

I. [64] Consider the following grammar G:

1. $S \rightarrow X \perp$
2. $X \rightarrow YX$
3. $X \rightarrow \epsilon$
4. $Y \rightarrow 1Y$
5. $Y \rightarrow 2$

(a) For each of the following grammar classes, if G is of that class, give the appropriate parsing table. If it is not of that class, fully explain why it isn't.

1) LL(0)

LL(0) grammars have deterministic parses with no choices so they generate only 1 string. Since G generates a language with more than 1 string in it, it is not LL(0).

2) LL(1)

For LL(1), if $X \rightarrow m$ and $X \rightarrow n$ are productions, then you must have:

$$\text{FIRST}(m \text{ Follow}(X)) \cap \text{FIRST}(n \text{ FOLLOW}(X)) = \emptyset$$

and since $X \rightarrow YX$ and $X \rightarrow \epsilon$

$$\text{FIRST}(YX \text{ Follow}(X)) \cap \text{FIRST}(\epsilon \text{ FOLLOW}(X)) = \{1,2\} \cap \{\perp\} = \emptyset$$

grammar will be LL(1).

Table:

	1	2	\perp
S	$S \rightarrow X \perp$	$S \rightarrow X \perp$	
X	$X \rightarrow YX$	$X \rightarrow YX$	$X \rightarrow \epsilon$
Y	$Y \rightarrow 1Y$	$Y \rightarrow 2$	

3) LR(0)

S0:

- $[S \rightarrow .X \perp]$
- $[X \rightarrow .YX]$
- $[X \rightarrow .\epsilon]$
- $[Y \rightarrow .1Y]$
- $[Y \rightarrow .2]$
- $[X \rightarrow \epsilon.]$

The completer step adds item $X \rightarrow \epsilon$, since ϵ has zero length from item $X \rightarrow \epsilon.$. This leads to a shift/reduce conflict in state S0, so cannot be LR(0).

4) LR(1)

Yes it is LR(1):

S0:

- $[S \rightarrow .X, \perp]$ X- S1
- $[X \rightarrow .YX, \perp]$ Y- S2
- $[X \rightarrow .\epsilon, \perp]$ 1- S3
- $[Y \rightarrow .1Y, 12\perp]$ 2- S4
- $[Y \rightarrow .2, 12\perp]$ \perp - R(3)
- $[X \rightarrow \epsilon., \perp]$

S1:

$$[S \rightarrow X., \perp] \quad \perp\text{- R(1)}$$

S2:

- $[X \rightarrow Y.X, \perp]$ X- S5
- $[X \rightarrow .\epsilon, \perp]$ Y- S2
- $[X \rightarrow .YX, \perp]$ 1- S3
- $[Y \rightarrow .1Y, 12\perp]$ 2- S4
- $[Y \rightarrow .2, 12\perp]$ \perp - R(3)
- $[X \rightarrow \epsilon., \perp]$

S3:

- $[Y \rightarrow 1.Y, 12\perp]$ Y- S6
- $[Y \rightarrow .1Y, 12\perp]$ 1- S3
- $[Y \rightarrow .2, 12\perp]$ 2- S4

S4:

$$[Y \rightarrow 2., 12\perp] \quad 12\perp\text{-R(5)}$$

S5:

$$[X \rightarrow YX., \perp] \quad \perp\text{- R(2)}$$

S6:

$$[Y \rightarrow 1Y., 12\perp] \quad 12\perp\text{- R(4)}$$

5) SLR(1)

No -

S0:

- $[S \rightarrow .X]$
- $[X \rightarrow .YX]$
- $[X \rightarrow .\epsilon]$
- $[Y \rightarrow .1Y]$
- $[Y \rightarrow .2]$
- $[X \rightarrow \epsilon.]$

(3) Showed that there was a shift reduce conflict in state S0. But from $[S \rightarrow .X\perp]$, $\text{First}(X) = \{1,2, \perp\}$. And from $[X \rightarrow \epsilon.]$, $\text{Follow}(X) = \{\perp\}$. Therefore a lookahead cannot resolve shift/reduce conflict.

6) LALR(1)

No 2 states in (4) have the same set of core items, so it is by default LALR(1).

7) General precedence

Not general precedence.

From production 1, $X=\perp$. From productions 1 and 2 we get $X>\perp$. Since precedence relations are not unique, it is not a precedence grammar.

(b) Show that the LANGUAGE generated by grammar G is a regular language.

Productions 2 and 3 generates Y^* and productions 4 and 5 generate 1^*2 . So language is $((1^*2)^* \perp)$, which is obviously regular.

II. [18] Answer each of the following:

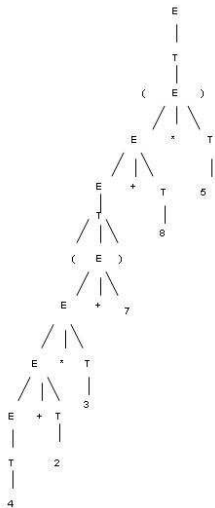
a) [3] Assuming the usual grammar for expressions, give the Polish Postfix for the expression:
 $((4+2*3+7)+8*5)$
 423^*+7+85^*+

b) [3] If the grammar for an expression is given by the grammar:

$$E \rightarrow E + T \mid E - T \mid E * T \mid E / T \mid T$$

$$T \rightarrow \text{digit} \mid (E)$$

Give the parse tree for the expression in (a).



c) [3] Using the grammar in (b), give the Postfix for the expression in (a).

$$42+3^*7+8+5^*$$

d) [3] Give a set of quads that would be the output for the Postfix in (c).

- (add, 4, 2, T1)
- (mult, T1, 3, T2)
- (add, T2, 7, T3)
- (add, T3, 8, T4)
- (mult, T5, 5, T6)

e) [6] What is the minimal value of k for which a Polish postfix grammar is LR(k) (i.e., a grammar that generates Polish postfix strings)? Explain why.

The grammar will be LR(0) so k=0. In a postfix grammar, the operator is the rightmost symbol of the grammar so upon seeing an operator, make a reduction with no lookahead. e. g.,

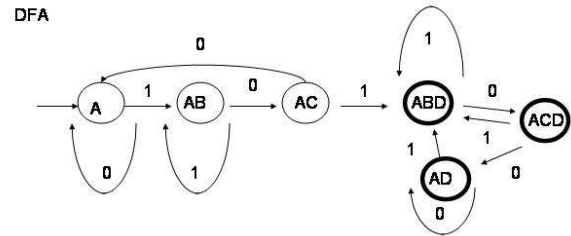
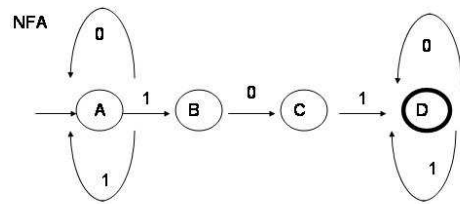
$$S \rightarrow \text{digit} \mid S S + \mid S S - \mid S S * \mid S S /$$

which is LR(0).

III. [15] For the regular expression $(0 \mid 1)^* 101 (0 \mid 1)^*$

(a) Give the minimal state DFA that recognizes the same set.

The NFA and DFA is given by the following figures:

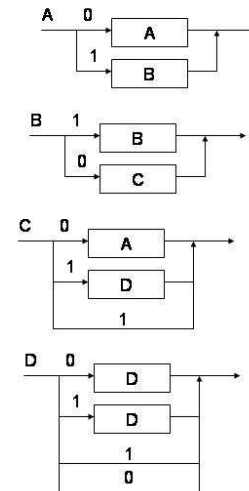


The states ABD ACD abd AD can be obviously combined since they are all accepting states and once you get into ABD, you will accept any string from then on.

(b) Give the regular grammar that recognizes the same set.

- $A \rightarrow 0A \mid 1B$
- $B \rightarrow 1B \mid 0C$
- $C \rightarrow 0A \mid 1D \mid 1$
- $D \rightarrow 0D \mid 1D \mid 0 \mid 1$

(c) Give the syntax diagrams for the regular grammar in (b) above.



IV. [3] Why do you want only synthesized attributes in YACC (or Bison or CUP)? What would happen if you defined YACC with inherited attributes?

YACC works as a bottom up parser so that synthesized attributes are known as reductions are made. Using inherited attributes, YACC would have to make multiple passes over the parse tree to evaluate the attributes top down.