ECE 468 & 573

Problem Set 2: Context-free Grammars, Parsers

1. For the following sub-problems, consider the following context-free grammar:

$$S \rightarrow A \quad (1)$$
$$A \rightarrow xAC \quad (2)$$
$$A \rightarrow B \quad (3)$$
$$B \rightarrow yBC \quad (4)$$
$$B \rightarrow \lambda \quad (5)$$
$$C \rightarrow z \quad (6)$$

(a) What are the terminals and non-terminals of this language?

Answer:

Terminals: $\{x, y, z\}$. Non-terminals: $\{S, A, B, C\}$.

(b) Describe the strings are generated by this language. Is this a regular language (*i.e.*, could you write a regular expression that generates this language)?

Answer:

This string is some number of xs, some number of ys, and then as many zs as xs and ys combined. In other words:

$$x^n y^m z^{(n+m)}$$

(c) Show the derivation of the string $xxyzzz$ starting from S (specify which production you used at each step), and give the parse tree according to that derivation.

Answer:

In each step, I have highlighted which non-terminal gets replaced.

$\mathbf{S}$	Rule 1
$\mathbf{A}$	Rule 2
$x\mathbf{A}C$	Rule 2
$xx\mathbf{A}CC$	Rule 3
$xx\mathbf{B}CC$	Rule 4
$xxy\mathbf{B}CCC$	Rule 5
$xxy\mathbf{C}CC$	Rule 6
$xxyz\mathbf{C}C$	Rule 6
$xxyzz\mathbf{C}$	Rule 6
$xxyzzz$	Done!

(d) Give the first and follow sets for each of the non-terminals of the grammar.

Answer:

Non-terminal	**First**	**Follow**
S	$\{x, y, \lambda\}$	$\{\}$
A	$\{x, y, \lambda\}$	$\{z\}$
B	$\{y, \lambda\}$	$\{z\}$
C	$\{z\}$	$\{z\}$

(e) What are the predict sets for each production?

Rule	**Predict set**
1	$\{x, y\}$ Note: this is tricky—we should also predict this rule if we see an EOF
2	$\{x\}$
3	$\{y, z\}$ Note: this is tricky—we should also predict this rule if we see an EOF
4	$\{y\}$
5	$\{z\}$ Note: this is tricky—we should also predict this rule if we see an EOF
6	$\{z\}$

(f) Give the parse table for the grammar. Is this an LL(1) grammar? Why or why not?

Answer:

Non-term:	**x**	**y**	**z**	**EOF**
S	1	1		1
A	2	3	3	3
B		4	5	5
C			6	

(g) Add one more production for C (i.e., of the form $C \rightarrow \alpha$) that makes this grammar *not* LL(1).

Answer: Many options, but one would be:

$$C \rightarrow x$$

Because then, while matching A, on an x the parser would not know whether to predict rule 2 or rule 3.

2. for the following sub-problems, consider the following grammar:

$$\begin{align} S &\rightarrow AB\$ \tag{7}\\ A &\rightarrow xB \tag{8}\\ A &\rightarrow xyB \tag{9}\\ B &\rightarrow zA \tag{10}\\ B &\rightarrow w \tag{11}\\ \tag{12} \end{align}$$

(a) Describe the strings generated by this language.

Answer:
A regular expression capturing the strings of this language would be:

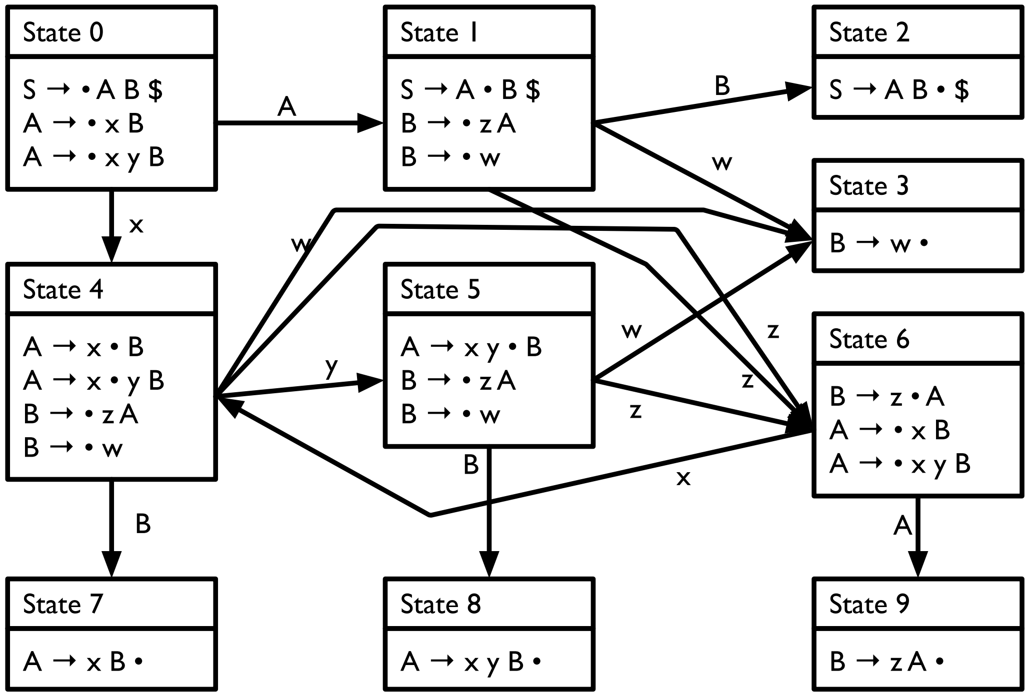
$$((xy?z)^{*}xy?w)(zxy?)^{*}w$$

(b) Is this language LL(1)? Why or why not?

Answer:
This is not LL(1); there would be a predict conflict when trying to match A if the parser sees an x.

(c) Build the CFSM for this grammar.

Answer:



(d) Build the goto and action tables for this grammar. Is it an LR(0) grammar? Why or why not?

Answer:

This is an LR(0) grammar: every state is either a shift state or a reduce state. States 3, 7, 8 and 9 are reduce states, State 2 is the accept state, and the others are shift states.

The goto table is:

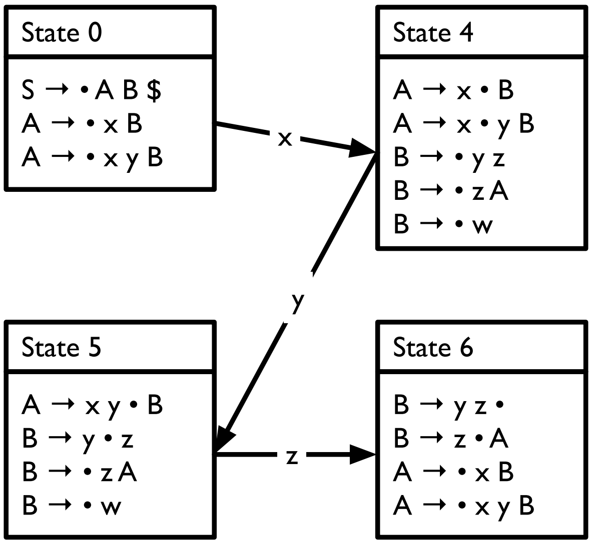
State	x	y	z	w	A	B
0	4				1	
1			6	3		2
2						
3						
4		5	6	3		7
5			6	3		8
6	4				9	
7						
8						
9						

(e) If we add the production

$$B \rightarrow yz$$

to the grammar, is it an LR(0) grammar? Why or why not?

This is no longer an LR(0) grammar, because it will produce a shift/reduce conflict. Here is a partial LR(0) machine that exhibits the conflict (not all states are shown):



(f) (ECE 573 only): Build the LR(1) machine for the grammar extended with the rule from (e).