TUTORIAL 3

DATE: 24th January 2023

---

1. Is the following grammar ambiguous? Give a proof of your answer.

   $S \rightarrow S\ (\ S\ )\ \mid\ \epsilon$

2. Consider the following fragment of C grammar fragment that we have seen earlier:

```
assignment_expression
: unary_expression
| unary_expression ASSIGN_OP assignment_expression
;
unary_expression
: postfix_expression
| INC_OP unary_expression
| DEC_OP unary_expression
        ;

postfix_expression
: primary_expression
| postfix_expression '[' assignment_expression ']'
| postfix_expression '(' ')'
| postfix_expression '(' argument_expression_list ')'
| postfix_expression '.' IDENTIFIER
| postfix_expression PTR_OP IDENTIFIER
| postfix_expression INC_OP
| postfix_expression DEC_OP
;
primary_expression
: IDENTIFIER
| CONSTANT
| STRING_LITERAL
| '('assignment_expression ')'
;

argument_expression_list
        : argument_expression_list ',' assignment_expression
        | assignment_expression
        ;
```

   And consider the following sentence:

   (a) Is the grammar ambiguous? Explain your answer.

   (b) Are all sentences generated from the grammar valid C program fragments?

3. Consider the context-free grammar

   $S \quad \rightarrow \quad S\,S\,*\mid S\,S\,+\mid a$

   (a) Describe the language described by this grammar.

   (b) Is this grammar ambiguous? Explain your answer.

4. Consider, once again, the ambiguous grammar:

   $\mathit{stmt} \quad \rightarrow \quad$ `if` expr `then` $\mathit{stmt}\ \mid\ \mathit{matched\_statement}$

   $\mathit{matched\_statement} \quad \rightarrow \quad$ `if` expr `then` $\mathit{matched\_statement}$ `else` $\mathit{stmt}\ \mid$ `assignment`

(a) Show that the grammar is ambiguous.

(b) If you were to construct a SLR(1) parsing table for this grammar, which items would result in conflicting entries. What would be the viable prefix(es) for the state containing the conflicting entries?

5. Explain with a grammar why the following rule for $FOLLOW(A)$ is wrong?

   "For the production $A \rightarrow \alpha B$, add everything in $FOLLOW(B)$ to $FOLLOW(A)$."

6. Consider the following grammar which is not $SLR(1)$, i.e. the $SLR(1)$ parsing table will have conflicts:

$$\begin{array}{lccl} 1. & S & \rightarrow & M\ \mathtt{a} \\ 2. & S & \rightarrow & \mathtt{b}\ M\ \mathtt{c} \\ 3. & S & \rightarrow & \mathtt{d}\ \mathtt{c} \\ 4. & S & \rightarrow & \mathtt{b}\ \mathtt{d}\ \mathtt{a} \\ 5. & M & \rightarrow & \mathtt{d} \end{array}$$

   (a) How many conflicts are there in the resulting parsing table?

   (b) Show each pair of items that cause a conflict.

   (c) What are the viable prefixes for the conflicting states.

   (d) Explain in a clear language the reason behind each of the conflicts.

7. Consider the grammar, and the state (set of items) constructed in a breadth-first manner starting with state 0. Each state will then get a unique number. For simplicity, the `if`, `cond` and the `then` have been combined together to a new terminal `ifcondthen`.

$$\begin{array}{lccl} 1. & S & \rightarrow & \mathtt{ass} \\ 2. & S & \rightarrow & \mathtt{ifcondthen}\ S \\ 3. & S & \rightarrow & \mathtt{ifcondthen}\ S\ \mathtt{else}\ S \end{array}$$

   (a) For which state and which symbol is there a conflict? Suppose the parser were to resolve the conflict by choosing the parse tree which matches an `else` with the closest `then`, which of the two conflicting actions should it choose?

   (b) Which state has a self-loop? What are the strings of grammar symbols (represented as regular expressions) that take you to the state.

   (c) Assuming that you have resolved the conflict, on what tokens will you reduce using (i) rule 1, (ii) rule 2, and (iii) rule 3.

8. For the grammar shown below, answer the following questions:

$$\begin{array}{lcl} S & \rightarrow & \mathtt{id}[\,E\,] := E \\ E & \rightarrow & E + T \mid T \\ T & \rightarrow & T * F \mid F \\ F & \rightarrow & (\,E\,) \mid \mathtt{id} \end{array}$$

   (a) In the automaton for the SLR(1) parser, there is a state with a self loop. Identify the state through all the viable prefixes that takes the automaton to this state. You may represent the set of viable prefixes by regular expressions over the grammar symbols.

   (b) Show an example string for which the SLR(1) and LR(1) parsers for the grammar behave differently and, in not more than five lines, clearly explain the difference.

   Hint: For (b) you do not have to deal with the entire grammar. Choose a suitable non-terminal, say A, and replace its productions $A \rightarrow \alpha \mid \beta$ by the production $A \rightarrow a$, where a is assumed to be a terminal. Use the reduced grammar for figuring the answer.

9. Consider the grammar

$$\begin{array}{lcl} S & \rightarrow & L = R \mid R \\ L & \rightarrow & *R \mid \mathtt{id} \\ R & \rightarrow & L \end{array}$$

   Explain why the LR(1) item $L \rightarrow * \bullet R, \{=\}$ is valid for the viable prefix $**$? Give viable prefixes for which the items $R \rightarrow L\bullet, \{=\}$ and $R \rightarrow L\bullet, \{\$\}$ are valid items.