

CMSC 330, Fall 2009, Practice Problems 3

1. Context Free Grammars
 - a. List the 4 components of a context free grammar.
 - b. Describe the relationship between terminals, non-terminals, and productions.
 - c. Define ambiguity.
 - d. Describe the difference between scanning & parsing.

2. Describing Grammars
 - a. Describe the language accepted by the following grammar:
 $S \rightarrow abS \mid a$
 - b. Describe the language accepted by the following grammar:
 $S \rightarrow aSb \mid \epsilon$
 - c. Describe the language accepted by the following grammar:
 $S \rightarrow bSb \mid A \quad A \rightarrow aA \mid \epsilon$
 - d. Describe the language accepted by the following grammar:
 $S \rightarrow AS \mid B \quad A \rightarrow aAc \mid Aa \mid \epsilon \quad B \rightarrow bBb \mid \epsilon$
 - e. Describe the language accepted by the following grammar:
 $S \rightarrow S \text{ and } S \mid S \text{ or } S \mid (S) \mid \text{true} \mid \text{false}$
 - f. Which of the previous grammars are left recursive?
 - g. Which of the previous grammars are right recursive?
 - h. Which of the previous grammars are ambiguous? Provide proof.

3. Creating Grammars
 - a. Write a grammar for $a^x b^y$, where $x = y$
 - b. Write a grammar for $a^x b^y$, where $x > y$
 - c. Write a grammar for $a^x b^y$, where $x = 2y$
 - d. Write a grammar for $a^x b^y a^z$, where $z = x+y$
 - e. Write a grammar for $a^x b^y a^z$, where $z = x-y$
 - f. Write a grammar for all strings of a and b that are palindromes.
 - g. Write a grammar for all strings of a and b that include the substring baa .
 - 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.
 - i. Write a grammar for the "if" statement in OCaml
 - j. Write a grammar for all lists in OCaml
 - k. Which of your grammars are ambiguous? Can you come up with an unambiguous grammar that accepts the same language?

4. Derivations, Parse Trees, Precedence and Associativity

For the following grammar: $S \rightarrow S \text{ and } S \mid \text{true}$

 - a. List 4 derivations for the string "true and true and true".
 - b. Label each derivation as left-most, right-most, or neither.
 - c. List the parse tree for each derivation
 - d. What is implied about the associativity of "and" for each parse tree?

For the following grammar: $S \rightarrow S \text{ and } S \mid S \text{ or } S \mid \text{true}$

- e. List all parse trees for the string “true and true or true”
- f. What is implied about the precedence/associativity of “and” and “or” for each parse tree?
- g. Rewrite the grammar so that “and” has higher precedence than “or” and is right associative

5. Left factoring & eliminating left recursion

Rewrite the following grammars so they can be parsed by a predictive parser by eliminating left recursion and applying left factoring where necessary

- a. $S \rightarrow S + a \mid b$
- b. $S \rightarrow S + a \mid S + b \mid c$
- c. $S \rightarrow a b c \mid a c$
- d. $S \rightarrow a a \mid a b \mid a$
- e. $S \rightarrow a S c \mid a S b \mid b$

6. Parsing

For the problem, assume the term “predictive parser” refers to a top-down, recursive descent, non-backtracking predictive parser.

- a. Consider the following grammar: $S \rightarrow S \text{ and } S \mid S \text{ or } S \mid (S) \mid \text{true} \mid \text{false}$
 - i. Compute First sets for each production and nonterminal
 - ii. Explain why the grammar cannot be parsed by a predictive parser
- b. Consider the following grammar: $S \rightarrow abS \mid acS \mid c$
 - i. Compute First sets for each production and nonterminal
 - ii. Show why the grammar cannot be parsed by a predictive parser.
 - iii. Rewrite the grammar so it can be parsed by a predictive parser.
 - iv. Write a predictive parser for the rewritten grammar.
- c. Consider the following grammar: $S \rightarrow Sa \mid Sc \mid c$
 - i. Show why the grammar cannot be parsed by a predictive parser.
 - ii. Rewrite the grammar so it can be parsed by a predictive parser.
 - iii. Write a recursive descent parser for your new grammar
- d. Describe an abstract syntax tree (AST)

7. Automata

- a. Compare finite automata and pushdown automata