ECE 468 & 573
Problem Set 2: Context-free Grammars, Parsers

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

$$
\begin{aligned}
S &\rightarrow AA\$ && (1)\\
A &\rightarrow xA && (2)\\
A &\rightarrow B && (3)\\
B &\rightarrow yB && (4)\\
B &\rightarrow \lambda && (5)\\
& && (6)
\end{aligned}
$$

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

   Answer: Terminals: {x, y, $}. Non-terminals: {S, A, B}

   (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)?

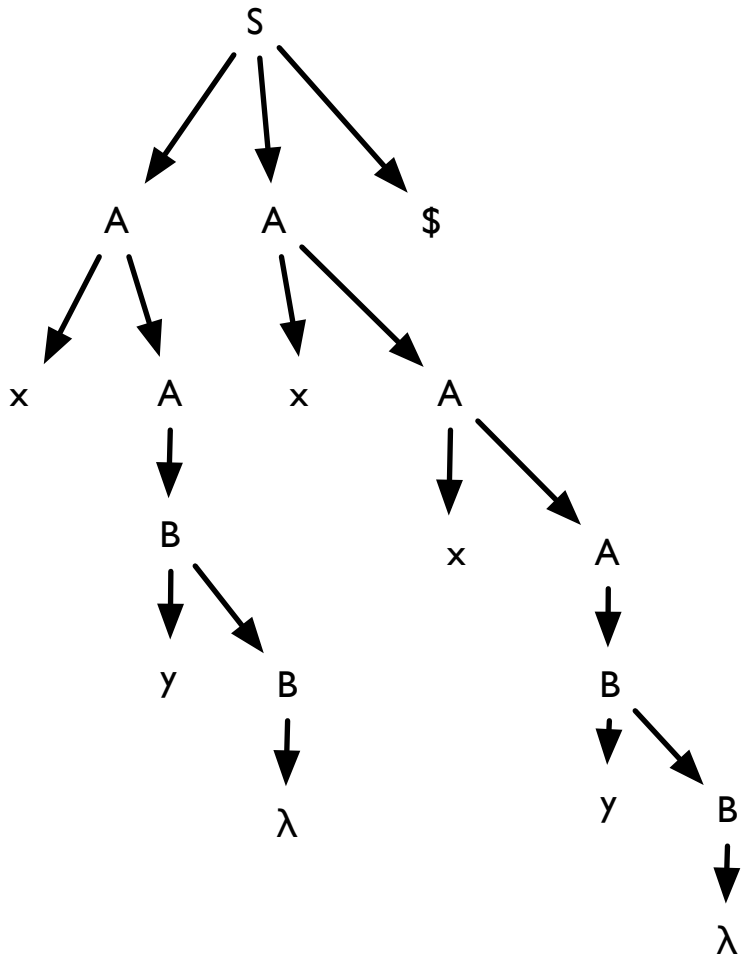
   Answer: Yes, this is a regular language. To see how, note that A turns into x^*y^*. Thus, the whole language can be captured by the following regular expression:

$$x^*y^*x^*y^*\$$$

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

   Answer:

$$
\begin{aligned}
S &\rightarrow AA\$ && \text{Rule 1}\\
&\rightarrow xAA\$ && \text{Rule 2}\\
&\rightarrow xBA\$ && \text{Rule 3}\\
&\rightarrow xyBA\$ && \text{Rule 4}\\
&\rightarrow xyA\$ && \text{Rule 5}\\
&\rightarrow xyxA\$ && \text{Rule 2}\\
&\rightarrow xyxxA\$ && \text{Rule 2}\\
&\rightarrow xyxxB\$ && \text{Rule 3}\\
&\rightarrow xyxxyB\$ && \text{Rule 4}\\
&\rightarrow xyxxy\$ && \text{Rule 5}
\end{aligned}
$$



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

Answer:

$$\begin{aligned} First(S) &= \{x, y, \$\} \\ First(A) &= \{x, y, \lambda\} \\ First(B) &= \{y, \lambda\} \end{aligned}$$

and:

$$\begin{aligned} Follow(S) &= \{\} \\ Follow(A) &= \{x, y, \$\} \end{aligned}$$

$$Follow(B) = \{x, y, \$\}$$

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

Answer:

$$\begin{aligned} Predict(1) &= \{x, y, \$\} \\ Predict(2) &= \{x\} \\ Predict(3) &= \{x, y, \$\} \\ Predict(4) &= \{y\} \\ Predict(5) &= \{x, y, \$\} \end{aligned}$$

Note that even though the first-set of B does not contain \$, \$ is in the predict set for Rule 3 because B can go to λ, and hence the predict set must include the follow-set of A.

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

Answer

	x	y	$
S	1	1	1
A	2, 3	3	3
B	5	4, 5	5

This is not LL(1) because there are conflicts in the parse table.

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

$$S \rightarrow AB\$ \quad (7)$$
$$A \rightarrow xB \quad (8)$$
$$A \rightarrow xw \quad (9)$$
$$B \rightarrow xyA \quad (10)$$
$$B \rightarrow z \quad (11)$$

(a) Describe the strings generated by this language.

Answer: There are many ways to describe this string. Here's one. Every B can turn into the following:

$$(xyx)^*(z|w)$$

And every A can turn into the following:

$$xB|xw$$

which is thus equivalent to:

$$x(xyx)^*(z|w) \mid xw$$

Putting it all together, we get:

$$(x(xyx)^*(z|w) \mid xw)\ (xyx)^*(z|w)$$

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

Answer: To figure this out, we must build the predict sets for these rules. So first, we compute the first and follow sets for each non-terminal:

$$\begin{aligned} First(S) &= \{x\} \\ First(A) &= \{x\} \\ First(B) &= \{x, z\} \end{aligned}$$

and:

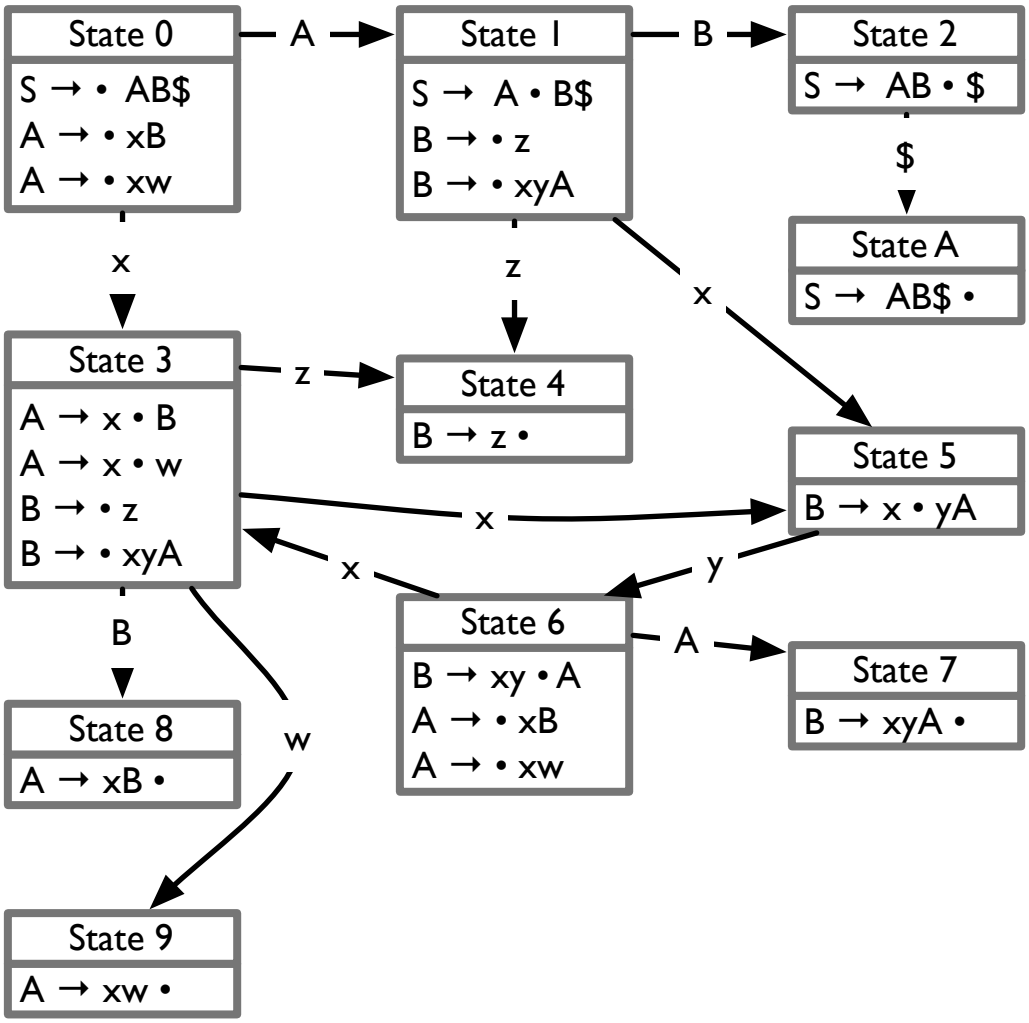
$$\begin{aligned} Follow(S) &= \{\} \\ Follow(A) &= \{x, z, \$\} \\ Follow(B) &= \{x, z, \$\} \end{aligned}$$

Note that rule 1 lets us know that $Follow(A) \supseteq First(B)$ and $Follow(B) \ni \$$, rule 2 lets us know that $Follow(B) \supseteq Follow(A)$ and rule 4 lets us know that $Follow(A) \supseteq Follow(B)$.
If we put these together, we can compute the predict sets:

$$\begin{aligned} Predict(7) &= \{x\} \\ Predict(8) &= \{x\} \\ Predict(9) &= \{x\} \\ Predict(10) &= \{x\} \\ Predict(11) &= \{z\} \end{aligned}$$

We can see that there will be a predict conflict between rules 8 and 9. If we are expanding an A, and our lookahead token is x, we do not know whether to predict $A \to xB$ or $A \to xw$. Thus, this is *not* an LL(1) grammar.

(c) Build the CFSM for this grammar.



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

Answer: The goto table can be read directly off the CFSM. Note how each state in the CFSM has one outgoing transition for each symbol to the right of the "read head." The following states are Shift states: 0, 1, 2, 3, 5, 6. The following states are Reduce states: 4, 7, 8, 9. State 10 (A) is the accept state. This is an LR(0) grammar because there are no Shift/Reduce or Reduce/Reduce conflicts. Every state is either a shift state or a reduce state (the accept state is just a special reduce state).

(e) Show the steps taken by the parser when parsing the string: $xzxyxz$. Give the action and show the state stack and remaining input for each step of the parse. Shift actions should be of the form "Shift X" where X is the state you are shifting to, and Reduce actions should be of the form "Reduce R, goto X" where "R" is the rule being used to reduce, and "X" is the state the parser winds up in.

Answer:

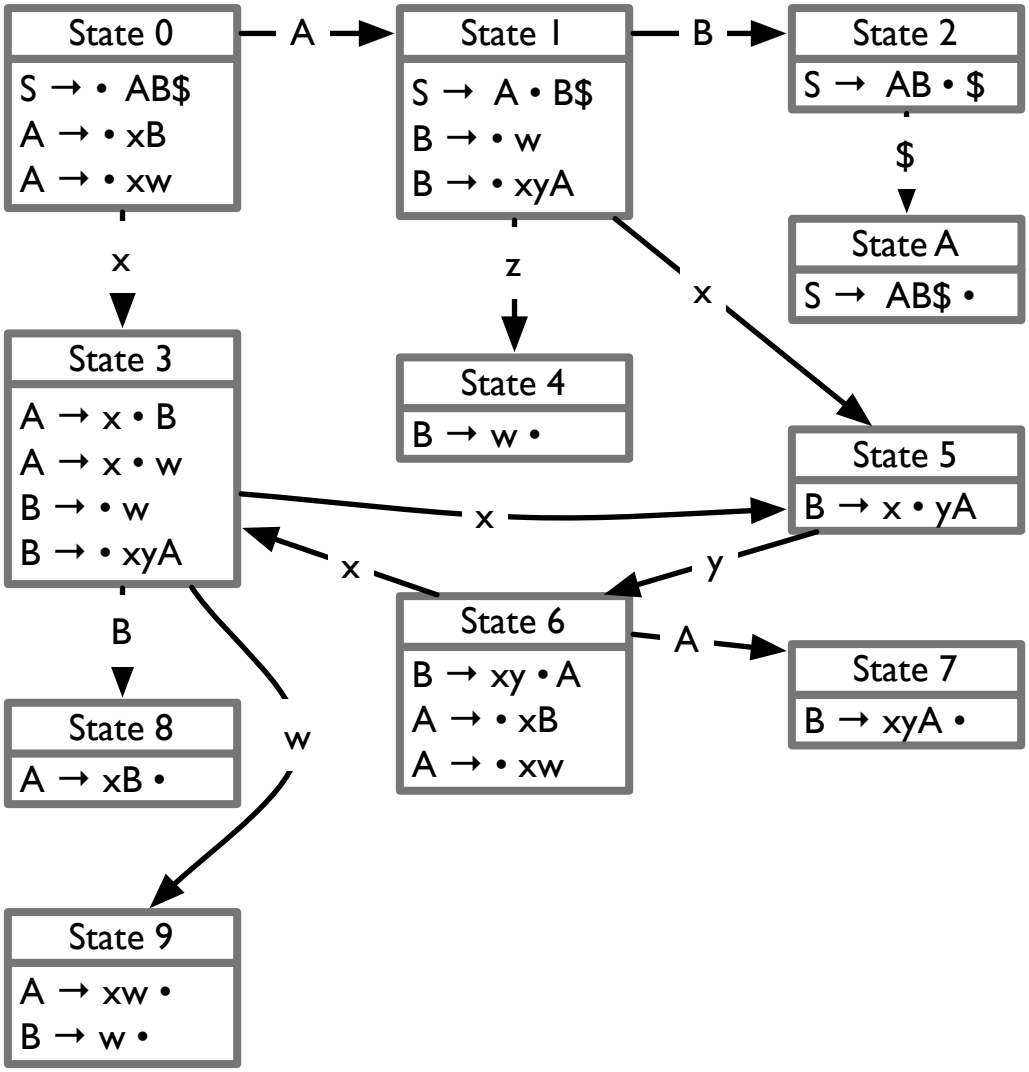
Parse stack	Symbol stack	Remaining input	Action
0		$xzxyxz$	Shift 3
0 3	x	$zxyxz$	Shift 4
0 3 4	xz	$xyxz$	Reduce 11, goto 8
0 3 8	xB	$xyxz$	Reduce 8, goto 1
0 1	A	$xyxz$	Shift 5
0 1 5	Ax	yxz	Shift 6
0 1 5 6	Axy	xz	Shift 3
0 1 5 6 3	$Axyx$	z	Shift 4
0 1 5 6 3 4	$Axyxz$		Reduce 11, goto 8
0 1 5 6 3 8	$AxyxB$		Reduce 8, goto 7
0 1 5 6 7	$AxyA$		Reduce 10, goto 2
0 1 2	AB		Shift A
0 1 2 A	AB\$		Accept

(f) Suppose we change the last production to:

$$B \rightarrow w$$

Is the resulting grammar LR(0)? Why or why not?

Answer: No. To see why, consider the CFSM that gets built:



State 9 has a Reduce/Reduce conflict. If the parser gets to that state, it does not know whether to reduce using $A \to xw$ or $B \to w$.