Fundamentele Informatica II

Answer to selected exercises 4

John C Martin: Introduction to Languages and the Theory of Computation

M.M. Bonsangue (and J. Kleijn)

Fall 2013

4.1 In each case say which language is generated by the CFG G with the productions as indicated.
a. $L(G) = a, b^*$.
b. $L(G) = \{a, b\}^*\{a\}$.
c. $L(G) = \{ba\}^*\{b\}$.
d. $L(G) = \{x \in \{a, b\}^* \mid bb$ does not occur in $x\}$.
e. $L(G) = \{a, b\}^*\{b\}$.
f. $L(G) = \{xaybx^r, xbyax^r \mid x, y \in \{a, b\}^* \wedge y = y^r\}$, i.e. the language of all words which are palindromes over $\{a, b\}$ with exactly one single "mistake".
g. $L(G) = \{x \in \{a, b\}^* \mid |x|$ is even $\}$.
h. $L(G) = \{x \in \{a, b\}^* \mid |x|$ is odd $\}$.

4.3 Find a context-free grammar generating the given language.
a. For $L = \{xay \mid x, y \in \{a, b\}^* \wedge |x| = |y|\}$ the CFG with productions
$S \to aSa \mid aSb \mid bSa \mid bSb \mid a$
b. For $L = \{xaay, xbby \mid x, y \in \{a, b\}^* \wedge |x| = |y|\}$ the CFG with
$S \to aSa \mid aSb \mid bSa \mid bSb \mid aa \mid bb$
c. For $L = \{axaya, bxbyb \mid x, y \in \{a, b\}^* \wedge |x| = |y|\}$ the CFG with
$S \to aAa \mid bBb, \quad A \to aAa \mid aAb \mid bAa \mid bAb \mid a, \quad B \to aBa \mid aBb \mid bBa \mid bBb \mid b$

4.4 The productions of two context-free grammars are given. Prove that neither one generates the language $L = \{x \in \{a, b\}^* \mid n_a(x) = n_b(x)\}$, the language consisting of all words with an equal number of a's and b's.
a. $S \to SabS \mid SbaS \mid \Lambda$
Clearly, every word generated by this grammar has an equal number of a's and b's, but it cannot generate every word of L: Every non-empty word generated by this grammar is of the form $xaby$ or $xbay$ with both x and y

also generated by S. Hence if x (or y) is non-empty it also contains at least one occurrence of ab or ba. This implies that $aabb$ cannot be generated even though it is in L.
b. $S \to aSb \,|\, bSa \,|\, abS \,|\, baS \,|\, Sab \,|\, Sba \,|\, \Lambda$
Clearly, every word generated by this grammar has an equal number of a's and b's, but it cannot generate every word of L: Every non-empty word generated by this grammar is of the form ayb, bya, aby, bay, yab, or yba with y also a word generated by the grammar. Consequently, $x = aabbbbaa$ cannot be generated even though $x \in L$.

4.5 $S \to aSbScS \,|\, aScSbS \,|\, bSaScS \,|\, bScSaS \,|\, cSaSbS \,|\, cSbSaS \,|\, \Lambda$.
Does the CFG G with these productions generate the language
$L = \{x \in \{a, b, c\}^* \mid n_a(x) = n_b(x) = n_c(x)\}$?
No. Since every production introduces an equal number of a's, b's, and c's, it is clear that $L(G)$ is included in L. Thus the question is whether G can generate *every* word in L. This turns out to be not the case.
Consider $aabbcc \in L$. Any derivation of this word has to start with an application of the production $S \to aSbScS$ because we need an a in the first place and b's have to precede c's. To derive $aabbcc$ from $aSbScS$, rewriting the first occurrence of S should lead to the terminal word a or ab, but this is impossible, because each word derivable from S has an equal number of a's, b's, and c's (as observed before).

- Give a CFG that generates all regular expressions over an alphabet Σ.

For simplicity, let us assume $\Sigma = \{a, b\}$. It is easy to generalize the result below to other aplphabets. De terminal symbols includes all elements of Σ, the operators $\{+, \cdot, ^*, (,)\}$, and distinct symbols for $\emptyset$ en Λ, say ϕ en λ, respetively. We construct a grammar with starting symbol S and with the following productions:

$$S \;\to\; (S + S) \mid (S \cdot S) \mid (S^*) \mid a \mid b \mid \lambda \mid \phi \;.$$

4.10 Find a CFG for each of the given langages.
a. $S \;\to\; aSb \mid B$ and $B \;\to\; bB \mid \Lambda$.
b. $S \;\to\; aSb \mid B$ and $B \;\to\; bB \mid b$.
c. $S \;\to\; aSbb \mid \Lambda$.
d. $S \;\to\; aSb \mid aSbb \mid \Lambda$.
e. $S \;\to\; aSBB \mid \Lambda$ and $B \;\to\; b \mid \Lambda$.
f. $S \;\to\; aSBB \mid a \mid ab$ and $B \;\to\; b \mid \Lambda$.

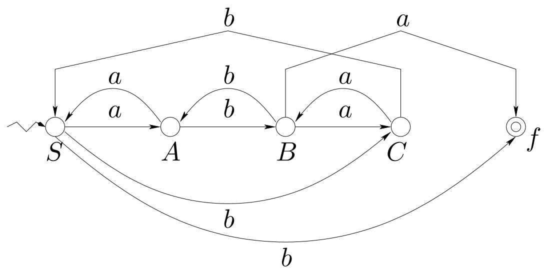
• Find a CFG for each of the given langages.
a. $L = \{a^i b^j c^k \mid i = j + k\}$. Thus each word in L has the form $a^k a^j b^j c^k$ and such words are exactly generated by the CFG with productions
$S \to aSc \,|\, T, \quad T \to aTb \,|\, \Lambda.$
e. $L = \{a^i b^j c^k \mid i < j \vee i > k\}$. Thus each word in L is of the form $a^i b^i b^n c^k$ or $a^k a^n b^j c^k$ for some $n \geq 1$. Such words are exactly generated by the CFG with productions
$S \to XC \,|\, A$
$X \to aXb \,|\, Xb \,|\, b, \quad C \to Cc \,|\, \Lambda,$
$Y \to aYc \,|\, aY \,|\, aZ, \quad Z \to bZ \,|\, \Lambda.$
h. $L = \{a^i b^j \mid i \leq j \leq 2i\}$. Thus each word in L is in $\{a\}^i \{b, bb\}^i$ for some $i \geq 0$. These words are exactly generated by the CFG with productions
$S \to aSb \,|\, aSbb \,|\, \Lambda.$

4.25 Given a language $L \subseteq \Sigma^*$ we need to prove that *a.*, *b.* and *c.* are equivalent.
a. implies *b.*: it follows directly because regular grammar are a special case of the grammar specified in *b.*
b. implies *a.*: Let L be a language generated by a grammar with productions of the form $A \to xB$ of $A \to \Lambda$ with A, B variables and $x \in \Sigma^*$. First we find an equivalent grammar without unit productions (i.e. without productions $A \to xB$ with $|x| = 0$) using Theorem 4.28. In the resulting grammar, we look at all its productions. If $A \to xB$ with $|x| = 1$ we leave at it is, but if $x = a_1 a_2 \cdots a_n$ for $n \geq 2$ and each $a_i \in \Sigma$, then we substitute $A \to xB$ by a sequence of production $A \to a_1 X_1$, $X_1 \to a_2 X_2$, $\ldots X_n \to a_n B$, with $X_1, \ldots, X_n$ new variable symbols.

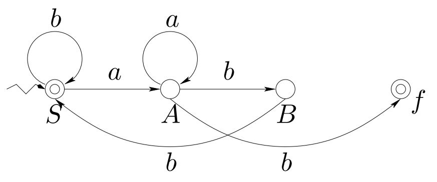
The new grammar so obtained is clearly regular and generates the same language as the original grammar.
a. implies *c.*: It is enough to change every production $A \to \sigma B$ into $A \to B\sigma$. The language of the new grammar is the reverse of the language of the language generated by the original grammar. Since the latter is regular, so is the language of our new grammar.
c. implies *a.*: We can use a construction similar to the one we have used for proving *a.* implies *b.*, by first transforming each production $A \to Bx$ into $A \to x^R B$, where x^R is the reverse of x. The new grammar generates a regular language (the reverse of the original one) because is of the form as specified in *b.*. Since regular language are closed under reversal, the original language must be regular too.

4.26 Describe the language generated by the given grammars.

a. $S \to aA \,|\, bC \,|\, b, \quad A \to aS \,|\, bB, \quad B \to aC \,|\, bA \,|\, a, \quad C \to aB \,|\, bS$
This is a regular grammar. Using the construction given in the proof of Theorem 4.14, we obtain the following NFA accepting $L(G)$.



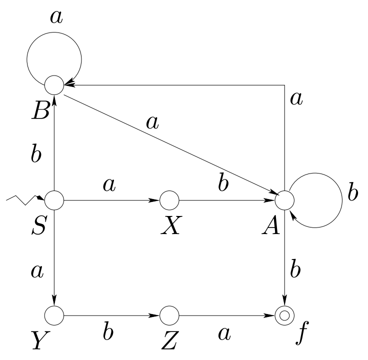
Now it is not difficult to see that
$L(G) = \{x \in \{a, b\}^* \mid n_a(x) \text{ is even and } n_b(x) \text{ is odd }\}$.
S corresponds to "even number of a's and even number of b's"
A corresponds to "odd number of a's and even number of b's"
B corresponds to "odd number of a's and odd number of b's"
C corresponds to "even number of a's and odd number of b's".
b. $S \to bS \,|\, aA \,|\, \Lambda, \quad A \to aA \,|\, bB \,|\, b, \quad B \to bS$
This is a regular grammar. Using the construction given in the proof of Theorem 4.14, we obtain the following NFA accepting $L(G)$.



From this automaton we can read the regular expression $(b^*aa^*bb)^*(\Lambda + b^*aa^*b)$ which describes $L(G)$.

4.27 See the FA M in Figure 4.33. The regular grammar G with $L(G) = L(M)$ constructed from M as in Theorem 4.4 has the productions:
$A \to aB \,|\, bD \,|\, \lambda, \quad B \to aB \,|\, bC \,|\, b, \quad C \to aB \,|\, bC \,|\, b, \quad D \to aD \,|\, bD.$
This grammar has A as its staring symbol. Note that the state D is a 'sink' state and that,consequently, the productions relating to D can be safely omitted from the grammar without affecting the successful derivations (and hence the generated language). This yields:
$A \to aB \,|\, \Lambda, \quad B \to aB \,|\, bC \,|\, b, \quad C \to aB \,|\, bC \,|\, b.$

4.28 Given is the CFG with productions:
$S \to abA \,|\, bB \,|\, aba, \quad A \to b \,|\, bA \,|\, aB, \quad B \to aA \,|\, aB.$
This grammar is not a regular grammar but we transform it into an equivalent regular CFG G:
$S \to aX \,|\, bB \,|\, aY, \quad X \to bA, \quad Y \to bZ, \quad Z \to a,$
$A \to b \,|\, bA \,|\, aB, \quad B \to aA \,|\, aB.$
Next we apply the method from the proof of Theorem 4.4 and obtain an NFA accepting $L(G)$:



4.29 Each of the given grammars, though not regular, generates a regular language. Find for each a regular grammar (a CFG with only productions of the form $X \to aY$ and $X \to a$) generating its language.
a. $S \to SSS \,|\, a \,|\, ab$
The only non-terminating production for S is $S \to SSS$, which means that the number of occurrences of S in the current string increases with 2 each time this production is used. Terminating productions can be postponed until no production $S \to SSS$ will be applied anymore. Since we begin with one S, this means that just before termination we will have an odd number of S's. Termination of S yields for every occurrence of S either a or ab. Hence $L(G)$ consists of an odd number of concatenated a or ab strings:
$L(G) = (\{a, ab\}\{a, ab\})^*\{a, ab\}$ which is indeed a regular language.
A regular grammar for this language would be (with starting symbol Z):
$Z \to aU \,|\, aV \,|\, a \,|\, aB, \quad B \to b, \quad V \to bU. \quad U \to aZ \,|\, aW, \quad W \to bZ$
b. $S \to AabB, \quad A \to aA \,|\, bA \,|\, \Lambda, \quad B \to Bab \,|\, Bb \,|\, ab \,|\, b$
It is easy to see that from A the language $\{a, b\}^*$ is generated.
From B we obtain the language $\{ab, b\}\{ab, b\}^* = \{ab, b\}^*\{ab, b\}$.
Consequently $L(G) = \{a, b\}^*\{ab\}\{ab, b\}^*\{ab, b\}$, a regular language.
A regular grammar for this language would be (with starting symbol Z):

$Z \to aZ \,|\, bZ \,|\, aB, \quad B \to bY, \quad Y \to aX \,|\, b \,|\, bY, \quad X \to b \,|\, bY$

c. $S \to AAS \,|\, ab \,|\, aab, \quad A \to ab \,|\, ba \,|\, \Lambda$

As long as no terminating productions have been used every string derived from S consists of an even number of A's followed by an S. Upon termination the S will be rewritten into ab or aab, while each A yields ab or ba or Λ. An even number of concatenated A's yields a string consisting of an arbitrary number of concatenated occurrences of ab and ba. Note that this number is not necessarily even, since any A may also be rewritten into Λ.
Consequently, $L(G) = \{ab, ba\}^*\{ab, aab\}$, a regular language.
A regular grammar for this language would be (with starting symbol Z):
$Z \to aY \,|\, bX, \quad X \to aZ, \quad Y \to bZ \,|\, b \,|\, aW, \quad W \to b$

d. $S \to AB, \quad A \to aAa \,|\, bAb \,|\, a \,|\, b, \quad B \to aB \,|\, bB \,|\, \Lambda$

From A we generate the language consisting of all odd-length palindromes over $\{a, b\}$, which is not a regular language! However B generates $\{a, b\}^*$. Thus $L(G)$ consists of words formed by an odd-length palindrome followed by an arbitrary word over $\{a, b\}$. Now note that *every* non-empty word over $\{a, b\}$ can be seen as an a or b (both odd-length palindromes) followed by an arbitrary word over $\{a, b\}$. Consequently, $L(G) = \{a, b\}^+$, a regular language after all!
A regular grammar for this (easy) language would be: $Z \to aZ \,|\, bZ \,|\, a \,|\, b$

e. $S \to AA \,|\, B, \quad A \to AAA \,|\, Ab \,|\, bA \,|\, a, \quad B \to bB \,|\, b$

Clearly, every occurrence of B generates $\{b\}^+$. Because of $S \to B$, this implies that $\{b\}^+ \subseteq L(G)$.
The other production for S is $S \to AA$. Each A can surround itself with any number of b's before either terminating as a or producing two more A's. Hence after $S \Rightarrow AA$ we can produce any word over $\{a, b\}$ with an even (non-zero) number of a's. Together with $\{b\}^+ \subseteq L(G)$, this implies that $L(G) = (\{b\}^*\{a\}\{b\}^*\{a\}\{b\}^*)^+ \cup \{b\}^+$.
A regular grammar for this language would be (with starting symbol Z):
$Z \to aY \,|\, bZ \,|\, b, \quad Y \to bY \,|\, aZ \,|\, a.$

4.34 $S \to a \,|\, Sa \,|\, bSS \,|\, SSb \,|\, SbS$. This grammar is ambiguous, the word $abaa$ has two different leftmost derivations:
$S \Rightarrow SbS \Rightarrow abS \Rightarrow abSa \Rightarrow abaa$ and $S \Rightarrow Sa \Rightarrow SbSa \Rightarrow abSa \Rightarrow abaa$.

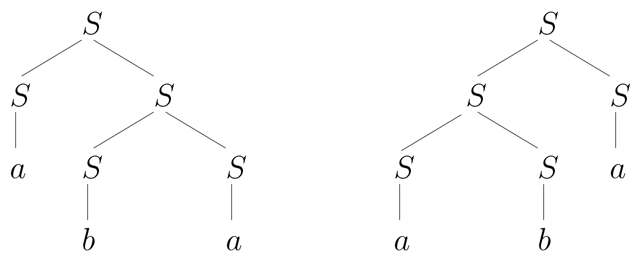
4.35 Consider the context-free grammar with productions
$S \to AB, \quad A \to aA \mid \Lambda, \quad B \to ab \mid bB \mid \Lambda$
This grammar is NOT unambiguous, even though every derivation of a string from S has to begin with $S \to AB$, and any string derivable from A has only one derivation from A and likewise for B.

There are strings in $L(G)$ which have more than one derivation tree (more than one leftmost derivation). Examples are ab and aab:
$S \Rightarrow AB \Rightarrow B \Rightarrow ab$ and $S \Rightarrow AB \Rightarrow aAB \Rightarrow aB \Rightarrow abB \Rightarrow ab$;
$S \Rightarrow AB \Rightarrow aAB \Rightarrow aB \Rightarrow aab$ and $S \Rightarrow AB \Rightarrow aAB \Rightarrow aaAB \Rightarrow aaB \Rightarrow aabB \Rightarrow aab$.

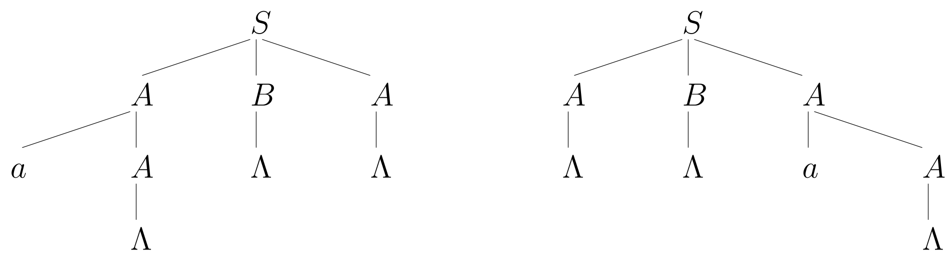
4.36 We look at the grammars given in Exercise 4.1. For each of them we have to decide if the grammar is ambiguous or not. We discuss here b, c, d, e, f and g. Grammars a and h are both not ambiguous, as it can be proved in a similar manner as for grammar g.
b The grammar given in b is ambiguous. This follows from the two different leftmost derivations for aaa: $S \Rightarrow SS \Rightarrow SSS \Rightarrow^3 aaa$ and $S \Rightarrow SS \Rightarrow aS \Rightarrow aSS \Rightarrow^2 aaa$.
c and **d** The grammar given c and d are ambiguous. This follows from the two different leftmost derivations for the word $babab$:
$S \Rightarrow SaS \Rightarrow SaSaS \Rightarrow^3 babab$ and $S \Rightarrow SaS \Rightarrow baS \Rightarrow baSaS \Rightarrow^2 babab$.
e This grammar is ambiguous. We have the following two leftmost derivations for $abab$:
$S \Rightarrow TT \Rightarrow aTT \Rightarrow aTaT \Rightarrow abaT \Rightarrow abab$ and $S \Rightarrow TT \Rightarrow TaT \Rightarrow aTaT \Rightarrow^2 abab$.
f We prove by induction on the length n of the derivation that $S \Rightarrow^n x$ for $x \in (V \cup \Sigma)^*$ has only one leftmost derivation from S.
(Induction base) $n = 1$ thus x is either aSa, bSb, aAb of bAa. In each case there is only one production that can be applied to obtain x from S.
(Induction step) For a given n, assume that there is only one leftmost derivation for $S \Rightarrow^n x$, If x containing at least a variable, then $x = uSv$ or $x = uAv$ (with u and v having length n). If we consider an $n+1$ leftmost derivation $S \Rightarrow^n x \Rightarrow y$, then if $y \neq uv$, the first two symbols immediately after u in y determine uniquely the production applied. For example, if $x = uSv$ and $y = uaSv'$ then we have applied the production $S \to aSa$. Similarly, if $x = uSv$ and $y = uaAv'$ then we have applied the production $S \to aAb$. When $y = uv$ then it must have been the case that $x = uAv$ and we applied the production $A \to \Lambda$. In every of these cases there is only one production that we can apply, so, using our induction hypothesis, all derivations are unique.
g We prove by induction on the length n of the derivation that $S \Rightarrow^n x$ for $x \in (V \cup \Sigma)^*$ has only one leftmost derivation from S.
(Induction base) $n = 1$ thus x is either Λ, aT or bT. In each case there is only one production that can be applied to obtain x from S.
(Induction step) For a given n, assume that there is only one leftmost deriva-

tion for $S \Rightarrow^n x$ and consider an $n+1$ leftmost derivation $S \Rightarrow^n x \Rightarrow y$. Because of the format of the production of the grammar, x is either of the form wS or wT, where $w \in \{a,b\}^*$. In the first case, either $y = w$ (i.e. we have applied the production $S \to \Lambda$), or $y = waT$ or wbT. In every of these cases there is only one production that we can apply, so, using our induction hypothesis, the all derivations are unique. The case when $y = waS$ or $y = wbS$ can be treated similarly.

4.38
Gevraagd wordt aan te tonen dat de gegeven grammatica dubbelzinnig is en vervolgens een equivalente niet dubbelzinnige grammatica te geven.
a. $S \to SS \,|\, a \,|\, b$
Volgens deze grammatica heeft het woord *aba* twee verschillende links-preferente (leftmost) afleidingen: $S \Rightarrow SS \Rightarrow aS \Rightarrow aSS \Rightarrow abS \Rightarrow aba$ en
$S \Rightarrow SS \Rightarrow SSS \Rightarrow aSS \Rightarrow abS \Rightarrow aba$.
De bijbehorende afleidingsbomen zien er zo uit:



De grammatica kan op het lege woord Λ na, alle woorden over $\{a,b\}$ genereren, d.w.z. de reguliere taal $\{a,b\}^+$.
Een equivalente (reguliere) grammatica is dan ook: $S \to aS \,|\, bS \,|\, a \,|\, b$.
Dat deze grammatica niet dubbelzinnig is volgt eenvoudig uit het feit dat hij bij een (deterministische!) FA hoort.
b. $S \to ABA \quad A \to aA \,|\, \Lambda \quad B \to \,|\, bB \,|\, \Lambda$
Volgens deze grammatica heeft het woord *a* twee verschillende links-preferente (leftmost) afleidingen: $S \Rightarrow ABA \Rightarrow aABA \Rightarrow^3 a$ en
$S \Rightarrow ABA \Rightarrow BA \Rightarrow A \Rightarrow aA \Rightarrow a$.
De bijbehorende afleidingsbomen zien er zo uit:



De grammatica genereert woorden bestaande uit 0 of meer a's gevolgd door 0 of meer b's gevolgd door 0 of meer a's, d.w.z. de reguliere taal $\{a\}^*\{b\}^*\{a\}^*$. Een equivalente grammatica is dan ook:
$S \to aS \,|\, bX \,|\, \Lambda \quad X \to bX \,|\, aY \,|\, \Lambda \quad Y \to aY \,|\, \Lambda$.
Deze grammatica is niet dubbelzinnig. (Vanwege de Λ-producties is deze grammatica strikt gesproken niet regulier, maar de onderliggende eindige automaat is ten duidelijkste deterministisch.) Zo heeft het woord a nu als enige afleiding $S \Rightarrow aS \Rightarrow \Lambda$.
d. $S \to aSb \,|\, aaSb \,|\, \Lambda$.
Volgens deze grammatica heeft het woord $aaab$ twee verschillende links-preferente (leftmost) afleidingen: $S \Rightarrow aSb \Rightarrow aaaSb \Rightarrow aaab$ en
$S \Rightarrow aaSb \Rightarrow aaaSb \Rightarrow aaab$.
De grammatica genereert woorden bestaande uit een aantal a's gevolgd door een aantal b's waarbij het aantal a's minstens even groot is als het aantal b's maar maximaal twee keer zo groot. d.w.z. de $\{a^ib^j \mid j \geq i \geq 2j\}$.
De dubbelzinnigheid van de gegeven grammatica wordt veroorzaakt doordat de extra a's op willekeurige momenten kunnen worden toegevoegd. De volgende grammatica genereert dezelfde taal, maar genereert eerst per b één a en als er eenmaal twee a's per b worden gegenereerd, gaat dat door totdat de afleiding stopt. We hebben dus een extra niet-terminaal nodig om die twee processen te kunnen scheiden:
$S \to aSb \,|\, \Lambda \,|\, aaAb \quad A \to aaAb \,|\, \Lambda$.
Deze grammatica is niet dubbelzinnig: de enige afleiding voor elk woord van de vorm $a^{j+k}b^j$ waarbij $0 \leq k \leq j$, is
$S \Rightarrow^j a^jSb^j \Rightarrow a^jb^j$ als $k = 0$ en
$S \Rightarrow^{j-k} a^{j-k}Sb^{j-k} \Rightarrow a^{j-k}aaAbb^{j-k} \Rightarrow^{k-1} a^{j-k+2}a^{2(k-1)}Ab^{k-1}b^{j-k+1} \Rightarrow a^{j+k}b^j$ als $k \geq 1$.

4.39 Let G be a regular grammar (note that $\Lambda \notin L(G)$). Convert G into an NFA M_G as in the proof of Theorem 4.14. Make M_G deterministic (using the subset construction) and transform the resulting FA M in an equivalent unambiguous regular grammar.

4.48 Let $G = (V, \Sigma, S, P)$ be a CFG. According to Definition 6.6, a variable is nullable if and only if it has a production with righthand-side Λ or a production with righthand-side consisting of nullable variables only.
We have to prove that for all $A \in V$ it holds that A is nullable if and only if $A \Rightarrow^* \Lambda$ in G.
Let $A \in V$. First assume that A is nullable. We use (structural) induction. If A is nullable, because of the production $A \to \Lambda$, then we have immediately that $A \Rightarrow \Lambda$. Otherwise there is a production $A \to B_1B_2 \dots B_n$ with $n \geq 1$ and the B_i nullable variables. By the induction hypothesis we have $B_i \Rightarrow^* \Lambda$ for all $1 \leq i \leq n$. Thus $A \Rightarrow B_1B_2 \dots B_n \Rightarrow^* B_2 \dots B_n \Rightarrow^* B_n \Rightarrow^* \Lambda$ as desired.
Next assume that $A \Rightarrow^m \Lambda$ in G for some $m \geq 1$ (the case $m = 0$ does not occur). We prove by induction on m that A is nullable. If $m = 1$, then $A \Rightarrow \Lambda$. This implies that $A \to \Lambda$ is a production of G and so A is nullable. Next assume (induction hypothesis) that whenever $B \Rightarrow^k \Lambda$ for some $k \leq m$, then B is nullable. Then consider the case $A \Rightarrow^{m+1} \Lambda$. This implies that the first production used in this derivation has been of the form $A \to B_1 \dots B_n$ for some $n \geq 1$. Thus $A \Rightarrow B_1 \dots B_n \Rightarrow^m \Lambda$. Consequently, for each $1 \leq i \leq n$, we have $B_i \Rightarrow^{k_i} \Lambda$ where $1 \leq k_i \leq m$. By the induction hypothesis each B_i is nullable and so also A is nullable.

4.49 Find a CFG without Λ-productions that generates the same language (except for Λ) as the given CFG. We apply Algorithm 6.1.
a. CFG G is given as $S \to AB \mid \Lambda, \quad A \to aASb \mid a, \quad B \to bS.$
The nullable variables are $N_0 = \{S\} = N_1$.
Modify the productions: $S \to AB \mid \Lambda, \quad A \to aASb \mid aAb \mid a, \quad B \to bS \mid b.$
Finally, remove the Λ productions to obtain G' with
$S \to AB, \quad A \to aASb \mid aAb \mid a, \quad B \to bS \mid b.$
Note that S is nullable. Thus (see exercise 6.33) $S \Rightarrow^* \Lambda$ which implies that $\Lambda \in L(G)$. Hence, in this case $L(G) - L(G') = \{\Lambda\}$.
b. CFG G is given as
$S \to AB \mid ABC, \quad A \to BA \mid BC \mid \Lambda \mid a,$
$B \to AC \mid CB \mid \Lambda \mid b, \quad C \to BC \mid AB \mid A \mid c.$
The nullable variables are obtained as $N_3 = N_2 = \{S, A, B, C\}$ from
$N_0 = \{A, B\}$, $N_1 = N_0 \cup \{C\}$, $N_2 = N_1 \cup \{S\}$.
Modify the productions (duplicates not included):
$S \to AB \mid A \mid B \mid \Lambda \mid ABC \mid BC \mid AC \mid C, \quad A \to BA \mid B \mid A \mid BC \mid C \mid \Lambda \mid a,$
$B \to AC \mid A \mid C \mid CB \mid B \mid \Lambda \mid b, \quad C \to BC \mid B \mid C \mid \Lambda \mid AB \mid A \mid c.$
Finally, remove the Λ productions and $X \to X$ productions to obtain G'
$S \to AB \mid A \mid B \mid ABC \mid BC \mid AC \mid C, \quad A \to BA \mid B \mid BC \mid C \mid a,$

$B \to AC \,|\, A \,|\, C \,|\, CB \,|\, b, \quad C \to BC \,|\, B \,|\, AB \,|\, A \,|\, c.$
Note that S is nullable and so $\Lambda \in L(G)$. Hence, also in this case $L(G) - L(G') = \{\Lambda\}$.

4.50 For each grammar G given, find a CFG G' without Λ-productions and without unit productions such that $L(G') = L(G) - \{\Lambda\}$. We apply Theorem 4.27 (Note that eliminating Λ-productions may introduce new unit productions, whereas eliminating unit productions does not introduce Λ-productions.)
a. G has productions $S \to ABA, \quad A \to aA \,|\, \Lambda, \quad B \to bB \,|\, \Lambda.$
Elimination of nullable productions: all variables of G are nullable, because $N_2 = N_1 = N_0 \cup \{S\}$ with $N_0 = \{A, B\}$.
Modifying the productions leads to
$S \to ABA \,|\, BA \,|\, AA \,|\, AB \,|\, B \,|\, A \,|\, \Lambda, \quad A \to aA \,|\, a \,|\, \Lambda, \quad B \to bB \,|\, b \,|\, \Lambda.$
Then we delete the Λ-productions and we obtain:
$S \to ABA \,|\, BA \,|\, AA \,|\, AB \,|\, B \,|\, A, \quad A \to aA \,|\, a, \quad B \to bB \,|\, b.$
Elimination of unit productions: Both A and B are S-derivable; since neither A nor B have unit productions, there are no variables that are A-derivable or B-derivable.
A is S-derivable, so we add $S \to aA$ and $S \to a$;
B is S-derivable, so we add $S \to bB$ and $S \to b$.
Then we delete all unit productions.
Consequently we arrive at the CFG G' defined by
$S \to ABA \,|\, BA \,|\, AA \,|\, AB \,|\, bB \,|\, b \,|\, aA \,|\, a, \quad A \to aA \,|\, a, \quad B \to bB \,|\, b.$

4.51, 4.52, 4.53 These exercises are all concerned with reducing CFGs in the sense that superfluous symbols (those that can never be used in a successful derivation) are removed. Let $G = (V, \Sigma, S, P)$ be a CFG.
4.51 Live variables:
A is live (in G) iff there exists an $x \in \Sigma^*$ such that $A \Rightarrow^* x$.
Recursive definition/algorithm:
$L_0 = \{A \in V \mid \exists x \in \Sigma^* \,.\, A \to x \in P\}$
$L_{k+1} = L_k \cup \{A \in V \mid \exists x \in (L_k \cup \Sigma)^* \,.\, A \to x \in P\}$ for all $k \geq 0$;
the algorithm terminates if $L_{m+1} = L_m$ for some $m \geq 0$.
4.52 Reachable variables:
A is reachable (in G) iff there exists $x, y \in (V \cup \Sigma)^*$ such that $S \Rightarrow^* xAy$.
Recursive definition/algorithm:
$R_0 = \{S\}$ and, for all $k \geq 0$,
$R_{k+1} = R_k \cup \{A \in V \mid \exists Z \in R_k \,.\, \exists x, y \in (V \cup \Sigma)^* \,.\, Z \to xAy \in P\}$;
the algorithm terminates if $R_{m+1} = R_m$ for some $m \geq 0$.

4.53 Useful variables:
A is useful (in G) iff there exists $x, y \in (V \cup \Sigma)^*$ and $z \in \Sigma^*$ such that $S \Rightarrow^* xAy \Rightarrow^* z$. Thus if A is useful, it is reachable and live.
6.38a. The converse does not hold. As an example, consider the CFG with productions $S \to AB$ and $A \to a$. Then, clearly A is reachable and live, but not useful (B cannot terminate).
6.38d. Note that only useful variables appear in successful derivations (and vice versa: all variables appearing in a successful derivation are useful). As discussed in **b.** and **c.** we can find for each CFG an equivalent CFG in which all variables are useful by first eliminating all dead variables and then all non-reachable ones. As an example consider the grammar G given by the productions
$S \to ABC \,|\, BaB, \quad A \to aA \,|\, BaC \,|\, aaa, \quad B \to bBb \,|\, a, \quad C \to CA \,|\, AC.$
First determine the live variables: $L_0 = \{A, B\}$, $L_1 = L_0 \cup \{S\}$, $L_2 = L_1$.
Eliminate the remaining ("dead") variables (in this case C) from G:
$S \to BaB, \quad A \to aA \,|\, aaa, \quad B \to bBb \,|\, a.$
Next determine (in the new grammar) the reachable variables: $R_0 = \{S\}$, $R_1 = R_0 \cup \{B\}$, $R_2 = R_1$.
Eliminate the remaining, unreachable, variables (in this case A) from G:
$S \to BaB, \quad B \to bBb \,|\, a.$
This grammar generates $L(G)$ and is "reduced" (all its variables are useful).
Finally, note that eliminating dead variables may make others unreachable: For the CFG given by $S \to AB$ and $A \to a$, eliminating $S \to AB$ makes A unreachable. On the other hand, eliminating non-reachable variables does not affect the liveness of the (reachable) others.

4.54 Construct for each grammar G given, a grammar G' in CNF with $L(G') = L(G) - \{\Lambda\}$.
a. G with productions $S \to SS \,|\, (S) \,|\, \Lambda$.
1. Eliminate the Λ-production from G which yields G_1 with productions $S \to SS \,|\, (S) \,|\, ()$. The newly introduced production $S \to S$ is removed together with the Λ-production. $L(G_1) = L(G) - \{\Lambda\}$.
2. There are no unit productions.
3. Finally, adapt to CNF; first we get $S \to SS \,|\, LSR \,|\, LR, \quad L \to (, \quad R \to)$;
next we have $S \to SS \,|\, LX \,|\, LR, \quad X \to SR, \quad L \to (, \quad R \to)$,
which are the productions of G' and $L(G') = L(G_1) = L(G) - \{\Lambda\}$.

• Let $G = (V, \Sigma, S, P)$ be a CFG in Chomsky normal form and $x \in L(G)$ with $|x| = k$ for some $k \geq 1$. We compute the number of derivation steps needed to generate x.

As in the beginning of section 6.6, we consider, for words $w \in (V \cup \Sigma)^*$, their length $|w|$ and the number of occurrences of terminals which appear in them: $t(w)$. Let $N(w) = |w| + t(w)$. Thus $N(S) = 1$ and $N(x) = |x| + t(x) = k + k = 2k$ for our given x. Since G is in CNF its productions are of the form $A \rightarrow BC$ or $A \rightarrow a$. Consequently, applying a production ia single derivation step $u \Rightarrow v$ either increases the length by 1 or increases the number of terminal occurrences by 1. In other words: $N(v) = N(u) + 1$. Since $N(x) - N(S) = 2k - 1$, it follows that a (each!) derivation of x from S in G consists of $2k - 1$ derivation steps.
