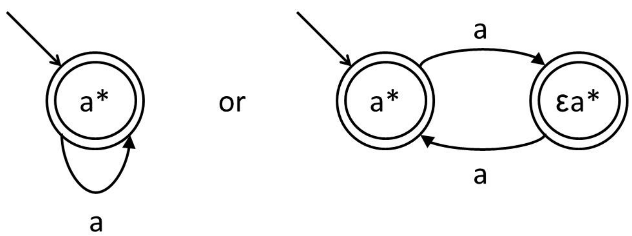
CMSC 330 Spring 2011
Practice Problems 2 (SOLUTIONS)

1. For each regular expression r below, (i) say whether or not r✓ holds and (ii) give the output of trans(r).
   a. r = a*
      (i) r✓ (ii) r –a-> a* (r –a-> εa* also acceptable)
   b. r = (a|ab)*a
      (i) not(r✓) (ii) r –a-> (a|ab)*a, r –a-> b(a|ab)*a, r –a-> ε (can have leading ε also for first two)
   c. r = a*(ba*ba*)*
      (i) r✓ (ii) r –a-> a*(ba*ba*)*, r –b-> a*ba*(ba*ba*)* (leading εs OK)

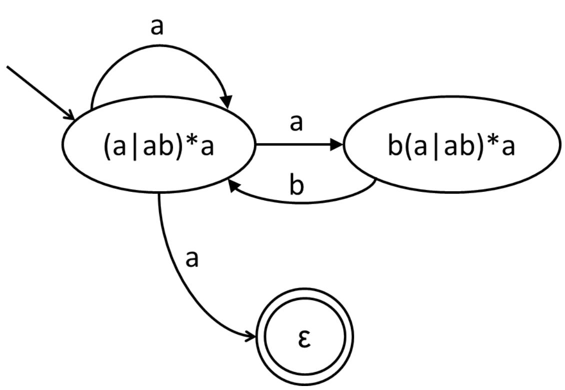
2. For each regular expression in question 1: 1*, (0|01)*0
   i. Reduce the RE to an NFA using the algorithm described in class.
   ii. Reduce the resulting NFA to a DFA using the subset algorithm.
   iii. Show whether the DFA accepts / rejects the strings "aba", "aa", "bbabb"

   a. **(i)**

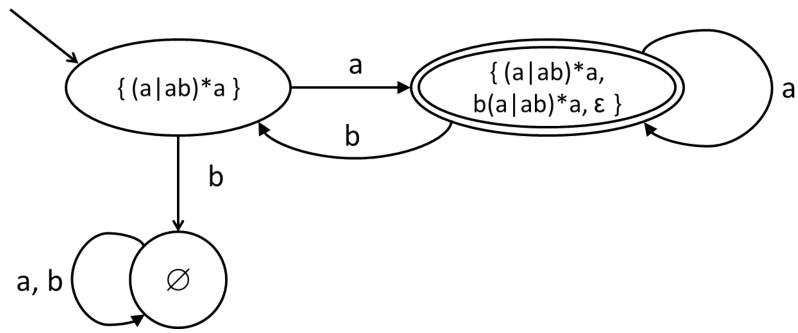


   (ii) Already deterministic!
   (iii) Accepts only aa

   b. **(i) Extra states involving ε at the beginning are also permissible.**

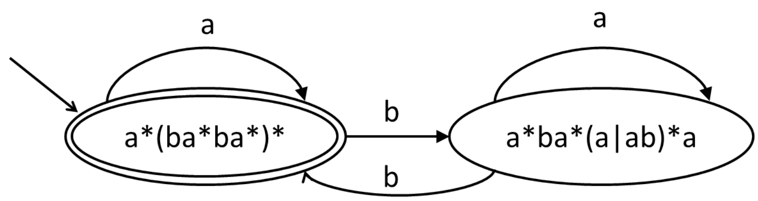


   (ii)



(iii) Accepts aa, aba, not bbabb

c. **(i) Extra states involving ε at the beginning are also permissible.**



(ii) Already deterministic!

(iii) Accepts aa, bbabb, not aba.

3. Context-Free Grammars
    a. List the 4 components of a context-free grammar.
        Terminals, non-terminals, productions, start symbol
    b. Describe the relationship between terminals, non-terminals, and productions.
        Productions are rules for replacing a single non-terminal with a string of terminals and non-terminals
    c. Define ambiguity.
        Multiple left-most (or right-most) derivations for the same string
    d. Describe the difference between scanning & parsing.
        Scanning matches input to regular expressions to produce terminals, parsing matches terminals to grammars to create parse trees

4. Describing Grammars
    a. Describe the language accepted by the following grammar:
        S → abS | a
        (ab)*a
    b. Describe the language accepted by the following grammar:
        S → aSb | ε
        a^nb^n, $n \geq 0$
    c. Describe the language accepted by the following grammar:
        S → bSb | A   A → aA | ε
        $b^na^*b^n$, $n \geq 0$
    d. Describe the language accepted by the following grammar:

$S \rightarrow AS \mid B$ $A \rightarrow aAc \mid Aa \mid \varepsilon$ $B \rightarrow bBb \mid \varepsilon$

Strings of a & c with same or fewer c's than a's and no prefix has more c's than a's, followed by an even number of b's

e. Describe the language accepted by the following grammar:

S → S and S | S or S | (S) | true | false

Boolean expressions of true & false separated by and & or, with some expressions enclosed in parentheses

f. Which of the previous grammars are left recursive?

2d, 2e

g. Which of the previous grammars are right recursive?

2a, 2c, 2d, 2e

h. Which of the previous grammars are ambiguous? Provide proof.

Examples of multiple left-most derivations for the same string

2d: **S => AS => AaS => aS => aB => a**

S => AS => S => AS => AaS => aS => aB => a

2e: **S => S and S => S and S and S => true and S and S**

=> true and true and S => true and true and true

S => S and S => true and S => true and S and S

=> true and true and S => true and true and true

5. Creating Grammars

a. Write a grammar for $a^x b^y$, where x = y

S → aSb | ε

b. Write a grammar for $a^x b^y$, where x > y

S → aL **L → aL | aLb | ε**

c. Write a grammar for $a^x b^y$, where x = 2y

S → aaSb | ε

d. Write a grammar for $a^x b^y a^z$, where z = x+y

S → aSa | L **L → bLa | ε**

e. Write a grammar for $a^x b^y a^z$, where z = x-y

S → aSa | L **L → aLb | ε**

f. Write a grammar for all strings of *a* and *b* that are palindromes.

S → aSa | bSb | L **L → a | b | ε**

g. Write a grammar for all strings of *a* and *b* that include the substring *baa*.

S → LbaaL **L → aL | bL | ε** **// L = any**

h. Write a grammar for all strings of *a* and *b* with an odd number of *a*'s and an odd number of *b*'s.

S → EaEbE | EbEaE **E → EaEaE | EbEbE | ε | SS** **// E = even #s**

i. Write a grammar for the set of regular expressions over the alphabet {a, b}.

1 possibility: S → a | b | eps | ∅ | SS | S + S | S*

(Here we are using "eps" for "ε" and "+" for "|" to avoid confusion with the symbols used in CFGs.)

Another possibility:

S → C | S+C
C → K | CK
K → K* | (S) | a | b | eps | ∅

j. Which of your grammars are ambiguous? Can you come up with an unambiguous grammar that accepts the same language?

Grammar for 3h is ambiguous. An unambiguous grammar must exist since the language can be recognized by a deterministic finite automaton, and DFA -> RE -> Regular Grammar.
Grammar 1 for 3i is ambiguous. Multiple derivations for "ab*". Grammar 2 is unambiguous, and enforces usual precedence rules for regular expressions.

6. Derivations, Parse Trees, Precedence and Associativity

For the following grammar: S → S and S | true

a. List 4 derivations for the string "true and true and true".

a) S => $\underline{S}$ and S => $\underline{S}$ and S and S => true and $\underline{S}$ and S => true and true and S => true and true and true

b) S => $\underline{S}$ and S => true and S => true and $\underline{S}$ and S => true and true and S => true and true and true

c) S => S and $\underline{S}$ => S and true => S and $\underline{S}$ and true => S and true and true => true and true and true

d) S => S and $\underline{S}$ => S and S and $\underline{S}$ => S and $\underline{S}$ and true => S and true and true => true and true and true

e) S => $\underline{S}$ and S => $\underline{S}$ and S and S => true and S and $\underline{S}$ => true and S and true => true and true and true

f) S => $\underline{S}$ and S => S and $\underline{S}$ and S => $\underline{S}$ and true and S => true and true and S => true and true and true

g) S => $\underline{S}$ and S => S and $\underline{S}$ and S => S and true and $\underline{S}$ => S and true and true => true and true and true

h) S => $\underline{S}$ and S => S and S and $\underline{S}$ => $\underline{S}$ and S and true => true and S and true => true and true and true

i) S => $\underline{S}$ and S => S and S and $\underline{S}$ => S and $\underline{S}$ and true => S and true and true => true and true and true

j) S => $\underline{S}$ and S => true and S => true and S and $\underline{S}$ => true and S and true => true and true and true

k) S => S and $\underline{S}$ => S and true => $\underline{S}$ and S and true => true and S and true => true and true and true

l) S => S and $\underline{S}$ => $\underline{S}$ and S and S => true and $\underline{S}$ and S => true and true and S => true and true and true

m) S => S and $\underline{S}$ => $\underline{S}$ and S and S => true and S and $\underline{S}$ => true and S and true => true and true and true

n) S => S and $\underline{S}$ => S and $\underline{S}$ and S =>$\underline{S}$ and true and S => true and true and S => true and true and true

o) S => S and $\underline{S}$ => S and $\underline{S}$ and S => S and true and $\underline{S}$ => S and true and true => true and true and true

p) S => S and $\underline{S}$ => S and S and $\underline{S}$ => $\underline{S}$ and S and true => true and S and true => true and true and true

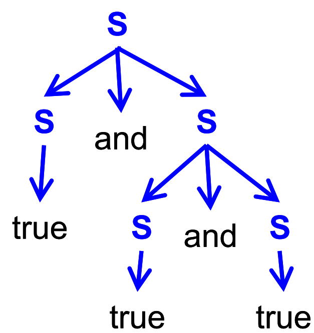
b. Label each derivation as left-most, right-most, or neither.

i and ii are left-most derivations, iii and iv are right-most derivations, remaining derivations are neither

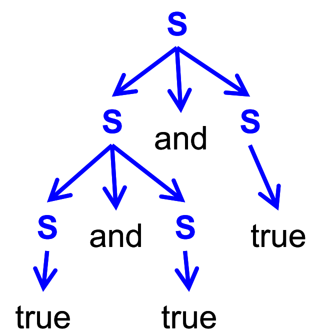
c. List the parse tree for each derivation

Tree 1 = ii, iii, x, xi, Tree 2 = rest

Tree 1



Tree 2



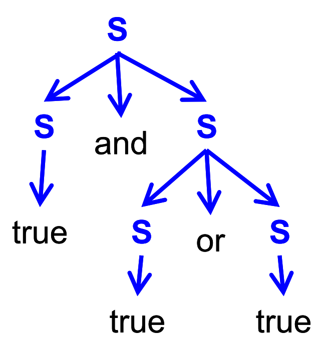
d. What is implied about the associativity of “and” for each parse tree?

Tree 1 => and is right-associative, Tree 2 => and is left-associative

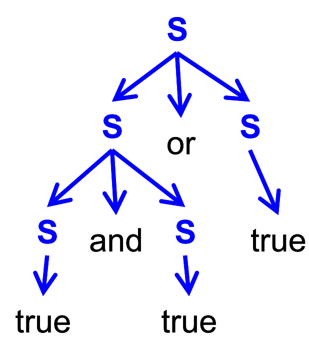
For the following grammar: S → S and S | S or S | true

e. List all parse trees for the string “true and true or true”

Tree 1



Tree 2



f. What is implied about the precedence/associativity of “and” and “or” for each parse tree?

Tree 1 => or has higher precedence than and

Tree 2 => and has higher precedence than or

g. Rewrite the grammar so that “and” has higher precedence than “or” and is right associative

S → S or S | L **// op closer to Start = lower precedence op**

L → true and L | true **// right recursive = right associative**