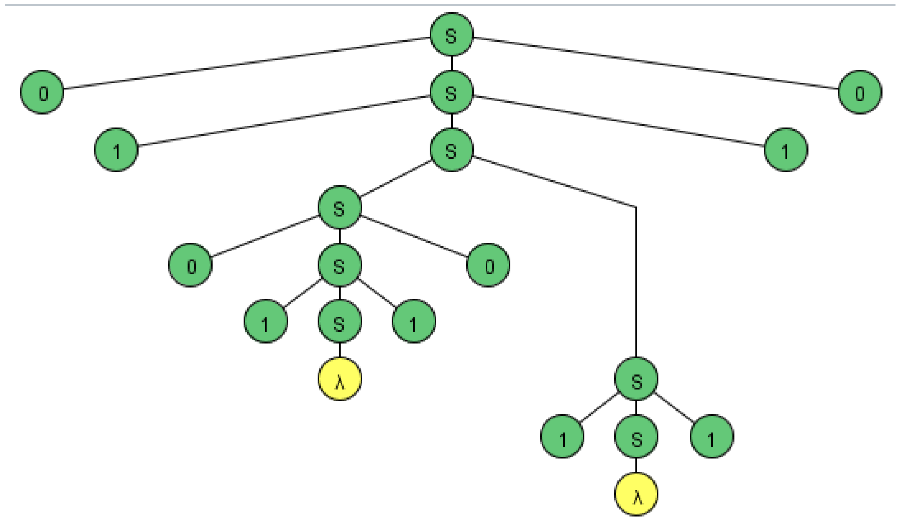
Chapter 7

Context-Free Language and Grammars

1. Consider the grammar $S \rightarrow 0S0 \mid 1S1 \mid SS \mid \lambda$. Given the string 0101101110, find a leftmost derivation and a rightmost derivation with corresponding parse trees.

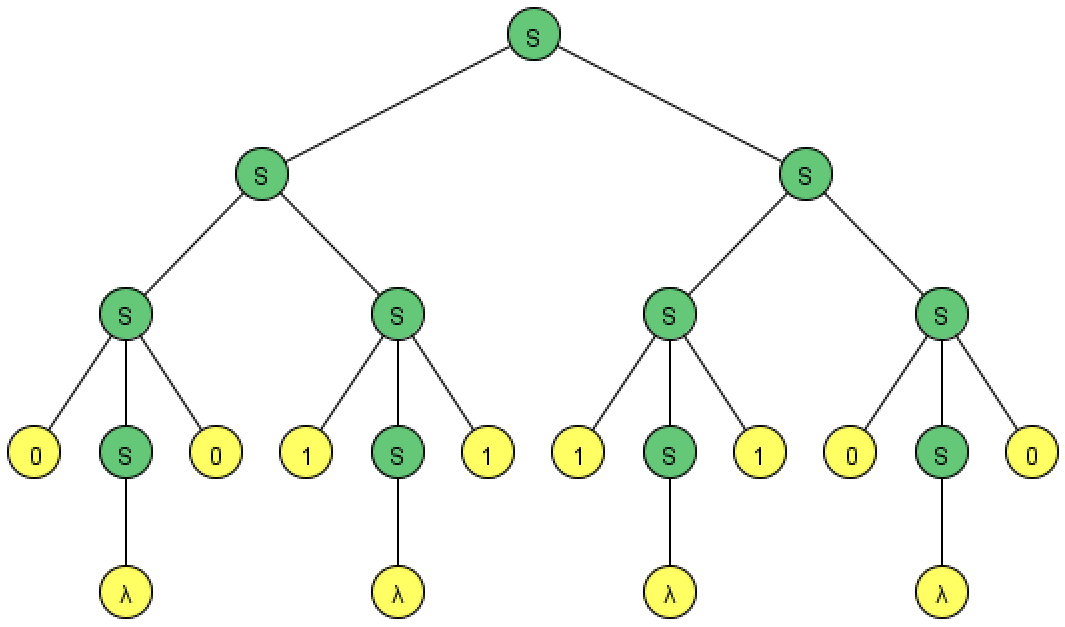
   Solution: Parse tree:

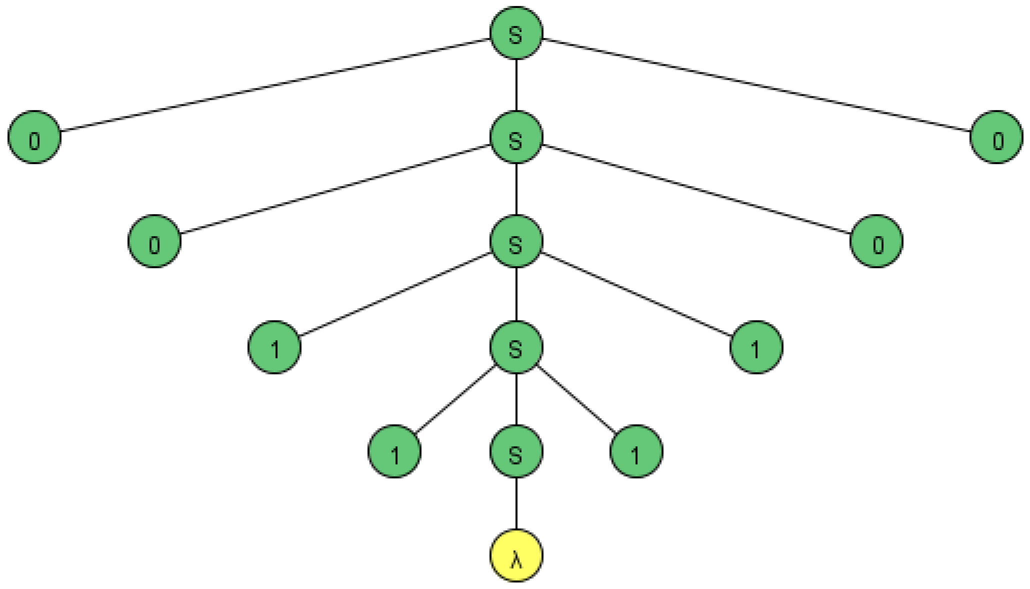


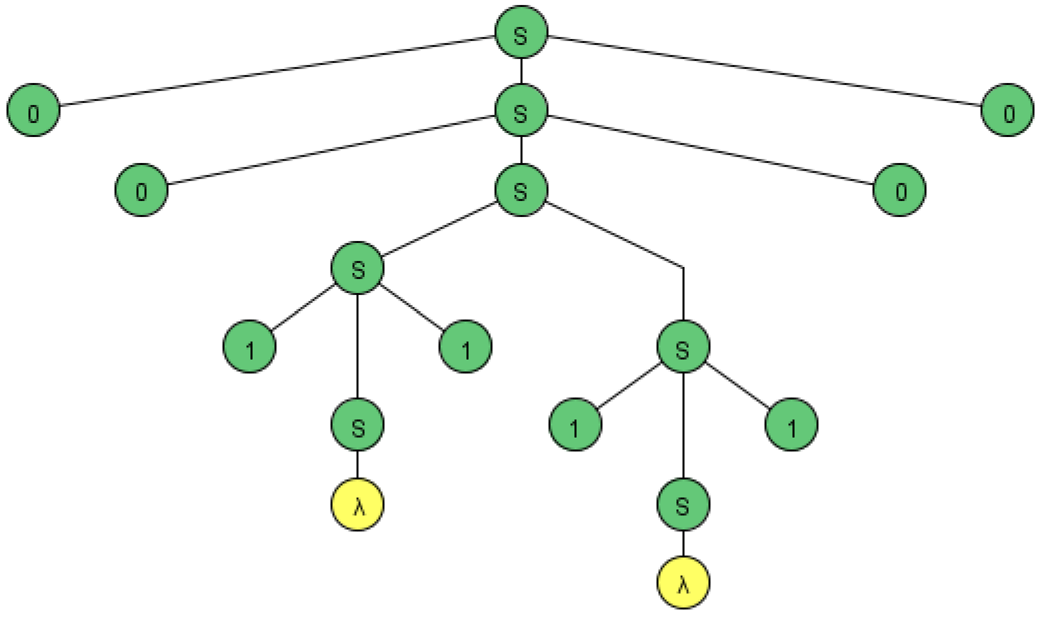
   Leftmost and rightmost derivations can be read from the parse tree.

2. For the above grammar, how many different parse trees can you construct for the string 00111100?

   Solution: Three parse trees are shown below. More combinations are possible..

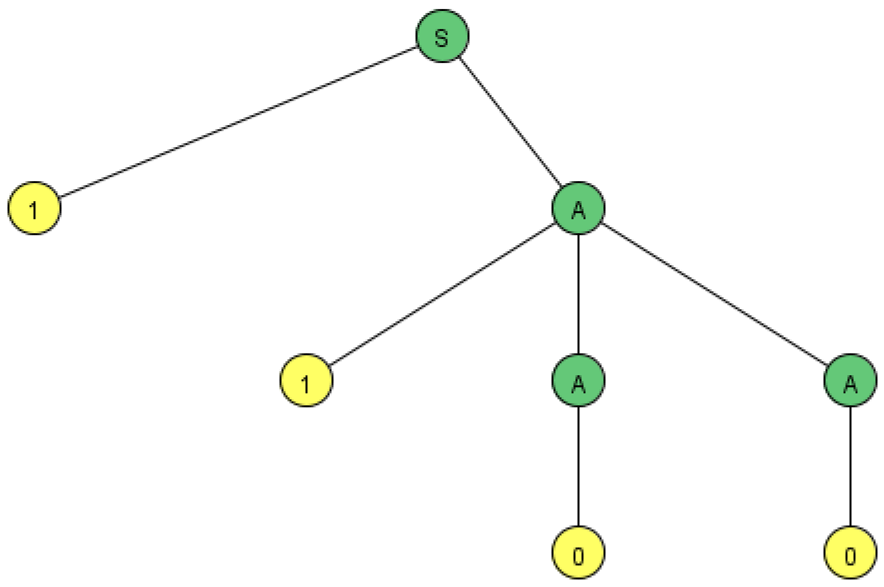


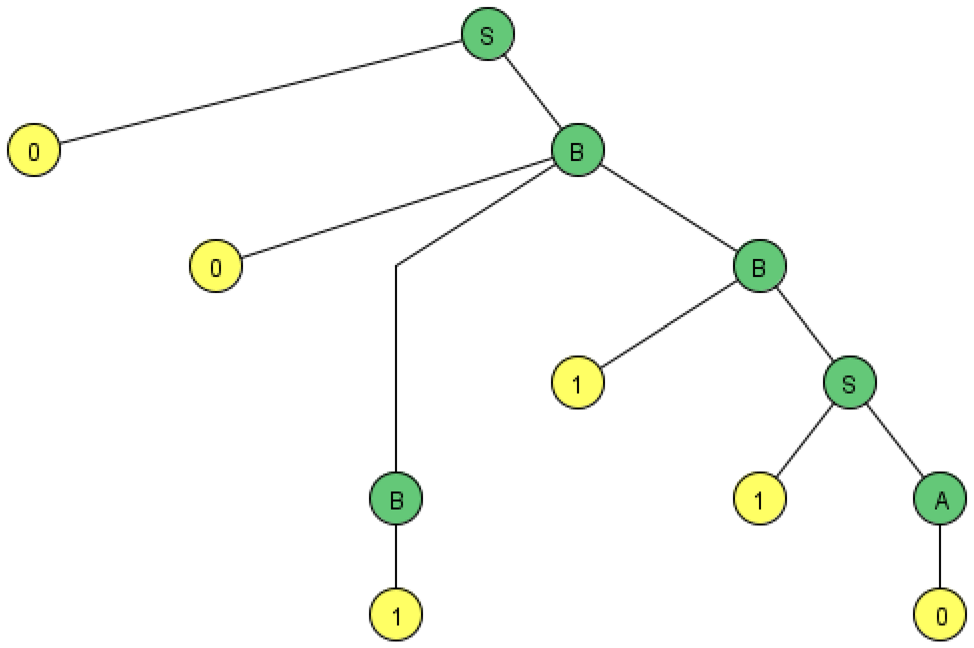


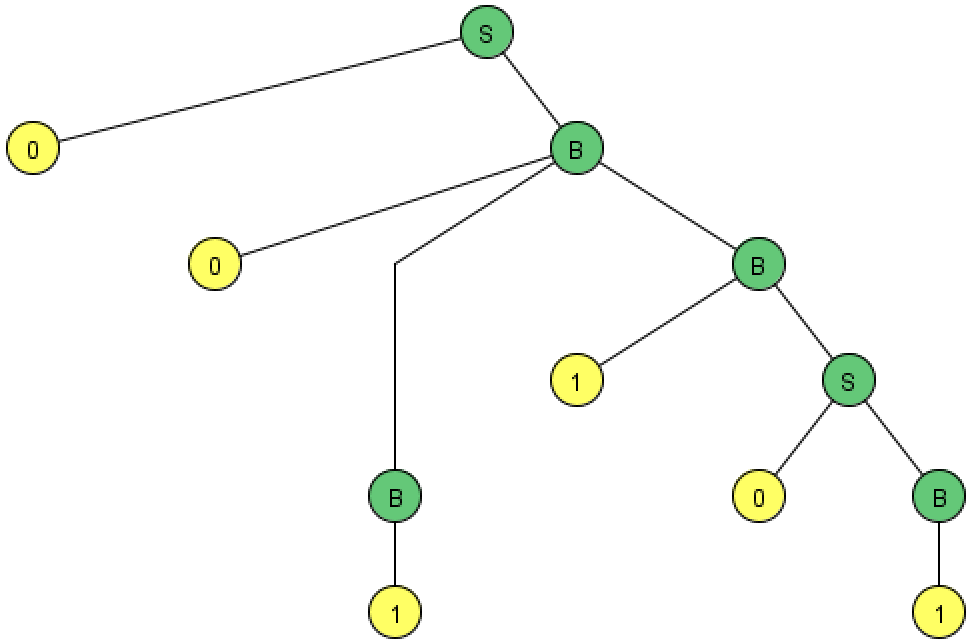


3. Consider the grammar $S \to 0B \mid 1A$, $A \to 0 \mid 0S \mid 1AA$, $B \to 1 \mid 1S \mid 0BB$. Given the string 1100, find a leftmost derivation and/or a rightmost derivation with corresponding parse trees. Repeat for the strings 001110 and 001101.

Solution: Parse trees are shown below. Derivations can be read from the parse trees:

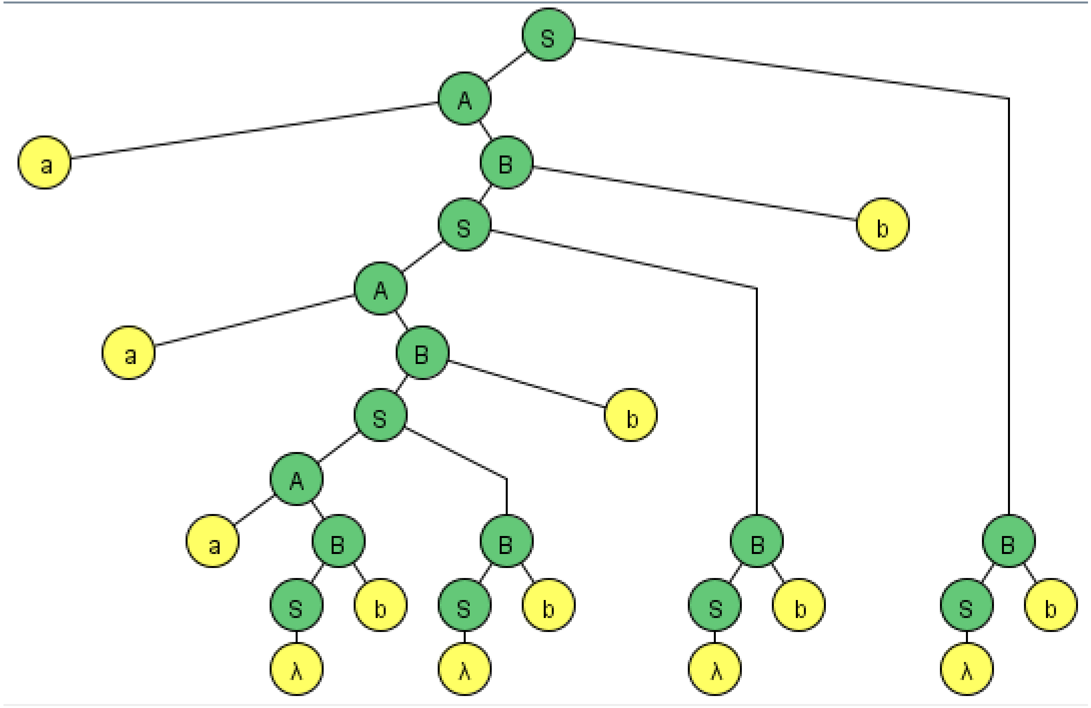

S
1
A
1
A
A
0
0


S
0
B
0
B
1
S
B
1
A
1
0



4. Given the CFG $S \to AB \mid \lambda$, $A \to aB$, $B \to Sb$, construct a derivation tree for *aaabbbbbb*.

Solution:



5. Prove that there exists a RegEx for every GNF grammar that is also linear.

Solution: A linear GNF grammar is actually a right-linear grammar whose language is regular. Therefore it has a RegEx.

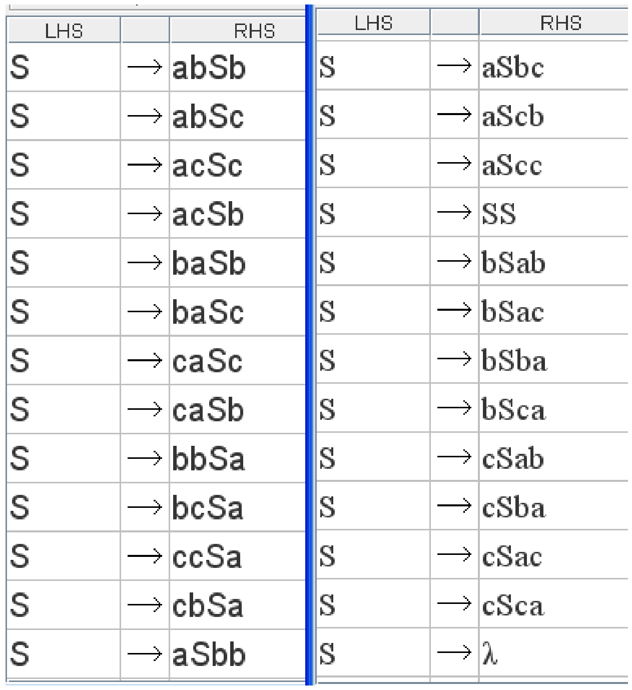
6. Consider the set of all strings that begin with a certain number of *x* s followed by a number of *y* s such that the number of *x* s is seven more than twice the number of *y* s. Is this a context-free language?

   Solution: Yes, we can construct a context-free grammar for this language (SEE VIDEO SOLUTION) and therefore it is a context-free language.
   $S \rightarrow xxxxxxxA,\ A \rightarrow xxAy \mid \lambda$

   Please see also the VIDEO SOLUTION for this exercise.

7. Show that *L* = {*w* | length of *w* = 3 × number of *a* s in *w*} is a context-free language. The alphabet is {*a, b, c*}.

   Solution: A context-free grammar for this language is:



LHS		RHS	LHS		RHS
S	→	abSb	S	→	aSbc
S	→	abSc	S	→	aScb
S	→	acSc	S	→	aScc
S	→	acSb	S	→	SS
S	→	baSb	S	→	bSab
S	→	baSc	S	→	bSac
S	→	caSc	S	→	bSba
S	→	caSb	S	→	bSca
S	→	bbSa	S	→	cSab
S	→	bcSa	S	→	cSba
S	→	ccSa	S	→	cSac
S	→	cbSa	S	→	cSca
S	→	aSbb	S	→	λ

8. Consider the simple nesting language a^nb^n for matching parentheses. Given that in no real program the nesting of parentheses can be to an infinite depth, let us limit nesting to, say, a depth of 10. If all other symbols are ignored, the language becomes finite. Since we know that a finite language is regular, should we model this language as a regular language or is there still some advantage in modeling it using a context-free grammar? Justify your answer by constructing both models.

   Solution: It is more efficient to model it as a context-free grammar. If we try to make it a RegEx, for example, it becomes unwieldy:

   ab + aabb + aaabbb + aaaabbbb + aaaaabbbbb + aaaaaabbbbbb + aaaaaaabbbbbbb + aaaaaaaabbbbbbbb + aaaaaaaaabbbbbbbbb + aaaaaaaaaabbbbbbbbbb

   Similarly, a DFA would need at least 10 states (see Figure-2.14 which used more states than necessary). The moreover, counting is an inherent characteristic of this language which will not be handled in either a RegEx or the automaton.

9. Repeat the above exercise for the proper nesting language.

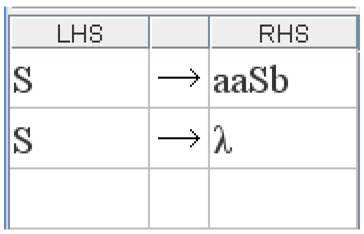
Solution: The details are left out here. In this case, there will be even more possibilities and it becomes rather impractical to model the language as a RegEx, finite automaton or regular grammar. For example, the RegEx must include expressions of the form:

ab + aabb + abab + aaabbb + ababab + aabbab + abaabb + …

A. Construct a context-free grammar for each of the following languages:

10. $a^{2n}b^n$

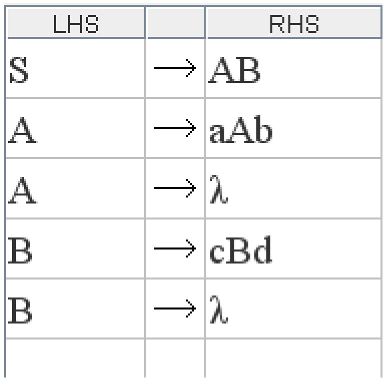
Solution:



LHS		RHS
S	$\rightarrow$	aaSb
S	$\rightarrow$	λ

11. $a^nb^nc^md^m$

Solution:

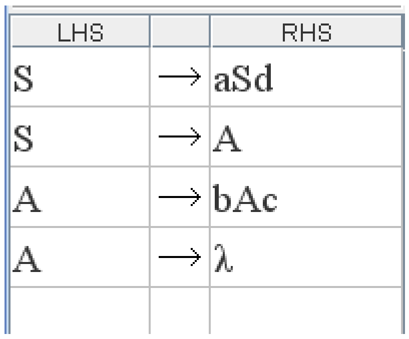


LHS		RHS
S	$\rightarrow$	AB
A	$\rightarrow$	aAb
A	$\rightarrow$	λ
B	$\rightarrow$	cBd
B	$\rightarrow$	λ

Please see also the VIDEO SOLUTION for this exercise.

12. $a^nb^mc^md^n$

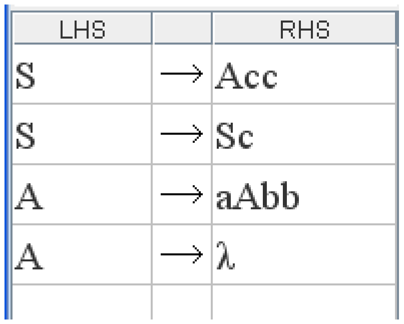
Solution:



LHS		RHS
S	$\rightarrow$	aSd
S	$\rightarrow$	A
A	$\rightarrow$	bAc
A	$\rightarrow$	λ

13. $a^n b^m c^k$ where $2n = m$ and $k \geq 2$.

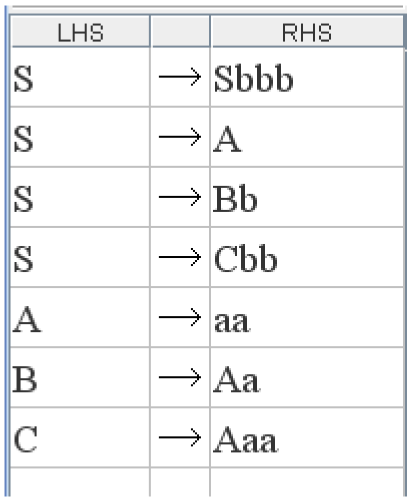
Solution:



LHS		RHS
S	$\longrightarrow$	Acc
S	$\longrightarrow$	Sc
A	$\longrightarrow$	aAbb
A	$\longrightarrow$	λ

14. $a^n b^m$ where $n = 2 + (m \bmod 3)$. Is this a regular language? If so, make sure that the grammar is regular.

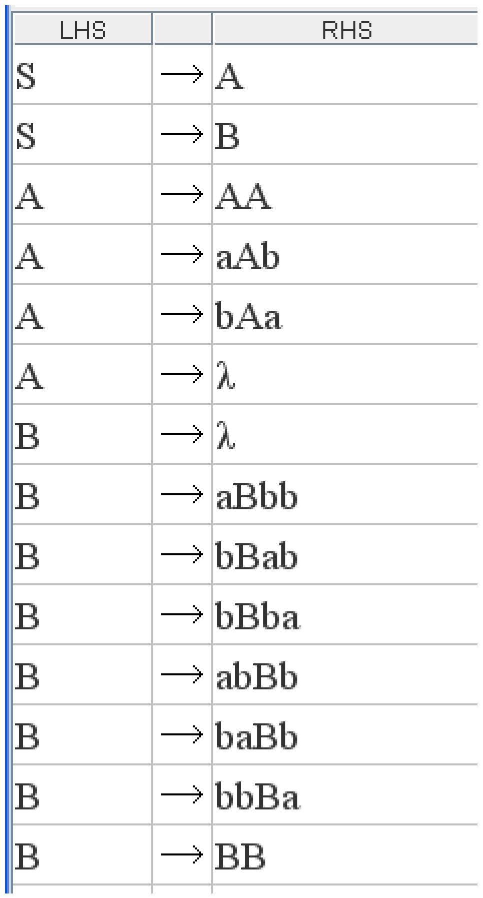
Solution: Yes, this is a regular language. The grammar below is left-linear.



LHS		RHS
S	$\longrightarrow$	Sbbb
S	$\longrightarrow$	A
S	$\longrightarrow$	Bb
S	$\longrightarrow$	Cbb
A	$\longrightarrow$	aa
B	$\longrightarrow$	Aa
C	$\longrightarrow$	Aaa

15. All strings over {*a, b*} with either equal numbers of *a* and *b* or twice as many *b* s as *a* s.

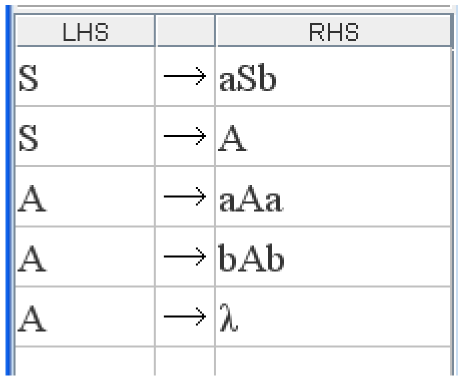
Solution:



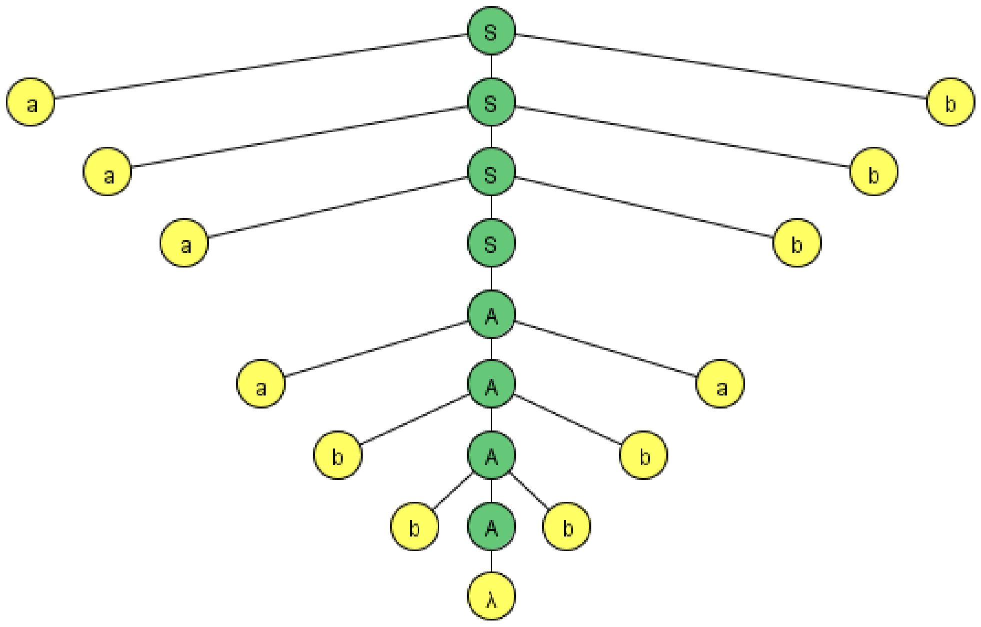
LHS		RHS
S	→	A
S	→	B
A	→	AA
A	→	aAb
A	→	bAa
A	→	λ
B	→	λ
B	→	aBbb
B	→	bBab
B	→	bBba
B	→	abBb
B	→	baBb
B	→	bbBa
B	→	BB

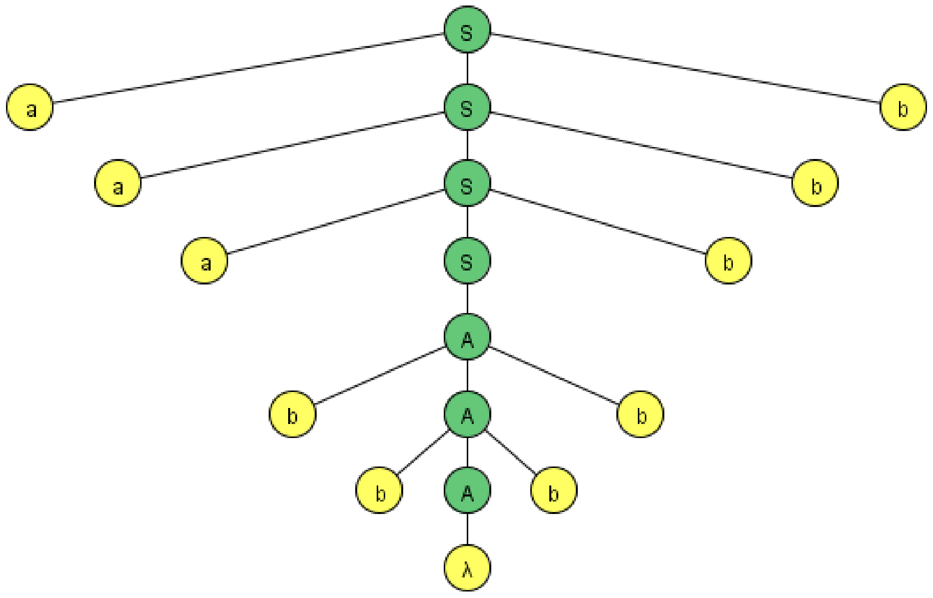
16. $a^n w w^R b^n$ where w is any string in $(a + b)^*$. Show the derivation for the string *aaaabbbbabbb*. Is this grammar ambiguous? How can it derive the string *aaabbbbbbb*?

Solution: It is not ambiguous. The parse trees for the two strings are shown below.



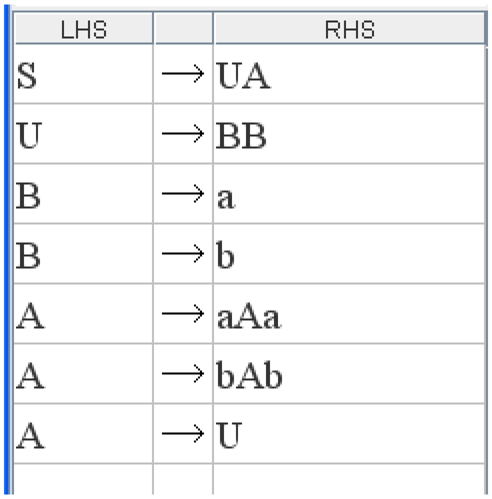
LHS		RHS
S	→	aSb
S	→	A
A	→	aAa
A	→	bAb
A	→	λ





17. $uvwv^R$ where v is of length at least 1 and u and w are of length 2. The alphabet is $\{a, b\}$.

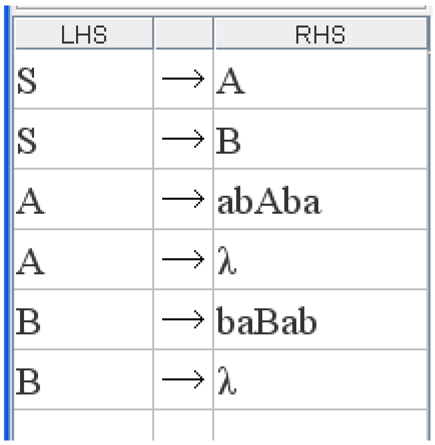
Solution:



LHS		RHS
S	$\longrightarrow$	UA
U	$\longrightarrow$	BB
B	$\longrightarrow$	a
B	$\longrightarrow$	b
A	$\longrightarrow$	aAa
A	$\longrightarrow$	bAb
A	$\longrightarrow$	U

18. ww^R where $w = (ab)^* + (ba)^*$ the alphabet being $\{a, b\}$.

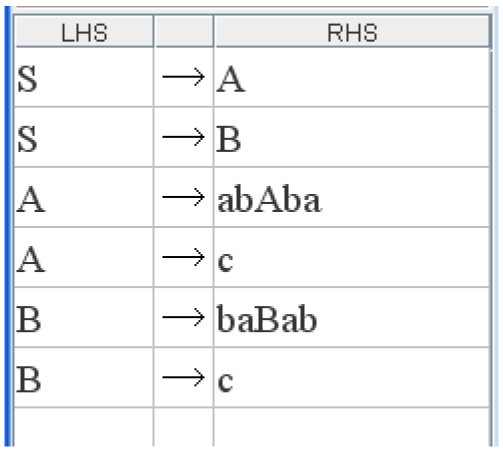
    Solution:



LHS		RHS
S	$\longrightarrow$	A
S	$\longrightarrow$	B
A	$\longrightarrow$	abAba
A	$\longrightarrow$	λ
B	$\longrightarrow$	baBab
B	$\longrightarrow$	λ

19. wcw^R where $w = (ab)^* + (ba)^*$ the alphabet being $\{a, b, c\}$.

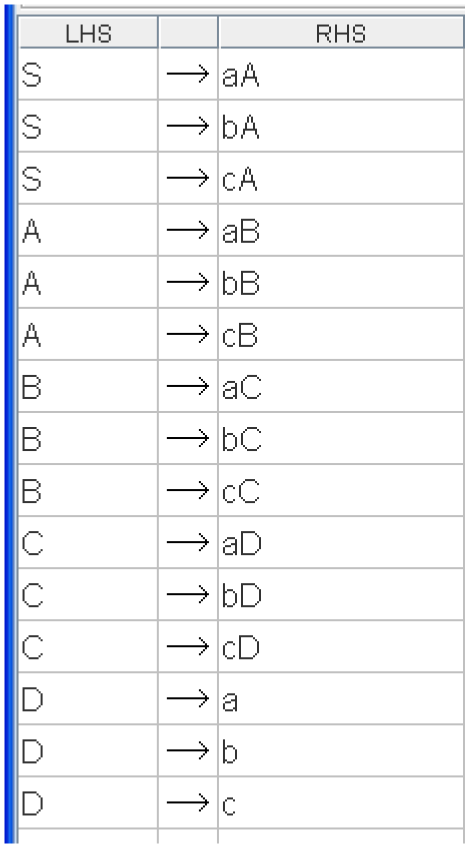
    Solution:



LHS		RHS
S	$\longrightarrow$	A
S	$\longrightarrow$	B
A	$\longrightarrow$	abAba
A	$\longrightarrow$	c
B	$\longrightarrow$	baBab
B	$\longrightarrow$	c

20. All strings of length 5 over the alphabet $\{a, b, c\}$. Can you make sure that the grammar is in Greibach Normal Form?

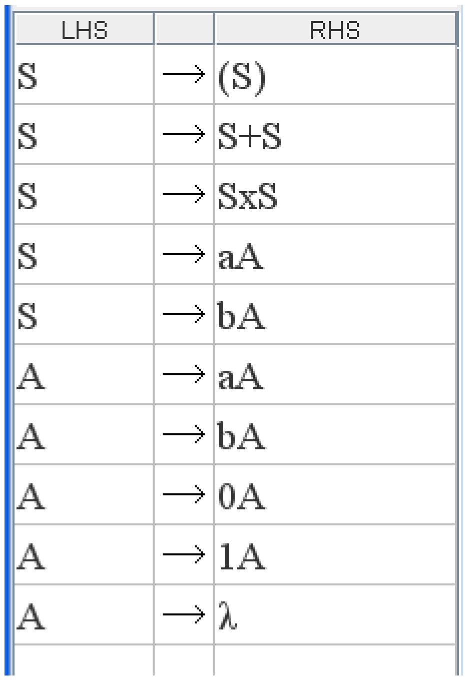
Solution:



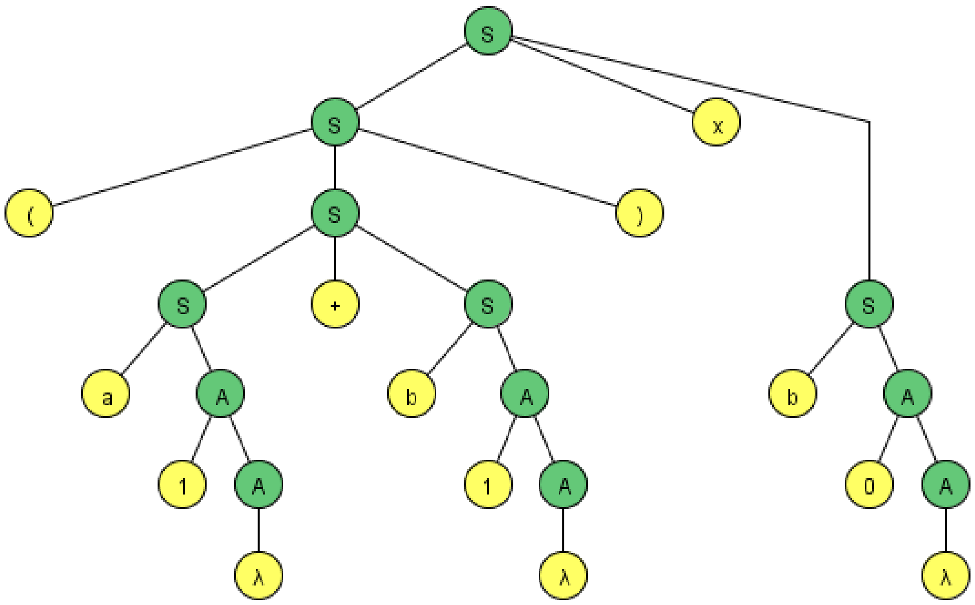
LHS		RHS
S	$\longrightarrow$	aA
S	$\longrightarrow$	bA
S	$\longrightarrow$	cA
A	$\longrightarrow$	aB
A	$\longrightarrow$	bB
A	$\longrightarrow$	cB
B	$\longrightarrow$	aC
B	$\longrightarrow$	bC
B	$\longrightarrow$	cC
C	$\longrightarrow$	aD
C	$\longrightarrow$	bD
C	$\longrightarrow$	cD
D	$\longrightarrow$	a
D	$\longrightarrow$	b
D	$\longrightarrow$	c

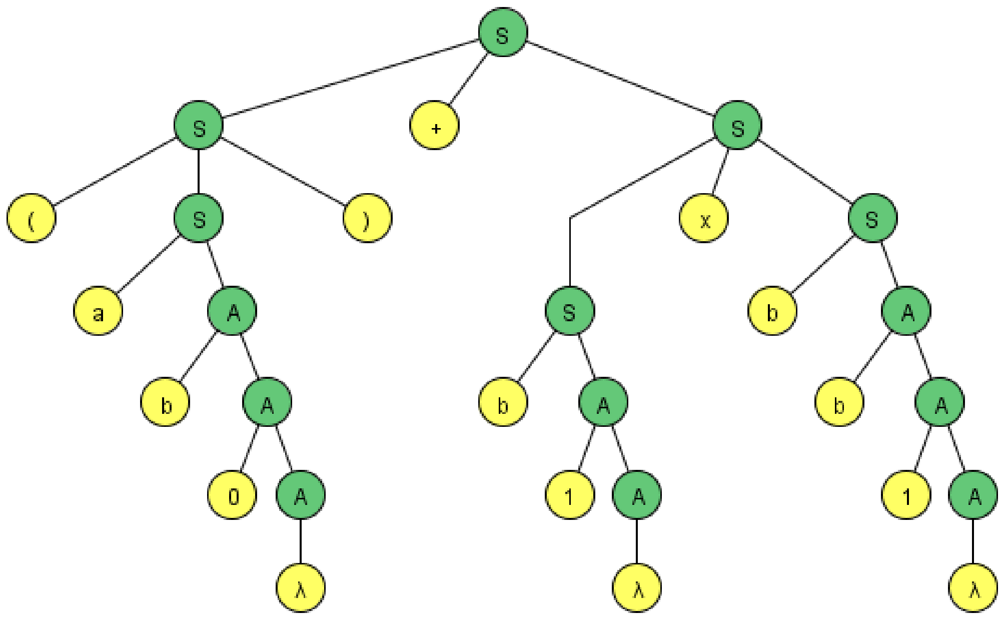
21. Arithmetic expressions that contain identifiers defined by the RegEx $(a + b).(a + b + 0 + 1)^*$. Expressions support the + and × operators and properly nested parentheses. For the CFG constructed, show derivations and parse trees for the following strings:
(a) $(a1+b1)\times b0$
(b) $(ab0)+b1\times bb1$

Solution: Parse trees for the two strings are shown below.



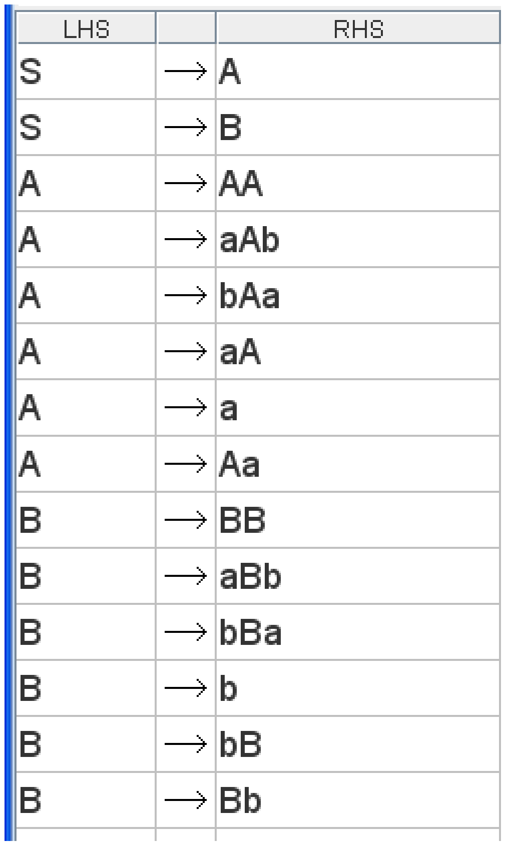
LHS		RHS
S	$\longrightarrow$	(S)
S	$\longrightarrow$	S+S
S	$\longrightarrow$	SxS
S	$\longrightarrow$	aA
S	$\longrightarrow$	bA
A	$\longrightarrow$	aA
A	$\longrightarrow$	bA
A	$\longrightarrow$	0A
A	$\longrightarrow$	1A
A	$\longrightarrow$	λ





22. Strings over {*a, b*} with either more *a* s than *b* s or less *a* s than *b* s. Is this language regular or context-free? Explain.

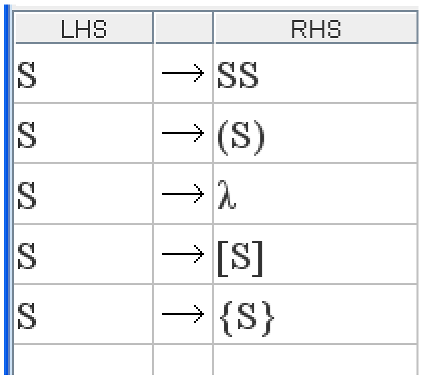
    Solution: This is context-free; it involves counting *a* s and *b* s. The grammar is below is not even linear.



LHS		RHS
S	→	A
S	→	B
A	→	AA
A	→	aAb
A	→	bAa
A	→	aA
A	→	a
A	→	Aa
B	→	BB
B	→	aBb
B	→	bBa
B	→	b
B	→	bB
B	→	Bb

23. Proper nesting of a combination of parentheses, square brackets and flower braces. For example,
    { [( ) ( )] { ([ ]) [ ] } }

Solution:



LHS		RHS
S	$\rightarrow$	SS
S	$\rightarrow$	(S)
S	$\rightarrow$	λ
S	$\rightarrow$	[S]
S	$\rightarrow$	{S}

Please see also the VIDEO SOLUTION for this exercise.

B. Describe the language of the following context-free grammars as concisely as possible:

24. $S \rightarrow pA \mid qB \mid rC,\ A \rightarrow Sp,\ B \rightarrow Sq,\ C \rightarrow \lambda$

Solution: Odd palindromes made up of *p* and *q* with a single *r* at the center.

25. $S \rightarrow AB \mid \lambda,\ A \rightarrow aB,\ B \rightarrow Bb \mid b$

Solution: Strings starting with one *a* and having two or more *b* s plus the empty string λ.

26. $S \rightarrow aS \mid bS \mid SS \mid \lambda$

Solution: $(a + b)^*$

27. $S \rightarrow aS \mid bS \mid SSS \mid \lambda$

Solution: $(a + b)^*$

28. $S \rightarrow aS \mid Sb \mid bS \mid Sa \mid SS \mid \lambda$

Solution: $(a + b)^*$

29. $S \rightarrow aA \mid Bb \mid \lambda,\ A \rightarrow Aa \mid a,\ B \rightarrow bB \mid b$

Solution: $(\lambda + aaa^* + bbb^*)$, that is, two or more *a* s or two or more *b* s.

30. $S \rightarrow aAb \mid aBb \mid \lambda,\ A \rightarrow aBb \mid a,\ B \rightarrow aAb \mid b$

Solution: $(\lambda + a^nab^n + a^nbb^n)$, that is, simple nesting language with either an extra *a* or an extra *b* at the center.

31. $S \rightarrow aSa \mid A, A \rightarrow bAb \mid \lambda$

Solution: $a^nb^ma^n$ where m is even.

32. $S \rightarrow aS \mid AB, A \rightarrow bA \mid B, B \rightarrow AA \mid \lambda$

Solution: a^nb^m.

33. $S \rightarrow a \mid bB \mid ccC, B \rightarrow bB \mid \lambda, C \rightarrow cC \mid \lambda, D \rightarrow d$

Solution: One *a* or one or more *b* s or two or more *c* s.

34. $S \rightarrow AaB \mid aaB, A \rightarrow \lambda, B \rightarrow bb \mid \lambda$

Solution: Strings starting with either one or two *a* s followed by zero or two *b* s. The language has only four strings: {*a*, *aa*, *abb*, *aabb*}

35. Consider the grammar called G_{mid}: $S \rightarrow aSa \mid \lambda$. Is G_{mid} right-linear or left-linear? Is the language a regular language? Convert the grammar to a regular grammar.

Solution: It is neither right-linear nor left-linear. However, the language is regular: it is just $(aa)^*$. An equivalent regular grammar is:

$$G_{\text{reg}}: S \rightarrow aaS \mid \lambda$$

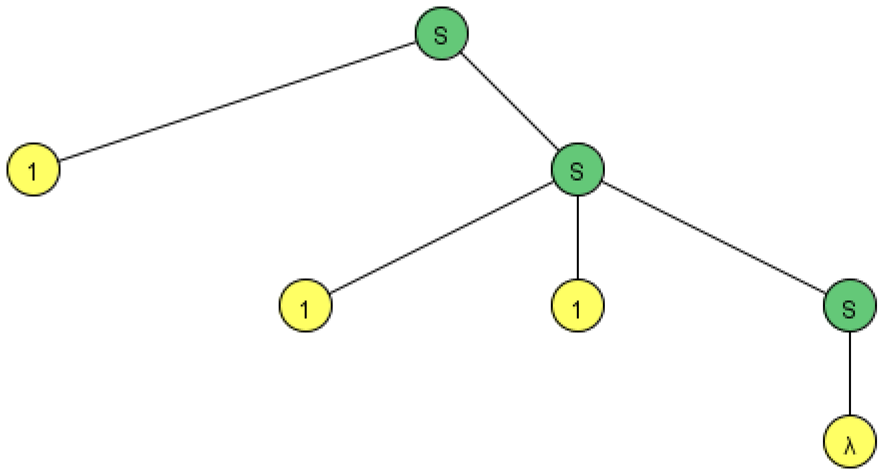
36. Consider $S \rightarrow aSb \mid \lambda$. Can this be converted to a regular grammar?

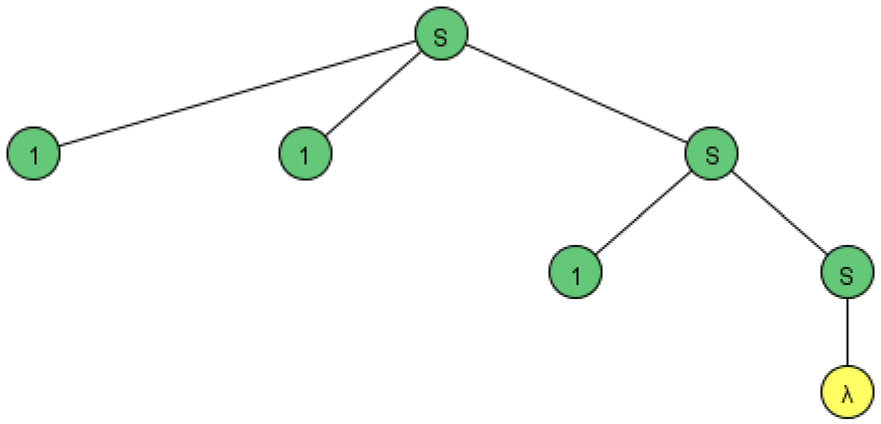
Solution: No, it cannot be made regular. This involves counting and matching the number of *a* s and *b* s. For a proof that it is not regular, see Pumping Lemma in Chapter 6.

C. The following problems are concerned with ambiguity in grammars:

37. Show that $S \rightarrow 1S \mid 11S \mid \lambda$ is ambiguous.

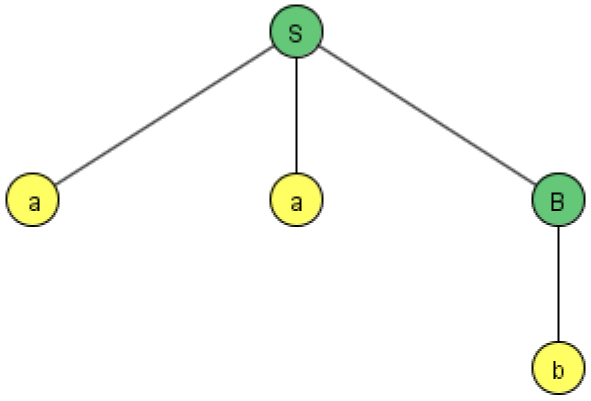
Solution: Consider the strings 111. There are two different leftmost derivations for this string as shown in the two parse trees below. Hence it is ambiguous.

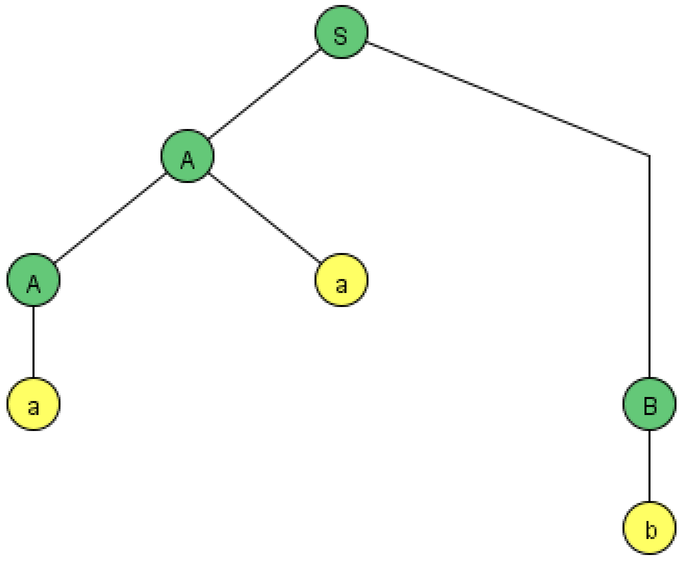




38. Is the following grammar ambiguous? $S \rightarrow AB \mid aaB, A \rightarrow a \mid Aa, B \rightarrow b$

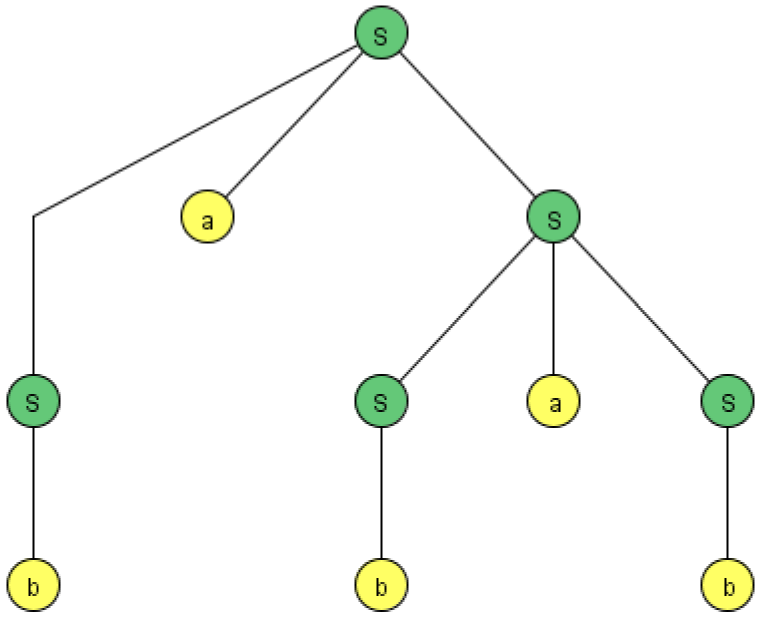
Solution: Consider the string *aab*. There are two different leftmost derivations for this string as shown in the two parse trees below. Hence it is ambiguous.

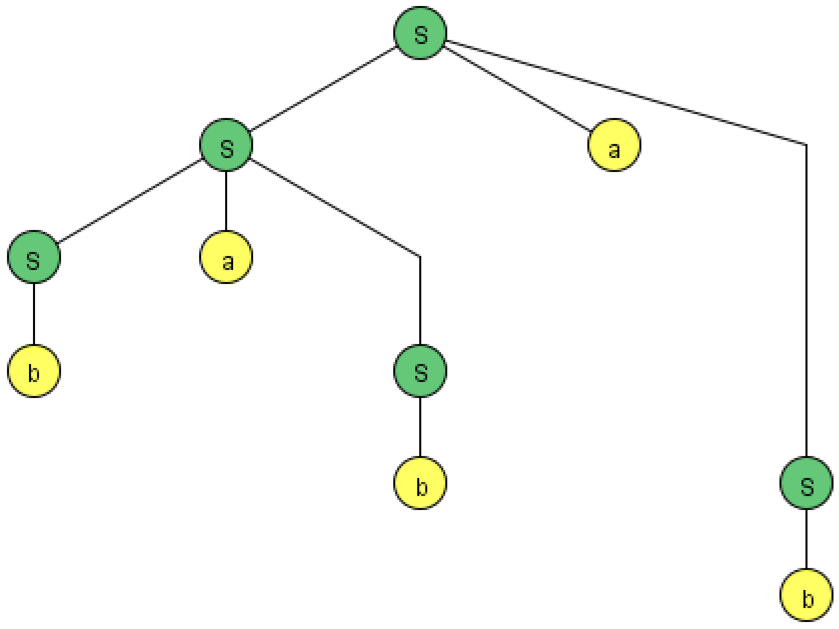




39. Show that $S \rightarrow SaS \mid b$ is ambiguous. Construct an unambiguous equivalent of the grammar.

Solution: Consider the string *babab*. There are two different leftmost derivations for this string as shown in the two parse trees below. Hence it is ambiguous.





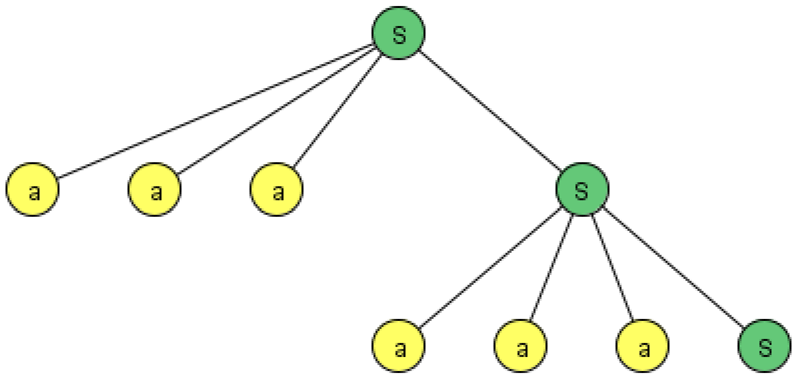
An unambiguous equivalent is the regular grammar: $S \rightarrow baS \mid b$

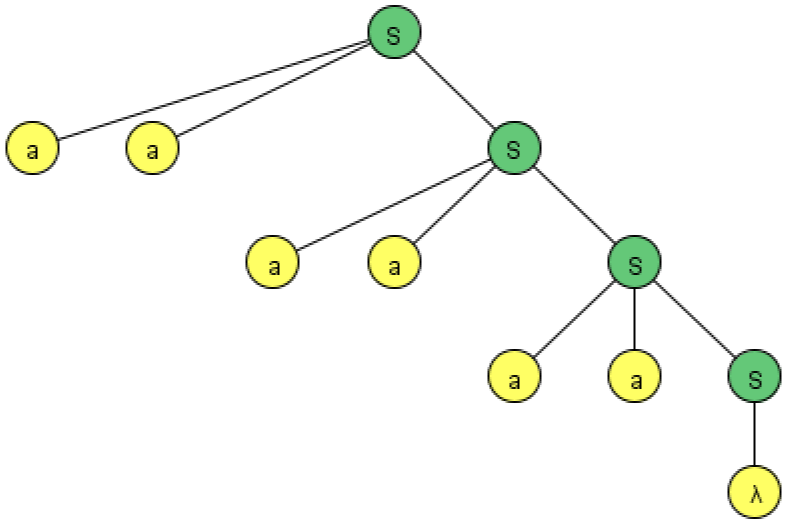
40. Show that the union of $a^n b^n c^m$ and $a^n b^m c^m$ is inherently ambiguous.

Solution: Consider the string $a^p b^p c^p$. This belongs to both sub-languages and can be parsed using the grammars for both languages. Hence the combined language is inherently ambiguous.

41. Consider the language of the grammar: $S \rightarrow aaS \mid aaaS \mid \lambda$. Is the language inherently ambiguous? Can you construct an unambiguous grammar for it?

Solution: Consider the string *aaaaaa*. There are two different leftmost derivations for this string as shown in the two parse trees below. Hence the given grammar is ambiguous.





However, this does not mean that the language itself is inherently ambiguous. In fact, the language is regular: $(aa + aaa)^*$ and we can easily construct an equivalent but unambiguous regular grammar: $S \rightarrow aaA \mid \lambda, A \rightarrow aA \mid \lambda$

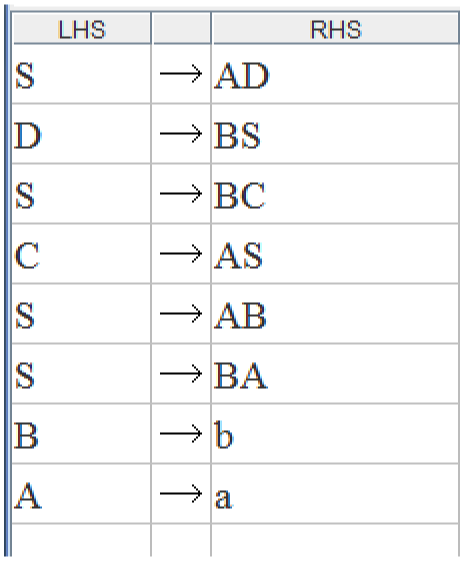
42. Consider the language of the grammar:
$S \rightarrow 0A \mid 1A \mid SS \mid \lambda, A \rightarrow 0B \mid 1B \mid 00S \mid 01S \mid 10S \mid 11S, B \rightarrow 0S \mid 1S \mid 00A \mid 01A \mid 10A \mid 11A$
Can you construct an unambiguous grammar for the same language?

Solution: The language of this grammar is just the set of all binary strings whose length is a multiple of 3. The variable A generates all strings whose length mod 3 = 1 and variable B those whose length mod 3 = 2. An equivalent grammar is:
$S \rightarrow 0A \mid 1A \mid \lambda, A \rightarrow 0B \mid 1B, B \rightarrow 0S \mid 1S$

D. Convert the following grammars to Chomsky Normal Form:

43. $S \rightarrow abS \mid baS \mid \lambda$

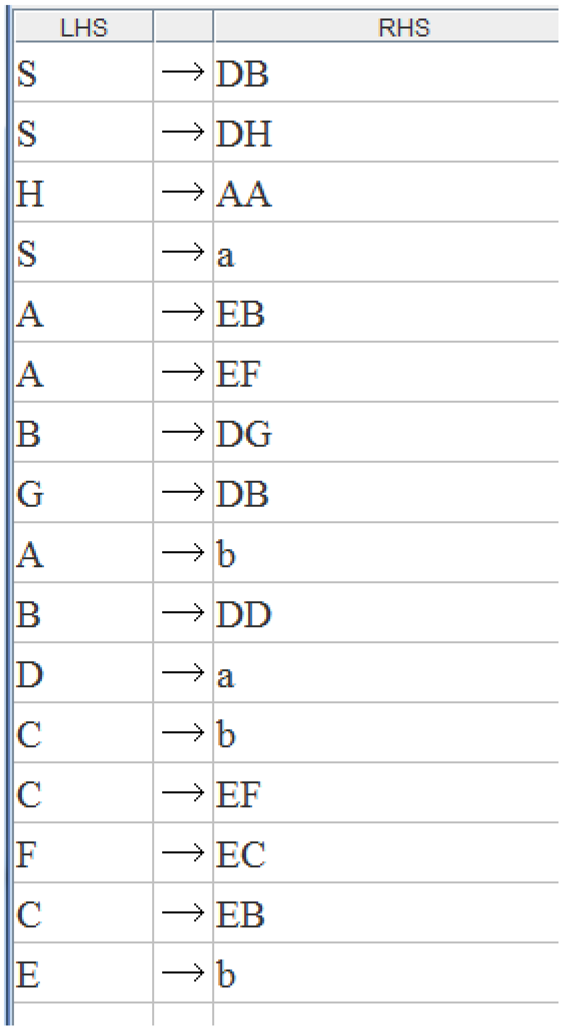
Solution: The CNF grammar is:



LHS		RHS
S	→	AD
D	→	BS
S	→	BC
C	→	AS
S	→	AB
S	→	BA
B	→	b
A	→	a

44. $A \rightarrow aB \mid aAA, A \rightarrow bB \mid bbC, B \rightarrow aaB \mid \lambda, C \rightarrow A$

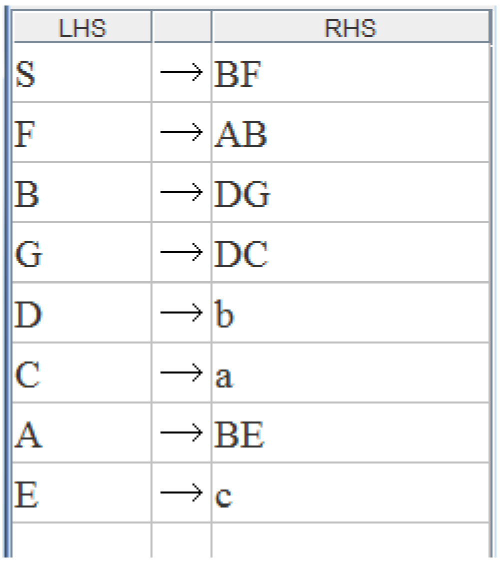
Solution: Assuming that the left-hand side of the first production is *S* not *A*, the CNF grammar is:



LHS		RHS
S	→	DB
S	→	DH
H	→	AA
S	→	a
A	→	EB
A	→	EF
B	→	DG
G	→	DB
A	→	b
B	→	DD
D	→	a
C	→	b
C	→	EF
F	→	EC
C	→	EB
E	→	b

45. $S \rightarrow BAB, B \rightarrow bba, A \rightarrow Bc$

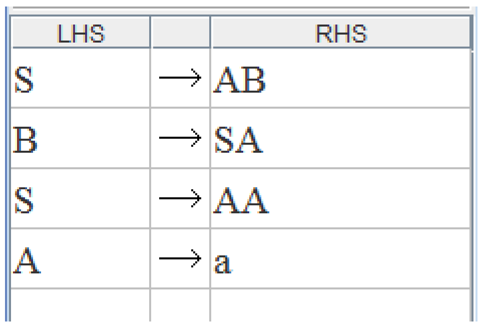
Solution:



LHS		RHS
S	$\rightarrow$	BF
F	$\rightarrow$	AB
B	$\rightarrow$	DG
G	$\rightarrow$	DC
D	$\rightarrow$	b
C	$\rightarrow$	a
A	$\rightarrow$	BE
E	$\rightarrow$	c

46. G_{mid} from Exercise 35 above.

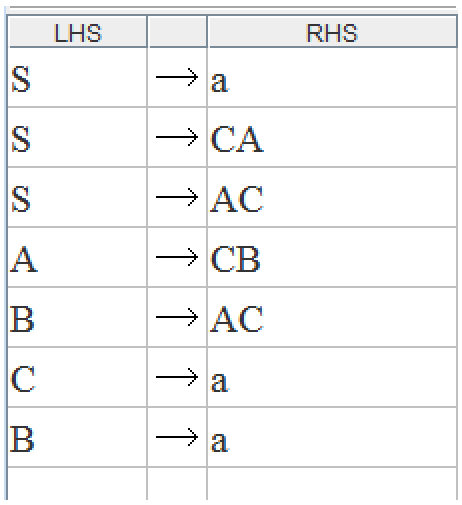
Solution:



LHS		RHS
S	$\rightarrow$	AB
B	$\rightarrow$	SA
S	$\rightarrow$	AA
A	$\rightarrow$	a

47. $S \rightarrow a \mid aA \mid B \mid C, A \rightarrow aB \mid \lambda, B \rightarrow Aa, C \rightarrow cCD, D \rightarrow$ add

Solution:



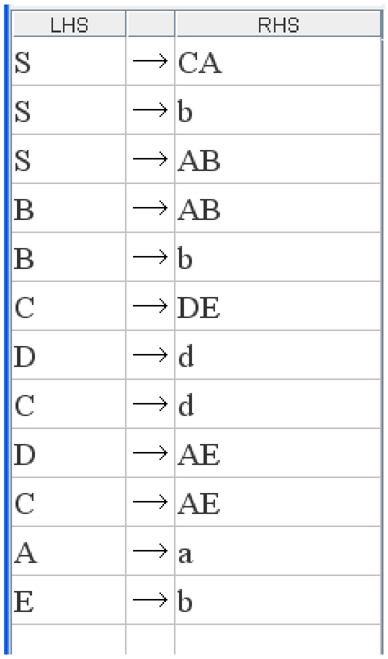
LHS		RHS
S	$\rightarrow$	a
S	$\rightarrow$	CA
S	$\rightarrow$	AC
A	$\rightarrow$	CB
B	$\rightarrow$	AC
C	$\rightarrow$	a
B	$\rightarrow$	a

48. $S \rightarrow AS \mid AAAS, A \rightarrow SA \mid aa \mid b$

Solution: The start variable S is itself useless! It does not generate any terminal string. The language of the grammar is empty. The equivalent CNF grammar has no production rules at all!

49. $S \rightarrow Aa \mid B \mid Ca, B \rightarrow aB \mid b, C \rightarrow Db \mid D, D \rightarrow E \mid d, E \rightarrow ab$

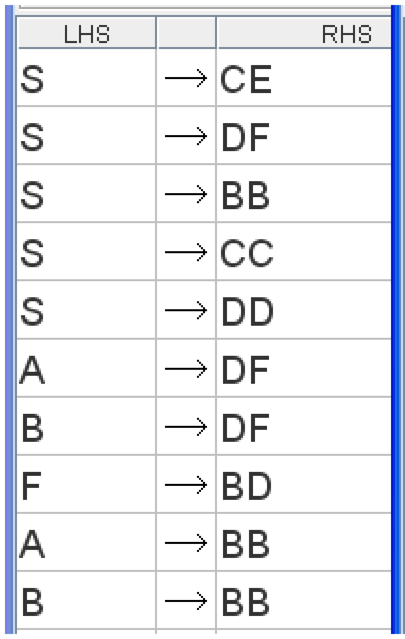
Solution: The CNF grammar is shown below:



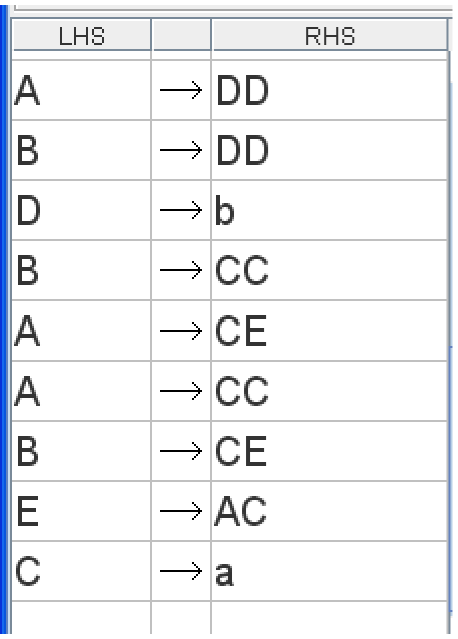
LHS		RHS
S	→	CA
S	→	b
S	→	AB
B	→	AB
B	→	b
C	→	DE
D	→	d
C	→	d
D	→	AE
C	→	AE
A	→	a
E	→	b

50. $S \rightarrow aAa \mid bBb \mid BB, A \rightarrow C, B \rightarrow S \mid A, C \rightarrow S \mid \lambda$

Solution: The CNF grammar is:



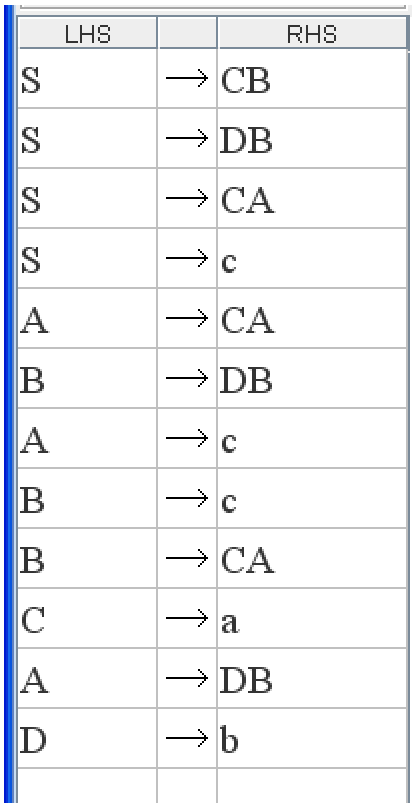
LHS		RHS
S	→	CE
S	→	DF
S	→	BB
S	→	CC
S	→	DD
A	→	DF
B	→	DF
F	→	BD
A	→	BB
B	→	BB



LHS		RHS
A	→	DD
B	→	DD
D	→	b
B	→	CC
A	→	CE
A	→	CC
B	→	CE
E	→	AC
C	→	a

51. $S \rightarrow A \mid aB \mid \lambda, A \rightarrow aA \mid B, B \rightarrow bB \mid C, C \rightarrow c \mid A$

Solution: The CNF grammar is:



LHS		RHS
S	$\rightarrow$	CB
S	$\rightarrow$	DB
S	$\rightarrow$	CA
S	$\rightarrow$	c
A	$\rightarrow$	CA
B	$\rightarrow$	DB
A	$\rightarrow$	c
B	$\rightarrow$	c
B	$\rightarrow$	CA
C	$\rightarrow$	a
A	$\rightarrow$	DB
D	$\rightarrow$	b

E. Convert the following grammars to Greibach Normal Form:

52. $S \rightarrow Sab \mid Sba \mid \lambda$

Solution: It really does not matter for this regular language whether we add the *ab* or *ba* on the right side or left side of *S.* An equivalent grammar in Greibach Normal Form is:
$S \rightarrow aBS \mid bAS \mid \lambda, A \rightarrow a, B \rightarrow b$

53. $S \rightarrow AB, A \rightarrow aA \mid bB \mid b, B \rightarrow b$

Solution: Only the *S* production is not in GNF. Substituting for *A*, we get:
$S \rightarrow aAB \mid bBB \mid bB, A \rightarrow aA \mid bB \mid b, B \rightarrow b$

54. $S \rightarrow abSb \mid aa$

Solution: An equivalent GNF grammar is:
$S \rightarrow aBSB \mid aA, A \rightarrow a, B \rightarrow b$

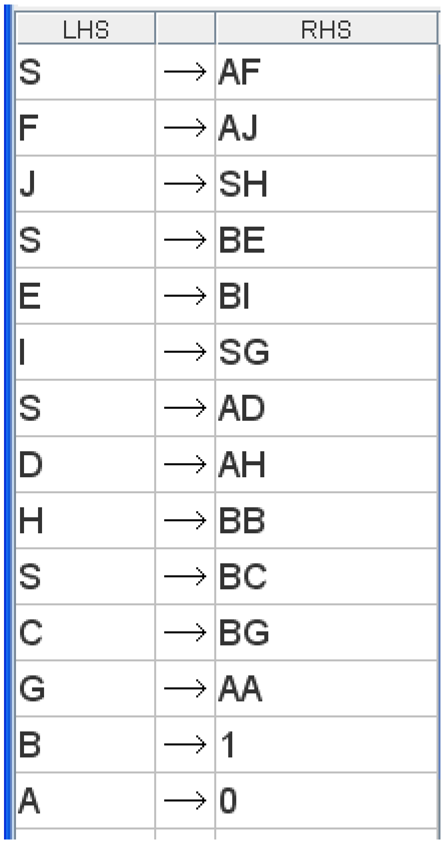
55. $S \rightarrow aSb \mid ab$

Solution: An equivalent GNF grammar is:
$S \rightarrow aSB \mid aB, B \rightarrow b$

F. Apply the CYK-algorithm to verify if the given string(s) can be derived by the grammar:

56. $S \rightarrow 00S11 \mid 11S00 \mid \lambda$, $w = 00111100$

Solution: An equivalent CNF grammar is:

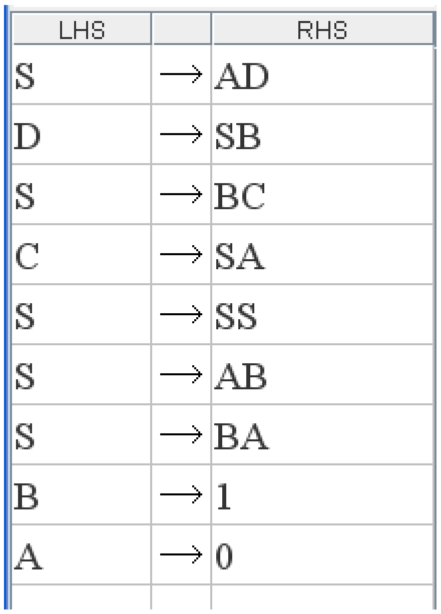


LHS		RHS
S	$\longrightarrow$	AF
F	$\longrightarrow$	AJ
J	$\longrightarrow$	SH
S	$\longrightarrow$	BE
E	$\longrightarrow$	BI
I	$\longrightarrow$	SG
S	$\longrightarrow$	AD
D	$\longrightarrow$	AH
H	$\longrightarrow$	BB
S	$\longrightarrow$	BC
C	$\longrightarrow$	BG
G	$\longrightarrow$	AA
B	$\longrightarrow$	1
A	$\longrightarrow$	0

Constructing the CYK table is left as an exercise for the reader.

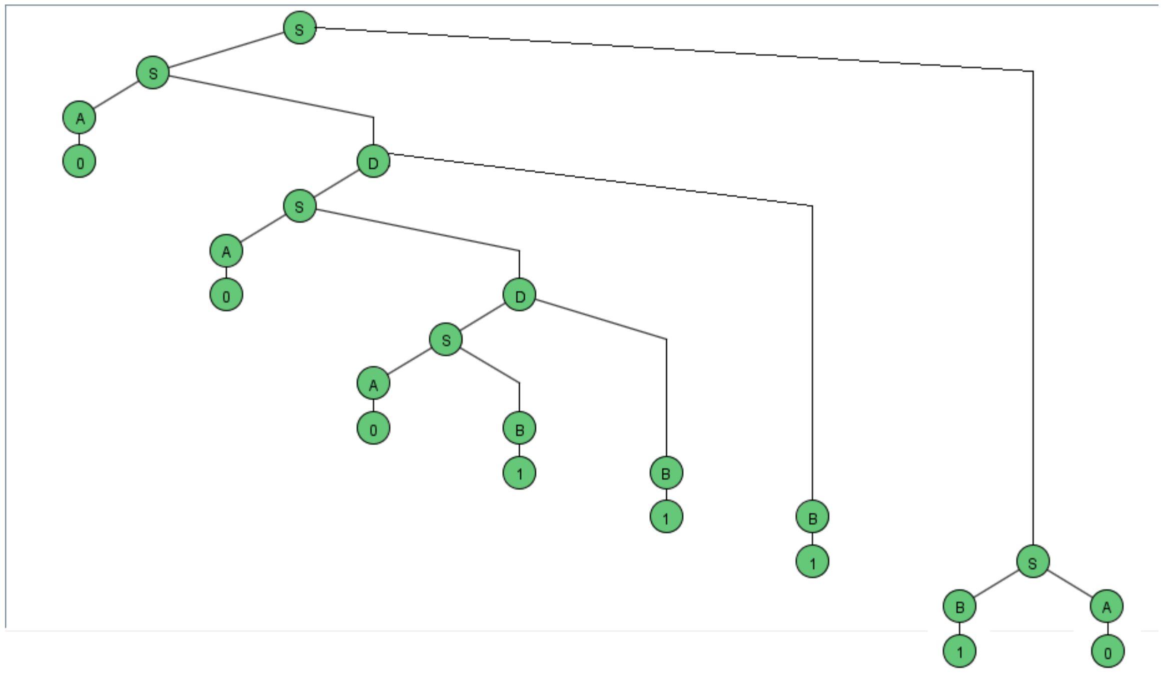
57. $S \rightarrow 0S1 \mid 1S0 \mid SS \mid \lambda$, $w = 00011110$

Solution: An equivalent CNF grammar is:



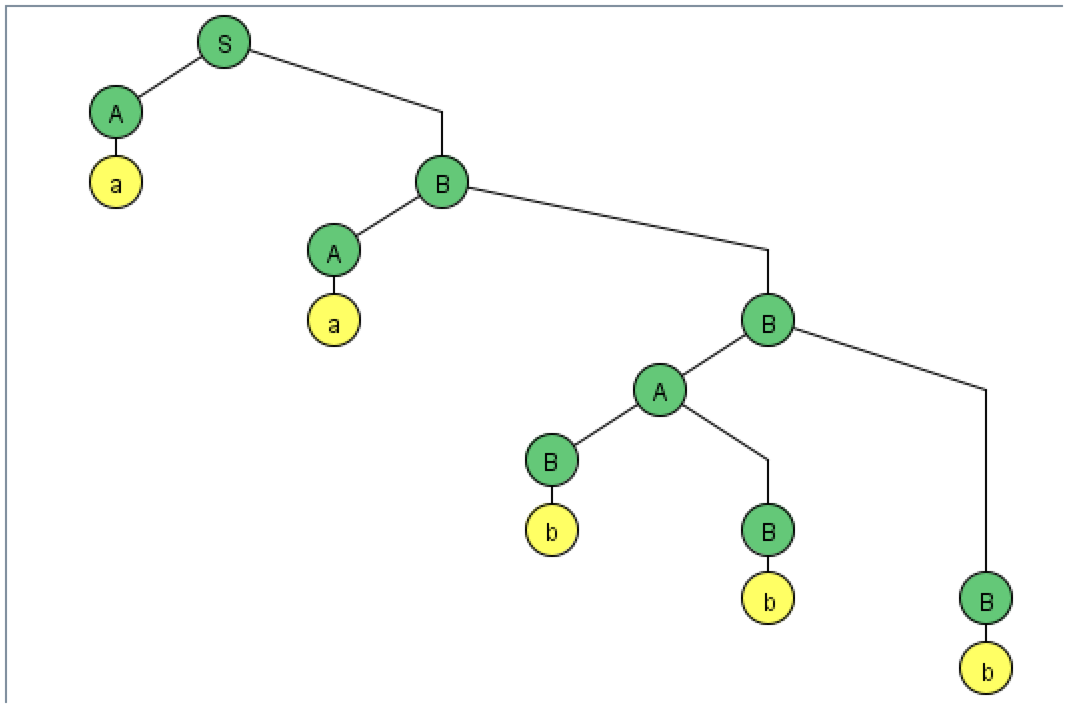
LHS		RHS
S	$\longrightarrow$	AD
D	$\longrightarrow$	SB
S	$\longrightarrow$	BC
C	$\longrightarrow$	SA
S	$\longrightarrow$	SS
S	$\longrightarrow$	AB
S	$\longrightarrow$	BA
B	$\longrightarrow$	1
A	$\longrightarrow$	0

The string is accepted and the resulting CYK parse tree is:



58. $S \to AB$, $A \to BB \mid a$, $B \to AB \mid b$, w = *aabbb, aabb, aabba, abbbb*

Solution: This grammar is already in CNF. The string *aabbb* is accepted and the CYK parse tree is:



The other three strings are all rejected.

59. $S \rightarrow AB, A \rightarrow BB \mid a, B \rightarrow AB \mid b, w = aabbb$

Solution: This happens to be the same as Exercise 58 (the first input string therein).

G. Debug and correct the following context-free grammars:

60. For the language $a^n b^m$ where $m = 3n$:
$S \rightarrow aS \mid B, B \rightarrow Sbbb \mid \lambda$

Solution: The correct grammar is:
$S \rightarrow aSbbb \mid \lambda$

61. For two more *a* s than *b* s (in any order):
$S \rightarrow aaA \mid Aaa \mid \lambda, A \rightarrow aAb \mid bAa \mid AA \mid \lambda$

Solution: The corrected grammar is (note that *A* generates all strings with equal numbers of *a* and *b*):
$S \rightarrow AaAaA, A \rightarrow aAb \mid bAa \mid AA \mid \lambda$

62. For $a^n b^m c^k$ where $k = n - m$ (the language of subtraction):
$S \rightarrow aSb \mid bSc \mid \lambda$

Solution: Note that the language is the same as $a^k a^m b^m c^k$. The corrected grammar is:
$S \rightarrow aSc \mid A, A \rightarrow aAb \mid \lambda$

63. For binary string in which every run of 1 s is of odd length:

$S \to 0S \mid 1A \mid \lambda, A \to 1A \mid 0A \mid \lambda$

Solution: The corrected grammar is:
$S \to 0S \mid 1A \mid \lambda, A \to 11A \mid 0S$

H. Open-ended exercises:

64. Write a context-free grammar for the structure of nested loops of various kinds in a modern programming language such as Java.

    Note: There is no unique answer to this exercise.

65. Consider a modern programming language such as Java and find out what elements of it, if any, are not context-free (i.e., context-sensitive).

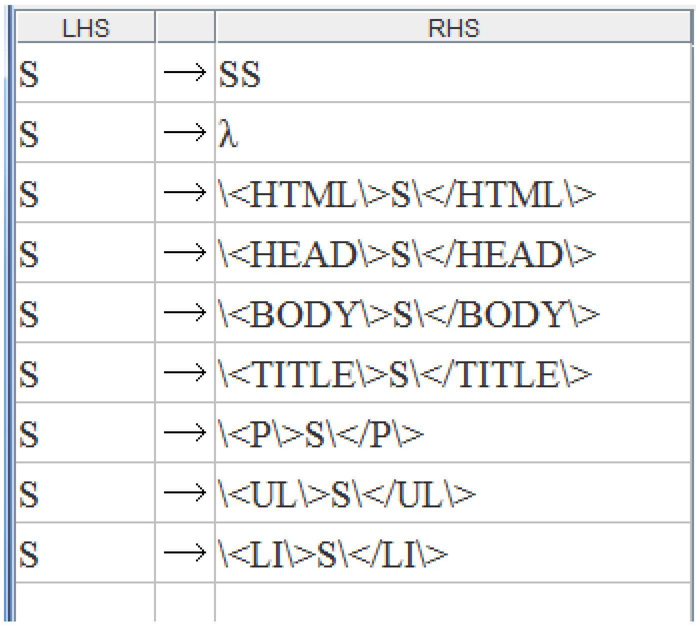
    Note: There is no unique answer to this exercise.

66. Write down a comprehensive set of production rules for the grammar of English considering most types of simple, assertive sentences (no questions, imperatives, etc.).

    Note: There is no unique answer to this exercise.

67. Consider HTML, HyperText Markup Language, the language of the Web. A valid HTML string has a set of matching start and end tags. Both are enclosed in angle brackets with the end tag having an extra symbol, the backslash at the beginning. For example, <P> is a start tag and </P> is the matching end tag. Tags must be properly nested with every start tag matching a corresponding end tag (although in reality most web browsers are rather forgiving and do not enforce this strictly). For example, a valid HTML string is
    <HTML><HEAD><TITLE></TITLE></HEAD>
    <BODY><P><UL><LI></LI><LI></LI></UL></P></BODY></HTML>
    ignoring of course the actual contents of a web page which appear in between start and end tags. Assuming that the only possible tags are {HTML, HEAD, BODY, TITLE, P, UL, LI}, construct a CFG for valid HTML tag strings.

    Solution:

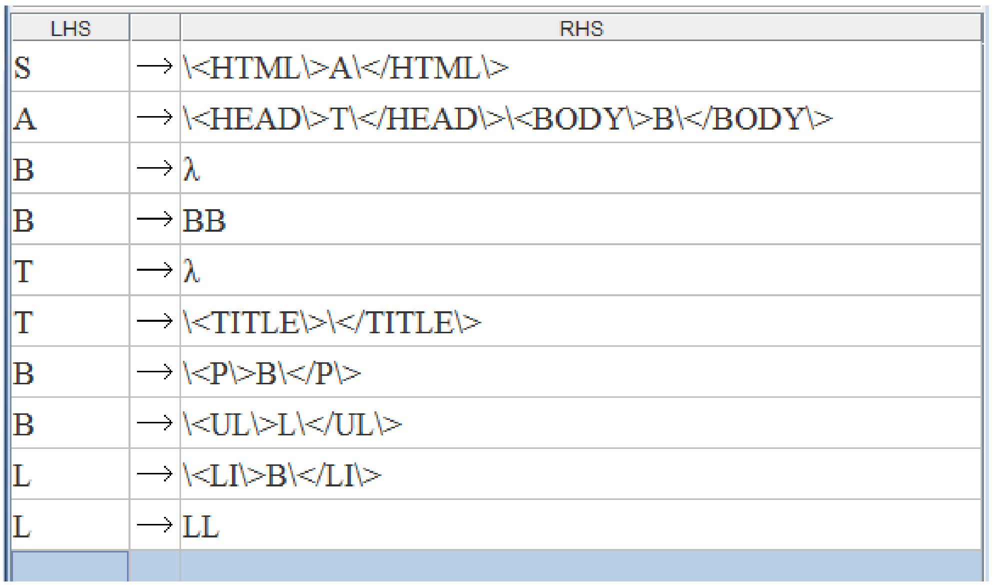


LHS		RHS
S	→	SS
S	→	λ
S	→	\<HTML\>S\</HTML\>
S	→	\<HEAD\>S\</HEAD\>
S	→	\<BODY\>S\</BODY\>
S	→	\<TITLE\>S\</TITLE\>
S	→	\<P\>S\</P\>
S	→	\<UL\>S\</UL\>
S	→	\<LI\>S\</LI\>

Please see also the VIDEO SOLUTION for this exercise.

68. Consider a slightly more accurate specification of HTML: the HTML tag must be the outermost and can occur only once. The HEAD tag must be the first tag inside the HTML tag and it must be followed by exactly one occurrence of the BODY tag. All other tags can only appear inside the BODY tag except the TITLE tag which may (or may not) appear once inside the HEAD tag. Modify your grammar to handle these requirements.

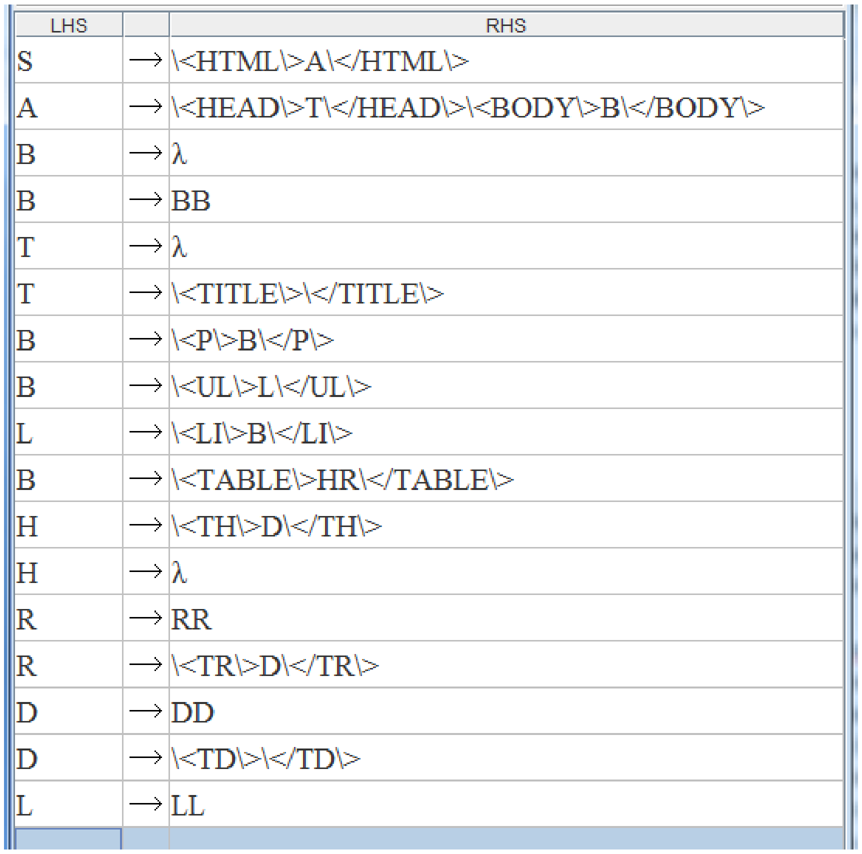
Solution:



LHS		RHS
S	→	\<HTML\>A\</HTML\>
A	→	\<HEAD\>T\</HEAD\>\<BODY\>B\</BODY\>
B	→	λ
B	→	BB
T	→	λ
T	→	\<TITLE\>\</TITLE\>
B	→	\<P\>B\</P\>
B	→	\<UL\>L\</UL\>
L	→	\<LI\>B\</LI\>
L	→	LL

69. Consider a more complete version of HTML where {TABLE, TH, TR, TD} are the additional tags for handling tables, table-headers, table-rows and table-data (or the individual cells in a table). These tags have the additional constraints that TH can only appear inside a TABLE; TR can appear any number of times but only inside a TABLE; TD can only appear inside a TH or TR. Extend the CFG to handle the table tags. (Notice that in the previous exercise, there should have been a similar constraint: LI can only appear inside a UL and a few other similar list structures in HTML.)

Solution:



LHS		RHS
S	→	\<HTML\>A\</HTML\>
A	→	\<HEAD\>T\</HEAD\>\<BODY\>B\</BODY\>
B	→	λ
B	→	BB
T	→	λ
T	→	\<TITLE\>\</TITLE\>
B	→	\<P\>B\</P\>
B	→	\<UL\>L\</UL\>
L	→	\<LI\>B\</LI\>
B	→	\<TABLE\>HR\</TABLE\>
H	→	\<TH\>D\</TH\>
H	→	λ
R	→	RR
R	→	\<TR\>D\</TR\>
D	→	DD
D	→	\<TD\>\</TD\>
L	→	LL

70. Consider a further extension the tables should have the same number of columns in each row, that is, the number of TDs in any TR cannot exceed the number of TDs in the TH of the table (while there being no limit on the number of columns in a table). Can your context-free grammar handle this? Explain.

    Solution: No, it is not possible to handle this in a context-free grammar. Having the same number of TDs across TH and a number of TRs is equivalent to a string of the form $a^nb^nc^nd^ne^nf^n$… which is not context-free. We need an elaborate context-sensitive grammar or some other mechanism outside the grammar to handle this.

71. Consider tags in XML, the eXtensible Markup Language. This is similar to HTML except that the tags themselves are user-extensible, that is, they no longer are limited to a fixed number of tags from a given set. Users can make up any tag on their own. For example, a valid XML string is:
    <a><b></b></a><a><a><b></b></a><b></b></a>
    where *a* and *b* are tag names. Assuming that tags are any alphabetical string of length less than 10 characters from the alphabet {*a, b, c, …, z*}, can you construct a context-free grammar for all valid XML strings (ignoring once again any content that appears between start and end tags)?

    Solution: This is also not possible in a context-free grammar. The total number of tags possible in valid XML strings with the above restrictions on tag names is $26 + 26^2 + 26^3 + … + 26^{10}$. We need rules of the type:

    S → \<T\>*S*\</*T*\> | *SS* | *λ*

where the two tags *T* should be the same. This is not possible to enforce by writing separate rules for expanding *T* into any of the possible tag strings. The only way would be write separate production rules for each of the above number of tags. This is impractical and we can conclude that it is not possible to construct a context-free grammar for this language.