ECE 468 & 573
Problem Set 2: Context-free Grammars, Parsers (**Solutions**)

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

$$\begin{aligned} S &\rightarrow A\$ && (1)\\ A &\rightarrow xBC && (2)\\ A &\rightarrow CB && (3)\\ B &\rightarrow yB && (4)\\ B &\rightarrow \lambda && (5)\\ C &\rightarrow x && (6) \end{aligned}$$

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

   Answer: Terminals: $\{x, y, \$\}$, Non-terminals: $\{S, A, B, C\}$. Note that λ is not a terminal.

   (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 is a regular language, and the strings generated by the grammar are matched by the regular expression:

$$xy^*(x?)$$

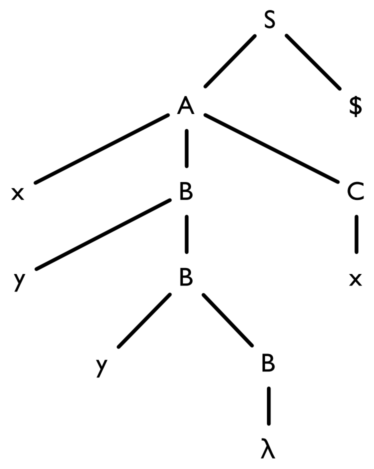
   To see how we came up with this, consider that A can be rewritten to either xBC or CB, that C *always* gets rewritten to x, and that B produces y^*. Hence, depending on which production you use for A, you will either get xy^*x or xy^*.

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

   Answer: We can generate $xyyx\$$ in one way:

$$\begin{aligned} S &\rightarrow A\$ && \text{rule 1}\\ &\rightarrow xBC\$ && \text{rule 2}\\ &\rightarrow xyBC\$ && \text{rule 4}\\ &\rightarrow xyyBC\$ && \text{rule 4}\\ &\rightarrow xyyC\$ && \text{rule 5}\\ &\rightarrow xyyx\$ && \text{rule 6} \end{aligned}$$

   The parse tree for this derivation is straightforward:



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

Answer: The first sets are:

$$\begin{array}{lcll}
First(S) & = & \{x\} & First(S) \supseteq First(A) \\
First(A) & = & \{x\} & First(A) \supseteq First(xBC) = \{x\} \text{ and} \\
 & & & First(A) \supseteq First(CB) = First(C) = \{x\} \\
First(B) & = & \{y, \lambda\} & First(B) \supseteq \{\lambda\} \text{ and} \\
 & & & First(B) \supseteq First(yB) = \{y\} \\
First(C) & = & \{x\} &
\end{array}$$

The follow sets are:

$$\begin{array}{lcll}
Follow(S) & = & \{\} & \text{(always)} \\
Follow(A) & = & \{\$\} & \\
Follow(B) & = & \{x, \$\} & Follow(B) \supseteq Follow(A) \text{ and} \\
 & & & Follow(B) \supseteq First(C) \\
Follow(C) & = & \{y, \$\} & Follow(C) \supseteq Follow(A) \text{ and} \\
 & & & Follow(C) \supseteq First(B) - \lambda = \{y\}
\end{array}$$

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

Answer:

$$\begin{array}{lcl}
Predict(1) & = & \{x\} \\
Predict(2) & = & \{x\} \\
Predict(3) & = & \{x\} \\
Predict(4) & = & \{y\} \\
Predict(5) & = & \{x, \$\} \\
Predict(6) & = & \{x\}
\end{array}$$

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

Answer:

	x	y	$
S	1		
A	2, 3		
B	5	4	5
C	6		

This is not an LL(1) grammar, because there is a conflict in the parse table: if the parser is trying to match an A, and the lookahead token is an x, we it doesn't know whether to pick rule 2 or rule 3.

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

$$\begin{aligned} S &\rightarrow AB\$ && (7) \\ A &\rightarrow xB && (8) \\ A &\rightarrow \lambda && (9) \\ B &\rightarrow xyA && (10) \\ B &\rightarrow w && (11) \end{aligned}$$

(a) Describe the strings generated by this language.

Answer:
This was a rather hard question to answer, because there is no straightforward regular expression that captures this language. There won't be anything quite so tricky on the exam.

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

Answer:
To answer this question, let us first build the first and follow sets for the non-terminals, then use those to build the predict sets for the productions of this grammar.
The first sets are:

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

The follow sets are:

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

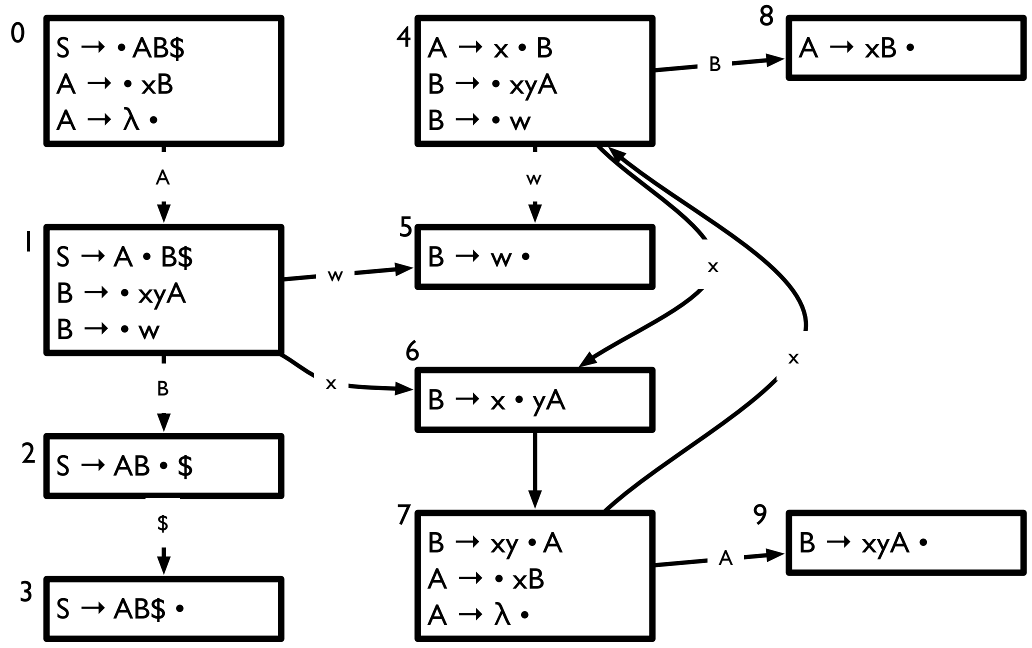
Thus, the predict sets are:

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

This is not an LL(1) grammar, because there would be a conflict in the parse table when parsing an A with a lookahead of 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:
The goto table can be read directly off from the CFSM. For the action table, states 1, 2, 4, and 6 are shift states, states 5, 8 and 9 are reduce states, state 3 is the accept state and states 0 and 7 have shift/reduce conflicts. The existence of shift reduce conflicts prevents this from being an LR(0) grammar.

(e) If we add the production

$$B \rightarrow x$$

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

Answer:
This does not resolve the existing shift/reduce conflicts. In addition, State 4 will add the configuration $B \rightarrow \cdot x$, and state 6 will add the configuration $B \rightarrow x\cdot$, which results in an additional shift/reduce conflict.

(f) Considering the original grammar, suppose we replaced rule 10 with the rule $B \to xA$. Argue that this grammar cannot be parsed at all by any LL or LR parser (hint: is the grammar ambiguous?).

Answer:

This grammar is now ambiguous. Consider the string $xxw\$$. This can be derived in two ways:

$$\begin{array}{lll} S & \to & AB\$ & \text{rule 7} \\ & \to & B\$ & \text{rule 9} \\ & \to & xA\$ & \text{rule 10} \\ & \to & xxB\$ & \text{rule 8} \\ & \to & xxw\$ & \text{rule 11} \end{array}$$

In this derivation, the first A is rewritten to λ.

$$\begin{array}{lll} S & \to & AB\$ & \text{rule 7} \\ & \to & xBB\$ & \text{rule 8} \\ & \to & xxAB\$ & \text{rule 10} \\ & \to & xxB\$ & \text{rule 9} \\ & \to & xxw\$ & \text{rule 11} \end{array}$$

In this derivation, the first A is rewritten to xB — this yields a different parse tree.