COMS W4115 Assignment 2

Programming Languages and Translators

Xijiao Li (xl2950)

November 6, 2020

Problem 1

a.

$$\begin{aligned}
\text{Given}\quad & S \to A \\
& A \to BBBwx \\
& B \to CCCyz \\
& C \to a|b \\
\text{then}\quad & S \to A \to BBBwx \to CCCyz\{3\}wx \to \big((a|b)\{3\}yz\big)\{3\}wx
\end{aligned}$$

So the total number is $2^9 = 512$.

b.

$$\begin{aligned}
\text{Given}\quad & S \to D \\
& D \to EEE \\
& E \to FFF \\
& F \to c|d|\epsilon \\
\text{then}\quad & S \to D \to EEE \to (FFF)\{3\} \to \big((c|d|\epsilon)\{3\}\big)\{3\} \to (c|d|\epsilon)\{9\}
\end{aligned}$$

So the total number is $\sum_{i=0}^{10} 2^i = 1023$.

Problem 2

a.

The context-free grammar G is given by the following set of production rules:

$$S \rightarrow \ aSa \mid cSc \mid eSe \mid rSr \mid a \mid c \mid e \mid r \mid \epsilon$$

b.

Proof by contradiction.
Suppose that G is regular. Let p be the pumping value from the pumping lemma. Consider the input string $s = a^p cca^p$, which is a palindrome and $|s| \geq p$. By the pumping lemma, there must exist strings x, y, and z satisfying the four constraints of the pumping lemma.

So, pick any x, y, z such that $s = xyz$, $|xy| \leq p$, and $|y| \geq 1$. Because $|xy| \leq n$, xy is entirely wrapped in the a^p at the start of s. So x and y consist entirely of a's, i.e. $x = a^i$, $y = a^j$. Then $z = a^k cca^p$, where $i + j + k = p$.

Now, by the pumping lemma, $xz = a^i a^k 11 a^p = a^{i+k} cca^p$ must be in the language. Since $|y| \geq 1, j \geq 1$, so $i + k < p$, meaning that xz is not a palindrome (because the numbers of zeros on the two ends don't match). Therefore there is a contradiction that the set of palindromes doesn't satisfy the pumping lemma, and thus, G is not regular.

Problem 3

Given that here we are referring to General Recursive Descent (i.e., backtracking is supported), as long as the grammar is not left recursive, it is possible to apply recursive descent parsing on all possible inputs. A grammar is left recursive if contains some left recursive nonterminal. A nonterminal is left-recursive if the leftmost symbol in one of its productions is itself (in the case of direct left recursion) or can be made itself by some sequence of substitutions (in the case of indirect left recursion).

a.

Given	$S \to E$	not left recursion
	$E \to XR \mid \epsilon$	not left recursion
	$X \to \epsilon \mid X$	left recursion
	$R \to int * E \mid \epsilon$	not left recursion

No, since there is a left recursion: $X \to X$.

Examples: $5 * 5*$ cannot be parsed using Recursive descent.

New grammar G':

$$S \to R$$
$$R \to int * R \mid \epsilon$$

b.

Given	$S \to E$	not left recursion
	$E \to AB \mid \epsilon$	not left recursion
	$A \to (A) \mid \text{int} \mid \ + E$	not left recursion
	$B \to \text{int}\ B \mid * B \mid * \mid \text{int}$	not left recursion

Yes, since there is no left recursion: we will always consume the first character of input and then continue the parsing.

Examples:

- $(5) * 5$
- $5 * 5$

c.

Given	$S \to E$	not left recursion
	$E \to WXY$	part of left recursion
	$W \to a \mid Yb$	part of left recursion
	$X \to (W) \mid Z * Z$	not left recursion
	$Y \to \epsilon \mid E + X$	part of left recursion
	$Z \to \text{int } Z \mid \text{int} \mid + \mid \epsilon$	not left recursion

No, since the grammar is left recursive. Given productions

$$\begin{aligned} E &\to WXY \\ W &\to a \mid Yb \\ Y &\to \epsilon \mid E + X \end{aligned}$$

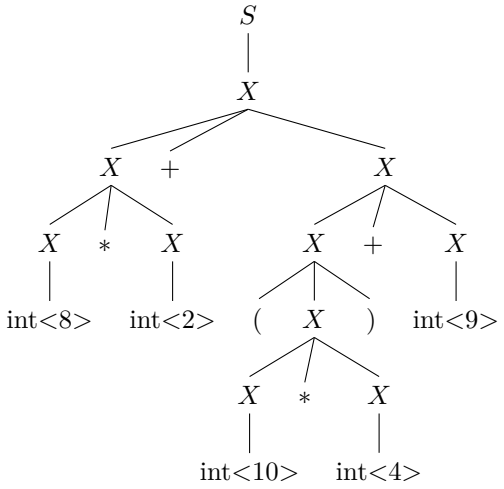
we have a left recursive nonterminal $E \to^+ E + XbXY$.
Examples: $b(a)$ cannot be parsed using recursive descent.
New grammar G':

$$\begin{aligned} S &\to E \\ E &\to WXY \\ W &\to bY \\ X &\to (W) \mid Z * Z \\ Y &\to \epsilon \mid E + X \mid aY \\ Z &\to \text{int } Z \mid \text{int} \mid + \mid \epsilon \end{aligned}$$

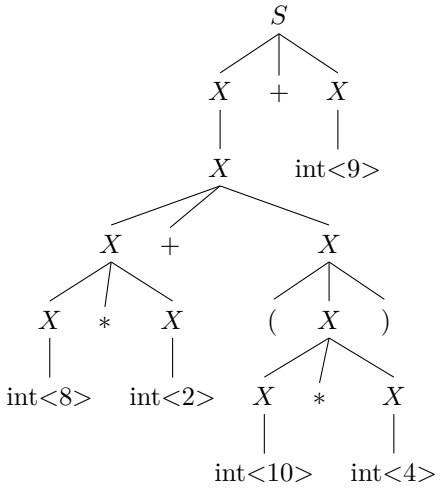
Problem 4

a.

First tree:



Second tree:



b.

5 unique parse trees:
$8 * \{2 + [(10 * 4) + 9]\}$
$8 * \{[2 + (10 * 4)] + 9\}$
$\{8 * [2 + (10 * 4)]\} + 9$
$\{[8 * 2] + (10 * 4)\} + 9$
$[8 * 2] + [(10 * 4) + 9]$

c.

Yes, G is ambiguous. To resolve the ambiguity, we can add some implicit mathematics rules when writing the productions:

$S \to X$	
$X \to X + Y \mid Y$	addtion is left associative, and has the lowest priority
$Y \to Y * Z \mid Z$	multiplication is left associative, and has the second priority
$Z \to (X) \mid \text{int}$	parentheses (simplify inside 'em) has the highest priority

Problem 5

a.

The context-free grammar G is given by the following set of production rules:

$$
\begin{aligned}
S &\to < word\ Y > X < /word > \mid \epsilon \\
X &\to S \mid word\ X \mid X\ word \\
Y &\to word = word\ Y \mid \epsilon
\end{aligned}
$$

b.

The terminals of G are $\{<, >, /, =, word\}$, and the nonterminals of G are $\{S, X, Y\}$.

$$
\begin{aligned}
FIRST(S) &= FIRST(_{<word\ Y>X</word>}) \cup FIRST(\epsilon) \\
&= \{<, \epsilon, \$\} \\
FIRST(X) &= FIRST(S) \cup FIRST(_{word\ X}) \cup FIRST(_{X\ word}) \\
&= \{<, \epsilon, word\} \\
FIRST(Y) &= FIRST(word = word\ Y) \cup FIRST(\epsilon) \\
&= \{word, \epsilon\} \\
FIRST(<) &= \{<\} \\
FIRST(>) &= \{>\} \\
FIRST(/) &= \{/\} \\
FIRST(=) &= \{=\} \\
FIRST(word) &= \{word\} \\
FOLLOW(S) &= \{\$, word, <\} \\
FOLLOW(X) &= \{<, word\} \\
FOLLOW(Y) &= \{>\} \\
FOLLOW(<) &= \{word, /\} \\
FOLLOW(>) &= \{word, <, \$\} \\
FOLLOW(/) &= \{word\} \\
FOLLOW(=) &= \{word\} \\
FOLLOW(word) &= \{word, >, =\}
\end{aligned}
$$

c.

Label the rules:

1. $S \rightarrow < word\ Y > X < /word >$
2. $S \rightarrow\ \epsilon$
3. $X \rightarrow\ S$
4. $X \rightarrow\ word\ X$
5. $X \rightarrow\ X\ word$
6. $Y \rightarrow\ word = word\ Y$
7. $Y \rightarrow\ \epsilon$

Parsing table:

	$<$	$>$	$/$	$=$	$word$	\$
S	1, 2				2	2
X	3, 5				3, 4, 5	
Y		7			6	

No, the G is not an $LL(1)$ grammar, since the parsing table has multiple entries.