Calculating FIRST sets and Nullable non-terminals

Rules for Calculating Nullable non-terminals

We use the following 2 rules are used to calculate nullable non-terminals

I.   If A -> ε is a grammar rule

      add A to Nullable

II. If A -> A_2 ... A_k is a grammar rule and
     A_1 is nullable and A_2 is nullable and ... and A_k is nullable

      add A to Nulllable

Rules for Calculating FIRST sets for terminals and non-terminals

The following three rules are used for calculating FIRST sets for terminals and non-terminals

I.   FIRST(a) = { a } for every terminal a

II. If A -> B α   is a grammar rule, where B is a terminal or non-terminal, then

      add FIRST(B) to FIRST(A)

III. If A -> A_2 ... A_k B α   is a grammar rule, where B is a terminal or non-terminal and
     A_1 is nullable and A_2 is nullable and ... and A_k is nullable

      add FIRST(B) to FIRST(A)

Calculating FIRST sets for a sequence of terminals and non-terminals

We can calculate FIRST sets for sequences of terminals and non terminals as follows

    FIRST(A1 A2 ... Ak) = if A1 is nullable then
                      FIRST(A1) U FIRST(A2 A3 ... Ak)
             else
                FIRST(A1)

Calculating FIRST sets and Nullable non-terminals

Algorithm for calculating nullable non-terminals

The nullable set is calculated iteratively. We start with an initialization step that adds all non-terminals with epsilon rules to nullable:

<u>Initialization</u> For every non-terminals A with a rule of the form A -> ε, add A to nullable

Then, we repeat multiple iterations of applying rule II for Nullable set until the Nullable sets does not change:

<u>Iterations</u>

```
change = true;
while changed
{
    changed = false
    for every grammar rule
        apply Nullable rule II to the grammar rule
    if the Nullable set changes, changed = true
}
```


Algorithm for calculating FIRST sets

To calculate FIRST sets, we need to calculate the Nullable set. Once the nullable set is calculated, FIRST sets are calculated iteratively.

<u>Initialization:</u> We start by applying rules I to get the initial FIRST sets for terminals. Next, we initialize all FIRST sets for non-terminals to the empty set.

After the initialization, we iteratively apply rules II and III of FIRST sets to all grammar rules as described in the following pseudocode

```
change = true;
while changed
{
    changed = false
    for every grammar rule
        apply FIRST set rules II and III to the grammar rule
    if any FIRST set changed
            changed = true
}
```

Calculating FOLLOW Sets

Rules for Calculating FOLLOW sets

We use the following 5 rules for calculating FOLLOW sets for non-terminals

I.   add $ to FOLLOW(S)

II.   If A -> α B   is a grammar rule, and B is a non-terminal,

add FOLLOW(A) to FOLLOW(B)

III.   If A -> α B A_1 A_2 ... A_k is a grammar rule, and B is a non-terminal, and A1, A2, ... Ak are ALL nullable, then

add FOLLOW(A) to FOLLOW(B)

IV.   If A -> α B A_1 A_2 ... A_k is a grammar rule, where B is non-terminal

add FIRST(A_1) to FOLLOW(B)

V.   If A -> α B A_1 A_2 ... A_i A_{i+1} A_k is a grammar rule, where B is non-terminal and A_1, A_2, ... A_i are ALL nullable, then

add FIRST(A_{i+1}) to FOLLOW(B)

Note: rules IV and V together are equivalent to:

IV.   IV-V: If A -> α B A_1 A_2 ... A_k is a grammar rule, where B is non-terminal

add FIRST(A_1 A_2 ... A_k) to FOLLOW(B)

For the calculation of FIRST(A_1 A_2 ... A_k), see page 30. This is also discussed on the next page.

Calculating FOLLOW sets (cont'd)

Algorithm for calculating FOLLOW sets

Calculating FOLLOW sets is done iteratively, not recursively. The following are the steps for calculating FOLLOW sets:

1. Start by applying rules I to add $ to FOLLOW(S)
2. Next initialize all FOLLOW sets for non-terminals to empty
3. Run the following pseudocode

```
// first pass apply rules IV and V to all grammar rules
for every grammar rule
    apply FOLLOW set rules  IV, and V to the grammar rule

// multiple passes until there is no change
// apply FOLLOW rules II and III to all grammar rules
change = true;
while changed
{
    changed = false
    for every grammar rule
            apply FOLLOW set rules II and III to the grammar rule
            if any FOLLOW set changes set changed = true

}
```

How do we apply rules IV and V to a grammar rule A -> A_1 A_2 ... A_k?

The straightforward, and less efficient, way is the following

```
for i = 1 to k-1 do
    if Aᵢ is a non-terminal then
        for j = i+1 to k do
            add FIRST(Aⱼ) to FOLLOW(Aᵢ)
            if  Aⱼ is not nullable then
                break;
```

for i = 1 **to** k-1 **do**
    if A_i is a non-terminal **then**
        for j = i+1 **to** k **do**
            add FIRST(A_j) to FOLLOW(A_i)
            if A_j is not nullable **then**
                break;

A more efficient calculation will proceed backward taking advantage of the fact that for any sequence A_1 A_2 ... A_k

FIRST(A_1 A_2 ... A_k) = FIRST(A_1) U FIRST(A_2 ... A_k) if A_1 is nullable and

FIRST(A_1 A_2 ... A_k) = FIRST(A_1) if A_1 is not nullable

The calculation can be done as follows

temp = FIRST(A_k)
for i = k-1 **downto** 1 **do**
    add temp to FOLLOW(A_i)
    if Ai is nullable then
      temp = temp U FIRST(A_i)
    else
      temp = FIRST(A_i)

Calculating Nullable Sets Example

S -> A B C   (1)   Nullable = { D, E, C, B }

A -> B C d D E   (2)

B -> b B   (3)

B -> D E   (4)

D -> Ɛ   (5)

E -> Ɛ   (6)

E -> e   (7)

C -> c C   (8)

C -> Ɛ   (9)

Initialization

Applying rules I for Nullable, we get the initial Nullable set for all non-terminals with "epsilon rules". The sets includes D, E and C.

Iteration

Now, we iterate over all grammar rules to apply Nullable rule II.

We examine grammar rule (1):
       Since A is not in Nullable, rule II does not apply

We examine grammar rule (2):
       Since B is not in Nullable, rule II does not apply

We examine grammar rule (3):
       Since b is not in Nullable, rule II does not apply

We examine grammar rule (4):
       Since both D and E are in Nullable, by rule II we add B to nullable   1

Examining rules (5) through (9) results in no change.

Since the Nullable set changed, we do one more pass over all grammar rules but that results in no change, so the final Nullable set is { D, E, C, B }

Calculating FIRST sets example

S -> A B C   (1)   <mark>Nullable = { D, E, C, B }</mark>

A -> B C d D E   (2)   FIRST(b) = { b }

B -> b B   (3)   FIRST(c) = { c }

B -> D E   (4)   FIRST(d) = { d }

D -> ε (5)   FIRST(e) = { e }

E -> ε (6)   FIRST(S) = {   }

E -> e   (7)   FIRST(A) = {   }

C -> c C   (8)   FIRST(B) = {   }

C -> ε (9)   FIRST(C) = {   }

FIRST(D) = {   }

FIRST(E) = {   }

To calculate FIRST sets, we should calculate the Nullable set, which we have already done.

Applying rule I gives us the FIRST sets of terminals

Initialization gives us empty FIRST sets for all non-terminals.

The sets shown above are what we get at the end of the initialization phase.

Now, we are going to start iterating over all grammar rules. This will be shown starting on the next page

Calculating FIRST sets example

S -> A B C   (1)

A -> B C d D E   (2)

B -> b B   (3)

B -> D E   (4)

D -> ε   (5)

E -> ε   (6)

E -> e   (7)

C -> c C   (8)

C -> ε   (9)

Nullable = { D, E, C, B }

FIRST(b) = { b }

FIRST(c) = { c }

FIRST(d) = { d }

FIRST(e) = { e }

FIRST(S) = {   }

FIRST(A) = { d[1]   }

FIRST(B) = {   }

FIRST(C) = {   }

FIRST(D) = {   }

FIRST(E) = {   }

We examine grammar rule (1)

        by rule II, we add FIRST(A) to FIRST(S)
        Since FIRST(A) is empty, there is no change

        since A ∉ Nullable, rule III does not apply

We examine grammar rule (2)

        by rule II, we add FIRST(B) to FIRST(A)
        Since FIRST(B) is empty, there is no change

        since B is in Nullable, rule III applies, and we can add FIRST(C) to FIRST(A)
        Since FIRST(C) is empty, there is no change

        Since both B and C are in Nullable, by rule III, we add FIRST(d) to FIRST(A)
        So, d is added to FIRST(A)   1

        Since d is not nullable, we cannot continue with rule III

Calculating FIRST sets example

S -> A B C   (1)

A -> B C d D E   (2)

B -> b B   (3)

B -> D E   (4)

D -> ℰ   (5)

E -> ℰ   (6)

E -> e   (7)

C -> c C   (8)

C -> ℰ   (9)

Nullable = { D, E, C, B }

FIRST(b) = { b }

FIRST(c) = { c }

FIRST(d) = { d }

FIRST(e) = { e }

FIRST(S) = {   }

FIRST(A) = {   d[1]   }

FIRST(B) = {   b[2]   }

FIRST(C) = {   }

FIRST(D) = {   }

FIRST(E) = {   }

We examine grammar rule (3)

       by rule II, we add FIRST(b) to FIRST(B)
       This adds b to FIRST(B)   2

       since b ∉ Nullable, rule III does not apply

We examine grammar rule (4)

       by rule II, we add FIRST(D) to FIRST(B)
       Since FIRST(D) is empty, there is no change

       Since D is nullable, by rule III we add FIRST(E) to FIRST(B)
       but FIRST(E) is empty, so there is no change   3

Calculating FIRST sets example

S -> A B C   (1)   <mark>Nullable = { D, E, C, B }</mark>

A -> B C d D E   (2)   FIRST(b) = { b }

B -> b B   (3)   FIRST(c) = { c }

B -> D E   (4)   FIRST(d) = { d }

D -> Ɛ   (5)   FIRST(e) = { e }

E -> Ɛ   (6)   FIRST(S) = {   }

E -> e   (7)   FIRST(A) = { d[1]   }

C -> c C   (8)   FIRST(B) = { b[2]   }

C -> Ɛ   (9)   FIRST(C) = {   }

FIRST(D) = {   }

FIRST(E) = {   }

We examine grammar rule (5)

        Since the righthand side is Ɛ , rules II and III don't apply

We examine grammar rule (6)

        Since the righthand side is Ɛ , rules II and III don't apply

Calculating FIRST sets example

S -> A B C   (1)

A -> B C d D E   (2)

B -> b B   (3)

B -> D E   (4)

D -> ε   (5)

E -> ε   (6)

E -> e   (7)

C -> c C   (8)

C -> ε   (9)

Nullable = { D, E, C, B }

FIRST(b) = { b }

FIRST(c) = { c }

FIRST(d) = { d }

FIRST(e) = { e }

FIRST(S) = {   }

FIRST(A) = { d^1 }

FIRST(B) = { b^2 }

FIRST(C) = { c^4 }

FIRST(D) = {   }

FIRST(E) = { e^3 }

We examine grammar rule (7)

     by rule II, we add FIRST(e) to FIRST(E)
     This adds e to FIRST(E)   3

     rule III does not apply

We examine grammar rule (8)

     by rule II, we add FIRST(c) to FIRST(E)
     This adds c to FIRST(C)   4

     rule III does not apply

We examine grammar rule (9)

     Since the righthand side is ε , rules II and III don't apply

Calculating FIRST sets example

S -> A B C   (1)   Nullable = { D, E, C, B }

A -> B C d D E   (2)   FIRST(b) = { b }

B -> b B   (3)   FIRST(c) = { c }

B -> D E   (4)   FIRST(d) = { d }

D -> ε (5)   FIRST(e) = { e }

E -> ε (6)   FIRST(S) = { d^5 }

E -> e   (7)   FIRST(A) = { d^1 , b^6 }

C -> c C   (8)   FIRST(B) = { b^2 }

C -> ε (9)   FIRST(C) = { c^4 }

     FIRST(D) = {   }

     FIRST(E) = { e^3 }

At this point we have finished examining all the grammar rules. **Since some FIRST sets have changed, we need to re-examine all grammar rules and apply rules II and III to them.**

We examine grammar rule (1)

> we add FIRST(A) to FIRST(S), so d gets added   5

We examine grammar rule (2)

> by rule III, we add FIRST(B) to FIRST(A)
> This adds { b } to FIRST(A)   6
>
> rules IV and V do not apply

We examine grammar rule (3)

> This will add b to FIRST(B), but b is already there, so no change
>
> Rules IV and V do not apply

Calculating FIRST sets example

S -> A B C   (1)

A -> B C d D E   (2)

B -> b B   (3)

B -> D E   (4)

D -> ε   (5)

E -> ε   (6)

E -> e   (7)

C -> c C   (8)

C -> ε   (9)

FIRST(b) = { b }

FIRST(c) = { c }

FIRST(d) = { d }

FIRST(e) = { e }

FIRST(S) = {. d^5 }

FIRST(A) = { d^1 , b^6 }

FIRST(B) = { b^2 , e^7 }

FIRST(C) = { c^4 }

FIRST(D) = {   }

FIRST(E) = { e^3 }

We examine grammar rule (4)

        Applying rule II results in no change
        Since D is nullable, by rule III, we add FIRST(E) to FIRST (B)
        This will add e to FIRST(B)   6

We examine grammar rule (5)-(9)

        No change!

Since some FIRST sets have changed, we need another pass.

Calculating FIRST sets example

S -> A B C(1)   <mark>Nullable = { D, E, C, B }</mark>

A -> B C d D E   (2)   FIRST(b) = { b }

B -> b B   (3)   FIRST(c) = { c }

B -> D E   (4)   FIRST(d) = { d }

D -> ε (5)   FIRST(e) = { e }

E -> ε (6)   FIRST(S) = { d^5 , b^8 }

E -> e   (7)   FIRST(A) = { d^1 , b^6 }

C -> c C   (8)   FIRST(B) = { b^2 , e^7 }

C -> ε (9)   FIRST(C) = { c^4 }

                                    FIRST(D) = {   }

                                    FIRST(E) = { e^3 }

Since some FIRST sets changed and Nullable changed in the previous pass, we need to do another pass (it is enough for wither some FIRST set or Nullable to change in order to do another pass).

We examine grammar rule (1)

    We add FIRST(A) to FIRST(S) by rule II
    This adds b to FIRST(S)   8

    rule III does not apply

Calculating FIRST sets example

S -> A B C   (1)

A -> B C d D E   (2)

B -> b B   (3)

B -> D E   (4)

D -> ε   (5)

E -> ε   (6)

E -> e   (7)

C -> c C   (8)

C -> ε   (9)

==Nullable = { D, E, C, B }==

FIRST(b) = { b }

FIRST(c) = { c }

FIRST(d) = { d }

FIRST(e) = { e }

FIRST(S) = { d^5 , b^8 }

FIRST(A) = { d^1 , b^6, e^9 , c^{10} }

FIRST(B) = { b^2 , e^7 }

FIRST(C) = { c^4 }

FIRST(D) = { }

FIRST(E) = { e^3 }

We examine grammar rule (2)

         We add FIRST(B) to FIRST(A) by rule II, so
e gets added to FIRST(A)   9

         B is nullable, so by rule III we add FIRST(C) to FIRST(A)
This adds c to FIRST(A)   10

         both B and C are nullable,
we add FIRST(d) to FIRST(A) by rule IV.
This adds d to FIRST(A)   11

         d is not nullable, we cannot continue applying rule III

We examine rules (3) – (9)

         no change!

Since some FIRST sets changed, we need another pass

Calculating FIRST sets example

S -> A B C`   (1)

A -> B C d D E   (2)

B -> b B   `   (3)

B -> D E   (4)

D -> Ɛ   (5)

E -> Ɛ   (6)

E -> e   (7)

C -> c C   (8)

C -> Ɛ   (9)

Nullable = { D, E, C, B }

FIRST(b) = { b }

FIRST(c) = { c }

FIRST(d) = { d }

FIRST(e) = { e }

FIRST(S) = { d^5 , b^8 , e^{11} , c^{11} }

FIRST(A) = { d^1 , b^6 , e^9 , c^{10} }

FIRST(B) = { b^2 , e^7 }

FIRST(C) = { c^4 }

FIRST(D) = { }

FIRST(E) = { e^3 }

We examine grammar rule (1)

       We add FIRST(A) to FIRST(S) by rule III
       This will add e , c FIRST(S) (b and d are already there)   11

       Rules IV and V do not apply

We examine grammar rules (2)–(9)

       no change!

Since some FIRST sets changed, we need to do one more pass.

In the last pass, nothing changes, and we are done.

Calculating FOLLOW sets example

S -> A B C   (1)   Nullable = { D, E, C, B, D }   FOLLOW(S) = { \$[1] ,   }

A -> B C d D E   (2)   FIRST(b) = { b }   FOLLOW(A) = { b[2], e[2], c[3] ,   }

B -> b B   (3)   FIRST(c) = { c }   FOLLOW(B) = { c[4]   }

B -> D E   (4)   FIRST(d) = { d }   FOLLOW(C) = {   }

D -> Ɛ   (5)   FIRST(e) = { e }   FOLLOW(D) = {   }

E -> Ɛ   (6)   FIRST(S) = { b , e, c , **d** }   FOLLOW(E) = {   }

E -> e   (7)   FIRST(A) = { b, e, c , **d** }

C -> c C   (8)   FIRST(B) = { b , e   }

C -> Ɛ   (9)   FIRST(C) = { c   }

FIRST(D) = {   }

FIRST(E) = { e   }

We look at the same grammar. We start with the calculated the Nullable set of non-terminals and the FIRST sets as shown above.

We initialize all FOLLOW sets to empty

We add \$ to FOLLOW(S) by rule I   1

Next, we do one pass by applying rules IV and V to all grammar rules.

Then we repeatedly apply rules II and III to all grammar rules until there is no change

_____
Applying rules IV and V

We start examining grammar rules

We examine rule (1) RHS

     We start with A

         we add FIRST(B) to FOLLOW(A) by rule IV   2
         this adds b and e to FOLLOW(A)

         Since B is nullable, we add FIRST(C) to FOLLOW(A)
         by rule V. This adds c to FOLLOW(A)   3

     Next, we look at B

         we add FIRST(C) to FOLLOW(B) by rule IV   4
         this adds c to FOLLOW(B)

         rule V does not apply for B

     We are done with grammar rule (1)

Calculating FOLLOW sets example

S -> A B C	(1)	Nullable = { D, E, C, B, D }		FOLLOW(S) = { $\$^1$,		}
A -> B C d D E	(2)	FIRST(b) = { b }		FOLLOW(A) = { b^2, e^2, c^3 ,		}
B -> b B	(3)	FIRST(c) = { c }		FOLLOW(B) = { c^4 , d^5		}
B -> D E	(4)	FIRST(d) = { d }		FOLLOW(C) = { d^6		}
D -> ε	(5)	FIRST(e) = { e }		FOLLOW(D) = { e^7		}
E -> ε	(6)	FIRST(S) = { b , e, c , **d** }		FOLLOW(E) = {		}
E -> e	(7)	FIRST(A) = { b, e, c , **d** }				
C -> c C	(8)	FIRST(B) = { b , e	}			
C -> ε	(9)	FIRST(C) = { c	}			
		FIRST(D) = {	}			
		FIRST(E) = { e	}			

<u>Applying rules IV and V</u>

We continue examining grammar rules

We examine rule (2) RHS

    We start with B

        we add FIRST(C) to FOLLOW(B) by rule IV
this results in no change

        Since C is nullable, we add FIRST(d) to FOLLOW(B)
by rule V. This adds d to FOLLOW(B)   5

        Since d is not nullable, we cannot keep on moving forward
with rule V

    We continue with C

        we add FIRST(d) to FOLLOW(C) by rule IV   6

        rule V does not apply

    We skip d because it is a terminal

    We continue with D

        We add FIRST(E) to FOLLOW(D) by rule IV
This adds e to FOLLOW(D)   7

        rule V does not apply

    we are done with grammar rule (2)

Calculating FOLLOW sets example

S -> A B C   (1)

A -> B C d D E   (2)

B -> b B   (3)

B -> D E   (4)

D -> Ɛ   (5)

E -> Ɛ   (6)

E -> e   (7)

C -> c C   (8)

C -> Ɛ   (9)

Nullable = { D, E, C, B, D }

FIRST(b) = { b }

FIRST(c) = { c }

FIRST(d) = { d }

FIRST(e) = { e }

FIRST(S) = { b , e, c , d }

FIRST(A) = { b, e, c , d }

FIRST(B) = { b , e   }

FIRST(C) = { c   }

FIRST(D) = {   }

FIRST(E) = { e   }

FOLLOW(S) = { \$[1] ,   }

FOLLOW(A) = { b[2], e[2], c[3] ,   }

FOLLOW(B) = { c[4] , d[5]   }

FOLLOW(C) = { d[6]   }

FOLLOW(D) = { e[7]   }

FOLLOW(E) = {   }

_____

Applying rules IV and V

Next, we examine rules (3) through (9)

      For grammar rules (3) and (5) – (9), FOLLOW set rules IV and V do not apply

      For rule (4), there is no change

At this point we are done with our pass applying rules IV and V

Next, we need to do multiple iterations of applying FOLLOW sets rules II and III

Calculating FOLLOW sets example

S -> A B C　　　　　(1)　　　Nullable = { D, E, C, B, D }　　FOLLOW(S) = { $[1] , 　　　　　}

A -> B C d D E　　　(2)　　　FIRST(b) = { b }　　　　　　FOLLOW(A) = { b[2] , e[2], c[3] , $[10] }

B -> b B　　　　　　(3)　　　FIRST(c) = { c }　　　　　　FOLLOW(B) = { c[4] , d[5] , $[9]　　}

B -> D E　　　　　　(4)　　　FIRST(d) = { d }　　　　　　FOLLOW(C) = { d[6] , $[8]　　　　}

D -> ε　　　　　　　(5)　　　FIRST(e) = { e }　　　　　　FOLLOW(D) = { e[7], b[12] , c[12] , $[12]}

E -> ε　　　　　　　(6)　　　FIRST(S) = { b , e, c , **d** }　FOLLOW(E) = {b[11], e[11], c[11] , $[11] }

E -> e　　　　　　　(7)　　　FIRST(A) = { b, e, c , **d** }

C -> c C　　　　　　(8)　　　FIRST(B) = { b , e　　　}

C -> ε　　　　　　　(9)　　　FIRST(C) = { c　　　　　}

　　　　　　　　　　　　　　FIRST(D) = {　　　　　　}

　　　　　　　　　　　　　　FIRST(E) = { e　　　　　}

<u>Applying rules II and III</u>

We examine grammar rule (1)

　　　　　by rule II, we add FOLLOW(S) to FOLLOW(C)
　　　　　This adds $ to FOLLOW(C)　　　　　　　　　　　　　　　　　　　　　　8

　　　　　Since C is nullable, we add FOLLOW(S) to FOLLOW(B)
　　　　　by rule III. This adds $ to FOLLOW(B)　　　　　　　　　　　　　　　　9

　　　　　Since both B and C are nullable, we add
　　　　　FOLLOW(S) to FOLLOW(A) by rule III. This adds $ to FOLLOW(A)　　　10

　　　　　we are done applying rules II and III to grammar rule (1)

Next, we examine grammar rule (2)

　　　　　by rule II, we add FOLLOW(A) to FOLLOW(E)
　　　　　This adds b, e , c, and $ to FOLLOW(E)　　　　　　　　　　　　　　11

　　　　　Since E is nullable, we add FOLLOW(A) to FOLLOW(D)
　　　　　by rule III. This adds b, e, c, and $ to FOLLOW(D), but
　　　　　e is already in FOLLOW(D) so it is not added again　　　　　　　　12

　　　　　We stop here because of d, which is a terminal, is not nullable

　　　　　we are done applying rules II and III to grammar rule (2)

Calculating FOLLOW sets example

S -> A B C   (1)

A -> B C d D E   (2)

B -> b B   (3)

B -> D E   (4)

D -> ε   (5)

E -> ε   (6)

E -> e   (7)

C -> c C   (8)

C -> ε   (9)

Nullable = { D, E, C, B, D }

FIRST(b) = { b }

FIRST(c) = { c }

FIRST(d) = { d }

FIRST(e) = { e }

FIRST(S) = { b , e, c , **d** }

FIRST(A) = { b, e, c , **d** }

FIRST(B) = { b , e   }

FIRST(C) = { c   }

FIRST(D) = {   }

FIRST(E) = { e   }

FOLLOW(S) = { \$[1] ,   }

FOLLOW(A) = { b[2] , e[2], c[3] , \$[10] }

FOLLOW(B) = { c[4] , d[5] , \$[8]   }

FOLLOW(C) = { d[6] , \$[7]   }

FOLLOW(D) = { e[7], b[12], c[12], \$[12], d[14] }

FOLLOW(E) = {b[11], e[11], c[11], \$[11], d[13] }

Applying rules II and III

next we examine grammar rule (3)

       no change

Next, we examine gramamr rule (4)

       by rule II, we add FOLLOW(B) to FOLLOW(E)
       This adds c, d, and $ to FOLLOW(E) but c and $ are already in FOLLOW(E)
       so, d gets added   13

       Since E is nullable, we add FOLLOW(B) to FOLLOW(D)
       This adds c, d, and $ to FOLLOW(E) but c and $ are already in FOLLOW(E)
       so, d gets added   14

Next, we examine grammar rules (5) through (9)

       none of them results in any change

Since some FOLLOW sets changed in this pass, we need to do one other pass of applying FOLLOW rules II and III to all grammar rules.

In the next pass there is no change and the answer we have above the final answer.