ASSIGNMENT 1

Programming Languages

Bibodi, Jay Nikhil
CSC 135, Section 1, Tu-Th: 1:30 – 2:45

Grammar:

block ::= B {statemt} E [D]
statemt ::= asignmt | ifstmt | while | inpout | block
asignmt ::= A ident ~ exprsn
ifstmt ::= I comprsn T block [L block]
while ::= W comprsn block
inpout ::= iosym ident {, ident}
comprsn ::= (oprnd opratr oprnd)
exprsn ::= factor {sumop factor}
factor ::= oprnd {prodop oprnd}
oprnd ::= integer | ident | (exprsn)
ident ::= letter {char}
char ::= letter | digit
integer ::= digit {digit}
iosym ::= R | O
opratr ::= < | = | > | !
sumop ::= + | -
prodop ::= * | /
letter ::= X | Y | Z
digit ::= 0 | 1

The tokens are: B E D A ~ I T L W , ( ) R O < + > ! + - * / X Y Z 0 1

Nonterminals are shown as lowercase words. Note that the following characters are NOT tokens (they are EBNF metasymbols): | { }

1. Compute the FIRST and FOLLOW for all the non-terminal in the above grammar.

Non Terminal	**First**	**Follow**
Digit	{ 0 , 1 }	Follow (char) U First (digit) U Follow (integer) = { ~ , , , E , < , = , > , ! ,) , * , / , + , - , 0 , 1 }
Letter	{ X , Y , Z }	First (char) U Follow (char) = { 0 , 1 , X , Y , Z , ~ , , , E , < , = , > , ! ,) , * , / , + , - }
Prodop	{ * , / }	First (oprnd) = { 0 , 1 , X , Y , Z , (}
Sumop	{ + , - }	First (factor) = { 0 , 1 , X , Y , Z , (}
Opratr	{ < , = , > , ! }	First (oprnd) = { 0 , 1 , X , Y , Z , (}
Iosym	{ R , O }	First (ident) = { X , Y , Z}
Integer	First (digit) = { 0 , 1 }	Follow (oprnd) = { < , = , > , ! ,) , * , / , + , - , E }

Char	First (letter) U First (digit) = { 0 , 1 , X , Y , Z }	Follow (ident) = { ~ , , , E , < , = , > , ! ,) , * , / , + , - }
Ident	First (letter) = { X , Y , Z }	{ ~ } U { , } U Follow (inpout) U Follow (oprnd) = { ~ , , , E , < , = , > , ! ,) , * , / , + , - }
Oprnd	First (integer) U First (letter) U { (} = { 0 , 1 , X , Y , Z , (}	First (opratr) U {) } U First (prodop) U Follow (factor) = { < , = , > , ! ,) , * , / , + , - , E }
Factor	First (oprnd) = { 0 , 1 , X , Y , Z , (}	First (sumop) U Follow (exprsn) = { + , - , E ,) }
Exprsn	First (factor) = { 0 , 1 , X , Y , Z , (}	Follow (asignmt) U {) } = { E ,) }
Comprsn	{ (}	{ T } U follow (block) = { T , B }
Inpout	First (iosym) = { R , O }	Follow (statemt) = { E }
While	{ W }	Follow (statemt) = { E }
Ifstmt	{ I }	Follow (statemt) = { E }
Asignmt	{ A }	Follow (statemt) = { E }
Statemt	First (asignmt) U First (ifstmt) U First (while) U First (inpout) U First (block) = { A , I , W , R , O , B }	{ E }
Block	{ B }	Follow (statemt) U { L } U follow (ifstmt) U follow (while) = { $, E , L }

2. Show that the grammar satisfies the two requirements for predictive parsing (it does, you just need to prove it). Make sure that you read the supplement regarding the rules for an EBNF grammar below.

Answer :

block ::=

first (statemt) ∩ { E } = { A , I , W , R , O , B } ∩ { E } = Ø
follow (block) ∩ { D } = { $, E , L } ∩ { D } = Ø

stetemt ::=

first (asignmt) ∩ first (ifstmt) ∩ first (while) ∩ first (inpout) ∩ first (block)
= { A } ∩ { I } ∩ { W } ∩ { R, O } ∩ { B } = Ø

asignmt ::= trivial case

ifstmt ::=

first (block) ∩ follow(ifstmt) = { B } ∩ { E } = Ø

while ::= trivial case

inpout ::=

first (ident) ∩ follow (inpout) = { X , Y , Z } ∩ { E } = Ø

comprsn ::= trivial case

exprsn ::=

first (sumop) ∩ first (factor) ∩ follow (exprsn)
= { +. - } ∩ { 0 , 1 , X , Y , Z , (} ∩ { E ,) } = Ø

first (sumop) ∩ first (factor) = Ø

first (factor) ∩ follow (exprsn) = Ø

factor ::=

first (prodop) ∩ first (oprnd) ∩ follow (factor) = { * , / } ∩ { 0 , 1 , X , Y , Z , (} ∩ { + , - , E ,) } = Ø

first (prodop) ∩ first (oprnd) = Ø

first (oprnd) ∩ follow (factor) = Ø

oprnd ::=

first (integer) ∩ first (ident) ∩ { (}
= { 0 , 1 } ∩ { X, Y , Z } ∩ { (} = Ø

first (integer) ∩ first (ident) ∩ {) } = Ø

first (integer) ∩ first (ident) ∩ first (exprsn) = Ø

ident ::=

first (char) ∩ follow (ident)
= { 0 , 1 , X , Y , Z } ∩ { ~ , , , E , < , = , > , ! ,) , * , / , + , - } = Ø

char ::=

first (letter) ∩ first (digit) = { X , Y , Z } ∩ { 0 , 1 } = Ø

integer ::=

first (digit) ∩ follow (integer) = { 0 , 1 } ∩ { < , = , > , ! ,) , * , / , + , - , E } = Ø

iosym ::= trivial case

opratr ::= trivial case

sumop ::= trivial case

prodop ::= trivial case

letter ::= trivial case

digit ::= trivial case