal form and has no useless, unproductive symbols. The new grammar \( G' \) is defined as follows:

- \( G' \) has all the rules and symbols from \( G \), plus some new stuff defined below.

- If \( X \) is a non-terminal in \( G \), then \( X' \) is also a non-terminal in \( G' \). So \( G' \) has twice as many non-terminals as \( G \).

The idea is that \( X' \) is supposed to generate all prefixes of strings generated by \( X \).

- \( X' \rightarrow \Lambda \) is a production in \( G' \) (for each non-terminal \( X \) in \( G \)).
• If $X \to \sigma$ is a production in $G$, then $X' \to \sigma$ is also a production in $G'$.

• If $X \to YZ$ is a production in $G$, then $X' \to Y'$ and $X' \to YZ'$ are also productions in $G'$.

• $S'$ is the start symbol of $G'$ (where $S$ is the start symbol of $G$).

It remains to prove that $G'$ accepts prefixes($L$). To do it, I will show that each symbol $X'$ generates the prefixes of the strings generated by $X$. As usual, there are two parts:

**Part 1** (everything generated by $X'$ is a prefix). Let $x'$ be generated by the non-terminal symbol $X'$. I will prove (by induction on the parse tree for $x'$) that $x'$ must be a prefix of some string $x$ generated by $X$:

- If the root production is $X' \to \Lambda$, then $x' = \Lambda$, which is a prefix of everything. In particular, since $X$ is not unproductive, we can take $x$ to be any string generated by $X$ in this case.

- If the root production is $X' \to \sigma$, then $x' = \sigma$. Additionally, we know that $X \to \sigma$ must also be in the grammar, so we can take $x = \sigma$.

- If the root production is $X' \to Y'$, then $x' = y'$ for some string $y'$ generated by $Y'$. Our induction hypothesis says that $y'$ is a prefix of some string $y$ generated by $Y$. Additionally, we know that $X \to YZ$ must also be in the grammar (for some symbol $Z$). Since $Z$ is not unproductive, we know that $Z$ generates some string $z$. Now we can take $x = yz$; we can easily see that $x'$ is a prefix of $x$, and we can also see that $x$ is generated by $X$ using the rule $X \to YZ$.

- If the root production is $X' \to YZ'$, then $x' = yz'$ for some string $y$ generated by $Y$ and some string $z'$ generated by $Z'$. Our induction hypothesis says that $z'$ is a prefix of some string $z$ generated by $Z$. Now we can take $x = yz$; again we can easily see that $x'$ is a prefix of $x$, and we can see that $x$ is generated by $X$ using the rule $X \to YZ$.

**Part 2** (every prefix is generated by $X'$). Let $x'$ be a prefix of a string $x$ generated by the symbol $X$. I will prove (by induction on the parse tree for $x$) that $x'$ is generated by $X'$. 

6
• If the root production is $X \rightarrow \sigma$, then $x = \sigma$. In this case, either $x' = \Lambda$ or $x' = \sigma$, so $x'$ can be generated using either $X' \rightarrow \Lambda$ or $X' \rightarrow \sigma$.

• If the root production is $X \rightarrow YZ$, then $x = yz$ for some strings $y$ and $z$ generated by $Y$ and $Z$ respectively. Our induction hypothesis says that every prefix of $y$ is generated by $Y'$ and that every prefix of $z$ is generated by $Z'$. Now, since $x'$ is a prefix of $x$, we know that either $x'$ is a prefix of $y$ or $x' = yz'$ for some prefix $z'$ of $z$. Therefore, we can generate $x'$ using either $X' \rightarrow Y'$ or $X' \rightarrow YZ'$. 