Chapter 6

Context-free Languages and Pushdown Automata

6.1. (a) The language of even-length palindromes over $\{a, b\}$.

(b) The language of odd-length palindromes over $\{a, b\}$.

(c) The set of even-length strings x over $\{a, b\}$ such that x^r is obtained from x by reversing a's and b's.

(d) The set of strings over $\{a, b\}$ that are not palindromes but could be made into palindromes by changing one symbol from a to b or vice versa.

(e) and (f) The set $\{a, b\}^*\{a\}$

(g) The set $\{ba\}^*\{b\}$

(h) The set of even-length strings in $\{a, b\}^*$

6.2. $S \to TS \mid \Lambda \qquad T \to a \mid bS \mid cUd \qquad U \to AeU \mid A \qquad A \to f \mid gBi \qquad B \to hjB \mid h$

6.3. One CFG is the one with start symbol P and productions

$$P \to pI; \mid pI(FF_1) \qquad F_1 \to ;FF_1 \mid \Lambda \qquad F \to G \mid vG \mid fG \mid pII_1$$
$$I_1 \to ,II_1 \mid \Lambda \qquad G \to II_1 : I \qquad I \to L \mid LL_2 \qquad L_2 \to L_1L_2 \mid \Lambda$$
$$L_1 \to L \mid D \qquad L \to a \mid b \qquad D \to 0 \mid 1$$

6.4. (a) $S \to aSa \mid aSb \mid bSa \mid bSb \mid a$

(b) $S \to aSa \mid aSb \mid bSa \mid bSb \mid aa \mid bb$

(c) $S \to aTa \mid bUb \qquad T \to aTa \mid aTb \mid bTa \mid bTb \mid a \qquad U \to aUa \mid aUb \mid bUa \mid bUb \mid b$

6.5. (a) If this CFG generated L, then every nonnull element of L would be either of the form $x01y$ or of the form $x10y$, where in either case $x, y \in L$. This is not the case, however, as the string 0011 illustrates.

(b) If this CFG generated L, then every nonnull element of L would have one of the forms $0y1$, $1y0$, $01y$, $10y$, $y01$, or $y10$, where y is an element of L. The string 00111100, however, does not fit any of these patterns.

6.6. No. The string $aabbcc$ cannot be obtained from this CFG. If it could, the first production in any derivation would have to be $S \to aSbScS$, because all the others would result in b's appearing before a's or c's appearing before a's or c's appearing before b's. The only way $aabbcc$ could be obtained from $aSbScS$ would be for the first S to eventually be replaced by either a or ab, and neither a nor ab has equal numbers of a's, b's, and c's.

6.7. We give an answer for the alphabet $\Sigma = \{a, b\}$, and it can easily be modified for the more general case.

(a) $S \to (S+S) \mid (SS) \mid (S^*) \mid a \mid b \mid \lambda$
(b) $S \to S+S \mid SS \mid S^* \mid (S) \mid a \mid b \mid \lambda$

6.8. None of these works. A counterexample to (a) is $S_1 \to aS_1b \mid \Lambda \qquad S_2 \to b$. The string abb is not in $L(G_1) \cup L(G_2)$, but the grammar obtained by the construction allows it. A counterexample to (b) is $S_1 \to a \qquad S_2 \to b$. The string abb is not in $L(G_1)L(G_2)$, but the construction allows the derivation $S_1 \Rightarrow S_1S_2 \Rightarrow S_1S_2S_2 \Rightarrow^* abb$. A counterexample to (c) is $S_1 \to aS_1a \mid b$. The string $abba$ is not in $L(G_1)^*$, but the grammar obtained by the construction allows it.

6.9. (a) $S \to aSc \mid T \qquad T \to aTb \mid \Lambda$
(b) $S \to AB \qquad A \to aAb \mid \Lambda \qquad B \to bBc \mid \Lambda$
(c) $S \to AT \mid UC \qquad A \to aA \mid \Lambda \qquad C \to cC \mid \Lambda \qquad T \to bTc \mid \Lambda \qquad U \to aUb \mid \Lambda$
(d) $S \to TC \mid U \qquad C \to cC \mid \Lambda \qquad T \to aTb \mid \Lambda \qquad U \to aUc \mid B \qquad B \to bB \mid \Lambda$
(e) $S \to TBC \mid AU \qquad T \to aTb \mid \Lambda \qquad B \to bB \mid b \qquad C \to cC \mid \Lambda$
$A \to aA \mid a \qquad U \to aUc \mid V \qquad V \to bV \mid \Lambda$
(f) $S \to aaSb \mid aSb \mid Sb \mid \Lambda$
(g) $S \to aS_1b \mid S_1b \qquad S_1 \to aaS_1b \mid aS_1b \mid S_1b \mid \Lambda$
(h) $S \to aSb \mid aSbb \mid \Lambda$

6.10 (a) Strings over $\{a, b\}$ containing an even number of a's and an odd number of b's.
(b) One description is the language corresponding to the regular expression $(b+aa^*bb)^*(\Lambda+aa^*)b$.

6.11. Statement (a) obviously implies statement (b). Suppose L can be generated by a grammar G in which all productions have either the form $A \to xB$ or the form $A \to x$, where $x \in \Sigma^+$. Construct the grammar G_1 as follows. All the productions $A \to aB$ or $A \to a$ in G (where A, B are variables and $a \in \Sigma$) are also in G_1. For each production $A \to xB$ in G with $|x| > 1$, say $x = a_1a_2 \ldots a_k$, put the productions $A \to a_1A_1$, $A_1 \to a_2A_2$, $\ldots$, $A_{k-2} \to a_{k-1}A_{k-1}$, $A_{k-1} \to a_kB$ in G_1. Here the variables A_1, A_2, $\ldots$, A_{k-1} are new variables that are used nowhere else. Similarly, for each production $A \to x$ in G with $x = a_1a_2 \ldots a_k$ where $k > 1$, put the productions $A \to a_1A_1$, $A_1 \to a_2A_2$, $\ldots$, $A_{k-2} \to a_{k-1}A_{k-1}$, $A_{k-1} \to a_k$ in G_1. Here also, these new variables are specific to this production and are used nowhere else. G_1 is clearly regular, and it is not hard to see that $L(G_1) = L$.

To show that (a) and (c) are equivalent, we observe that the grammar obtained from the one in (c) by reversing the right sides of the productions has the property in (b) and generates the language L^r. Therefore, L can be generated by a grammar of the type in (c) if and only if L^r is regular. But this is true if and only if L is regular.

6.13. $A \to aB \qquad B \to aB \mid bC \mid b \qquad C \to cC \mid aB \mid b$

6.16. The "only if" part is trivial. The converse is false: the CFG with productions

$S \to aT \mid \Lambda$ and $T \to Sb$ is equivalent to the one with productions $S \to aSb \mid \Lambda$, and this grammar generates the nonregular language $\{a^n b^n \mid n \geq 0\}$.

6.17. If L is regular, then $L - \{\Lambda\}$ can be generated by a grammar G that is not self-embedding, by Theorem 6.2. If $\Lambda \in L$, we can generate L by the grammar G_1, whose start symbol is S (not in G) and whose productions are those in G as well as the two additional productions $S \to \Lambda \mid S_1$ (where S_1 is the start symbol of G). G_1 is obviously not self-embedding.

6.18. (a) A regular expression is $(aa + aab + aba + abab)^*(a + ab)$, and a regular grammar is

$$S \to aT \mid a \qquad T \to aU \mid bV \mid b \qquad U \to aT \mid a \mid bS$$
$$V \to aW \qquad W \to aT \mid a \mid bS$$

(b) A regular expression is $(a + b)^*ab(ab + b)^+$. A regular grammar is

$$A \to aA \mid bA \mid aB \qquad B \to bC \qquad C \to aD \mid bE$$
$$D \to bE \qquad E \to aD \mid bE \mid \Lambda$$

(c) A regular expression is $(ab + ba)^*(ab + aab)$, and a regular grammar is

$$S \to aT \mid bU \qquad T \to aV \mid bW \mid b$$
$$U \to aS \qquad W \to aT \mid bU \qquad V \to bX \mid b$$

(d) Although A generates the language of odd-length palindromes, which is not regular, every nonnull string begins with an odd-length palindrome. Therefore, the language is $\{a, b\}^+$, and a regular grammar is $S \to aS \mid bS \mid a \mid b$.

(e) A generates all strings with an odd number of a's. S generates the strings that have either a positive even number of a's or no a's—in other words, all strings with an even number of a's. A regular expression is $(b^*ab^*a)^*b^*$, and a regular grammar is the one with productions

$$A \to bA \mid aB \mid \Lambda \qquad B \to bB \mid aA$$

6.19. (c) 5 (d) 14 (e) 1

6.20. Let the productions be $S \to ABA \qquad A \to \Lambda \qquad B \to b$. Then the string B has the derivation $S \Rightarrow ABA \Rightarrow BA \Rightarrow B$, but B has neither a leftmost derivation nor a rightmost derivation.

6.21. (a) If $a = 3$, the resulting value of x is 3, and if $a = 1$, the resulting value of x is 1.

(b) If $a = 3$, the resulting value of x is 1, and if $a = 1$, the resulting value of x is 3.

6.22. The string $abaa$ has two leftmost derivations, one beginning $S \Rightarrow SbS$, the other beginning $S \Rightarrow Sa$.

6.23. False. The string ab has the two derivations $S \Rightarrow AB \Rightarrow aAB \Rightarrow aB \Rightarrow abB \Rightarrow ab$ and $S \Rightarrow AB \Rightarrow B \Rightarrow ab$. If x is a string derivable from AB, then for each way of writing $x = yz$, where $A \Rightarrow^* y$ and $B \Rightarrow^* z$, there is only one way of deriving y from A and only one way of deriving z from B. However, there is more than one choice for writing $x = yz$ in this way.

6.24. All the grammars except those in (f) and (g) are unambiguous. We prove this result for (c). We will show that for every $n \geq 0$, and every x with $|x| = n$, i) if $S \Rightarrow^* x$, x has only one leftmost derivation from S; and ii) if $A \Rightarrow^* x$, x has only one leftmost derivation from A.

For the basis step, it is clear that Λ can be derived only from the variable A, and it has only one derivation from A. Suppose that $k \geq 0$ and that any string of length k or less that can be derived from one of the two variables has only one leftmost derivation from that variable. Now suppose $|x| = k+1$. If $S \Rightarrow^* x$, then it is clear that $k+1 \geq 2$. The first step of a derivation of x from S is determined by the beginning and ending symbols of x. In the first case, where the first step is $S \Rightarrow aSa$, it follows that $x = aya$, where $S \Rightarrow^* y$. By the induction hypothesis, y has only one leftmost derivation from S, and it follows that x has only one leftmost derivation from S. The other three cases are similar.

If $A \Rightarrow^* x$, there are five cases: i) $x = aya$; ii) $x = byb$; iii) $x = a$; iv) $x = b$; v) $x = \Lambda$. In each of these cases, the first step in any derivation is determined. In the last three cases the first step is the only one, and in the first two the induction hypothesis allows us to conclude that x has only one leftmost derivation from A.

(f) Ambiguous. The string aaa has two leftmost derivations.

(g) Ambiguous. The string $babab$ has two leftmost derivations.

6.25. (a) (Example 6.3) Unambiguous. The proof is similar to part (c) of Exercise 6.24.

(b) (Example 6.9) Unambiguous. We will not present a detailed proof, but it is not hard to see that for any string x corresponding to the regular expression, x can be written uniquely as yz, where y corresponds to $(011 + 1)^*$ and z corresponds to $(01)^*$. This might not have been the case: for example, if the regular expression had been $(0101 + 1)^*(01)^*$, the string 0101 could match in two different ways, and the corresponding grammar would be ambiguous.

(c) (Example 6.11) Unambiguous. Again we only sketch the argument. If $x = 0^i1^j0^k$, where $j > i + k$, the first step of the derivation of x is $S \rightarrow ABC$. The string to be derived from A must be 0^i1^i, and this string has only one derivation from A. The string to be derived from C must be 1^k0^k, and it also has only one derivation from C. The string to be derived from B is 1^{j-i-k}, and it also has only one derivation from B.

6.26. (a) The string aaa has two different leftmost derivations. An equivalent unambiguous grammar is $S \rightarrow aS \mid bS \mid \Lambda$.

(b) The string a can be derived $S \Rightarrow ABA \Rightarrow aBA \Rightarrow^* a$ or $S \Rightarrow ABA \Rightarrow BA \Rightarrow A \Rightarrow a$. It's easy to see, however, that any string with at least one b has only one leftmost derivation. Therefore, an equivalent unambiguous grammar is $S \rightarrow A \mid ABA \qquad A \rightarrow aA \mid \Lambda \qquad B \rightarrow bB \mid b$.

(c) ab has two leftmost derivations, but it's the only string that does. Therefore, an equivalent unambiguous grammar is one that allows ab to be derived only one way, such as $S \to A \mid B \qquad A \to aAb \mid aabb \qquad B \to abB \mid \Lambda$.

(d) The grammar is ambiguous because in deriving $aaabb$, for example, we could use either $S \Rightarrow aSb \Rightarrow aaaSbb \Rightarrow aaabb$ or $S \Rightarrow aaSb \Rightarrow aaaSbb \Rightarrow aaabb$. In other words, we could use the two productions in either order. To make it unambiguous, fix it so that we have to use one first, as many times as necessary. One way to do this is $S \to aaSb \mid T$ $T \to aTb \mid \Lambda$. In this grammar, if we want to end up with 5 more a's than b's, for example, we have to use the first production 5 times and then the second as many times as there are b's.

(e) The ambiguity here seems to be directly related to the Λ-production, since aSb and abS yield the same thing when S is replaced by Λ. We must keep Λ in the language, but aside from that we can eliminate it from the grammar, as follows:

$$S \to T \mid \Lambda \qquad T \to aTb \mid abT \mid ab$$

It is reasonably clear that this generates the same language as the original grammar. We show that the new grammar is unambiguous by showing that for any $n \geq 1$, if x is any string that can be derived from T in n steps, then x has only one derivation from T. The basis step, $n = 1$, is clear. Suppose that $k \geq 1$ and that a string derivable from T in k or fewer steps can be derived in only one way. Let x be a string derivable from T in $k + 1$ steps.

We consider two cases. If $x = aby$ for some y, then any derivation of x from T must begin $T \Rightarrow abT$, and the remainder of the derivation consists of deriving y from T in k steps. By the induction hypothesis, there is only one way to do this; therefore, x has only one derivation from T. Otherwise, $x = ayb$ for some y derivable from T in k steps. Again the induction hypothesis implies that x has only one derivation from T.

6.27. The nonnegative integers that can be expressed in the form $4i + 7j$, where i and j are nonnegative, are 0, 4, 7, 8, 11, 12, 14, 15, 16, and (by Exercise 2.50) all the integers greater than 17. An unambiguous grammar is therefore

$$S \to \Lambda \mid a^4 \mid a^7 \mid a^8 \mid a^{11} \mid a^{12} \mid a^{14} \mid a^{15} \mid a^{16} \mid T \qquad T \to aT \mid a^{18}$$

6.28. The grammar obtained from an FA as in the proof of Theorem 6.2 is always unambiguous. The grammar obtained from an NFA (see Exercise 6.15) may be ambiguous.

6.29. Convert the regular grammar into an NFA, as in the proof of Theorem 6.2. Use the subset construction to find an equivalent FA. Then convert the FA into a regular grammar, as in the proof of Theorem 6.2. It will be unambiguous.

6.30. Consider a left parenthesis $(_0$ in a balanced string x. First, there must be at least one right parenthesis in x after $(_0$. This is true because x has equal numbers of left and right parentheses, and if there were no right parentheses following $(_0$, the prefix of x ending

just before $(_0$ would have more right than left.

Second, of the right parentheses following $(_0$, at least one must have the property that the substring beginning with $(_0$ and ending with this one has equal numbers of left and right parentheses. To see this, consider substrings beginning with $(_0$. The substring of length 1 has more left parentheses than right, and every subsequent symbol changes by 1 the difference between the number of left and the number of right. Therefore, if there were no such right parenthesis, all the substrings beginning with $(_0$ would have more left parentheses than right. In particular, the suffix of x beginning with $(_0$ would, and therefore, the prefix ending just before $(_0$ would have more right than left.

Let $)_0$ be the first right parenthesis following $(_0$ such that the substring y beginning with $(_0$ and ending with $)_0$ has equal numbers of left and right parentheses. Then every shorter nonnull prefix of $y$has more left than right. (Otherwise, there would have been a right parenthesis before $)_0$ having the same property as $)_0$.) Therefore, y is balanced.

6.31. First observe that two distinct left parentheses a_1 and a_2 can't have the same mate. Otherwise the string beginning with a_1, the leftmost of the two, and extending up to but not including a_2, would be a balanced string, which would imply that the mate of a_1 was within that string.

Now, suppose there is a left parenthesis a_1 to the right of a so that the string from a_1 to b was balanced. We may assume a_1 is the rightmost left parenthesis having this property. Since the mate b_1 of a_1 can't be b, it must appear before b. The string now looks like axa_1yb_1zb, where the entire string, the substring a_1yb_1, and the substring a_1yb_1zb are all balanced. This implies, however, that the substring zb is also balanced. This substring must start with a left parenthesis, and now we have contradicted the assumption that a_1 is the rightmost left parenthesis for which the string from a_1 to b is balanced.

6.32. $S \to S+T \mid S-T \mid T \qquad T \to T*F \mid T/F \mid F \qquad F \to (S) \mid a$

6.33. First we show that if A is nullable, then $A \Rightarrow^* \Lambda$. Using structural induction, it is sufficient to show that if there is a production $A \to \Lambda$, then $A \Rightarrow^* \Lambda$ (which is obvious), and that if $A \to B_1B_2 \ldots B_n$ is a production and each B_i satisfies $B_i \Rightarrow^* \Lambda$, then $A \Rightarrow^* \Lambda$. This is also obvious.

For the converse, we show that if $A \Rightarrow^* \Lambda$, then A is nullable. $A \Rightarrow^* \Lambda$ means there is a derivation by which Λ is derived from A. The proof is on the number of steps in the derivation. The basis step is easy, since if $A \Rightarrow \Lambda$ then $A \to \Lambda$ must be a production. Suppose that $k \geq 0$ and that any variable from which Λ can be derived in k or fewer steps is nullable. Now suppose we have a derivation of Λ from A in $k+1$ steps, and let the first step be $A \to B_1B_2 \ldots B_n$. Since $B_1B_2 \ldots B_n \Rightarrow^* \Lambda$, each B_i must be a variable from which Λ can be derived in k or fewer steps. By the induction hypothesis, each B_i is nullable. Therefore, A is nullable by the second part of the definition.

6.34. (a) $S \to AB \qquad A \to aASb \mid aAb \mid a \qquad B \to bS \mid b.$

(b) All the variables are nullable. Following the algorithm, we obtain

$$S \to AB \mid A \mid B \mid ABC \mid AC \mid BC \mid C$$
$$A \to BA \mid B \mid BC \mid C \mid a$$
$$B \to AC \mid A \mid C \mid CB \mid b$$
$$C \to BC \mid B \mid AB \mid A \mid c$$

6.35. (a) $S \to ABA \mid AB \mid BA \mid AA \mid aA \mid a \mid bB \mid b \qquad A \to aA \mid a \qquad B \to bB \mid b$
(c)

$$S \to aAa \mid bB \mid bb \mid aCaa \mid baD \mid abD \mid aa$$
$$A \to aAa \mid bB \mid bb \qquad B \to bB \mid bb$$
$$C \to aCaa \mid baD \mid abD \mid aa \qquad D \to baD \mid abD \mid aa$$

6.36. A definition of live variables is the following: 1. Any variable A for which there is a production $A \to x$, with $x \in \Sigma^*$, is live. 2. Any variable A for which there is a production $A \to \alpha$, where every symbol of α is either a terminal or a live variable, is live.

A corresponding algorithm to find the live variables is the following.

$L_0 = \{A \in V \mid P$ contains a production $A \to x$ with $x \in \Sigma^*\}$;
$i = 0$;
do
    $i = i + 1$;
    $L_i = L_{i-1} \cup \{A \mid P$ contains $A \to \alpha$, where $\alpha \in (\Sigma \cup L_{i-1})^*\}$;
while $L_i \neq L_{i-1}$;
L_i is the set of live variables.

6.37. A definition of reachable variables is the following: 1. S is reachable. 2. If A is reachable, and P contains a production $A \to \alpha B \beta$, where $B \in V$, B is reachable.

A corresponding algorithm to find the reachable variables is the following.

$R_0 = \{S\}$;
$i = 0$;
do
    $i = i + 1$;
    $R_i = R_{i-1} \cup \{B \in V \mid P$ contains $A \to \alpha B \beta$, where $A \in R_{i-1}\}$;
while $R_i \neq R_{i-1}$;
R_i is the set of reachable variables.

6.38. (a) Consider the grammar with productions $S \to AB \qquad A \to a$. A is live, since $A \to a$ is a production, and A is reachable, since $S \to AB$ is a production, but A is

obviously useless, since no strings can be derived from this grammar.

(b) We know that $L(G_1) = L(G)$, because the only productions left out of G_1 are those containing variables of G from which no strings of terminals can be derived in G. Similarly, $L(G_1) = L(G_2)$, since in constructing G_2 from G_1 the only productions omitted are those containing variables never obtained in any derivation of G_1 beginning with S.

We wish to show every variable in G_2 is useful. Let A be such a variable. A is reachable in G_1; i.e., $S \Rightarrow^* \alpha A \beta$ in G_1. This means that not only A, but any variables in α and β as well, are reachable in G_1. Therefore, this is a legitimate derivation in G_1, since any production in G_1 that involves only variables in G_1 is still present in G_1. On the other hand, since this is a derivation in G_1, all variables involved (in particular, A, as well as all variables in α and β) are variables in G_1. This means they are live variables in G. Thus there is a derivation $\alpha A \beta \Rightarrow^* x$ in G, for some $x \in \Sigma^*$. Any variable involved in this derivation is also a live variable in G, so all the productions involved are present in G_1; thus this is a legitimate derivation in G_1. In fact, it's a derivation in G_2 as well: $\alpha A \beta$ is derivable from S in G_1, so anything derivable from $\alpha A \beta$ in G_1 is also derivable from S in G_1. Since all variables involved are therefore present in G_2, all the productions involved must also be present in G_2. We have $S \Rightarrow^*_{G_2} \alpha A \beta \Rightarrow^*_{G_2} x$, which shows that A is a useful variable in G_2.

(c) The example in (a) also illustrates this fact. Nothing is eliminated in the first step, and only the first production is eliminated in the second step.

(d)(ii) The live variables are S, A, and B, and the grammar obtained from the first step has the productions

$$S \to AB \qquad A \to aAb \mid bAa \mid a \qquad B \to bbA \mid aaB \mid AB$$

Since all the variables in this grammar are reachable, this is the final result.

6.39. (b) Eliminating Λ-productions yields the productions

$$S \to S(S) \mid (S) \mid S()$$

There are obviously no unit productions. Converting to Chomsky normal form gives

$$\begin{aligned} &S \to SY_1 \qquad Y_1 \to X_(Y_2 \qquad Y_2 \to SX_) \\ &X_(\to (\qquad X_) \to) \\ &S \to X_(Y_2 \mid SY_3 \qquad Y_3 \to X_(X_) \end{aligned}$$

(The variables are S, Y_1, Y_2, Y_3, $X_($, and $X_)$.)

6.40. As we did at the beginning of Section 6.6., we consider the quantity $s = l + t$, the sum of the length of the current string in the derivation and the number of terminal symbols in the string. Initially, when the current string is S, $s = 1$. At the end of the derivation, $s = 2k$. Since the grammar is in Chomsky normal form, s increases by 1 at each step. Therefore, there are exactly $2k - 1$ steps in the derivation.

6.41. Strings over $\{a, b\}$ in which every prefix has at least as many a's as b's. We prove by induction that every string having this property can be derived from the grammar. If $|x| = 0$, x can obviously be derived. Suppose that $k \geq 0$ and that for any string x of length $\leq k$ in which every prefix has at least as many a's as b's, x can be derived. We wish to show that any string x of length $k + 1$ having this property can be derived.

Suppose that $|x| = k + 1$ and every prefix of x has at least as many a's as b's. Then $x = ay$ for some y. If every prefix of y has at least as many a's as b's, then y can be derived from the grammar, by the induction hypothesis; therefore, $x = ay$ can, because we can start the derivation $S \Rightarrow aS$ and continue by deriving y from S.

If y has a prefix with more b's than a's, let y_1 be the smallest such prefix. Then y_1 must end in b; let $y_1 = y_2 b$. The string y_2 has the same number of a's as b's, and every prefix of y_2 has at least as many a's as b's. We can write $x = ay_2bz$ for some string z. Since every prefix of x has at least as many a's as b's, and since ay_2b has equal numbers of a's and b's, every prefix of z must have at least as many a's as b's. Therefore, by the induction hypothesis, $S \Rightarrow^* y_2$ and $S \Rightarrow^* z$. It follows that $S \Rightarrow^* ay_2bz$, since we can start the derivation $S \Rightarrow aSbS$ and continue by deriving y_2 from the first S and z from the second.

If we think of a and b as (and), respectively, this language is the set of all prefixes of balanced strings of parentheses.

6.42. If these two productions were the only ones with variables on the right side, then we might as well assume that S is the only variable (any other variable could never be involved in a derivation from S), and so the only other productions are $S \rightarrow x_1 \mid x_2 \mid \ldots \mid x_n$, where each x_i is an element of $\{a, b\}^*$. In this case, however, no string that begins with a and ends with b and is not one of the x_i's can be derived. Therefore, the grammar could not generate all nonpalindromes.

6.43. The only string of length 1 produced by the grammar is 0, which has more 0's than 1's. Suppose that $k \geq 1$ and that every string of length k or less produced by the grammar has more 0's than 1's. Let x be a string of length $k + 1$ produced by the grammar, and consider a derivation of x. If the first step in the derivation is either $S \Rightarrow S0$ or $S \Rightarrow 0S$, then $x = y0$ or $x = 0y$, where in either case y is a string of length k produced by the grammar. By the induction hypothesis, y has more 0's than 1's, and so x obviously does also. If the first step in the derivation of x is one of the last three productions, then x is the concatenation of three strings, two of which are derivable from the grammar and one of which is 1. By the induction hypothesis, the two strings derivable from the grammar both have more 0's than 1's; therefore, since the two of them together have at least two more 0's than 1's, the result of adding a 1 still has more 0's than 1's.

6.44. We consider a string x satisfying $d(x) > 0$, and we complete the induction step of the proof that if x starts with 1, x can be derived in G_0. (The other remaining case, in which x ends with 1, is very similar.) We have $x = 1y$, where $d(y) = n_0(y) - n_1(y) \geq 2$. Since each symbol appended to a string changes the d value by 1, there must be some prefix z of y for which $d(z) = 1$, and if $y = zw$, it follows that $d(w) \geq 1$. The induction hypothesis implies that z and w can both be derived from S; it follows that x can, because we can

start a derivation of x with the production $S \to 1SS$ and continue by deriving z from the first S and w from the second.

6.46. We can show that the CFG generates $\{a, b\}^*$. Clearly Λ can be generated. Let us show by induction that for every $n \geq 1$, any string of length n can be derived from T. Suppose $k \geq 1$ and every string of length k can be derived from T. Let $|x| = k + 1$. If $x = ay$, then by the induction hypothesis $T \Rightarrow^* y$. It follows that $S \Rightarrow^* y$, because $S \Rightarrow ST \Rightarrow T$. Therefore, $T \Rightarrow^* x$, because we can start the derivation $T \Rightarrow aS$ and continue by deriving y from S. If $x = by$, then it follows from the induction hypothesis that $T \Rightarrow^* y$, and therefore $T \Rightarrow^* x$ because we can start the derivation $T \Rightarrow bT$ and continue by deriving y from T.

6.48. The grammar generates the language of strings in which the number of 0's and the number of 1's are both even. The proof that every string generated has this property is straightforward. In the other direction, the induction step uses the observation that if x is a nonnull string for which $n_0(x)$ and $n_1(x)$ are even, then for some y having the same property, one of the statements $x = 00y$, $x = 11y$, $x = y00$, $x = y11$, $x = 01y01$, $x = 01y10$, $x = 10y10$, $x = 10y01$ must be true.

6.50. Let $d(x) = n_0(x) - n_1(x)$. We can show by induction that for every $n \geq 0$, if $|x| = n$, these three statements hold: (i) if $S \Rightarrow^* x$, then $d(x) = 0$; (ii) if $A \Rightarrow^* x$, then $d(x) = 1$; and (iii) if $B \Rightarrow^* x$, then $d(x) = -1$. Assume all three statements are true for $n \leq k$, and consider $n = k + 1$. If $S \Rightarrow^* x$, then either $x = 0y$, where $B \Rightarrow^* y$, or $x = 1y$, where $A \Rightarrow^* y$. In the first case, $d(y) = -1$ by the induction hypothesis, and therefore $d(x) = 0$; in the second case, $d(y) = 1$ by the induction hypothesis, and therefore $d(x) = 0$. The proofs in the cases $A \Rightarrow^* x$ and $B \Rightarrow^* x$ are similar.

In the opposite direction, we can show that for every $n \geq 0$, if $|x| = n$, the converses of the three statements above hold. Suppose this is true for $n \leq k$, and consider $n = k + 1$. If $d(x) = 0$, then either $x = 0y$ for some y with $d(y) = -1$ or $x = 1y$ for some y with $d(y) = 1$. In the first case, it follows from the induction hypothesis that $B \Rightarrow^* y$, and therefore $S \Rightarrow^* x$ because we can start the derivation $S \Rightarrow 0B$. In the second case, it follows from the induction hypothesis that $A \Rightarrow^* y$, and so $S \Rightarrow^* x$ because we can start the derivation $S \Rightarrow 1A$.

If $d(x) = 1$ and $x = 0y$, then $d(y) = 0$; by the induction hypothesis, $S \Rightarrow^* y$; and so $A \Rightarrow^* x$, because we can start the derivation $A \Rightarrow 0S$. If $d(x) = 1$ and $x = 1y$, then it follows from the discussion in Example 6.8 that $y = wz$ for some w and z with $d(w) = d(z) = 1$. By the induction hypothesis, both w and z can be derived from A. Therefore, x can be derived from A, since we can start the derivation $A \Rightarrow 1AA$.

The proof in the case where $d(x) = -1$ is similar.

6.51. We show by induction on n that every string in L of length n can be generated by the grammar. The basis step is easy, since $S \to \Lambda$ is a production in the grammar. Suppose that $k \geq 0$ and that every string of length k or less having equal numbers of 0's and 1's can be generated by the grammar. Suppose x is a string of length $k + 1$ having equal numbers

of 0's and 1's. If x starts with either 01 or 10, then the induction hypothesis implies that the suffix y of length $k-1$ can be generated by the grammar. It follows that x can, since in the first case we can start the derivation with the production $S \rightarrow 0S1S$ and continue by replacing the first S by Λ and deriving y from the second. The argument is similar in the second case. If x starts with neither of these strings, then it begins with either 00 or 11. We complete the proof in the first case.

Suppose that $x = 00y$. Let $d(z) = n_0(z) - n_1(z)$, and consider $d(z)$ for the prefixes z of x. $d(00) = 2$ and $d(x) = 0$. Therefore, there must exist a prefix z for which $d(z) = 1$. Let z_1 be the longest such prefix. Then x must be $z_1 1 z_2$ for some string z_2—otherwise, since $d(z_1 0) = 2$, there would be a longer prefix for which $d = 1$. We now know that $x = 00y_1 1z_2$, and $d(0y_1) = d(00y_1 1) = 0$. Therefore, $d(z_2) = 0$ as well. By the induction hypothesis, both $0y_1$ and z_2 can be derived from S. Therefore, $x = 0(0y_1)1z_2$ can be also, since we can start a derivation with the production $S \rightarrow 0S1S$.

6.52. (a) The language of balanced strings of parentheses.

(b) Let G be the CFG with productions $S \rightarrow SS \mid (S) \mid \Lambda$, and let G_1 be the CFG with productions $S_1 \rightarrow S_1(S_1) \mid \Lambda$.

First we show that $L(G_1) \subseteq L(G)$, by showing that for any $n \geq 1$, if x can be derived from S_1 in n steps, then $x \in L(G)$. The basis step, $n = 1$, is clear. We assume that $k \geq 0$ and that any string derivable from S_1 in k or fewer steps is derivable from S. Now suppose x can be derived from S_1 in $k+1$ steps. The first step must be $S_1 \Rightarrow S_1(S_1)$, and thus $x = y(z)$, where y and z can both be derived from S_1 in k or fewer steps. By the induction hypothesis, both y and z can be derived from S. Therefore, x can, since we can take the first two steps of the derivation to be $S \Rightarrow SS \Rightarrow S(S)$, and we can continue by deriving y from the first S and z from the second.

Next we show that $L(G) \subseteq L(G_1)$, by showing that for any $n \geq 1$, if x can be derived from S in n steps, then $x \in L(G_1)$. Again the basis step is easy. Assume that $k \geq 1$ and that any string derivable from S in k or fewer steps is derivable from S_1. Let x be a string derivable from S in $k+1$ steps. We may assume there is no derivation with k or fewer steps, since otherwise the induction hypothesis would give us the result. Fix a specific $(k+1)$-step derivation. If the first step is $S \Rightarrow (S)$, then $x = (y)$, where y is derivable from S in k steps. By the induction hypothesis, y is derivable from S_1. Therefore, we can derive x from S_1, by starting the derivation $S_1 \Rightarrow S_1(S_1) \Rightarrow \Lambda(S_1) = (S_1)$ and continuing to derive y from S_1. We are left with the case in which the first step in the derivation of x is $S \Rightarrow SS$, which implies that $x = yz$, where y and z are nonnull strings in $L(G)$. (If either were null, x would have a derivation with k or fewer steps.) Suppose y and z are strings like this for which y is as short as possible. If y had a derivation in G that started $S \rightarrow SS$, in which nonnull strings y_1 and y_2 were subsequently derived from both S's, then we could write $x = y_1 y_2 z = y_1 z'$, where y_1 and z' are nonnull strings in $L(G)$ with $|y_1| < |y|$. Therefore, y must have a derivation beginning with $S \rightarrow (S)$. This implies that $x = (w)z$, where w and z are in $L(G)$ and both w and z can be derived in G in k or fewer steps. The induction hypothesis implies that w and z are in $L(G_1)$. It follows that $x \in L(G_1)$, because we may start a derivation $S_1 \rightarrow (S_1)S_1$ and continue by deriving w from the first S_1 and z from the second.

6.53. Call this grammar G, and let L be the language of strings x with $n_a(x) = 2n_b(x)$. The proof that $L(G) \subseteq L$ is a straightforward induction proof. We show, also using induction, that $L \subseteq L(G)$. It is clear that $\Lambda \in L(G)$. Suppose that $k \geq 0$ and that any string in L of length k or less is in $L(G)$. Let $x \in L$ with $|x| = k + 1$.

For any string z, let $d(z) = n_a(z) - 2n_b(z)$. We consider six cases, one of which must hold: i) x begins with aab; ii) x begins with ab; iii) x begins with aaa and ends with b; iv) x begins with aaa and ends with a; v) x begins with ba; and vi) x begins with bb.

In case i), we write $x = aaby$. Since $d(y)$ must also be 0, the induction hypothesis implies that $y \in L(G)$. Therefore, $x \in L(G)$, because we can start the derivation $S \Rightarrow aSaSbS \Rightarrow^* aabS$ and continue by deriving y from S.

In case ii), $x = aby$. We observe that $d(ab) = -1$ and $d(x) = d(aby) = 0$; consider the first prefix of x for which $d \geq 0$. Adding the last symbol of this prefix causes d to increase, and thus the prefix ends with a. This means that $x = abwaz$, where $d(w) = d(z) = 0$. By the induction hypothesis, w and z are both in $L(G)$. Therefore, x is also, because we can start the derivation $S \Rightarrow aSbSaS \Rightarrow abSaS$ and continue to derive w from the first S and z from the second.

In case iii), $x = aaayb$. All we really need in this case is that $x = aazb$. By the induction hypothesis, $z \in L(G)$; therefore, so is x, since we can start the derivation $S \Rightarrow aSaSbS \Rightarrow^* aaSb$ and continue by deriving z from S.

In case iv), $x = aaaya$. This time, since $d(aaa) > 0$, we consider the smallest nonnull prefix of x for which $d \leq 0$. This prefix must end with b, so that $x = aaawbza$. The possible values of $d(aaaw)$ are 1 and 2. If it is 1, then $d(aaw) = 0$, so that $x = a(aaw)bza$. In this case, the induction hypothesis implies that $aaw \in L(G)$ and $z \in L(G)$, and therefore x is also, since we can start the derivation $S \Rightarrow aSbSaS \Rightarrow aSbSa$ and continue by deriving aaw from the first S and z from the second. If the value is 2, then $d(aw) = 0$, and $x = aa(aw)b(za)$, so that $d(za)$ must be 0. The induction hypothesis then implies that aw and za are in $L(G)$. Therefore, x is, because we can start the derivation $S \Rightarrow aSaSbS \Rightarrow aaSbS$ and continue by deriving aw from the first S and za from the second.

In case v), $x = bay$. Since $d(ba) = -1$, we consider the smallest nonnull prefix for which $d \geq 0$. As in case ii), this prefix ends in a, which means that $x = bawaz$, where $d(w) = d(z) = 0$. By the induction hypothesis, w and z are in $L(G)$. Therefore, so is x, since we can start the derivation $S \Rightarrow bSaSaS \Rightarrow baSaS$ and continue by deriving w from the first S and z from the second.

Finally, in case vi) $x = bby$. Since $d(bb) < -1$, d must increase to -1 at some point, and later increase to 0. In both cases, the symbol causing this to happen is a. We may therefore write $x = bwaza$, where $d(w) = d(z) = 0$. By the induction hypothesis, w and z are in $L(G)$, and so starting a derivation $S \Rightarrow bSaSaS \Rightarrow bSaSa$ allows us to conclude that x is also.

6.54. No. The string $aabbaa$ cannot be generated by this grammar.

6.55. These two statements can be proved by induction: i) For every $n \geq 0$, and every x with $|x| = n$, if $S \Rightarrow^* x$ then $n_a(x) = 2n_b(x)$ and if $T \Rightarrow^* x$ then $n_a(x) = 2n_b(x) + 1$.

ii) For every $n \geq 0$, and every x with $|x| = n$, if $n_a(x) = 2n_b(x)$ then $S \Rightarrow^* x$ and if $n_a(x) = 2n_b(x) + 1$ then $T \Rightarrow^* x$. We omit the proof.

6.56. We describe the proof in the case when $\Sigma_1 = \Sigma_2 = \{0, 1\}$, and it can easily be adapted to the general case. Suppose $f : \{0,1\}^* \to \{0,1\}^*$ is a homomorphism, and let $f(0) = s_0$ and $f(1) = s_1$. Then f is determined by the two values s_0 and s_1. (For example, $f(011) = f(0)f(1)f(1)$, since f is a homomorphism, and so $f(011) = s_0s_1s_1$.) Suppose $G = (V, \{0,1\}, S, P)$ is a CFG generating L. Consider the new CFG $G_f = (V, \{0,1\}, S, P_f)$, where the productions in P_f are obtained by taking the productions in P and replacing all occurrences of 0 and 1 on the right sides by s_0 and s_1, respectively. Then it is not hard to see that G_f generates precisely the strings $f(x)$, for $x \in L(G)$.

For example, the language *pal* is generated by $S \to 0S0 \mid 1S1 \mid 0 \mid 1 \mid \Lambda$. If $f(0) = 01$ and $f(1) = 110$, the grammar G_f has the productions $S \to 01S01 \mid 110S110 \mid 01 \mid 110 \mid \Lambda$. The string 010 has the derivation $S \Rightarrow 0S0 \Rightarrow 010$, and $f(010) = 0111001$ has the derivation $S \Rightarrow 01S01 \Rightarrow 0111001$.

6.57. We prove by induction that for any $n \geq 0$, and any x with $S \Rightarrow^* x$ and $|x| = n$, x has only one leftmost derivation. This is clear for $n = 0$. Suppose $k \geq 0$ and the statement is true for $n \leq k$. We must show that if $S \Rightarrow^* x$ and $|x| = k + 1$, x has only one leftmost derivation. The first step of any derivation of x must be $S \to (S)S$. Therefore, $x = (y)z$, where $S \Rightarrow^* y$ and $S \Rightarrow^* z$ and $|y|, |z| \leq k$. According to the induction hypothesis, both y and z have only one leftmost derivation. In order to conclude that x does, however, we must first eliminate the possibility that x can be written in *two different ways* as $(y)z$, where $S \Rightarrow^* y$ and $S \Rightarrow^* z$. The reason this is impossible is that if $x = (y)z$, where both y and z are balanced strings of parentheses, then the right parenthesis shown is the unique right parenthesis that matches the left parenthesis shown. Therefore, the grammar is unambiguous.

6.58. (a) We use the fact that this is $\{a^ib^jc^k \mid i < j + k\} \cup \{a^ib^jc^k \mid i > j + k\}$. For the first part, we can modify the CFG in Exercise 6.9(a) by requiring that at some point either some c's are generated to which no a's correspond, or some b's, or both. The variable B will stand for one or more b's, and C for one or more c's.

$$S \to aSc \mid TC \mid UC \mid C \qquad T \to aTb \mid \Lambda$$
$$U \to aUb \mid B \qquad B \to bB \mid b \qquad C \to cC \mid c$$

For the second part, this modification of the CFG in Exercise 6.9(a) will work:

$$S \to aSc \mid T \qquad T \to aTb \mid A \qquad A \to aA \mid a$$

We can now use the standard construction to form a CFG for the union of these two languages.

(b) This is the union of the two languages $L_1 = \{a^ib^jc^k \mid j > i+k\}$ and $L_2 = \{a^ib^jc^k \mid j < i + k\}$. Let $E = \{a^ib^i \mid i \geq 0\}$ and $F = \{b^jc^j \mid j \geq 0\}$, and let A, B, and C denote the languages $\{a^i \mid i > 0\}$, $\{b^i \mid i > 0\}$, and $\{c^i \mid i > 0\}$, respectively. Then $L_1 = EBF$ (this

was described in Example 6.11 for the alphabet $\{0,1\}$), and $L_2 = AEF \cup EFC \cup AEFC$. The second equality means that a string in L_2 consists of a string of the form $a^i b^i$ followed by a string $b^j c^j$, with one or more additional a's at the beginning, one or more additional c's at the end, or both. Constructing CFGs for the languages E, F, A, B, and C is straightforward.

6.59. (a) $S \rightarrow aSb \mid aaSbbb \mid \Lambda$. It is easy to see that every string obtained by this CFG satisfies the given inequalities. We show that if i and j are nonnegative integers with $i \leq j \leq 3i/2$, then $a^i b^j$ can be obtained from the grammar. The proof is by induction on i. If $i = 0$, then the inequalities are satisfied only if $j = 0$, and Λ can be obtained from the grammar. Suppose $k \geq 0$ and that for any i and j satisfying $0 \leq i \leq k$ and $i \leq j \leq 3i/2$, $a^i b^j$ can be obtained from the grammar. We wish to show that if $i = k+1$ and $i \leq j \leq 3i/2$, then $a^i b^j$ can be obtained from the grammar.

In the case $j = k+1$, $a^i b^j = a^{k+1} b^{k+1}$ can be obtained by using the first production $k+1$ times. Otherwise, $k+2 \leq j \leq 3(k+1)/2$. Provided $k \geq 1$, therefore, it follows that $i_1 = i - 2 = k - 1$ and $j_1 = j - 3$ are nonnegative integers satisfying $i_1 \leq j_1 \leq 3i_1/2$. By the induction hypothesis, $a^{i_1} b^{j_1}$ can be obtained from the grammar. Therefore, by using the production $S \rightarrow aaSbbb$ one extra time, so can the string $a^{i_1+2} b^{j_1+3} = a^{k+1} b^j$. If $k = 0$, however, the inequalities $k+2 \leq j \leq 3(k+1)/2$ are impossible, and so the proof is complete.

(b) $S \rightarrow aaSb \mid aaSbbb \mid aSb \mid \Lambda$. Again it is easy to see that every string $a^i b^j$ obtained from this CFG satisfies the inequalities $i/2 \leq j \leq 3i/2$. We show that if i and j are nonnegative integers satisfying these inequalities, then $a^i b^j$ can be obtained from the grammar. It is straightforward to check that this is correct if $i \leq 2$. For the induction step, suppose that $k \geq 2$, and for any $i \leq k$ and any j satisfying $i/2 \leq j \leq 3i/2$, $a^i b^j$ can be obtained from the grammar. Now we wish to show that if $(k+1)/2 \leq j \leq 3(k+1)/2$, then $a^{k+1} b^j$ can be obtained. We may do this by considering several cases.

If k is odd and $j = (k+1)/2$, $a^{k+1} b^j = (aa)^j b^j$, and thus the string can be obtained by using the first production j times.

If k is even and $j = k/2 + 1$, $a^{k+1} b^j = (aa)^{k/2} a b^{k/2} b$, and so the string can be obtained by using the first production $k/2$ times and the third production once.

If k is odd and $j = (k+3)/2$, $a^{k+1} b^j = (aa)^{(k-1)/2} a^2 b^{(k-1)/2} b^2$, and the string can be obtained by using the first production $(k-1)/2$ times and the third production twice.

If k is even and $j = k/2+2$, $a^{k+1} b^j = (aa)^{k/2-1} a^3 b^{k/2-1} b^3$, and the string can be obtained by using the first production $k/2 - 1$ times and the third production three times.

The only remaining case is the one in which $(k+5)/2 \leq j \leq 3(k+1)/2$ (and k is either even or odd). In this case, it is easy to check that $(k-1)/2 \leq j - 3 \leq 3(k-1)/2$. By the induction hypothesis, therefore, the string $a^{k-1} b^{j-3}$ can be obtained from the grammar. Therefore, the string $a^{k+1} b^j$ can be, by using the same derivation except that one more step is added in which the third production is used.

6.60. (a) The string $aacbc$ has the two derivations $S \Rightarrow aS \Rightarrow aaSbS \Rightarrow^* aacbc$ and $S \Rightarrow aSbS \Rightarrow aaSbS \Rightarrow^* aacbc$.

(b) We show first that if x can be derived in G_1, then x can be derived in G. First, it is clear that if $T \Rightarrow^*_{G_1} x$, then $S \Rightarrow^*_G x$, since both the productions $S \rightarrow aSbS \mid c$ are present

in G. Let us prove by induction on $|x|$ that if either $S_1 \Rightarrow^*_{G_1} x$ or $U \Rightarrow^*_{G_1} x$, then $S \Rightarrow^*_G x$. The basis step, when $|x| = 1$, is easy, since the only string of length 1 that can be derived from either S_1 or U is c, and $S \rightarrow c$ is a production in G. Suppose that $k \geq 1$ and that every string x with $|x| \leq k$ that can be derived in G_1 from either S_1 or U can be derived in G from S. Now consider a string x with $|x| = k + 1$.

If $S_1 \Rightarrow T \Rightarrow^*_{G_1} x$, then we have already observed that $S \Rightarrow^*_G x$. If $S_1 \Rightarrow U \Rightarrow aS_1 \Rightarrow^*_{G_1} x$, then $x = ay$, where $S_1 \Rightarrow^*_{G_1} y$ and $|y| = k$. By the induction hypothesis, $S \Rightarrow^*_G y$. Therefore, $S \Rightarrow^*_G x$, because we can begin the derivation with the production $S \rightarrow aS$. Finally, if $S_1 \Rightarrow U \Rightarrow aTbU \Rightarrow^*_{G_1} x$, then $x = aybz$, where $T \Rightarrow^*_{G_1} y$, $U \Rightarrow^*_{G_1} z$, and $|y|$ and $|z|$ are both no larger than k. We know that y can be derived from S in G, and the induction hypothesis tells us that z can. Therefore, since we have the production $S \rightarrow aSbS$ in G, the string x can be derived from S in G.

In the other direction, we first observe that $L(G)$ can be described in a way similar to the language in Exercise 6.41—a slightly more complicated way, simply because the production $S \rightarrow \Lambda$ has been replaced by $S \rightarrow c$. It is not hard to see that for $x \in L(G)$, x matches the regular expression $a^*c(ba^*c)^*$, and every prefix of x has at least as many a's as b's. We can prove in a way similar to the solution to Exercise 6.41 that every x having this property is in $L(G)$. In the induction step, if x is a string of length $k + 1$ with this property, then x must start with a. The case $x = ay$, where y also has this property, is straightforward. Otherwise $x = ay$, where y also matches the regular expression but some prefix of y has more b's than a's. If y_1 is the smallest such prefix, then y_1 ends in b, and so y matches the regular expression $a^*c(ba^*c)^*b$. This means, as in the solution to 6.41, that the prefix of y up to the last b is a string matching $a^*c(ba^*c)^*$, of which every prefix has at least as many a's as b's, and that $x = aybz$, where z also has this property. Therefore, by the induction hypothesis, y and z can both be generated in G, and thus x can, since G contains the production $S \rightarrow aSbS$.

Now we wish to prove that $L(G) \subseteq L(G_1)$. This is implied by the following three statements, which we prove simultaneously by induction: (i) If $x \in L(G)$ and x has a derivation in G involving only the productions $S \rightarrow aSbS$ and $S \rightarrow c$ (this is the same as saying that $x \in L(G)$ and x has equal numbers of a's and b's), then $S_1 \Rightarrow T \Rightarrow^*_{G_1} x$; (ii) if $x \in L(G)$ and x has a derivation in G that begins with the production $S \rightarrow aS$, then $S_1 \Rightarrow U \Rightarrow^*_{G_1} x$; and (iii) if $x \in L(G)$, x has more a's than b's, and every derivation of x in G begins with the production $S \rightarrow aSbS$, then $S_1 \Rightarrow U \Rightarrow^*_{G_1} aTbU \Rightarrow^*_{G_1} x$.

The basis step is straightforward. Suppose that all three statements hold for strings of length $\leq k$, and suppose $x \in L(G)$ and $|x| = k + 1$. If x has a derivation involving only $S \rightarrow aSbS$ and $S \rightarrow c$, then either $x = c$, or $x = aybz$ where both y and z are in $L(G)$ and have derivations involving only these two productions. Since G_1 contains the productions $T \rightarrow aTbT \mid c$, it follows from the induction hypothesis that $S_1 \Rightarrow T \Rightarrow^*_{G_1} x$. If x has a derivation beginning with $S \rightarrow aS$, then $x = ay$ for some $y \in L(G)$. The induction hypothesis implies that $y \in L(G_1)$, and thus $x \in L(G_1)$ because G_1 contains the productions $S_1 \Rightarrow U \Rightarrow aS_1$. Finally, suppose that $x \in L(G)$, x has more a's than b's, and every derivation in G begins with the production $S \rightarrow aSbS$. Here we use the characterization of $L(G)$ described above. The string x matches the regular expression $a^*c(ba^*c)^*$, every prefix has at least as many a's as b's, and there is at least one b. Let

$x = x_1y_1$, where x_1 is of the form a^*cba^*c. If x started with aa, then x_1 would be ax_2 for some $x_2 \in L(G)$, so that x would have a derivation starting $S \rightarrow aS$. Therefore, $x = acbz$ for some z. The string z also matches the regular expression $a^*c(ba^*c)^*$, every prefix of z has at least as many a's as b's, and z has more a's than b's. The induction hypothesis implies that $U \Rightarrow^*_{G_1} z$, and therefore, $S_1 \Rightarrow U \Rightarrow aTbU \Rightarrow^*_{G_1} x$.

(c) We show that for any $n \geq 1$, and any x, if x can be derived from one of the three variables in n steps, then x has only one leftmost derivation from that variable. The basis step, $n = 1$, is clear. Suppose that $k \geq 1$ and that any string derivable from one of the variables in k or fewer steps has only one leftmost derivation from that variable. We wish to show the same result for a string x derivable from one of the variables in $k + 1$ steps.

It is easy to see that strings derivable from T have equal numbers of a's and b's, and those derived from U have an excess of a's. Therefore, if $S_1 \Rightarrow^* x$, the first step in a derivation of x from S_1 must be $S_1 \Rightarrow T$ if $n_a(x) = n_b(x)$, and $S_1 \Rightarrow U$ otherwise. Therefore, the induction hypothesis implies the result for strings derivable in $k + 1$ steps from S_1.

Suppose x is derivable from T in $k + 1$ steps. Since $k \geq 1$, the first step in a derivation from T must be $T \Rightarrow aTbT$. Thus $x = aybz$ for some strings y and z derivable from T in k or fewer steps. It is also easy to verify that in any string derivable from S, or in particular from T, every prefix has at least as many a's as b's. For this reason, the prefix ayb in the formula $x = aybz$ must be the smallest prefix ending with b having equal numbers of a's and b's. (Otherwise, $y = y_1by_2$, where ay_1b has equal numbers of a's and b's; but then the prefix y_1b of y has more b's than a's.) The induction hypothesis implies that both y and z have only one leftmost derivation from T. Since there is no choice for the strings y and z, it follows that x also has only one leftmost derivation from T.

Finally, suppose x is derivable from U in $k + 1$ steps. If some prefix of x is of the form ayb, where y has equal numbers of a's and b's, then the first step in a derivation of x from U must be $U \Rightarrow aTbU$. (If a derivation started $U \Rightarrow aS_1$, some string derivable from S_1 would have a prefix with more b's than a's.) Furthermore, if no prefix of x has this form, the first step in a derivation of x from U must be $U \Rightarrow aS_1$. If the first step is $U \Rightarrow aS_1$, the induction hypothesis tells us that there is only one way to continue a leftmost derivation, and therefore in this case x has only one leftmost derivation from U. If the first step is $U \Rightarrow aTbU$, then $x = aybz$, where y and z can be derived from T and U, respectively, in k or fewer steps. By the induction hypothesis, y has only one leftmost derivation from T and z has only one from U. Furthermore, just as in the preceding paragraph, there is only one choice for the string y. Therefore, x has only one leftmost derivation from U.