CMPSC 461: Programming Language Concepts, Fall 2024
Assignment 2 Practice Notes Packet

Prof. Suman Saha

September 7, 2025

Problem 1: Context Free Grammar - Creation 1 $[5 + 1.5 + 1.5 = 8$ pts$]$

As a CMPSC 461 student, you wish to write a rudimentary CFG for parsing roman numerals from 1 to 99 (i,ii,iii,iv,v,. . . ,ix,x,. . . ,xl,. . . ,lxxx,. . . ,xc,. . . ,xcix). If you are unfamiliar with roman numerals, please have a look at http://literacy.kent.edu/Minigrants/Cinci/romanchart.htm.).

- Your grammar should comprise of terminals {c, l, x, v, i}.
- c = 100, l = 50, x = 10, v = 5, i = 1.
- Notice that we use lowercase characters here to represent the numerals, to distinguish them from the non-terminals.

1. Define a **context-free grammar** to model this language.
2. What are **terminals** and **nonterminals** in the context of CFGs? Based on your answer for part 1 answer label/describe terminals and non-terminals in your answer.
3. Do context-free grammars represent all regular languages? Do regular languages translate all grammar? Are there any languages CFGs may not be able to represent? (Explain each answer in 10 words)

Solution

1.
```
S -> DB
A -> i | ii | iii | ϵ
B -> A | iv | vA | ix
C -> x | xx | xxx | ϵ
D -> C | xl | lC | xc
```

2. Terminals are actual symbols or characters that appear in the language described by the grammar like in part 1 $\{c, l, x, v, i\}$ are terminals.
   Non-terminals serve as placeholders for sequences of terminals and other non-terminals. Therefore, in the above question are all non-terminals $\{S, T, X, U, Y, Z\}$ because they are placeholders for the terminals.

3. Yes, because context-free grammars are more powerful and can express all patterns that regular languages can;
   No, because regular languages are simpler and can't handle complex structures that context-free grammars can;
   Yes, context-free grammars can't represent languages that need more complex rules, like context-sensitive languages.

Problem 2: Context Free Grammar - Creation 2 [8 pts]

Create context free grammars for each of the following languages.

1. The set of strings which contains palindromic binary numbers. The strings can have leading zeros (for example, 101, 010 or 00).
2. The set of strings which have a number of "a"s followed by twice the number of "b"s (for example, "abb", "aabbbb" and so on).

Solution

1. A palindrome reads the same forwards and backwards.
   Starting at S, the grammar is as follows :

   $S \rightarrow 0S0 \mid 1S1 \mid (0|1)$

   Rule 1 : This rule says that if you start with an S, you can generate a palindrome by placing a 0 at the beginning and a 0 at the end, and recursively generating a palindrome in between.

   Rule 2: Similarly, this rule generates palindromes by placing a 1 at both ends of the string and recursively generating a palindrome in the middle.

   Rule 3: These are the base cases. A single 0 or 1 is a palindrome, and the empty string (denoted by ϵ) is also considered a palindrome.

2. Starting at S, the grammar is as follows :

   $S \rightarrow aSbb \mid \epsilon$

Problem 3: Context Free Grammar - Creation 3 [9 pts]

Give context free grammars that generate the following languages.

1. { w $\in \{0,1\}^*$ | w contains at least three **1s**.}
2. { w $\in \{a,b\}^*$ | the length of w is odd and middle symbol is b.}
3. { $a^i b^j c^k$ | $i,j,k \geq 0$ and i = j or i = k }

Solution

1. G = (V, Σ, R, S) with set of variables V = {S, X}, where S is the start variable; set of terminals Σ = {0, 1}; and rules

   $S \rightarrow X1X1X1X$
   $X \rightarrow 0X \mid 1X \mid \epsilon$

2. G = (V, Σ, R, S) with set of variables V = {S}, where S is the start variable; set of terminals Σ = {a, b}; and rules

   $S \rightarrow bSb|bSa|aSb|aSa|b$

3. G = (V, Σ, R, S) with set of variables V = {S,W,X,Y,Z}, where S is the start variable; set of terminals Σ = {a, b,c}; and rules

   $S \rightarrow XY \mid W$
   $X \rightarrow aXb \mid \epsilon$
   $Y \rightarrow cY \mid \epsilon$
   $W \rightarrow aWc \mid Z$
   $Z \rightarrow bZ \mid \epsilon$

Problem 4: Context Free Grammar - Creation 4 [12 pts]

Give context free grammars that generate the following languages

1. $\{a^i b^j c^k \mid i, j, k \geq 0$ and i + j = k$\}$
2. $\{ab^n acab^n a \mid n \geq 0\}$

Solution

1. Assuming a start variable S and a set of terminals {a,b,c},
   S $\rightarrow aSc|X$
   $x \rightarrow bXc|\epsilon$
2. Assuming a start variable S and a set of terminals {a,b,c},
   S $\rightarrow aTa$
   $T \rightarrow bTb|$ aca

Problem 5: Context Free Grammar - Knowledge Check [10 pts]

For each of the following statements, respond with either "True" or "False" to indicate whether the statement is correct. Provide reasoning for each answer provided.

1. A regular grammar can generate context-free languages.
2. An NFA (Nondeterministic Finite Automaton) can recognize languages that a DFA cannot.
3. When two regular languages are concatenated, the resulting language is still a regular language.
4. An automaton with multiple initial states can be considered a DFA.
5. A NFA (Nondeterministic Finite Automaton) can have epsilon (ε) transitions, allowing it to move to the next state without consuming any input symbol.

Solution

1. **False:** A regular grammar cannot generate context-free languages. Regular grammars generate regular languages, which are a proper subset of context-free languages.
2. **False:** An NFA and a DFA can recognize the same class of languages(regular languages).
3. **True:** When you concatenate (join end-to-end) two regular languages, the resulting language is also regular, as regular languages are closed under concatenation operation.
4. **False:** An automaton with multiple initial states is not considered a Deterministic Finite Automaton (DFA). A DFA should have exactly one initial state.
5. **True:** An NFA can have epsilon (ε) transitions, allowing it to move to the next state without consuming any input symbol. Epsilon transitions are a feature of NFAs, which allow for more flexibility in recognizing certain languages.+

Problem 6: BNF-EBNF Conversion [16 pts]

1. Convert the following BNF to an EBNF

```
<goal>      ::= <a>
<goal>      ::= x <b> <a>
<a>         ::= y
<a>         ::= x <a>
<b>         ::= <a>
<b>         ::= <a> <b>
<b>         ::= y <b>
```

2. Convert the following EBNF to a BNF

```
<N> ::= A [B]
<Q> ::= [-] <num>
<P> ::= A { A }
<X> ::= { A }
<blk> ::= begin <cmd> { ; <cmd> } end
<nws> ::= (+|-) <num>
<SN> ::= [(+|-)] <num>
```

Solution

1. One possible solution is :
   $< goal >::= [x < b >] < a >$
   $< a >::= y|x < a >$
   $< b >::=< a > | < a >< b > |y < b >$

2. One possible solution is :
   $< N >::= A|AB$
   $< Q >::= - < num > | < num >$
   $< P >::=< P > A|A$
   $< X >::=< X > A|\epsilon$
   $< blk >::= begin < sts > end$
   $< sts >::=< cmd > | < cmd >; < sts >$
   $< nws >::= + < num > |- < num >$
   $< SN >::= + < num > |- < num > | < num >$

Problem 7: Context-Free-Grammar Derivation [16 pts]

Answer the following questions with this following grammar starting with S.

$$
\begin{aligned}
S &\rightarrow S - S \mid P \mid T \\
P &\rightarrow P + P \mid V \mid T \\
V &\rightarrow V * V \mid S \mid T \\
T &\rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9
\end{aligned}
$$

1. Give the left-most derivation of the following: $4 - 6 + 1 * 5$
2. Give the right-most derivation of the following: $4 - 6 + 1 * 5$
3. Is there more than one right-most derivation? Explain.

Solution

1.
```
S -> S - S
  -> T - S (Expand S -> T)
  -> 4 - S (Expand T -> 4)
  -> 4 - P (Expand S -> P)
  -> 4 - P + P (Expand P -> P + P)
  -> 4 - T + P (Expand P -> T)
  -> 4 - 6 + P (Expand T -> 6)
  -> 4 - 6 + V (Expand P -> V)
  -> 4 - 6 + V * V (Expand V -> V * V)
  -> 4 - 6 + T * V (Expand V -> T)
  -> 4 - 6 + 1 * V (Expand T -> 1)
  -> 4 - 6 + 1 * T (Expand V -> T)
  -> 4 - 6 + 1 * 5 (Expand T -> 5)
```

2. Similar to above, start derivating from right-side for right-most derivation.

```
S -> P
-> V
-> V * V
-> V * T
-> V * 5
-> S * 5
-> P * 5
-> P + P * 5
-> P + T * 5
-> P + 1 * 5
-> V + 1 * 5
-> S + 1 * 5
-> S - S + 1 * 5
-> S - T + 1 * 5
-> S - 6 + 1 * 5
-> T - 6 + 1 * 5
-> 4 - 6 + 1 * 5
```

3. Yes, there are more than one right-most derivation. If the first step is from S to S - S instead of P, it can still be right-most as long as it works its way to a terminal on the right side.

```
S -> S - S
  -> S - P + P
  -> S - P + V
  -> S - P + V * V
  -> S - P + V * T
  -> S - P + V * 5
  -> S - P + T * 5
  -> S - P + 1 * 5
  -> S - T + 1 * 5
  -> S - 6 + 1 * 5
  -> T - 6 + 1 * 5
  -> 4 - 6 + 1 * 5
```

Problem 8: Context-Free-Grammar Parse Tree [16 pts]

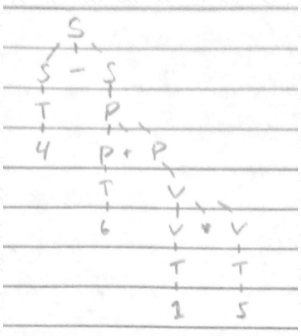
Answer the following questions with this following grammar starting with S.

$$S \to S - S \mid P \mid T$$
$$P \to P + P \mid V \mid T$$
$$V \to V * V \mid S \mid T$$
$$T \to 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$$

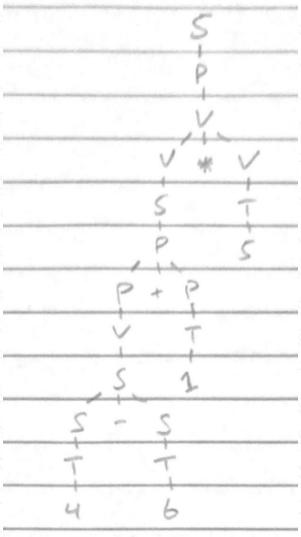
1. Draw the parse tree of the left-most derivation for $4 - 6 + 1 * 5$
2. Draw the parse tree of the right-most derivation for $4 - 6 + 1 * 5$

Solution Refer to state transitions from Question-7 solutions to draw these parse trees.

1.



2.



Problem 9: Context-Free-Grammar Ambiguity [16 pts]

The following problems help with some common ambiguity problems. Take the following grammars:

$$Grammar1:$$
$$S \to S - S \mid T$$
$$T \to 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$$

$$Grammar2:$$
$$S \to S - T \mid T - S \mid T$$
$$T \to 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$$

$$Grammar3:$$
$$S \to S - P \mid S + P \mid P$$
$$P \to V/P \mid V * P \mid V$$
$$V \to (S) \mid T$$
$$T \to 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$$

1. Is Grammar 1 Ambiguous? If so, how do you fix it? If not, why?
2. Is Grammar 2 Ambiguous? If so, how do you fix it? If not, why?
3. Is Grammar 3 Ambiguous? If so, how do you fix it? If not, why?

Solution

1. Grammar 1 is ambiguous because you can construct $4 - 6 - 1$ in two different ways. Both start from S to S - S, but then can derive another S - S from either side. To fix this, make it so for any operation, only have one symbol that goes back to itself. Making it S - T or T - S would solve the ambiguity.
2. Grammar 2 is ambiguous because you can construct $4 - 6 - 1$ in two different ways. In the beginning, S can be either T - S or S - T, then it can go down to T on either side. By taking out either S - T or T - S would solve the ambiguity.
3. Grammar 3 is unambiguous. The reason would be because there is no string that is formed by this grammar can be done in two different derivation. Many times, if you cannot find a counter example, it is most likely unambiguous, but this takes extensive practice.

Problem 10: Context-Free-Grammar Parse Tree [16 pts]

Consider the following grammar:

$$S \to TT$$
$$T \to TTT \mid a$$
$$T \to bT \mid Tb$$

1. Give at least four distinct strings generated by derivations of four or fewer steps.
2. Give at least four distinct parse trees to generate the string *babbab*.
3. Is this grammar ambiguous? Explain.

Solution

1. S $\to TT \to Ta \to aa$
   $S \to TT \to TTb \to Tab \to aab$
   $S \to TT \to Ta \to bTa \to baa$
   $S \to TT \to TbT \to Tba \to aba$
2. S $\to TT \to bTT \to bTbT \to babT \to babTb \to babbTb \to babbab$
   $S \to TT \to TbT \to TbTb \to Tbab \to Tbbab \to bTbbab \to babbab$
   $S \to TT \to TbT \to bTbT \to babT \to babTb \to babbTb \to babbab$
   $S \to TT \to TTb \to TbTb \to Tbab \to Tbbab \to bTbab \to babbab$
3. This grammar is ambiguous because there is more than one way to generate the string *babbab* (reference parse trees from previous question).

Problem 11: Context-Free-Grammar Parse Tree [16 pts]

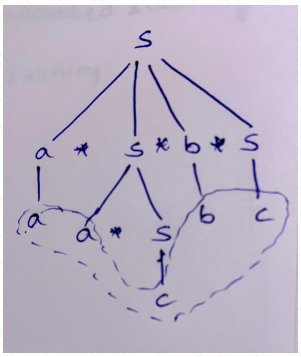
Consider the following grammar:

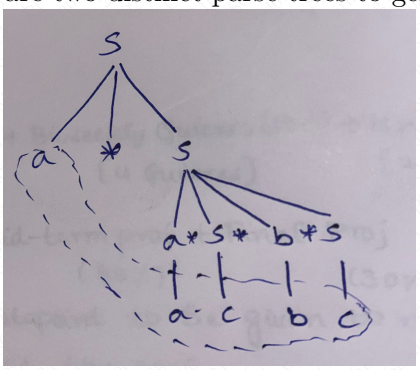
$$S \to aS \mid aSbS \mid c$$

1. Is this grammar ambiguous? Show by drawing parse trees for the string *aacbc*
2. Design an unambiguous CFG for the language above.

Solution

1. Yes, this grammar is ambiguous because there are two distinct parse trees to generate *aacbc*.





2. To eliminate the ambiguity in the grammar, we introduce two new non-terminals, X and Y.

   X: This handles patterns where every 'a' is paired with a corresponding 'b', creating balanced sequences like a...b. It also covers the simple case where X produces just 'c'.

   Y: This deals with sequences that start with 'a' and are followed by either another sequence (created recursively) or a balanced a...b sequence derived from X.

   S $\to X \mid Y$
   $X \to aXbX \mid c$
   $Y \to aS \mid aXbY$

Problem 12: Context-Free-Grammar Parse Tree [10 pts]

Consider the following:

1. What is ambiguity in CFG and what makes a grammar ambiguous?
2. Give a real-world example portraying how ambiguous grammar can be bad.

Solution

1. Ambiguity happens due to grammar rules that allow multiple "meanings" or derivations for the same string of terminals.
2. Answers may vary. Imagine you are designing a new programming language that allows prefix and postfix arithmetic. However, you did not account for the order of evaluation for prefix and postfix in your programming language's grammar which makes the language ambiguous.

   x = 2
   y = ++x * x−−

   For the above snippet, the order for evaluation could be prefix first or postfix first due to ambiguity and can both lead to two different values for y.