There are six parts of the grammar where branches lead to different parts of the grammar , I will procede to show that their possible options do not overlap and they are disjoint

statemt : asignmt | ifstmt | for | input | output

= first(asignmt) ∩ first(ifstmt) ∩ first(for) ∩ first(input) ∩ first(output)

= [first(indent) ∩ {I} ∩ {F} ∩ {N} ∩ {O}]

= [first(letter) ∩ {I} ∩ {F} ∩ {N} ∩ {O}]

= [{X,Y,Z}∩ {I} ∩ {F} ∩ {N} ∩ {O}] no overlap of paths

sumop: + | -

={+} ∩ {-} no overlap of paths

prodop: \* | /

={+} ∩ {-} no overlap of paths

Oprnd: integer | ident | ( exprsn )

= first(integer) ∩ first(indent) ∩ ( first(exprsn) )

= first(digit) ∩ first(letter) ∩ ( first(factor) )

= {0,1,2,3,4,5,6,7} ∩ {Z,X,Y} ∩ ( first(oprnd) )

= {0,1,2,3,4,5,6,7} ∩ {Z,X,Y} ∩ ( first(integer) U first(indent) U {(} )

= {0,1,2,3,4,5,6,7} ∩ {Z,X,Y} ∩ ( first(digit) U first(letter) U {(} )

= {0,1,2,3,4,5,6,7} ∩ {Z,X,Y} ∩ { (} U {0,1,2,3,4,5,6,7} U {Z,X,Y} U {(} U{ )}

no overlap in paths

opratr: < | = | > | !

= {<} ∩ {>} ∩ {=} ∩ {!} no overlap in paths

Char: letter | digit

=first(digit) ∩ first(letter)

= {0,1,2,3,4,5,6,7} ∩ {Z,X,Y}

letter : X | Y | Z

= {Z,X,Y}

Digit: 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7

={0,1,2,3,4,5,6,7}

I will now also show how the rules that lead to an item within the rule, or

exits the rule are disjoint

program ::= statemt {statemt} $

first(branch)= first (statement)

= first(asignmt) U first(ifstmt) U first(for) U first(input) U first(output)

= [first(indent) U {I} U {F} U {N} U {O}]

= [first(letter) U {I} U {F} U {N} U {O}]

= [{X,Y,Z}U {I} U {F} U {N} U {O}]

Follow(statement) = ‘$’

ifstmt ::= I comprsn @ {statemt} [% {statemt}] &

first(branch)= first (statemt)

= first(asignmt) U first(ifstmt) U first(for) U first(input) U first(output)

= [first(indent) U {I} U {F} U {N} U {O}]

= [first(letter) U {I} U {F} U {N} U {O}]

= [{X,Y,Z}U {I} U {F} U {N} U {O}]

Follow(brach)= Second(branch)

Second(branch)= {%} ∩ {&}

Follow(ifstmt)= ‘$’

for ::= F ( asignmt ) ( comprsn) L {statemt} \

first(branch)= first (statemt)

= first(asignmt) U first(ifstmt) U first(for) U first(input) U first(output)

= [first(indent) U {I} U {F} U {N} U {O}]

= [first(letter) U {I} U {F} U {N} U {O}]

= [{X,Y,Z}U {I} U {F} U {N} U {O}]

Follow(for)= “$’

input ::= N ident {, ident} ; AND output ::= O ident {, ident } ;

first(branch) = {,}

follow(input)= ‘$’

exprsn ::= factor { sumop factor }

first(branch) = first(sumop)

= {+, -}

Follow(exprsn)= $

factor ::= oprnd { prodop oprnd }

first(branch) = first(prodep)

= {\*,/}

Follow(factor) = ‘$’ U sumop

Follow(sumop)= factor U ‘$ //eventually it will reach the terminating char

ident ::= letter {char}

first(branch)= firs(char)

= first(letter) U first(digit)

= {Z,X,Y} U {0,1,2,3,4,5,6,7}

Follow(indent) = {;} U {,}

integer ::= digit {digit}

first(branch)= firs(digit)

= {0,1,2,3,4,5,6,7}

Follow(integer) = prodop U sumop U ‘$’

Follow(prodop)= oprnd

Follow(oprnd)= prodop U sumop U ‘$’

Follow(sumop)= factor

Follow(factor) = sumop U ‘$’ eventually it will get to the terminating char