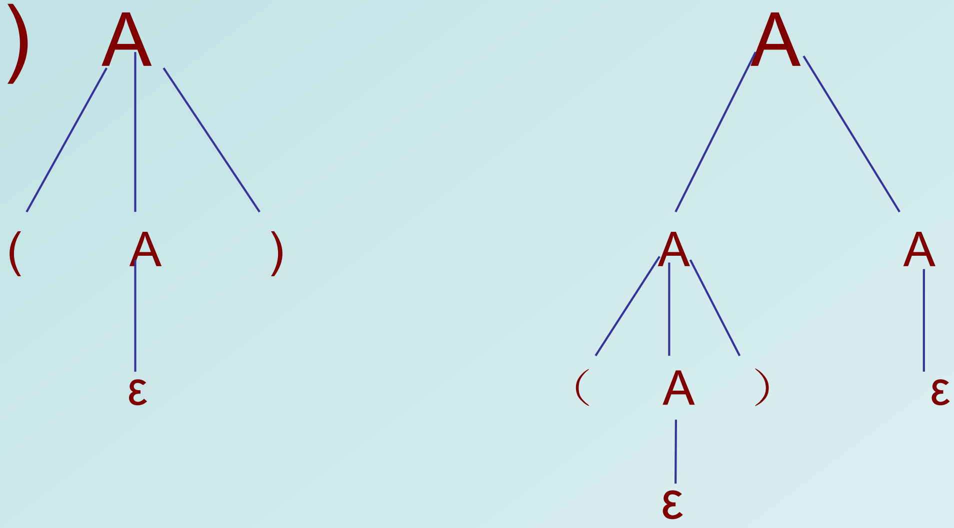
3.2 Given the grammar $A \to AA|(A)|\varepsilon$,

a. Describe the language it generates.

b. Show it is ambiguous.

(a) Generates a string of balanced parenthesis,including the empty string.

(b)



So the grammar is ambiguous.

3.3 Given the grammar:

exp → exp addop term | term

addop → + | -

term → term mulop factor | factor

mulop → *

factor → (exp) | factor

Write down leftmost derivation, parse trees, and absctract syntax trees for the following expressions:

(a) 3+4*5-6

3.3 (a) Solution:

The leftmost derivations for the expression

3+4*5-6;

exp=> exp addop term =>exp addop term addop term

=>term addop term addop term =>factor addop term addop term

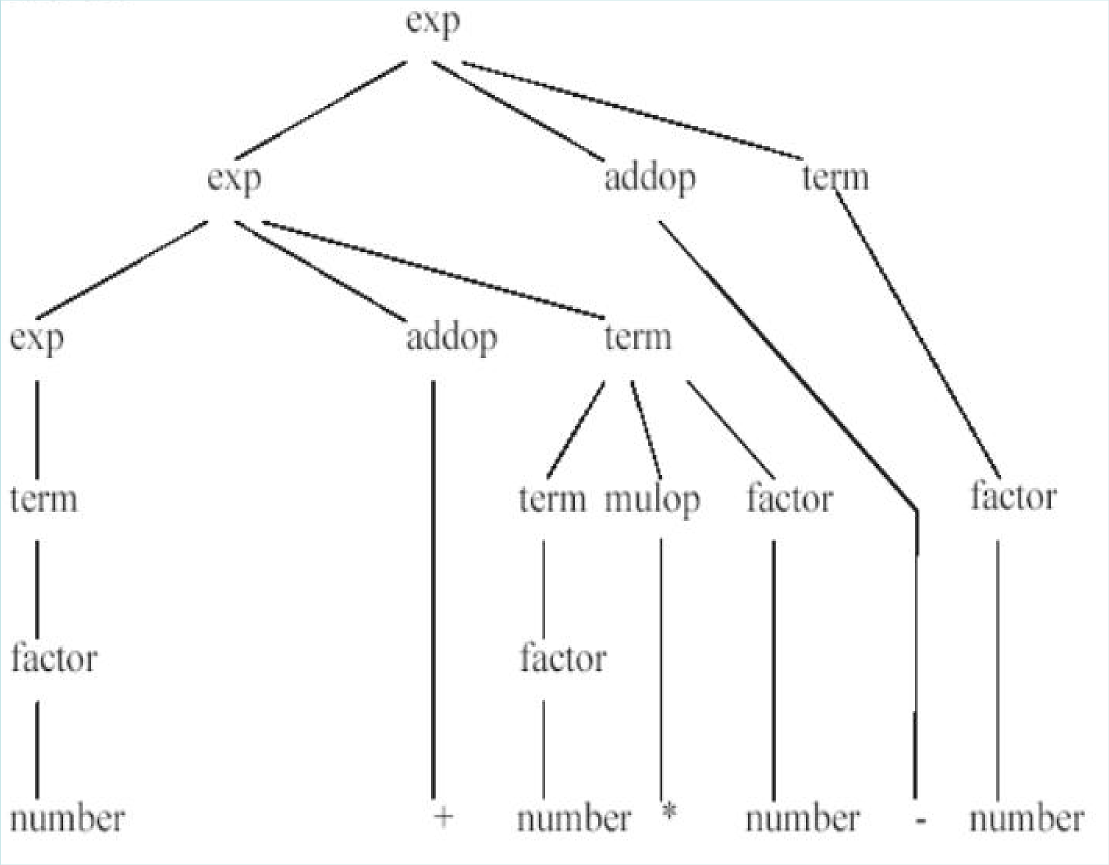
=>3 addop trem addop term => 3+term addop term

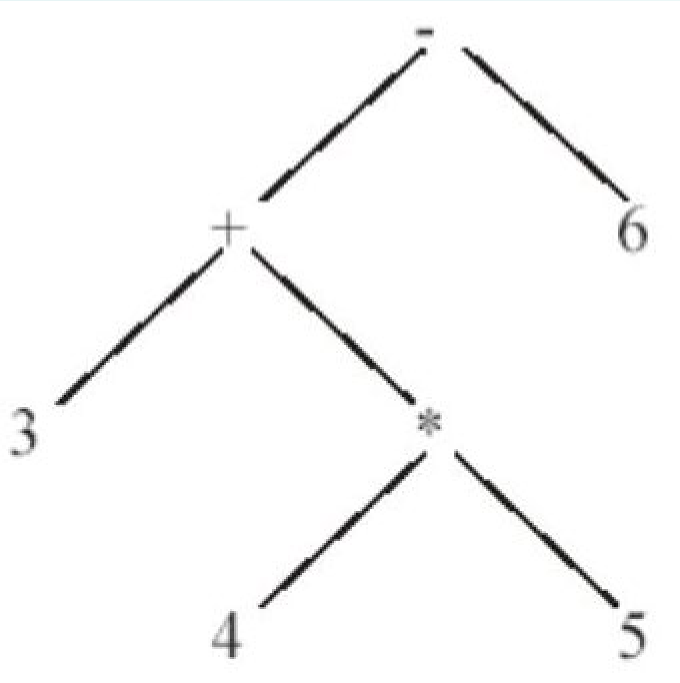
=>3 + term mulop factor addop term =>3 + factor mulop factor addop term

=> 3 + 4 mulop factor addop term =>3 + 4*factor addop term

=>3 + 4*5 addop term =>3 + 4*5 –term

=>3 + 4*5- factor = > 3 + 4*5-6;


exp
exp
addop
term
exp
addop
term
term
term
mulop
factor
factor
factor
factor
number
+
number
*
number
-
number


-
+
6
3
*
4
5

3.4 The following grammar generates all regular expressions over the alphabet of letters (we have to use quotes to surround operators, since the vertical bar is an operator as well as a metasymbol):

rexp → rexp “|” rexp
| rexp rexp
| rexp “*”
| “(” rexp “)”
| letter

a. Give a derivation for the regular expression (ab|a)* using this grammar.

b. Show that this grammar is ambiguous.

c. Rewrite this grammar to establish the correct precedences for the operations (see chapter 2).

d. What associativity does your answer in part (c) give to the binary operations? Why?

3.4 answer

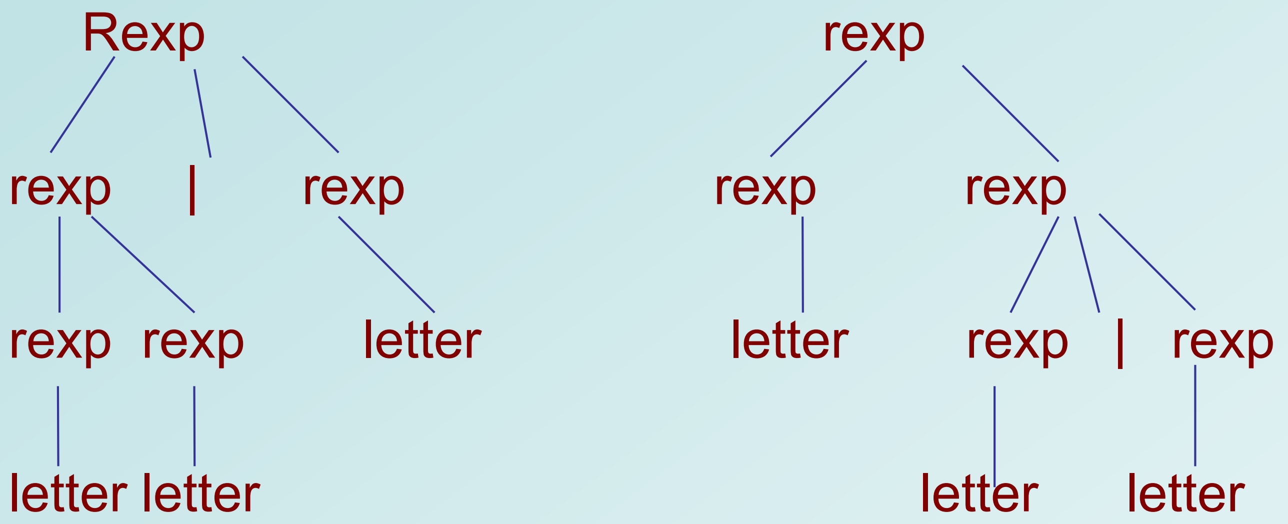
(a)

rexp=>rexp*=>(rexp)*=>(rexp|rexp)*

=>(rexp rexp|rexp)*=>(a rexp|rexp)*

=>(ab|rexp)*=>(ab|b)*

(b) Parse trees of ab|b:



(c)

rexp -> rexp “|” rexp1 | rexp1

rexp1 -> rexp1 rexp2 | rexp2

rexp2 -> rexp3 “*” | rexp3

rexp3 -> “(“rexp”)” | letter

(d)

left associativity