YOURNAME - SUNETID

CS143 - Written Assignment 2

1. Give the context-free grammar for each of the following languages:

   (a) The set of all strings over the alphabet $\{2, 7, +, (,)\}$ representing valid arithmetic expressions in base-10 that evaluate to an odd number. Your CFG should allow for multi-digit numbers.
   Examples of strings in the language:

   $$7 \qquad (72+72)+27 \qquad (((777)))$$

   Examples of strings **not** in the language:

   $$22+22 \qquad (72+27)) \qquad 22+77722+22++7$$

   Answer:

   $$\begin{aligned} S &\to S+E \mid E+S \mid (S) \mid D7 \\ E &\to E+E \mid S+S \mid (E) \mid D2 \\ D &\to D2 \mid D7 \mid \epsilon \end{aligned}$$

(b) The set of all strings over the alphabet $\{a, b\}$ where the number of b's is at most one greater than the number of a's.
Examples of strings in the language:

$aaaba$ $\qquad b \qquad$ $bbaabaaaaaaabaa$

Examples of strings **not** in the language:

$babb$ $\qquad ababababbbb \qquad$ $bbbbb$

Answer:

$$\begin{aligned} S &\to T \mid UbU \\ T &\to TaT \mid U \\ U &\to UU \mid aUb \mid bUa \mid \epsilon \end{aligned}$$

(c) The set of all strings over the alphabet $\{0, 1\}$ in the language $L : \{0^i 1^j 0^k \mid j = i + k\}$.
Examples of strings in the language:

001110 000111 111000

Examples of strings **not** in the language:

01110 101 11111

Answer:

$$\begin{aligned} S &\rightarrow TU \\ T &\rightarrow 0T1 \mid \epsilon \\ U &\rightarrow 1U0 \mid \epsilon \end{aligned}$$

(d) The set of all strings over the alphabet $\{[,], \{, \}, ,\}$ which are sets (for clarification, "," is in the alphabet). We define a set to be a collection of zero or more comma-separated arrays enclosed in an open brace and a close brace. Similarly, we define an array to be a collection of zero or more comma-separated sets enclosed in an open bracket and a close bracket.

Examples of sets:

$$\{\} \qquad \{[], []\} \qquad \{[], [\{\}]\}$$

Examples of arrays:

$$[] \qquad [\{\}, \{[]\}, \{\}] \qquad [\{[\{\}]\}]$$

Examples of strings in the language:

$$\{\} \qquad \{[], []\} \qquad \{[\{\}], [], [\{\}, \{\}]\}$$

Examples of strings **not** in the language:

$$[] \qquad \{[], \{\}\} \qquad \{\}, []$$

Answer:

$$\begin{aligned} S &\to \{T\} \mid \{\} \\ T &\to U, T \mid U \\ U &\to [V] \mid [] \\ V &\to S, V \mid S \end{aligned}$$

2. (a) Left factor the following grammar:

$$\begin{aligned} S &\rightarrow S^* \mid S \cup S \mid S? \mid [T] \\ T &\rightarrow Ta \mid Tb \mid Tc \mid \epsilon \end{aligned}$$

Answer:

$$\begin{aligned} S &\rightarrow SV \mid [T] \\ V &\rightarrow ^* \mid \cup S \mid ? \\ T &\rightarrow TW \mid \epsilon \\ W &\rightarrow a \mid b \mid c \end{aligned}$$

(b) Eliminate left recursion from the following grammar:

$$\begin{aligned} S &\to Sab \mid S! \mid (T) \mid bTb \\ T &\to Ta \mid Tb \mid Tc \mid \epsilon \end{aligned}$$

Answer:

$$\begin{aligned} S &\to (T)S' \mid bTbS' \\ S' &\to abS' \mid !S' \mid \epsilon \\ T &\to T' \\ T' &\to aT' \mid bT' \mid cT' \mid \epsilon \end{aligned}$$

3. Consider the following CFG, where the set of terminals is $\Sigma = \{a, b, \#, \%, !\}$:

$$\begin{aligned} S &\to \%aT \mid U! \\ T &\to aS \mid baT \mid \epsilon \\ U &\to \#aTU \mid \epsilon \end{aligned}$$

(a) Construct the FIRST sets for each of the nonterminals.
Answer:

- S: $\{\%, \#, !\}$
- T: $\{a, b, \epsilon\}$
- U: $\{\#, \epsilon\}$

(b) Construct the FOLLOW sets for each of the nonterminals.
Answer:

- S: $\{\#, !, \$\}$
- T: $\{\#, !, \$\}$
- U: $\{!\}$

(c) Construct the LL(1) parsing table for the grammar.

Answer:

	a	b	#	%	!	$
S			$S \to U!$	$S \to \%aT$	$S \to U!$	
T	$T \to aS$	$T \to baT$	$T \to \epsilon$		$T \to \epsilon$	$T \to \epsilon$
U			$U \to \#aTU$		$U \to \epsilon$	

(d) Show the sequence of stack, input and action configurations that occur during an LL(1) parse of the string "#*abaa*%*aba*!". At the beginning of the parse, the stack should contain a single S.

Answer:

Stack	Input	Action
$S\$$	$\#abaa\%aba!\$$	output $S \to U!$
$U!\$$	$\#abaa\%aba!\$$	output $U \to \#aTU$
$\#aTU!\$$	$\#abaa\%aba!\$$	match #
$aTU!\$$	$abaa\%aba!\$$	match a
$TU!\$$	$baa\%aba!\$$	output $T \to baT$
$baTU!\$$	$baa\%aba!\$$	match b
$aTU!\$$	$aa\%aba!\$$	match a
$TU!\$$	$a\%aba!\$$	output $T \to aS$
$aSU!\$$	$a\%aba!\$$	match a
$SU!\$$	$\%aba!\$$	output $S \to \%aT$
$\%aTU!\$$	$\%aba!\$$	match %
$aTU!\$$	$aba!\$$	match a
$TU!\$$	$ba!\$$	output $T \to baT$
$baTU!\$$	$ba!\$$	match b
$aTU!\$$	$a!\$$	match a
$TU!\$$	$!\$$	output $T \to \epsilon$
$U!\$$	$!\$$	output $U \to \epsilon$
$!\$$	$!\$$	match !
$\$$	$\$$	accept

4. What advantage does left recursion have over right recursion in shift-reduce parsing?
   Hint: Consider left and right recursive grammars for the language a^*. What happens if your input has a million a's?
   Answer: Consider what happens with the right recursive grammar for a^*: $S \to aS \mid \epsilon$. The resulting shift-reduce machine will shift the entire input onto the stack, before performing the first reduction. It will accept all strings in the language, but it requires an unbounded stack size. Now consider $S \to Sa \mid \epsilon$. This machine will initially reduce by $S \to \epsilon$ and then alternate between shifting an "a" on to the stack and reducing by $S \to Sa$. The stack space used is thus constant. In general, using left recursion in grammars for shift-reduce parsers helps limit the size of the stack.

5. Consider the following grammar G over the alphabet $\Sigma = \{a, b, c\}$:

$$\begin{aligned} S' &\to S \\ S &\to Aa \\ S &\to Bb \\ A &\to Ac \\ A &\to \epsilon \\ B &\to Bc \\ B &\to \epsilon \end{aligned}$$

You want to implement G using an SLR(1) parser (note that we have already added the $S' \to S$ production for you).

(a) Construct the first state of the LR(0) machine, compute the FOLLOW sets of A and B, and point out the conflicts that prevent the grammar from being SLR(1).
Answer: Here is the first state of the LR(0) machine:

$$\begin{aligned} S' &\to .S \\ S &\to .Aa \\ S &\to .Bb \\ A &\to .Ac \\ A &\to .\epsilon \\ B &\to .Bc \\ B &\to .\epsilon \end{aligned}$$

We have that FOLLOW(A) = $\{a, c\}$ and FOLLOW(B) = $\{b, c\}$. We have a reduce-reduce conflict between production 5 ($A \to \epsilon$) and production 7 ($B \to \epsilon$), so the grammar is not SLR(1).

(b) Show modifications to production 4 ($A \to Ac$) and production 6 ($B \to Bc$) that make the grammar SLR(1) while having the same language as the original grammar G. Explain the intuition behind this result.

Answer: We change productions 4 and 6 to be right recursive, as follows:

$$A \to cA$$
$$B \to cB$$

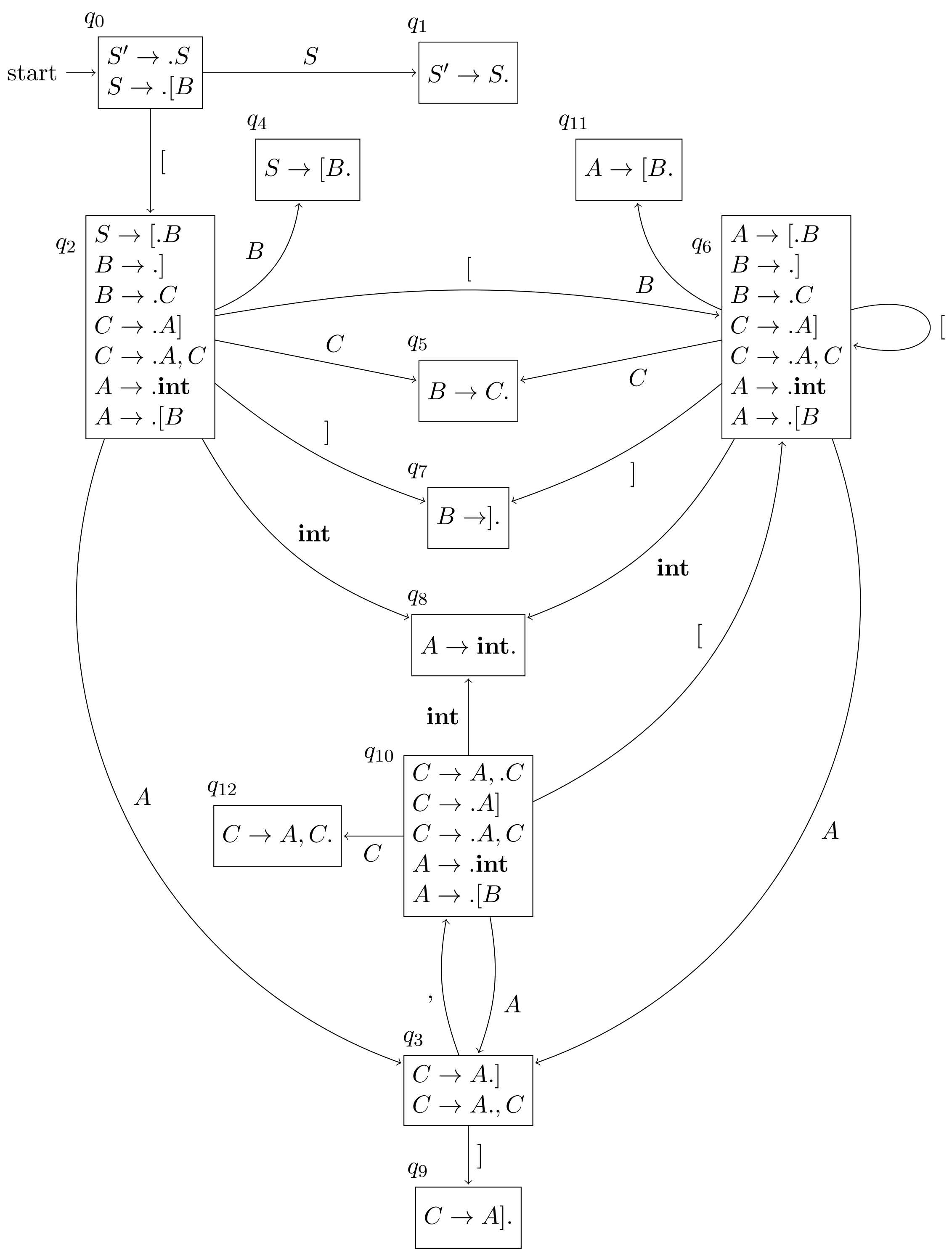
As a result, c is no longer in the follow sets of either A nor B. Since the follow sets are now disjoint, the reduce-reduce conflict in the SLR(1) table is removed. Intuitively, using right recursion causes the parser to defer the decision of whether to reduce a string of c's to A's or B's. When we reach the end of the input, the final character gives us enough information to determine which reduction to perform.

6. Consider the following CFG, where the set of terminals is $\Sigma = \{[,], ,, \textbf{int}\}$:

$$\begin{aligned} S' &\to S \\ S &\to [B \\ A &\to \textbf{int} \mid [B \\ B &\to] \mid C \\ C &\to A] \mid A, C \end{aligned}$$

The grammar describes how arrays may be declared. Arrays consist of zero or more elements where each element is either an integer (represented by **int**) or an array (thus we may have nested arrays). Note that comma is a terminal, and we have already added a dummy production $S' \to S$ for you.

(a) Construct a DFA for viable prefixes of the grammar using LR(0) items.


start
q_0
$S' \to .S$
$S \to .[B$
S
q_1
$S' \to S.$
$[$
q_4
$S \to [B.$
q_{11}
$A \to [B.$
q_2
$S \to [.B$
$B \to .]$
$B \to .C$
$C \to .A]$
$C \to .A, C$
$A \to .\textbf{int}$
$A \to .[B$
B
$[$
B
q_6
$A \to [.B$
$B \to .]$
$B \to .C$
$C \to .A]$
$C \to .A, C$
$A \to .\textbf{int}$
$A \to .[B$
$[$
C
q_5
$B \to C.$
C
$]$
q_7
$B \to].$
$]$
$\textbf{int}$
$\textbf{int}$
q_8
$A \to \textbf{int}.$
$[$
$\textbf{int}$
q_{10}
$C \to A, .C$
$C \to .A]$
$C \to .A, C$
$A \to .\textbf{int}$
$A \to .[B$
A
q_{12}
$C \to A, C.$
C
A
$,$
A
q_3
$C \to A.]$
$C \to A., C$
$]$
q_9
$C \to A].$

(b) Identify any shift-reduce and reduce-reduce conflicts in the grammar under the SLR(1) rules.
Answer: There are no conflicts.

(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 "[**int**,[]]". Your table should include a "Configuration" column, a "DFA Halt State" column, and an "Action" column.
Answer:

Configuration	DFA Halt State	Action
$\mid [\textbf{int}, []]\$$	q_0	shift
$[\mid \textbf{int}, []]\$$	q_2	shift
$[\textbf{int} \mid , []]\$$	q_8, $',' \in FOLLOW(A)$	reduce $A \rightarrow \textbf{int}$
$[A \mid , []]\$$	q_3	shift
$[A, \mid []]\$$	q_{10}	shift
$[A, [\mid]]\$$	q_6	shift
$[A, [] \mid]\$$	q_7, $']' \in FOLLOW(B)$	reduce $B \rightarrow]$
$[A, [B \mid]\$$	q_{11}, $'] \in FOLLOW(A)$	reduce $A \rightarrow [B$
$[A, A \mid]\$$	q_3	shift
$[A, A] \mid \$$	q_9, $'\$' \in FOLLOW(C)$	reduce $C \rightarrow A]$
$[A, C \mid \$$	q_{12}, $'\$' \in FOLLOW(C)$	reduce $C \rightarrow A, C$
$[C \mid \$$	q_5, $'\$' \in FOLLOW(B)$	reduce $B \rightarrow C$
$[B \mid \$$	q_4, $'\$' \in FOLLOW(S)$	reduce $S \rightarrow [B$
$S \mid \$$	q_1, $'\$' \in FOLLOW(S')$	reduce $S' \rightarrow S$
$S' \mid \$$	q_0	accept