

Solutions to Selected Exercises for Chapter 11 of R. Gregory Taylor, *Models of Computation and Formal Languages* (Oxford University Press: New York, 1998)

© Oxford University Press

Solutions to Exercises for § 11.1

11.1.1 (a)

| | |
|--------------------------|-----------|
| $S \Rightarrow aAB$ | by (i) |
| $\Rightarrow aaAXB$ | by (iii) |
| $\Rightarrow aaaXXB$ | by (iv) |
| $\Rightarrow aaaXXbBd$ | by (v) |
| $\Rightarrow aaaXXbbYdd$ | by (vi) |
| $\Rightarrow aaaXbXbYdd$ | by (vii) |
| $\Rightarrow aaaXbbXYdd$ | by (vii) |
| $\Rightarrow aaabXbXYdd$ | by (vii) |
| $\Rightarrow aaabbXXYdd$ | by (vii) |
| $\Rightarrow aaabbXYcdd$ | by (viii) |
| $\Rightarrow aaabbYccdd$ | by (viii) |
| $\Rightarrow aaabbccdd$ | by (ix) |

(b)

| | |
|--------------------|---------|
| $S \Rightarrow aB$ | by (i) |
| $\Rightarrow abYd$ | by (vi) |
| $\Rightarrow abcd$ | by (ix) |

11.1.2 (a) We number the productions of this grammar as follows.

| | |
|-------------|---------------------------------------|
| (i)–(ii) | $S \rightarrow aS'bXlabX$ |
| (iii)–(vii) | $S' \rightarrow aS'bCIS'bCIS'CIBICIC$ |
| (viii) | $Cb \rightarrow bC$ |
| (ix) | $CX \rightarrow Xc$ |
| (x) | $X \rightarrow c$ |

We then have the following derivation:

| | |
|---------------------------|-----------------------------------|
| $S \Rightarrow aS'bX$ | by (i) |
| $\Rightarrow aaS'bCbX$ | by (iii) |
| $\Rightarrow aaS'bCbCbX$ | by (iv) |
| $\Rightarrow aaCbCbCbX$ | by (vii) |
| $\Rightarrow^* aabbbCCCX$ | by several applications of (viii) |
| $\Rightarrow^* aabbbXccc$ | by three applications of (ix) |
| $\Rightarrow aabbbccccc$ | by (x) |

(b)

| | |
|---------------------|---------|
| $S \Rightarrow abX$ | by (ii) |
| $\Rightarrow abc$ | by (x) |

11.1.3 (a) $\{a^n(bc)^n | n > 0\}$

- (b) $S \rightarrow aAC$
 $A \rightarrow aAC|aC$
 $C \rightarrow Bc$
 $aB \rightarrow ab$
 $bB \rightarrow bb$
 $cB \rightarrow Bc$

(c) We refer to the productions of (b) using numbers (i) through (vii) in the usual way.

| | |
|----------------------|----------|
| $S \Rightarrow aAC$ | by (i) |
| $\Rightarrow aaCC$ | by (iii) |
| $\Rightarrow aaBcC$ | by (iv) |
| $\Rightarrow aaBcBc$ | by (iv) |
| $\Rightarrow aabcBc$ | by (v) |
| $\Rightarrow aabBcc$ | by (vii) |
| $\Rightarrow aabbcc$ | by (vi) |

It can be seen that generating word $a^n b^n c^n$ requires a total of n initial applications of productions (i)–(iii). Then n applications of (iv) serve to introduce n copies of string Bc . A total of n applications of (v) and (vi) convert B s to bs . However, before a B can be rewritten as b , it must be moved to the left past any and all occurrences of c using (vii). And it is this step that dominates our analysis, although our example does not illustrate this very well. To be precise, the n^{th} occurrence of B from the left in

$aaaa\dots aaaaBcBcBcBc\dots BcBcBcBc$

must be moved to the left past $n-1$ occurrences of c before it can be rewritten. By Gauss, this step requires $O(n^2)$ applications of production (vii) total. Since all other steps as described above require only $O(n)$ steps, our overall analysis is $O(n^2)$ production applications.

The grammar of Example 11.1.1 functions in essentially the same way and also requires $O(n^2)$ production applications to generate word $a^n b^n c^n$.

11.1.5. (a)

| | |
|------------|--------------------------|
| (i)–(ii) | $S \rightarrow aABW aBW$ |
| (iii)–(iv) | $A \rightarrow aXY aXY$ |
| (v)–(vi) | $B \rightarrow bBd bYd$ |
| (vii) | $Xb \rightarrow bX$ |
| (viii) | $XY \rightarrow YX$ |
| (ix) | $XY \rightarrow Yc$ |
| (x) | $Y \rightarrow c$ |
| (xi) | $Yb \rightarrow bX$ |
| (xii) | $Yc \rightarrow cX$ |
| (xiii) | $Yd \rightarrow dX$ |
| (xiv) | $YW \rightarrow We$ |
| (xv) | $W \rightarrow e$ |

(b) We have the following derivations within the grammar at (a) above.

| | |
|----------------------------|--|
| $S \Rightarrow aABW$ | by (i) |
| $\Rightarrow aaXZBW$ | by (iv) |
| $\Rightarrow aaXZBW$ | by (iv) |
| $\Rightarrow aaXZbBdW$ | by (v) |
| $\Rightarrow aaXZbbYddW$ | by (vi) |
| $\Rightarrow aaZXbbYddW$ | by (viii) |
| $\Rightarrow^* aaZbbXYddW$ | by two applications of (vii) |
| $\Rightarrow aaZbbYcdaW$ | by (ix) |
| $\Rightarrow aaZbbccddW$ | by (x) |
| $\Rightarrow^* aabbccddZW$ | by two applications each of (xi), (xii), and (xiii) |
| $\Rightarrow aabbccddWe$ | by (xiv) |
| $\Rightarrow aabbccdde$ | by (xv) |
| | |
| $S \Rightarrow aBW$ | by (ii) |
| $\Rightarrow abYdW$ | by (vi) |
| $\Rightarrow abcdW$ | by (x) |
| $\Rightarrow abcde$ | by (xv) |

(c) We could modify the grammar given at (a) above so as to permit the generation of words such as $aaccee$ and $bbbddd$. There is an easier approach, however. First of all, recall that our goal is a context-sensitive grammar G^* that generates the language $L = \{a^i b^j c^k d^l e^m \mid i, j \geq 0\} \setminus \{\epsilon\}$. Also, note that L is the union of the three languages

$$\begin{aligned} L_1 &= \{a^i b^j c^k d^l e^m \mid i, j \geq 1\} \\ L_2 &= \{a^i c^k e^m \mid i \geq 1\} \\ L_3 &= \{b^j d^l \mid j \geq 1\} \end{aligned}$$

and that each of L_1 , L_2 , and L_3 is generated by a context-sensitive grammar:

- Language L_1 is generated by the grammar—call it G_1 —given at (a) above.
- Language L_2 is generated by a context-sensitive grammar—call it G_2 —that is a trivial variant of the grammar of Example 11.1.1.
- Language L_3 is easily seen to be generated by a context-free—and hence context-sensitive—grammar G_3 (cf. Example 9.8.1).

Given grammars G_1 , G_2 , and G_3 , we can construct context-sensitive grammar G^* generating L as follows. First, reletter the nonterminals of grammars G_1 , G_2 , and G_3 in such a way that no nonterminal occurs in any two of them. Suppose, further, that the start symbols of G_1 , G_2 , and G_3 are S_1 , S_2 , and S_3 , respectively. Then the productions of G^* will be all those of G_1 , G_2 , and G_3 together with the three productions

$$S \rightarrow S_1 S_2 S_3$$

where S is the new start symbol of G^* . It should be clear that G^* will generate precisely $L = L_1 \cup L_2 \cup L_3$. (With this example we have in essence demonstrated the closure of the family of context-sensitive languages under union—see Theorem 11.5.)

Solutions to Exercises for § 11.3

- 11.3.4. (a) We have only to modify the linear-bounded automaton M of Example 11.3.2 slightly. Namely, if copies of divisor k should ever fit perfectly between λ and ρ , then $k|i$ holds, in which case M has only to erase everything except λ and ρ and halt scanning an accepting I . On the other hand, if division by successive divisors never produces remainder 0, then i is not composite and M may halt in any nonaccepting configuration.

Solutions to Exercises for § 11.4

11.4.2. We take the forward direction first. Suppose that $M = \langle Q, \Sigma, \Gamma, q_{init}, \delta_M \rangle$ accepts language L "by I ." As in the past, we describe the construction of linear-bounded automaton $M' = \langle Q', \Sigma, \Gamma, q_{init}, \{q_t\}, \delta_{M'} \rangle$ in terms of a modification of the transition diagram of M . As in the proof of Theorem 2.3, for every state $q \in Q$ for which there is no arc from q for symbol I , we add to the transition diagram of M a routine starting at q that checks to see whether M has halted reading a I on a tape that is otherwise blank except for endmarkers λ and ρ . If so, then M' enters its unique terminal state q_t and halts (cf. Figure 2.2.6). The foregoing makes clear that if M accepts w , then M' accepts w as well. Suppose, on the other hand, that M' accepts w . Then M' must have halted in its unique terminal state q_t . But, by the construction of M' , the only way for this to happen is for M to have halted scanning an accepting I . So M accepts w as well. In short, we have $L(M) = L(M')$.

As for the backward direction, suppose that $M' = \langle Q', \Sigma, \Gamma, q_{init}, F, \delta_{M'} \rangle$ is a linear-bounded automaton that accepts L by terminal state. We can assume without loss of generality that if M' does not accept a given input word, then M' never halts for that input word. (Cf. the solution to Exercise 2.10.8(a).) In other words, we can assume that if M' does halt, then it halts in a terminal state. Continuing, we construct M from M' as follows. For each terminal state $q' \in F$ and each symbol $\sigma \in \Gamma$ such that no arc from q' is labeled by σ , we add to the transition diagram of M' a routine beginning at q' whose initial arc is labeled by σ . This routine erases the tape contents of M' 's tape except for endmarkers λ and ρ and writes a single I adjacent to λ before halting. It is hoped that the reader can see that if M' accepted w , then, by the described construction, so does M . On the other hand, if M accepts w , then M' , given w as input, must have halted in a terminal state, thereby accepting w as well. In other words, $L(M) = L(M')$ holds. Q.E.D.

Solutions to Exercises for § 11.6

- | | | |
|---------|----------------------------|----------|
| 11.6.1. | $S \Rightarrow aSS_1$ | by (i) |
| | $\Rightarrow aaSS_1S_1$ | by (i) |
| | $\Rightarrow aaaS_2S_1S_1$ | by (ii) |
| | $\Rightarrow aaabS_2cS_1$ | by (iii) |
| | $\Rightarrow aaabS_2S_1c$ | by (iv) |
| | $\Rightarrow aaabbS_2cc$ | by (iii) |
| | $\Rightarrow aaabbbccc$ | by (v) |

11.6.3. We start by noting that, whereas the nodes of a parse tree are labeled by individual symbols, the nodes of the derivation tree of Figure 11.6.1 are labeled by entire derivations.

- In addition, a parse tree for word w assumes some context-free grammar G and, in fact, such a parse tree can exist if and only if $w \in L(G)$.
- A derivation tree, on the other hand, assumes some context-sensitive grammar G' and some (maximum) length n . Typically, n is the length of some word w' that is of interest. Such a tree may or may not have a derivation of w' labeling one of its nodes.

11.6.4. Let L be a context-sensitive language over Σ and let $G = \langle \Sigma, \Gamma, \sigma, \Pi \rangle$ be a context-sensitive grammar that generates L . We describe a multitape Turing machine M that recognizes L . Suppose

that word w over Σ appears on M 's input tape. M writes G 's start symbol σ on its worktape₁, say. Then M proceeds in accordance with the decision algorithm of the proof of Theorem 11.7. Namely, assuming some ordering of the members of Π , individual productions are examined in sequence for potential application to the current contents of worktape₁ and, if applicable, are applied to the string on worktape₁. After any such application, M checks to see whether the resulting string is identical to w and, if so, writes an accepting I to its output tape. If not, M determines whether the length of the new string is at least as great as $|w|$. If so, then M backtracks in order to consider other possible paths to w . (One imagines that other worktapes will be involved in implementation of this backtracking.) Eventually, assuming that w never appears on worktape₁, M will have exhausted all possibilities for deriving w within G . In this case, M will write a rejecting O to its output tape and halt. (The foregoing description coincides with a depth-first approach. An alternative breadth-first approach is available.)

Solutions to Exercises for § 11.7

11.7.1 We show that the language L_{diag} is Turing-recognizable by describing a multitape Turing machine M that recognizes it. So suppose that word w over Σ appears initially on M 's input tape. We may imagine that M first determines w 's position within the enumeration (11.7.1). Suppose that, as it turns out, $w=w_j$. Next, M must proceed through the enumeration (11.7.2) of the generative grammars far enough so as to obtain the j^{th} context-sensitive grammar in that enumeration. (This determination is effective since it is merely a matter of verifying that every production of a given grammar is length-preserving.) In any case, the j^{th} context-sensitive grammar at (11.7.2) is what we are calling G_j at (11.7.3). So now M must determine whether word $w=w_j$ is in $L(G_j)$. It can do this by implementing the algorithm of the proof of Theorem 11.7. (See our solution to Exercise 11.6.4.) If it turns out that $w=w_j$ is in $L(G_j)$, then M writes a rejecting O to its output tape and halts (cf. (11.7.4)). On the other hand, if $w=w_j$ is not in $L(G_j)$, then M writes an accepting I instead.

Solutions to Exercises for § 11.8

11.8.3 (a) All eight productions are length-preserving.

(b) $S \Rightarrow YXXY$ by (i)
 $\Rightarrow^* aaaa$ by two applications each of (vii) and (viii)

(c) $S \Rightarrow YXXY$ by (i)
 $\Rightarrow YXXXZXY$ by (iv)
 $\Rightarrow YXXXXXXXXZY$ by (v)
 $\Rightarrow YXXXXXXXXXY$ by (vi)
 $\Rightarrow^* a^8$ by several applications each of (vii) and (viii)

11.8.4 (b)

$S \Rightarrow X_1AAAAAAS_3$ by (i)
 $\Rightarrow X_1AAAAAAAAS_3$ by (vi)
 $\Rightarrow S_1AAAS_1AAAAAS_3$ by (vii) /* Note that this string contains 11 symbols.

*/

$\Rightarrow S_1ADX_2Y_1EAAAAS_3$ by (viii) /* Begin division by 3. */
 $\Rightarrow S_1DX_2AY_1EAAAAS_3$ by (x)
 $\Rightarrow S_1BX_3AY_1EAAAAS_3$ by (xii)
 $\Rightarrow S_1BAX_3Y_1EAAAAS_3$ by (xiii)
 $\Rightarrow S_1BAY_1X_3EAAAAS_3$ by (xv)
 $\Rightarrow^* S_1BAY_1AAAAX_3ES_3$ by four applications of (xvii)
 $\Rightarrow S_1BAY_1AAAAX_4CS_3$ by (xix)
 $\Rightarrow^* S_1BAY_1X_4AAAACS_3$ by four applications of (xx)

$\Rightarrow S_1BAAS_2AAAACS_3$
 $\Rightarrow S_1BDX_2Y_1EAAACS_3$
 $\Rightarrow S_1DX_2BY_1EAAACS_3$
 $\Rightarrow S_1BX_3BY_1EAAACS_3$
 $\Rightarrow S_1BX_3BY_1EAAACS_3$
 $\Rightarrow S_1BBX_3Y_1EAAACS_3$
 $\Rightarrow S_1BBY_1X_3EAAACS_3$
 $\Rightarrow^* S_1BBY_1AAAX_3ECS_3$
 $\Rightarrow S_1BBY_1AAAX_4CCS_3$
 $\Rightarrow^* S_1BBY_1X_4AAACCS_3$
 $\Rightarrow S_1BBAS_2AAACCS_3$
 $\Rightarrow S_1BDX_2Y_2EAACCS_3$
 $\Rightarrow S_1DX_2BY_2EAACCS_3$
 $\Rightarrow S_1BX_3BY_2EAACCS_3$
 $\Rightarrow S_1BBX_3Y_2EAACCS_3$
 $\Rightarrow S_1BBY_2X_3EAACCS_3$
 $\Rightarrow^* S_1BBY_2AAX_3ECCS_3$
 $\Rightarrow S_1BBY_2AAX_4CCS_3$
 $\Rightarrow^* S_1BBY_2X_4AACCCS_3$
 $\Rightarrow S_1BBBS_2AACCCS_3$
 $\Rightarrow S_1BBBS_2AACCCS_3$
 $\Rightarrow S_1X_3BBS_2AACCCS_3$
 $\Rightarrow^* S_1BBX_5S_2AACCCS_3$
 $\Rightarrow S_1BBX_6S_2AACCCS_3$
 $\Rightarrow^* S_1X_6AAS_2AACCCS_3$
 $\Rightarrow S_1AAAS_2AACCCS_3$
 $\Rightarrow S_1ADX_2Y_1EACCCS_3$
 $\Rightarrow S_1DX_2AY_1EACCCS_3$
 $\Rightarrow S_1BX_3AY_1EACCCS_3$
 $\Rightarrow S_1BAX_3Y_1EACCCS_3$
 $\Rightarrow S_1BAY_1X_3EACCCS_3$
 $\Rightarrow S_1BAY_1AX_3ECCCS_3$
 $\Rightarrow S_1BAY_1AX_4CCCS_3$
 $\Rightarrow S_1BAY_1X_4ACCCS_3$
 $\Rightarrow S_1BAAS_2ACCCS_3$
 $\Rightarrow S_1BDX_2Y_1ECCCS_3$
 $\Rightarrow S_1DX_2BY_1ECCCS_3$
 $\Rightarrow S_1BX_3BY_1ECCCS_3$
 $\Rightarrow S_1BBX_3Y_1ECCCS_3$
 $\Rightarrow S_1BBY_1X_3ECCCS_3$
 $\Rightarrow S_1BBY_1X_4CCCCS_3$
 $\Rightarrow S_1BBAS_2CCCCS_3$
 $\Rightarrow S_1BBAS_2CCCCS_3$
 $\Rightarrow S_1BBAS_2CCCCX_7S_3$
 $\Rightarrow^* S_1BBAS_2X_7AAAAS_3$
 $\Rightarrow S_1BX_8AAS_2AAAAS_3$
 $\Rightarrow S_1X_8AAAAS_2AAAAS_3$
 $\Rightarrow S_1AAAAS_2AAAAS_3$
...
 $\Rightarrow^* S_1AAAAAS_2AAAS_3$

by (xxii)
 by (viii)
 by (xi)
 by (xii)
 by (xii)
 by (xiv)
 by (xv)
 by three applications of (xvii)
 by (xix)
 by three applications of (xxi)
 by (xxiii)
 by (ix)
 by (xi)
 by (xii)
 by (xiv)
 by (xvi)
 by two applications of (xvii)
 by (xx)
 by two applications of (xxi)
 by (xxiv)
 by (xxiv)
 by (xxv)
 by two applications of (xxvi)
 by (xxvii)
 by two applications of (xxviii)
 by (xxix)
 by (viii)
 by (x)
 by (xii)
 by (xiii)
 by (xv)
 by (xvii)
 by (xx)
 by (xxi)
 by (xxiii)
 by (ix)
 by (xi)
 by (xii)
 by (xiv)
 by (xv)
 by (xx)
 by (xxiii)
 by (xxiii) /* Division by 3 unsuccessful. */
 by (xxx)
 by four applications of (xxxii)
 by (xxxiii)
 by (xxxvii)
 by (xxxviii) /* Ready to begin division by 4. */
 by (viii)-(xxxviii) /*Ready to begin division by 5.

Neither production (viii) nor production (ix) can now be applied again. Only production (xxx) is useful now. (Applying production (xxv) leads

*/

| | | |
|----|---|---|
| */ | $\Rightarrow^* S_1 A A A A A A S_2 A A S_3$ | by (viii)–(xxxviii) /*Ready to begin division by 6. |
| | ... | |
| */ | $\Rightarrow^* S_1 A A A A A A A A S_2 A S_3$ | by (viii)–(xxxviii) /*Ready to begin division by 7. |
| | $\Rightarrow S_1 A A A A A D X_2 Y_1 E S_3$ | by (viii) |
| | $\Rightarrow^* S_1 D X_2 A A A A A Y_1 E S_3$ | by five applications of (x) |
| | $\Rightarrow S_1 B X_3 A A A A A Y_1 E S_3$ | by (xii) |
| | $\Rightarrow^* S_1 B A A A A A X_3 Y_1 E S_3$ | by five applications of (xiii) |
| | $\Rightarrow S_1 B A A A A A Y_1 X_3 E S_3$ | by (xv) |
| | $\Rightarrow S_1 B A A A A A Y_1 X_4 C S_3$ | by (xix) |
| | $\Rightarrow S_1 B A A A A A A S_2 C S_3$ | by (xxiii) /* Division by 7 unsuccessful. */ |
| | $\Rightarrow S_1 B A A A A A A S_2 X_7 S_3$ | by (xxx) |
| | $\Rightarrow S_1 B A A X_8 A A A A S_2 S_3$ | by (xxxv) |
| | $\Rightarrow^* S_1 B X_8 A A A A A A S_2 S_3$ | by two applications of (xxxvi) |
| | $\Rightarrow S_1 X_8 A A A A A A A S_2 S_3$ | by (xxxvii) |
| | $\Rightarrow S_1 A A A A A A A A S_2 S_3$ | by (xxxviii) |
| | $\Rightarrow S_1 A A A A A A A A X_9 a$ | by (xxxix) |
| | $\Rightarrow^* S_1 X_9 a a a a a a a a a$ | by eight applications of (xL) |
| | $\Rightarrow a a a a a a a a a a$ | by (xLi) |

| | | |
|------------|-------------------------------------|-------------|
| 11.8.5 (a) | $S \rightarrow XW \mid YZ$ | (i)–(ii) |
| | $X \rightarrow aXA \mid bXB \mid a$ | (iii)–(v) |
| | $Y \rightarrow aYA \mid bYB \mid b$ | (vi)–(viii) |
| | $Aa \rightarrow aA$ | (ix) |
| | $Ab \rightarrow bA$ | (x) |
| | $Ba \rightarrow aB$ | (xi) |
| | $Bb \rightarrow bB$ | (xii) |
| | $AW \rightarrow aW$ | (xiii) |
| | $BW \rightarrow bW$ | (xiv) |
| | $AZ \rightarrow aZ$ | (xv) |
| | $BZ \rightarrow bZ$ | (xvi) |
| | $W \rightarrow a$ | (xvii) |
| | $Z \rightarrow b$ | (xviii) |

| | | |
|-----|------------------------------|-------------------|
| (b) | $S \Rightarrow XW$ | by (i) |
| | $\Rightarrow aXAW$ | by (iii) |
| | $\Rightarrow^* ababaXABABAW$ | by (iii) and (iv) |
| | $\Rightarrow ababaaABABAW$ | by (v) |
| | $\Rightarrow ababaaABABaW$ | by (xiii) |
| | $\Rightarrow ababaaABAaBW$ | by (xi) |
| | $\Rightarrow ababaaABAabW$ | by (xiv) |
| | $\Rightarrow^* ababaaababaw$ | by (ix)–(xiv) |
| | $\Rightarrow ababaaababaa$ | by (xvii) |

11.8.6 (a) (\Leftarrow). Suppose that both L and L^c are LBA-acceptable. Let M and M^c be linear-bounded automata that accept L and L^c , respectively. We construct a new linear-bounded automaton M^* that recognizes L . M^* 's tape will be divided into two tracks in the familiar way. Given input word w , M^* begins by replicating w on both tracks and then proceeds to simulate M on the upper track and M^c on the lower track, say. M^* would alternate between its two simulations: perhaps five steps on the upper track and then five on the lower track. Eventually, one of the two simulations will halt in an accepting configuration. If this occurs on the upper track, then M^* will eliminate the two tracks and write an accepting 1 before halting. On the other hand, if it is the lower simulation

that halts in an accepting configuration, then M^* eliminates the two tracks, writes a rejecting 0, and halts.

(\Rightarrow). Suppose that L is LBA-recognizable. It follows immediately that L is LBA-acceptable since any language recognizer is a fortiori a language acceptor as well. As for L^c , a simple transformation turns a machine M recognizing L into a machine M^c recognizing, and hence accepting, L^c : whenever M would write a 1, let M^c write a 0, and whenever M would write a 0, let M^c write a 1.

(b) (\Rightarrow). Suppose that every LBA-acceptable language is LBA-recognizable and let L be an arbitrary context-sensitive language. By Theorem 11.2, L is LBA-acceptable and hence LBA-recognizable. But then, by (a), L^c is LBA-acceptable and hence context-sensitive by Theorem 11.4. But L was an arbitrary context-sensitive language, and we have now seen that its complement is context-sensitive as well.

(\Leftarrow). Suppose that the complement of any context-sensitive language is itself context-sensitive and let L be an arbitrary LBA-acceptable language. By Theorem 11.4, L is context-sensitive. By hypothesis, L^c is context-sensitive as well. Further, by Theorem 11.2, L^c is LBA-acceptable. So we have established that both L and L^c are LBA-acceptable. By (a), we have that L is LBA-recognizable. But we started by assuming that L is an arbitrary LBA-acceptable language, and we have now seen that it is LBA-recognizable as well.

11.8.7. The argument here should be perfectly familiar by now. Let G be a context-sensitive grammar with $L=L(G)$. Form grammar G^R by replacing each production

$$\alpha \rightarrow \beta$$

of G with the new production

$$\alpha' \rightarrow \beta'$$

Grammar G^R is context-sensitive and $L(G^R)=L^R$.

11.8.8. We give what is essentially the same proof as was given for the class of context-free languages. (Cf. the solution to Exercise 10.6.1.) Suppose that both L_1 and L_2 are context-sensitive languages. Without loss of generality, we may assume that $L_1 \setminus \{\varepsilon\} = L(G_1)$ and $L_2 \setminus \{\varepsilon\} = L(G_2)$, where G_1 and G_2 are context-sensitive grammars in normal form in the sense of Theorem 11.1. (In particular, this means that neither involves any productions whose left-hand sides consist of terminals only.) We imagine that the nonterminals of G_1 and G_2 are relettered so as to avoid conflicts. Finally, let grammar G^* be that context-sensitive grammar whose productions are those of G_1 together with those of G_2 as well as a new production

$$S \rightarrow S_1 S_2$$

where S_1 and S_2 are the start symbols of G_1 and G_2 , respectively, and S is the start symbol of grammar G^* . It is easy to see that $L(G^*) = L(G_1) \cdot L(G_2) = L_1 \setminus \{\varepsilon\} \cdot L_2 \setminus \{\varepsilon\}$. There are several cases to consider.

- If, in fact, neither L_1 nor L_2 contains ε , then $L(G^*) = L_1 \setminus \{\varepsilon\} \cdot L_2 \setminus \{\varepsilon\} = L_1 \cdot L_2$, from which it follows that $L_1 \cdot L_2$ is context-sensitive.
- If L_1 contains ε but L_2 does not, then $L_2 \subseteq L_1 \cdot L_2$. Adding production $S \rightarrow S_2$ in turn ensures that $L_2 \subseteq L(G^*)$ so that $L(G^*) = (L_1 \setminus \{\varepsilon\} \cdot L_2) \cup (L_2) = L_1 \cdot L_2$ so that, again, $L_1 \cdot L_2$ is seen to be context-sensitive.
- If L_2 contains ε but L_1 does not, then $L_1 \subseteq L_1 \cdot L_2$. We add production $S \rightarrow S_1$.

- If both L_1 and L_2 contain ϵ , then $L_1 \cup L_2 \subseteq L_1.L_2$. We add production $S \rightarrow S_1$ as well as production $S \rightarrow S_2$ so that $L(G^*) = (L_1 \setminus \{\epsilon\}.L_2 \setminus \{\epsilon\}) \cup (L_1 \cup L_2) \setminus \{\epsilon\} = (L_1.L_2) \setminus \{\epsilon\}$. In this case, too, language $L_1.L_2$ is seen to be context-sensitive by Definition 11.2.

Incidentally, we could have avoided the appeal to Theorem 11.1 by adding nonterminal representatives X_a , X_b , and so forth, throughout G_1 and G_2 before relettering.

11.8.9. Suppose that $L \setminus \{\epsilon\} = L(G)$ for context-sensitive grammar G with start symbol S . Let G' be the result of adding one production, namely,

$$S \rightarrow SS$$

to those of G . It should be clear that $L(G') = (L \setminus \{\epsilon\})^+ = L^* \setminus \{\epsilon\}$ so that language L^* is seen to be context-sensitive by Definition 11.2.

11.8.10 (a) We do have closure under relative complementation since

$$L_1 \setminus L_2 = L_1 \cap L_2^c$$

(Recall that the family of context-sensitive languages is closed under intersection.)

(b) We do have closure under symmetric difference since

$$L_1 \oplus L_2 = (L_1 \setminus L_2) \cup (L_2 \setminus L_1)$$

(Recall that the family of context-sensitive languages is closed under union, and use (a).)

11.8.11 (a)

$$\begin{aligned} S &\rightarrow X_a S B C \mid X_a B C \mid \\ C B &\rightarrow C Y_2 \\ C Y_2 &\rightarrow Y_1 Y_2 \\ Y_1 Y_2 &\rightarrow Y_1 C \\ Y_1 C &\rightarrow B C \\ X_a B &\rightarrow X_a X_b \\ X_b B &\rightarrow X_b X_b \\ X_b C &\rightarrow X_b X_c \\ X_c C &\rightarrow X_c X_c \\ X_a &\rightarrow a \\ X_b &\rightarrow b \\ X_c &\rightarrow c \end{aligned}$$

(b) (\Rightarrow). Suppose that language L is context-sensitive₁. Then L is generated by a context-sensitive₁ grammar G with start symbol S , say. Grammar G may or may not contain production $S \rightarrow \cdot$.

- Suppose, first, that it does not contain production $S \rightarrow \cdot$. Then all productions of G are length-preserving so that G itself is context-sensitive in the sense of Definition 11.1, and hence L is context-sensitive in the sense of Definition 11.2.
- On the other hand, suppose that G does contain production $S \rightarrow \cdot$. Then, since S can have no occurrence on the right-hand side of any production, it follows that omitting production $S \rightarrow \cdot$ only yields a new grammar G' with $L(G') = L \setminus \{\epsilon\}$. Since each production of G' is length-preserving, it follows

that G' is context-sensitive and that L itself is context-sensitive in the sense of Definition 11.2.

(\Leftarrow). Suppose that L is context-sensitive. Then, by Theorem 11.1, we have $L=L(G)$ for some context-sensitive grammar G in normal form. But any such G is context-sensitive₁, from which it follows that L is context-sensitive₁.