Homework Solutions: Context-Free Grammar

Learning Objectives:

In this homework, we will exercise the following important knowledge points on context-free grammar:

1. Understanding the relations of strings and grammars
2. Performing derivations and constructing parse trees
3. Understanding the relation of parse trees and program semantics
4. Determining and resolving ambiguity
5. Designing a grammar to describe given string patterns

Instructions:

1. Total points: 45 pt
2. Early deadline: Feb 1 (Wed) 11:59 pm, Regular deadline Feb 3 (Fri) 11:59 pm (you can continue working on the homework till TA starts to grade the homework)
3. How to submit:
   - Submit your document to Canvas under Assignments, Homework 1
   - Please provide the complete solutions in one PDF file
   - You can write your solutions in latex or word and then convert it to PDF; or you can submit a scanned document with legible handwritten solutions

Questions:

1. (8 pt) [Grammar, strings, derivations, and parse tree] Given the string

   $$\text{if (a} < \text{b) then } \{ \text{ b} + \text{a } \} \text{ else } \{ \text{ b} = 0 \ \}$$

   and the context free grammar G below:

   $S \rightarrow F \mid T\ N\ T$
   $F \rightarrow \text{if } B \text{ then } \{ S \} \mid \text{if } B \text{ then } \{ S \} \text{ else } \{ S \}$
   $B \rightarrow (T\ E\ T)$
   $T \rightarrow a|b|0$
   $E \rightarrow > \mid <$
   $N \rightarrow + \mid * \mid =$

   (a) (2 pt) What are the terminals and non-terminals of the grammar?
   (b) (2 pt) Write a derivation for the string.

(c) (2 pt) Construct a parse tree for the string.

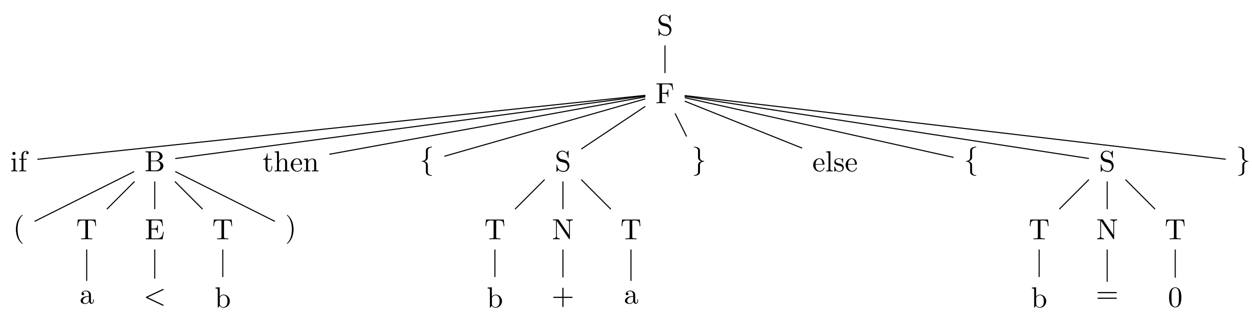
(d) (2 pt) Write 2 strings that do not belong to the language $L(G)$ but use only the terminals from $L(G)$.

Solution:

(a) **Terminals**: $a, b, 0, >, <, +, *, =, (,), \{, \}$, if, then, else
Non-Terminals: S, F, B, T, E, N

(b) $S \Rightarrow F$
$\Rightarrow$ if B then { S } else { S }
$\Rightarrow$ if (TET) then { S } else { S }
$\Rightarrow$ if (aET) then { S } else { S }
$\Rightarrow$ if $(a < T)$ then { S } else { S }
$\Rightarrow$ if $(a < b)$ then { S } else { S }
$\Rightarrow$ if $(a < b)$ then { TNT } else { S }
$\Rightarrow$ if $(a < b)$ then { bNT } else { S }
$\Rightarrow$ if $(a < b)$ then { $b + T$ } else { S }
$\Rightarrow$ if $(a < b)$ then { $b + a$ } else { S }
$\Rightarrow$ if $(a < b)$ then { $b + a$ } else { TNT }
$\Rightarrow$ if $(a < b)$ then { $b + a$ } else { bNT }
$\Rightarrow$ if $(a < b)$ then { $b + a$ } else { $b = T$ }
$\Rightarrow$ if $(a < b)$ then { $b + a$ } else { $b = 0$ }

(c)



(d)

i. if (a == 0) then { a = b }
ii. a = a + b

2. (16 pt) [Ambiguity] Consider the following grammar:

$$S \rightarrow aS|SbS|a|c$$

(a) (2 pt) Give an example string that has two different parse trees

(b) (4 pt) Give a leftmost derivation and a rightmost derivation for the string from 2(a).

(c) (4 pt) Give two different parse trees for the string from 2(a).

(d) (4 pt) Modify the grammar to remove ambiguity.

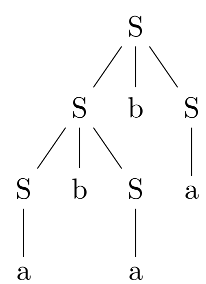
(e) (2 pt) Explain how your new grammar modifies the parse tree you drew in the first step to remove ambiguity.

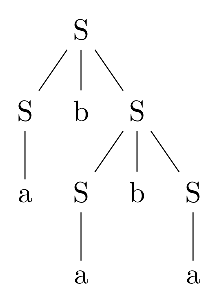
Solution:

(a) ababa

(b) **Leftmost derivation**:
$S \Rightarrow SbS$
$\Rightarrow SbSbS$
$\Rightarrow abSbS$
$\Rightarrow ababS$
$\Rightarrow ababa$
Rightmost derivation:
$S \Rightarrow SbS$
$\Rightarrow SbSbS$
$\Rightarrow SbSba$
$\Rightarrow Sbaba$
$\Rightarrow ababa$

(c)





(d) We modify the grammar to remove ambiguity as follows:

$S \to SbT \mid T$
$T \to aT \mid F$
$F \to a \mid c$

(e) We found that the current parse trees of ababa have the operator associativity problem. In our previous grammar, S can be expanded on both sides, which causes ambiguity. However, with the modification, S can only be expanded on the left side. We also removed the ambiguity by incorporating the precedence.

3. (10 pt) [Grammar analysis and semantics] In the following, we redefine the grammar for bitwise operations:

   - Terminals: 0, 1, `not`, `and`, `or`, `xor`
   - Non-terminals: E, T, F, V
   - Start symbol: E
   - Production rules:
     $E \rightarrow E$ `and` $T \mid E$ `or` $T \mid$ T
     $T \rightarrow F$ `xor` $T \mid$ `not` $T \mid$ F
     $F \rightarrow 0 \mid 1$

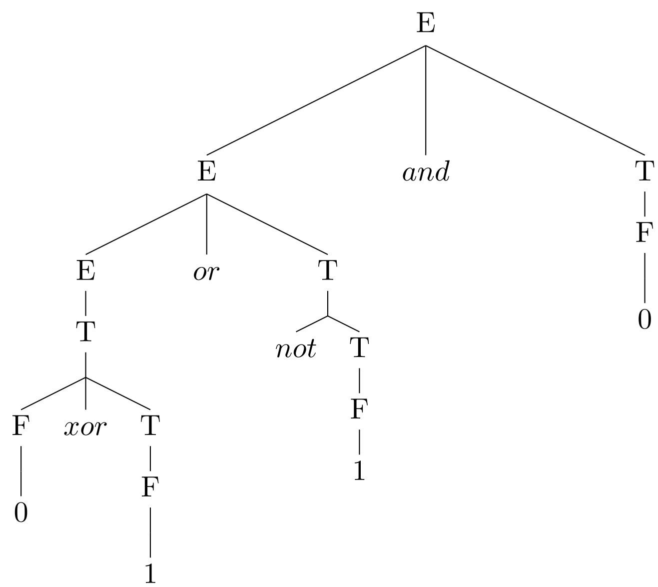
   (a) (3 pt) What is the associativity of the operators "and", "or" and "xor"? Explain why.

   (b) (3 pt) What is the precedence of "not","or" and "xor"? Explain why.

   (c) (4 pt) Consider the parse tree given below and answer the following questions.

   i. (2 pt) What is the value of the string generated by the parse tree? (For the single operations of "and", "or", "not", and "xor", use their usual semantics)

   ii. (2 pt) Explain how the value of the string is generated from the parse tree step by step.



Solution:

(a) The operators "and" is left-associative because the start symbol E appears on the left side of the operator. Similarly, the operator "or" is left-associative because the start symbol E appears on the left side of the operator. Finally, "xor" is right-associative because starting from the start symbol E, "xor" can only be reached through T, that appears on the right side of the operator.

(b) The operator precedence is : "xor" = "not" > "or", because "xor" and "not" are further down the grammar rules than "or".

(c) i. 0

ii. $(and\ (or\ (0\ xor\ 1),\ (not\ 1)),\ 0) \rightarrow 0$

(1) 0 "xor" 1

(2) "not" 1

(3) 0 "xor" 1 "or" "not" 1

(4) 0 "xor" 1 "or" "not" 1 "and" 0

4. (11 pt) [Designing Grammar] Design CFGs for the given languages:

(a) (3 pt) Write a grammar that describes the strings $(8)^*(7|4)^+$.

(b) (3 pt) Write a grammar that describes all the palindrome strings on the alphabet (terminals) of a, b and c. See the examples of valid strings; aca, bab, abba, acccca ...

(c) (5 pt) In most programming languages, array elements can be represented using a name followed by any number of indices. The name might be a letter selected between 'a' to 'z', and an index can be represented using '[' integer ']'. For example, a[-9], b[10][1], c[0][1][2] are valid array elements. Write a grammar to describe such array elements.

Solution:

(a) $S \rightarrow\ XY$
$X \rightarrow\ 8X|\epsilon$
$Y \rightarrow\ 7Y|4Y|7|4$

(b) $S \rightarrow aSa|bSb|cSc|a|b|c|\epsilon$

(c) $S \rightarrow aT|bT|\dots|zT$
$T \rightarrow [Sign\ Digits]\ T \mid [Sign\ Digits] \mid [Zero]\ T \mid [Zero]$
$Sign \rightarrow -|\epsilon$
$Digits \rightarrow NonZero \mid NonZero\ Digits\ |NonZero\ Zero$
$NonZero \rightarrow 1|2|\dots|9$
$Zero \rightarrow 0$