

Solutions to Selected Exercises for Chapter 10 of R. Gregory Taylor, *Models of Computation and Formal Languages* (Oxford University Press: New York, 1998)

© Oxford University Press

Solutions to Exercises for § 10.1

- 10.1.1 (b) Language $\{a^n b^m \mid m=n \text{ or } m=2n \text{ with } n \geq 0\}$ does contain ϵ .
(c) Language $\{a^n b^m \mid m < n \text{ with } m \geq 0 \text{ or } m > 2n \text{ with } n \geq 0\}$ does not contain ϵ .

10.1.2 (b) $S \rightarrow aSa \mid bSb \mid a \mid b \mid$

(c) Consider the following inductive definition first.

- (i) ϵ is in L .
- (ii) If w is in L , so are $waab, awab, aawb, aabw,$
 $waba, awba, abwa, abaw,$
 $wbaa, bwaa, bawb, and baaw.$
- (iii) Nothing else is in L .

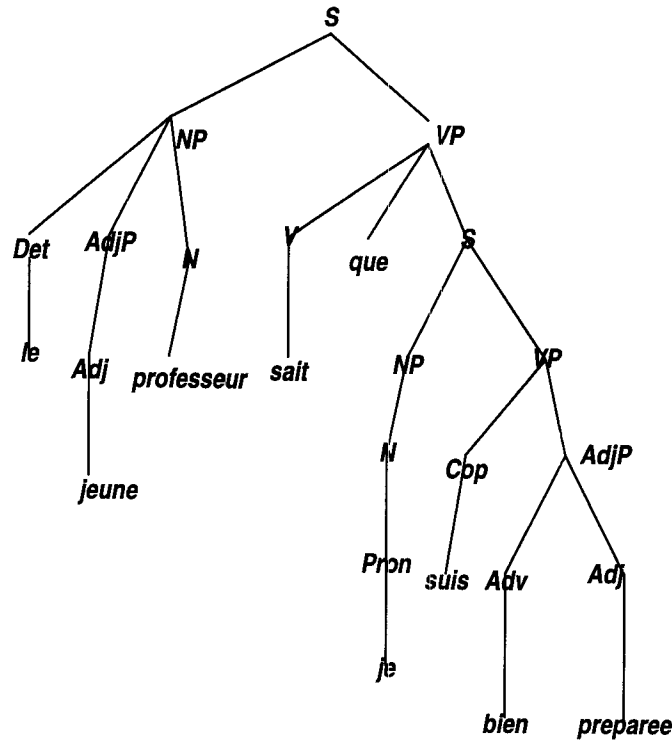
Clause (ii) gives us twelve productions

$$\begin{aligned} S \rightarrow & Saab \mid aSab \mid aaSb \mid aabS \mid \\ & Saba \mid aSba \mid abSa \mid abaS \mid \\ & Sbaa \mid bSaa \mid baSa \mid baaS \end{aligned}$$

whereas clause (i) yields the single production

$$S \rightarrow$$

10.1.3.



10.1.5. [[**[el]**]_{Det}[[**[muy]**]_{Adv}[[**[famoso]**]_{Adj}]_{AdjP}[[**[poeta]**]_N[[**[nicaragüense]**]_{Adj}]_{AdjP}]_{NP}[[**[fue]**]_{Cop}[[**[enterrado]**]_{Adj}[[**[en]**]_{Prep}[[**[la]**]_{Det}[[**[gran]**]_{Adj}]_{AdjP}[[**[catedral]**]_N[[**[de]**]_{Prep}[[**[León]**]_N]_{NP}]_{PrepP}]_{AdjP}]_{NP}]_{PrepP}]_{AdvP}]_{VP}]_S

Solutions to Exercises for § 10.2

10.2.1 (a) $\{a^n b^m c^{2k} b^m a^n \mid n \geq 1, m \geq 0, k \geq 0\}$

(b) $S \rightarrow aSa \mid aXa \mid aa$
 $X \rightarrow bXb \mid Y \mid bb$
 $Y \rightarrow cYc \mid cc$

10.2.2 (a) $\{a^n b^m c^{2k} b^m a^n \mid n \geq 0, m \geq 0, k \geq 0\}$

- (b) $S \rightarrow aSa \mid X \mid aa$
 $X \rightarrow bXb \mid Y \mid bb$
 $Y \rightarrow cYc \mid cc$

One sees that $L(G) = L(G) \setminus \{\epsilon\}$.

- 10.2.3. $S \rightarrow aSa \mid aXa$
 $X \rightarrow bXb \mid cYc \mid$
 $Y \rightarrow cYc \mid$

- 10.2.4. $S \rightarrow aSa \mid bXb \mid cYc \mid$
 $X \rightarrow bXb \mid cYc \mid$
 $Y \rightarrow cYc \mid$

- 10.2.5 (a) $\{a^n b^m \mid 1 \leq n \leq m \leq 2n\}$
 (b) First, eliminating ϵ -productions produces the grammar with productions

$$S \rightarrow aSXb \mid aSb \mid aXb \mid ab$$

$$X \rightarrow b$$

Applying step (2) yields the grammar

$$S \rightarrow Z_a SX Z_b \mid Z_a S Z_b \mid Z_a X Z_b \mid Z_a Z_b$$

$$X \rightarrow b$$

$$Z_a \rightarrow a$$

$$Z_b \rightarrow b$$

In Step (3), we decompose the first three productions above so as to obtain an equivalent grammar in Chomsky normal form.

$$S \rightarrow Z_a W_1 \mid Z_a Y_1 \mid Z_a U_1 \mid Z_a Z_b$$

$$W_1 \rightarrow S W_2$$

$$W_2 \rightarrow X Z_b$$

$$Y_1 \rightarrow S Z_b$$

$$U_1 \rightarrow X Z_b$$

$$X \rightarrow b$$

$$Z_a \rightarrow a$$

$$Z_b \rightarrow b$$

10.2.7. Here is an algorithm for eliminating cycles from a context-free grammar. First, order the grammar nonterminals as

$$X_n > X_{n-1} > \dots > X_1$$

where X_n is the grammar start symbol. We define a *descending production* as one of the form $X_i \rightarrow \alpha$ where α is either empty or contains at least one terminal symbol or consists solely of terminals with indices strictly lower than i .

The algorithm itself amounts to starting with the productions for X_n and making substitutions for any that are not descending, then doing the same for the productions for X_{n-1} , and so forth. As for rendering productions descending, suppose that $i \leq j$ and that α and β consist solely of nonterminals so that production

$$X_i \rightarrow \alpha X_j \beta$$

is not descending. (Incidentally, X_j may have other occurrences in either α or β or both.) We then replace this production by new productions that result from replacing the displayed occurrence of X_j by the right-hand sides of the various descending productions for X_j . (And if none of the productions for X_j is descending so that no such substitutions can be made, then production $X_i \rightarrow \alpha X_j \beta$ is merely discarded.) This process may have to be repeated for some of the newly introduced productions.

Applied to grammar

$$S \rightarrow SS \mid (S) \mid$$

our algorithm requires replacing the nondescending production $S \rightarrow SS$ by the two productions $S \rightarrow S(S) \mid S$, say. This yields grammar

$$S \rightarrow S(S) \mid S \mid (S) \mid$$

The second production here is still nondescending, however. So the process must be repeated. The result is the grammar with the three descending productions

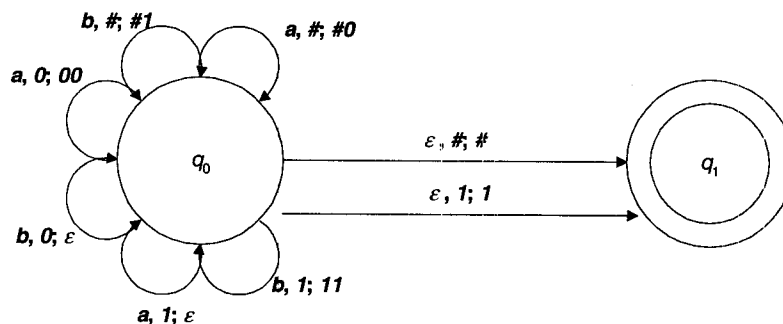
$$S \rightarrow S(S) \mid (S) \mid$$

Solutions to Exercises for § 10.3

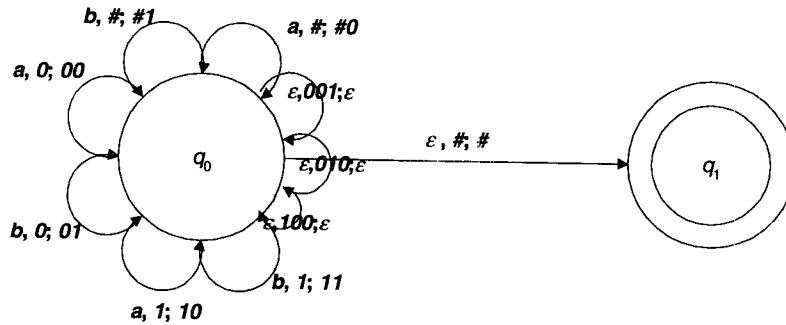
10.3.2. Here is the transition function of a machine M that accepts $\{a^n b^n \mid n \geq 0\}$. M 's one accepting state is q_2 .

$$\begin{aligned} \delta(q_0, a, \#) &= (\#, q_0) \\ \delta(q_0, a, 0) &= (00, q_0) \\ \delta(q_0, b, 0) &= (\varepsilon, q_1) \\ \delta(q_1, b, 0) &= (\varepsilon, q_1) \\ \delta(q_1, b, \#) &= (\#, q_2) \end{aligned}$$

10.3.3.



10.3.4. Our solution takes advantage of the fact that, in any nonempty word w having twice as many a s as b s, there must exist a substring containing two a s and one b (see proof below). Moreover, removing this substring from w results in a new word w' that, if nonempty, must also have such a substring.



As for our proof, suppose that nonempty word $w = a_1 a_2 \dots a_n$ contains twice as many a s as b s but that every substring of length 3 contains at least two b s. Clearly, we have $n \geq 3$. Consider now the array

$$(1) \quad \begin{array}{cccccccccccc} a_1 & a_2 & a_3 & a_4 & a_5 & \dots & \dots & a_{n-1} & a_n & & & \\ & & a_1 & a_2 & a_3 & a_4 & a_5 & \dots & \dots & a_{n-1} & a_n & \\ & & & a_1 & a_2 & a_3 & a_4 & a_5 & \dots & a_{n-2} & a_{n-1} & a_n \end{array}$$

where each of the three lines here consists of w . Of the $n+2$ columns here, precisely $n-2$ of them, read from bottom to top, consist of a substring of w of length 3. By assumption, then, each of these columns contains at least two b s, so that the $n-2$ columns of length 3 contain no fewer than $2n-4$ b s total. Moreover, it is easy to see that at least one b occurs in the first two partial columns, and at least one b must occur in the last two partial columns. Hence, of the $3n$ characters at (1), no fewer than $2n-2$ are b s. Further, by assumption, precisely $2n$ of the $3n$ characters at (1) are a s. It follows that there are at least $4n-2$ characters total at (1). But, since $n \geq 3$, we have

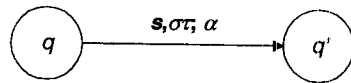
$$\begin{aligned} 4n-2 &= 3n + (n-2) \\ &> 3n \end{aligned}$$

which is a contradiction. We conclude that w must contain a substring consisting of two a s and a b .

10.3.6. Languages (a), (b), (c), and (e) are PSA-acceptable languages. In fact, accepting pushdown automata may be described that are even deterministic in the sense of § 10.8. For example, in the case of (c), symbol 0 might be pushed for each a read. Later symbol 0 is popped for each pair of c s read.

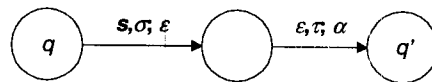
Language (d) is beyond anything of which a pushdown automaton is capable, as are (f), (g), and (h). Having emptied its stack in processing bs , say, a pushdown automaton can no longer remember anything regarding previously processed input. On the other hand, it is easy to see that language (i) is PSA-acceptable. Further, our work in § 10.6 will show that the family of PSA-acceptable languages is closed under concatenation. But (i)—as well as (e)—is the concatenation of two such languages.

10.3.8. Suppose that language L is accepted by a pushdown automaton M that pops one or more symbols from its symbol stack during any one instruction. We sketch the construction of a new machine M^* in the sense of Definition 10.4 such that $L(M^*) = L$. So suppose that, when in state q reading input symbol s with symbol σ and, below that, symbol τ on the symbol stack, M pops both σ and then τ , pushes symbol string α , and then enters state q' .



in M

In constructing M^* , we add a new intermediate state between q and q' and instructions for popping first σ and then τ . Thus

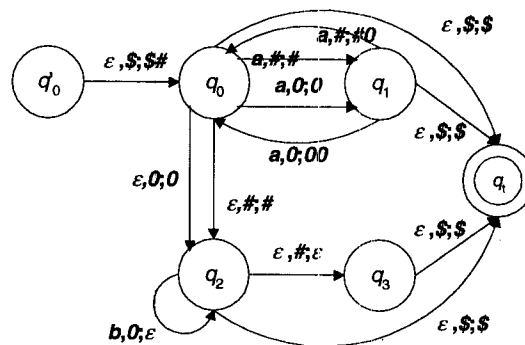


in M^*

In similar manner, any instruction of M that involves popping k symbols with $k > 1$ is replaced in M^* with a sequence of k instructions each of which involves popping exactly one symbol. (The construction of this instruction sequence will involve $k-1$ new states as well.) It should be clear that M^* will accept L .

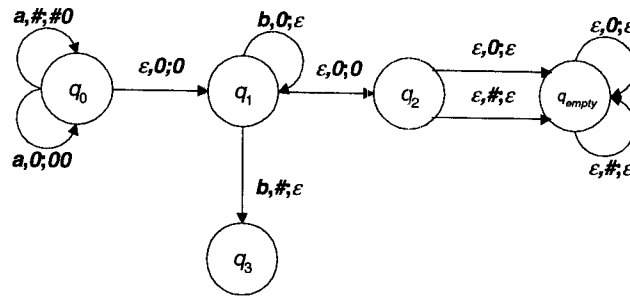
Solutions to Exercises for § 10.4

10.4.3. Note that, in the transition diagram appearing below, the four arcs leading to state q_i could just as well be labeled $\epsilon, \$, \epsilon$



10.4.4 (a) M accepts language $\{a^n b^m | n \geq 1 \text{ and } n > m\}$. Note that M does not accept word abb .

(b) The transition diagram of machine M' appears below.



(c) M' can be seen to accept word abb by empty stack. (It will find itself in state q_3 with its stack empty.)

Solutions to Exercises for § 10.6

10.6.4. It is necessary to modify both M_1 and M_2 before applying the construction of Theorem 10.9. It is easiest to talk about M_2 . So let us consider modifications to M_2 's transition diagram first. For every symbol s in $\Sigma_1 \setminus \Sigma_2$ and every state q of M_2 , we add an arc, labeled by s , leading from q to M_2 's trap state. This has the effect of making M_2 fully defined for $\Sigma_1 \cup \Sigma_2$, although it can be seen that M_2 will not accept any word containing a symbol of $\Sigma_1 \setminus \Sigma_2$. Something analogous must be done to the transition diagram of M_1 . Namely, for every symbol s in $\Sigma_2 \setminus \Sigma_1$, every symbol s' in Γ , and every state q of M_1 , we add an arc, labeled by s and s' and ϵ , say, leading from q to M_1 's trap state. This will mean that M_1 is fully defined for $\Sigma_1 \cup \Sigma_2$. At the same time, M_1 will accept no word containing any symbol of $\Sigma_2 \setminus \Sigma_1$.

The construction of the proof of Theorem 10.9 can now be applied to M_1 and M_2 as modified in the manner described above.

Solutions to Exercises for § 10.7

10.7.3. Suppose, for the sake of obtaining a contradiction, that the class of context-free languages were closed under complementation. Let L_1 and L_2 be two, arbitrary context-free languages. Then, by the assumption of closure under complementation, L_1^c and L_2^c are context-free as well. By Theorem 10.8 and closure under complementation, $(L_1^c \cup L_2^c)^c = L_1 \cap L_2$ is context-free, contradicting Theorem 10.11. Q.E.D.

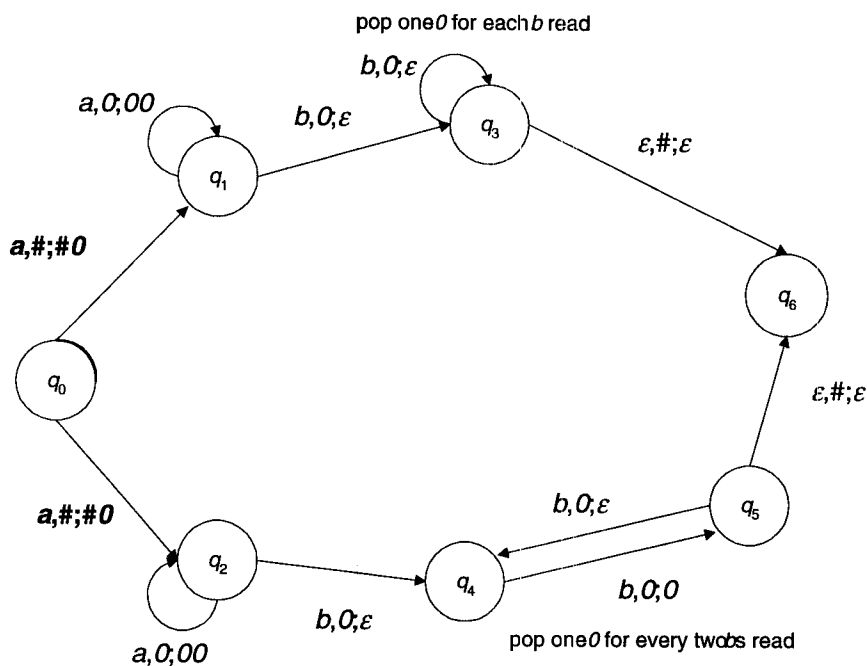
10.7.4. Suppose that the class of context-free languages were closed under set difference. Let L be an arbitrary context-free language over Σ . Of course, language Σ^* is regular. (If $\Sigma = \{a, b\}$, say, then Σ^* is $L((ab)^*)$.) Hence, Σ^* is context-free by Remark 10.1.1. But then, by the assumed closure under set difference, we have that $\Sigma^* \setminus L = L^c$ is context-free. But L was an arbitrary context-free language, and its complement has now been shown to be context-free as well. In other words, we have been led to conclude that the class of context-free languages is closed under complementation, which contradicts Theorem 10.12. Q.E.D.

Solutions to Exercises for § 10.8

10.8.1. Suppose, for the sake of proving a contradiction, that any language accepted by a nondeterministic pushdown automaton is accepted by some deterministic machine as well. Let L be an arbitrary context-free language. Then by Theorem 10.6 there exists a nondeterministic machine M_{nd} that accepts L . By

hypothesis, there also exists a deterministic machine M_d that accepts L . By Theorem 10.13, language L^c is context-free. But L was an arbitrary context-free language, which means that the class of context-free languages is closed under complementation, contradicting Theorem 10.12. We are forced to conclude that there exist languages that are accepted by some nondeterministic pushdown automata but by no deterministic pushdown automaton. Q.E.D.

10.8.2. First of all, the nondeterministic machine whose state diagram appears below accepts L by empty stack. As for the rest, assume for the sake of deriving a contradiction that there exists some deterministic pushdown automaton M that accepts L by accepting state. We shall use M as the basis for constructing a new nondeterministic M' that accepts $\{a^n b^n c^n | n > 0\}$ by accepting state, thereby contradicting Theorem 10.10.



We make two copies of M , which we refer to as M_1 and M_2 . Hence each state in M_1 has its counterpart in M_2 , and vice versa. The states of the new M' will be those of M_1 together with those of M_2 . The start state of M_1 becomes the start state of M' , and the accepting states of M_2 will be the accepting states of M' . In addition, we make two modifications to the transitions of M_1 and M_2 .

- (1) Any arc formerly emanating from an accepting state q_i and leading to some state q in M_1 will now lead to the counterpart of q in M_2 .
- (2) The label of any arc that leads to a state within the former M_2 , including any that were redirected at (1), are now changed so as to process c s instead of b s. In other words, the arc labels of the former M_2 now make no reference to symbol b . (Arc labels involving a s are unchanged.)

We claim that $L(M')$ is none other than $\{a^n b^n c^n | n > 0\}$. First, suppose that $w \in L(M')$. By the construction of M' , some prefix of w of the form $a^n b^n$ with $n > 0$ must label a unique path from M' 's initial state to what was formerly an accepting state in M_1 . Moreover, by (1) and (2), only symbol c can follow this prefix in w . Now let us consider that state q^* in M_2 to which the new path labeled $a^n b^n c$ leads. It

should be clear that a path labeled $a^n b^{n+1}$ formerly led to q^* in M_2 . Moreover, by the determinism of the original M again, q^* is the only state in the former M_2 that was labeled by this path. Again by determinism, the only path that formerly led from q^* to an accepting state of M_2 would have been labeled by b^{n-1} . (Why?) But now this path is labeled by c^{n-1} . Hence w must be of the form $a^n b^n c c^{n-1} = a^n b^n c^n$. We have shown that $L(M') \subseteq \{a^n b^n c^n | n > 0\}$.

As for the other direction, suppose that $w \in \{a^n b^n c^n | n > 0\}$. Prefix $a^n b^n$ labels a unique path from initial state to a former accepting state within M_1 . The subsequent c labels the transition into the former M_2 , arriving at that unique state q^* to which a path labeled $a^n b^{n+1}$ formerly led in the M_2 . But now there is a path from q^* to an accepting state of M' labeled c^{n-1} , and we have that $\{a^n b^n c^n | n > 0\} \subseteq L(M')$. Q.E.D.

Solutions to Exercises for § 10.9

10.9.2. First, note that only productions whose right-hand sides contain no terminals are relevant to this issue. Let us refer to such productions as *terminal-free*. Start by considering all terminal-free productions for nonterminal A that are nonrecursive, i.e., contain no occurrence of A itself on the right. If, among them, there is an ϵ -production, then we are done and $A \Rightarrow^* \epsilon$. Otherwise, $A \Rightarrow^* \epsilon$ just in case there exists a terminal-free, nonrecursive production

$$A \rightarrow X_1 X_2 \dots X_n$$

such that, for each i with $1 \leq i \leq n$, $X_i \Rightarrow^* \epsilon$. Deciding this new question will have us considering all terminal-free, nonrecursive productions for each of the X_i . Moreover—and this is crucial—we need not consider productions for X_i that have an occurrence of A on the right. (Why?) Since G has but finitely many productions, this process comes to an end eventually.

10.9.3. If $w = \epsilon$, then the algorithm of Exercise 10.9.2 can be applied. If $w \neq \epsilon$, we first apply the algorithm of the proof of Theorem 10.4 so as to obtain a new grammar G' in Greibach normal form with $L(G') = L(G) \setminus \{\epsilon\}$. Suppose that w begins with terminal symbol s followed by terminal symbol s' followed by word w' . It follows that we need consider only those productions for A whose right-hand sides begins with s . Suppose, for the sake of simplicity, that the only such production is

$$A \rightarrow s X_1 X_2 \dots X_n$$

We must now determine whether some production for nonterminal X_1 has right-hand side beginning with terminal s' . If not, then we see immediately that $A \not\Rightarrow^* w$. Suppose, on the other hand, that there is one such production

$$X_1 \rightarrow s' Y_1 Y_2 \dots Y_m$$

Now the question becomes whether $Y_1 Y_2 \dots Y_m X_2 \dots X_n \Rightarrow^* w'$, and to find the answer we first consider productions for Y_1 . This process must eventually terminate, since w is of finite length and G' has but finitely many productions.

10.9.4. If $w = \epsilon$, then, letting $\beta = A$ itself, $A \Rightarrow^* w\beta$ trivially. Otherwise, we proceed essentially as in the solution to Exercise 10.9.3 except that, after two steps, the question becomes whether $Y_1 Y_2 \dots Y_m X_2 \dots X_n \Rightarrow^* w'\beta$. In other words, it is enough to ascertain whether $Y_1 Y_2 \dots Y_m X_2 \dots X_n$ generates a string beginning with w' .

10.9.5. It should be clear that, for any nonterminal X , it is decidable whether some string containing X is derivable from A . We then have only to check, using the algorithm of Exercise 10.9.3, to see whether $X \Rightarrow^* w$.

10.9.6. It should be clear that, for any nonterminal X , it is decidable whether some string ending in X is derivable from A . We then have only to check, using the algorithm of Exercise 10.9.3, to see whether $X \Rightarrow^* w$.

Solutions to Exercises for § 10.10

10.10.5 (a) Lemma 10.3 itself is sufficient for this one. Suppose, for the sake of producing a contradiction, that $L = \{a^i b^j c^k \mid 1 \leq i \leq j \leq k\}$ is context-free. Since L is infinite, there exists an n such that for any word $z = uvwxy$ with $|z| > n$ we have that $uv^i wx^i y$ is in L , for $i \geq 0$, where v and x are not both empty. Clearly $a^n b^n c^n$ is in L with length exceeding n . So $a^n b^n c^n$ must be of the form $uvwxy$ and $uv^i wx^i y$ is also in L , for $i \geq 0$, where v and x are not both empty. It is apparent that v must consist of as alone, bs alone, or cs alone. Similarly, x must consist just of as , just of bs , or just of cs . (Otherwise pumping would produce words with bs before as , cs before as , or even cs before as .) So suppose that v is just as and x is just bs . In this case, pumping will produce words in L with more bs than cs , which contradicts the definition of L . Similarly, suppose that v is just bs and x is just cs . In this case, pumping puts $uv^0 wx^0 y$ in L with fewer bs and cs than as , which contradicts the definition of L . To consider a different sort of case, suppose that v is ϵ and that x consists of as only. Then pumping puts words in L with more as than bs and cs . On the other hand, suppose that v is ϵ and that x consists of bs only. Then pumping puts $uv^0 wx^0 y$ in L with more as than bs . There are many other cases to consider. It should be clear, however, that each leads to contradiction. We must conclude that $L = \{a^i b^j c^k \mid 1 \leq i \leq j \leq k\}$ is not context-free. Q.E.D.

(b) We use the strengthened version of Lemma 10.3 presented in § 10.9. Suppose, for the sake of producing a contradiction, that $L = \{a^i b^j c^i d^j \mid 1 \leq i, j\}$ is context-free. Let G be a context-free grammar with k nonterminals that accepts L . Since L is infinite, we have, for any word $z = uvwxy$ with $|z| > 2^{k-1}$, that $uv^i wx^i y$ is in L , for $i \geq 0$, where v and x are not both empty and $|vwx| \leq 2^k$. Clearly $a^{2^k} b^{2^k} c^{2^k} d^{2^k}$ is in L with length exceeding 2^k . So $a^{2^k} b^{2^k} c^{2^k} d^{2^k}$ must be of the form $uvwxy$, where $uv^i wx^i y$ is also in L , for all $i \geq 0$, where v and x are not both empty and $|vwx| \leq 2^k$. Apparently, v must consist of as alone, bs alone, cs alone, or ds alone. Similarly, x must consist just of as , bs , cs , or ds . (Otherwise, pumping would produce words with bs before as or cs before bs and so on.) The fact that $|vwx| \leq 2^k$ is crucial now. For it follows that v cannot consist of as only while x consists of cs only, say, since this would imply $|vwx| > 2^k$ (why?). We see that the only remaining possibilities are

- (i) v consists of as only and x consists of bs only
- (ii) v consists of bs only and x consists of cs only
- (iii) v consists of cs only and x consists of ds only
- (iv) v and x both consist of as only
- (v) v and x both consist of bs only
- (vi) v and x both consist of cs only
- (vii) v and x both consist of ds only

But each of (i)–(vii) leads to contradiction even if one of v or x is empty. We conclude that $L = \{a^i b^j c^i d^j \mid 1 \leq i, j\}$ is not context-free. Q.E.D.

(c) We present an indirect argument. That is, we suppose for the sake of producing a contradiction that $L = \{a^i \mid i \text{ is composite}\}$ is context-free. Since L is infinite, the Pumping lemma applies and there exists k such that, for any word $z = uvwxy$ with $|z| > 2^{k-1}$, we have that (i), (ii), and (iii) hold. Let us set $N = 2^k$. Since the set of primes is infinite and hence unbounded, there exist primes greater than $(N!+1)+1$. Let p be the least such prime.

Suppose for the moment that $(N!+1)+N!+1 \geq p$. That is,

$$(N!+1)!+N!+1 \geq p > (N!+1)+1$$

Now subtracting $(N!+1)!$ from all three sides of this inequality, we obtain

$$N!+1 \geq p - (N!+1)! > 1$$

But then p differs from $(N!+1)!$ by a factor of $(N!+1)!$. Moreover, since $(N!+1)!$ is composite, p must be composite also, contradicting our assumption. We conclude that $p > (N!+1)! + N! + 1$ after all.

But now we can see that $p - N!$ is composite. For suppose not, i.e., suppose that $p - N!$ is prime. Then we can write

$$p > p - N! > (N!+1)! + 1$$

contradicting the definition of p as the least prime greater than $(N!+1)+1$. So $p - N!$ is composite. This puts $a^{p-N!}$ in L . Since it can be seen that $|a^{p-N!}| > 2^{k-1}$, it follows that $a^{p-N!}$ is of the form $uvwxy$ such that (i), (ii), and (iii) of the lemma apply. In particular, by (ii), $|vwx| \leq 2^k = N$. It follows that $|vx| \leq N$ as well, so that $|vx|$ is a factor of $N!$. But then, since $a^{p-N!}$ is in L , pumping v and x puts a^p in L as well, contradicting the primality of p . We conclude that $L = \{a^i \mid i \text{ is composite}\}$ cannot be context-free after all. Q.E.D.

10.10.9 (a) (i) Not letter-equivalent
(ii) Letter-equivalent

(b) $w = w'$

(c) (i) Letter-equivalent
(ii) Letter-equivalent
(iii) Letter-equivalent

(d) (i) $L((aa)^*(bb)^*(alb)^2)$
(ii) $L((()))^*$, where (and) are the two symbols of alphabet Σ .

(e) Suppose that L is a context-free language over alphabet Σ with $\text{card}(\Sigma) = 1$. By the cited theorem there exists a regular language L' that is letter-equivalent to L . But by (b), every word of L is in L' , and vice versa. That is, $L = L'$. Thus, L is not merely context-free. It is regular as well.