Solutions to Written Assignment 2

1. Give a context-free grammar (CFG) for each of the following languages over the alphabet $\Sigma = \{0, 1\}$:

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

$$\begin{aligned} S &\rightarrow 0 \mid 1 \mid 0A0 \mid 1A1 \\ A &\rightarrow A0 \mid A1 \mid \varepsilon \end{aligned}$$

   (b) All strings that contain more 1s than 0s.

$$\begin{aligned} S &\rightarrow A1 \mid MS \mid SMA \\ A &\rightarrow A1 \mid \varepsilon \\ M &\rightarrow \varepsilon \mid MM \mid 0M1 \mid 1M0 \end{aligned}$$

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

$$S \rightarrow \varepsilon \mid 0 \mid 1 \mid 0S0 \mid 1S1$$

2. Consider the following CFG.

$$\begin{aligned} S &\rightarrow AED \mid F \\ A &\rightarrow Aa \mid a \\ B &\rightarrow Bb \mid b \\ C &\rightarrow Cc \mid c \\ D &\rightarrow Dd \mid d \\ E &\rightarrow bEc \mid bc \\ F &\rightarrow aFd \mid BC \end{aligned}$$

   (a) What is the language generated by this grammar?

$$\{a^i b^j c^k d^i \mid (i \geq 0) \wedge (j, k > 0)\} \cup \{a^i b^j c^j d^k \mid i, j, k > 0\}$$

   (b) Show that this grammar is ambiguous by giving a string that can be parsed in two different ways. Draw both parse trees.
   The string *abcd* has the two parse trees shown in Figure **??**.

   (c) Give an unambiguous grammar that generates the same language as the grammar above.

$$\begin{aligned} S &\rightarrow AM \mid MD \mid F \\ A &\rightarrow Aa \mid a \\ B &\rightarrow Bb \mid b \\ C &\rightarrow Cc \mid c \\ D &\rightarrow Dd \mid d \\ E &\rightarrow bEc \mid bc \\ F &\rightarrow aFd \mid BC \\ M &\rightarrow aMd \mid aEd \end{aligned}$$

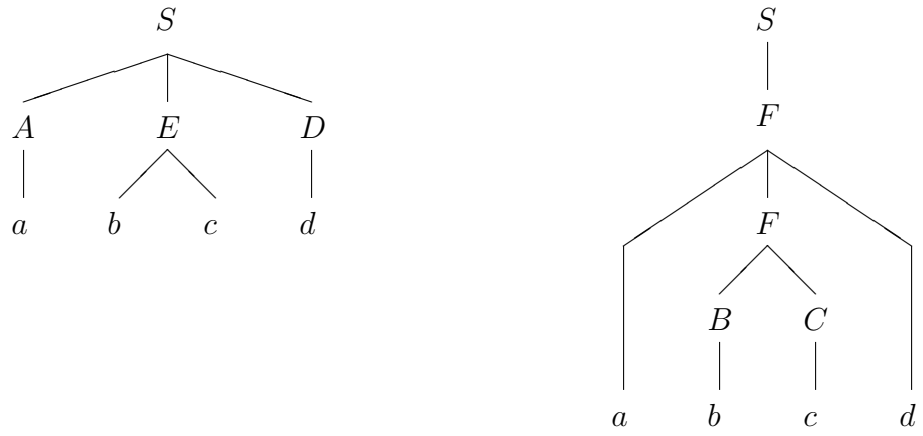

Figure 1: Two parse trees for the string *abcd*.

3. The Cool Reference Manual, in Chapter 11, has the context-free grammar defining the syntax for Cool. Create the parse tree from the grammar definition for the following class definition.

```
class FOO inherits BAR {
  i : Int <- 42;
  baz (x:Int) : Foo { i <- x + i };
};
```

Figure **??** shows a sample parse tree for this class definition. In the Cool Reference Manual, the specification of the syntax for Cool is not given as a pure CFG, and so some productions have been introduced here to capture the regular expression notation used in the specification. The goal of this question was to practice working with the general structure of a parse tree for a language such as Cool, and as such this sample tree is primarily intended to illustrate the overall structure of the parse tree, as opposed to the details of all the nodes in the tree.

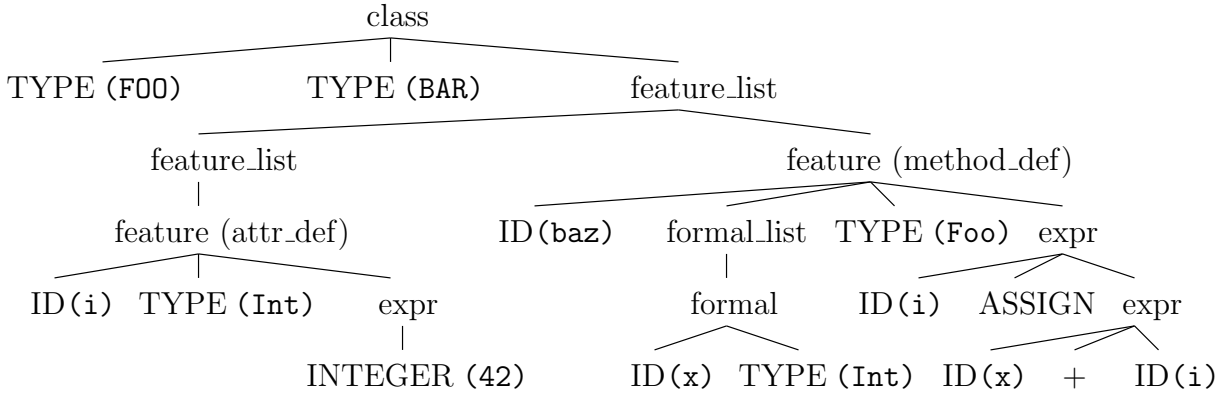

Figure 2: An example parse tree for the class definition.

4. Give a one-sentence description of the language generated by each of the following CFGs.

(a)

$$\begin{aligned} S &\rightarrow Z1Z1Z1A \\ Z &\rightarrow Z0 \mid \varepsilon \\ A &\rightarrow A0 \mid A1 \mid \varepsilon \end{aligned}$$

All binary strings that contain at least three 1s.

(b)

$$S \rightarrow 0 \mid 1 \mid 0S0 \mid 0S1 \mid 1S0 \mid 1S1$$

All binary strings that contain an odd number of symbols.

(c)

$$\begin{aligned} S &\rightarrow DC \mid AE \\ A &\rightarrow Aa \mid \varepsilon \\ C &\rightarrow Cc \mid \varepsilon \\ D &\rightarrow aDb \mid \varepsilon \\ E &\rightarrow bEc \mid \varepsilon \end{aligned}$$

$$\{a^i b^j c^k \mid i, j, k \geq 0 \wedge (i = j \vee j = k)\}$$

5. Consider the following CFG, which has the set of terminals $T = \{\mathbf{id}, (,), [,], ;\}$.

$$\begin{aligned} E &\rightarrow \mathbf{id} \mid \mathbf{id}(A) \mid \mathbf{id}[E] \\ A &\rightarrow E \mid E \,;\, A \end{aligned}$$

(a) Left-factor this grammar so that no two productions with the same left-hand side have right-hand sides with a common prefix.

$$\begin{aligned} E &\rightarrow \mathbf{id}X \\ X &\rightarrow \varepsilon \mid (A) \mid [E] \\ A &\rightarrow EY \\ Y &\rightarrow \varepsilon \mid ; A \end{aligned}$$

(b) Construct an LL(1) parsing table for the left-factored grammar.
The First and Follow sets of the non-terminals are as follows.

$$\begin{array}{ll} \text{First}(E) = \{\mathbf{id}\} & \text{Follow}(E) = \{\$,], ;,)\} \\ \text{First}(X) = \{\varepsilon, (, [\} & \text{Follow}(X) = \{\$,], ;,)\} \\ \text{First}(A) = \{\mathbf{id}\} & \text{Follow}(A) = \{)\} \\ \text{First}(Y) = \{\varepsilon, ;\} & \text{Follow}(Y) = \{)\} \end{array}$$

Here is an LL(1) parsing table for the grammar.

	id	(	)	[	]	;	**$**
E	$E \rightarrow \mathbf{id}X$						
X		$X \rightarrow (A)$	$X \rightarrow \varepsilon$	$X \rightarrow [E]$	$X \rightarrow \varepsilon$	$X \rightarrow \varepsilon$	$X \rightarrow \varepsilon$
A	$A \rightarrow EY$						
Y			$Y \rightarrow \varepsilon$			$Y \rightarrow ; A$	

(c) Show the operation of an LL(1) parser on the input string **id(id[id]; id)**.

Stack	Input	Action
E**\$**	**id(id[id]; id)\$**	$E \rightarrow$ **id**X
id$X$**\$**	**id(id[id]; id)\$**	terminal **id**
X**\$**	**(id[id]; id)\$**	$X \rightarrow$ **(**A**)**
(A**)\$**	**(id[id]; id)\$**	terminal **(**
A**)\$**	**id[id]; id)\$**	$A \rightarrow EY$
EY**)\$**	**id[id]; id)\$**	$E \rightarrow$ **id**X
idXY**)\$**	**id[id]; id)\$**	terminal **id**
XY**)\$**	**[id]; id)\$**	$X \rightarrow$ **[**E**]**
[E**]**Y**)\$**	**[id]; id)\$**	terminal **[**
E**]**Y**)\$**	**id]; id)\$**	$E \rightarrow$ **id**X
id$X$**]**Y**)\$**	**id]; id)\$**	terminal **id**
X**]**Y**)\$**	**]; id)\$**	$X \rightarrow \varepsilon$
]Y**)\$**	**]; id)\$**	terminal **]**
Y**)\$**	**; id)\$**	$Y \rightarrow$ **;** A
; A**)\$**	**; id)\$**	terminal **;**
A**)\$**	**id)\$**	$A \rightarrow EY$
EY**)\$**	**id)\$**	$E \rightarrow$ **id**X
idXY**)\$**	**id)\$**	terminal **id**
XY**)\$**	**)\$**	$X \rightarrow \varepsilon$
Y**)\$**	**)\$**	$Y \rightarrow \varepsilon$
)\$	**)\$**	terminal **)**
\$	**\$**	Accept

6. Consider the following CFG, which has the set of terminals $T = \{\mathbf{a}, \mathbf{b}\}$.

$$\begin{aligned} S &\rightarrow X\mathbf{a} \\ X &\rightarrow \mathbf{a} \mid \mathbf{a}X\mathbf{b} \end{aligned}$$

(a) Construct a DFA for viable prefixes of this grammar using LR(0) items.
Figure **??** shows a DFA for viable prefixes of the grammar.

(b) Identify a shift-reduce conflict in this grammar under the SLR(1) rules.
The First and Follow sets of the non-terminals in the grammar are as follows.

$$\begin{array}{ll} \text{First}(S) = \{\mathbf{a}\} & \text{Follow}(S) = \{\$\} \\ \text{First}(X) = \{\mathbf{a}\} & \text{Follow}(X) = \{\mathbf{a}, \mathbf{b}\} \end{array}$$

In the DFA state 2, one valid action for an SLR(1) parser would be to shift **a**. Also, the parser could reduce using the production $X \rightarrow \mathbf{a}$ on the lookahead symbol **a**, because **a** is in the Follow set of the non-terminal X.

(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 **aaba**.

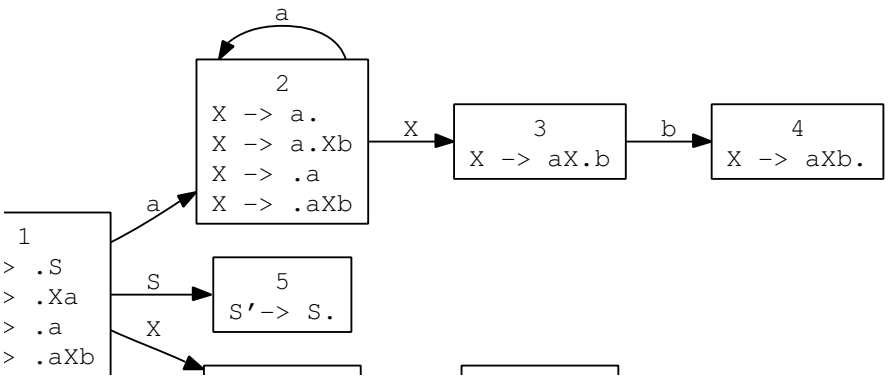

Figure 3: A DFA for viable prefixes of the grammar in Question **??**.

Configuration	DFA Halt State		Action
| **a a b a $**	1		Shift **a**
a | **a b a $**	2	shift-reduce conflict	Shift **a**
a a | **b a $**	2		Reduce $X \rightarrow \mathbf{a}$
a X | **b a $**	3		Shift **b**
a X **b** | **a $**	4		Reduce $X \rightarrow \mathbf{a}X\mathbf{b}$
X | **a $**	6		Shift **a**
X **a** | **$**	7		Reduce $S \rightarrow X\mathbf{a}$
S | **$**	5		Reduce $S' \rightarrow S$
S' | **$**			Accept

(d) Suppose that the production $X \rightarrow \varepsilon$ is added to this grammar. Identify a reduce-reduce conflict in the resulting grammar under the SLR(1) rules.

The item $X \rightarrow \mathbf{.}$ will appear in the DFA state 2. Now, in state 2, there will be two possible reductions on the lookahead symbol **a**: $X \rightarrow \mathbf{a}$ and $X \rightarrow \varepsilon$.