CS 331 Compilers **Fall 2017**

Written Assignment 2

Prof. Szajda *Due Thursday, October 19, 5:00 pm*

This assignment asks you to prepare written answers to questions on context-free grammars, parse trees, and parsing. Each of the questions has a short answer. You may discuss this assignment with other students and work on the problems together. However, as usual, your write-up should be your own individual work. The assignment need not be completed in electronic form, but if you decide to hand write your submission, please take the time to write legibly – if I can't read your solution, I'll conclude that it is wrong.

1. Give a context-free grammar (CFG) for each of the following languages over the alphabet $\Sigma = \{a, b\}$:

(a) All strings in the language $L : \{a^n b^m a^{2n} | n, m \geq 0\}$.

(b) All nonempty strings that start and end with the same symbol.

(c) All strings with more as than bs.

(d) All palindromes (a palindrome is a string that reads the same forwards and backwards).

2. A history major taking CS 331 decides to write a rudimentary CFG to parse the roman numerals 1-99 $(i, ii, iii, iv, v, \ldots, ix, x, \ldots, xl, \ldots, lxxx, \ldots, xc, \ldots, xcix)$. If you are unfamiliar with roman numerals, please have a look at `http://en.wikipedia.org/wiki/Roman_numerals` and `http://literacy.kent.edu/Minigrants/Cinci/romanchart.htm`. Consider the grammar below, with 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 non-terminals.

```
S → xTU | lX | X
T → c | l
X → xX | U
U → iY | vI | I
Y → x | v
I → iI | ϵ
```

(a) Draw a parse tree for 47: "xlvii".

(b) Is this grammar ambiguous?

(c) Write semantic actions for each of the 14 rules in the grammar (remember `X → A | B` is short for `X → A` and `X → B`) to calculate the decimal value of the input string. You can associate a synthesized attribute `val` to each of the non-terminals to store their value. The final value should be returned in `S.val`. Hint: have a look at the calculator examples presented in class.

3.

(a) Left factor the following grammar:

$$E \rightarrow int \mid int + E \mid int - E \mid E - (E)$$

(b) Eliminate left-recursion from the following grammar:

$$\begin{aligned} A &\rightarrow A + B \mid B \\ B &\rightarrow int \mid (A) \end{aligned}$$

4. Consider the following LL(1) grammar, which has the set of terminals $T = \{\texttt{a},\texttt{b},\texttt{ep},\texttt{+},\texttt{*},\ \texttt{(},\texttt{)}\}$. This grammar generates regular expressions over $\{a, b\}$, with + meaning the RegExp OR operator, and `ep` meaning the ϵ symbol. (Yes, this is a context free grammar for generating regular expressions!)

$$\begin{aligned} E &\rightarrow TE' \\ E' &\rightarrow +E \mid \epsilon \\ T &\rightarrow FT' \\ T' &\rightarrow T \mid \epsilon \\ F &\rightarrow PF' \\ F' &\rightarrow *F' \mid \epsilon \\ P &\rightarrow (E) \mid \texttt{a} \mid \texttt{b} \mid \texttt{ep} \end{aligned}$$

(a) Give the first and follow sets for each non-terminal.

(b) Construct an LL(1) parsing table for the left-factored grammar.

(c) Show the operation of an LL(1) parser on the input string `ab*`.

5. Consider the following CFG, which has the set of terminals $T = \{\texttt{stmt}, \{, \}, ;\}$. This grammar describes the organization of statements in blocks for a fictitious programming language. Blocks can have zero or more statements as well as other nested blocks, separated by semicolons, where the last semicolon is optional. (P is the start symbol here.)

$$\begin{aligned} P &\rightarrow S \\ S &\rightarrow \texttt{stmt} \mid \{B \\ B &\rightarrow \} \mid S\} \mid S\texttt{;}B \end{aligned}$$

(a) Construct a DFA for viable prefixes of this grammar using LR(0) items.

(b) Identify any shift-reduce and reduce-reduce conflicts in this grammar under the SLR(1) rules.

(c) Assuming that an SLR(1) parser resolves shift-reduce conflicts by choosing to shift, show the operation of such a parser on the input string $\{\texttt{stmt;}\}$.