Q1. Consider the following non-left-recursive grammar

S -> ABCDE

A -> a | ε

B -> b| ε

C -> c

D -> d| ε

E -> e| ε

1. Find The first and follow of the grammar
2. Construct the predictive parsing table

**Find The first and follow of the grammar**

Computing the First Sets.

1. Let’s start with the Start Symbol S :

**First(S) = First(A)**

This is obvious and clear. Hence we need to determine what is the First Set of A. However, because A is a Non-Terminal and is joined back-to-back with B, we may need to later examine First(B) in case First(A) contains ε. And in the same sense, if First(B) contains ε, we may need to examine First(C) and so on.

2. To determine the First Set of A, we look at the A Production Rule and observe its RHS. At its RHS, we see that A can be broken down to either ‘a’ or ‘ε’ (epsilon, i.e. the empty statement). Hence the First Set of A contains ‘a’ and ‘ε’ :

**First(A) = { a, ε }**

Now because First(A) includes ε, and based on the Production Rule S -> ABCDE,

S -> BCDE is a valid derivation (the A being substituted by ε).

Hence First(S) includes First(B) or :

**First(S) = First(A) U First(B) = { a } U First(B)**

Note that the ε in First(A) is removed as it has been used to work out the inclusion of First(B).

3. Let’s now work out the First Set of B :

**First(B) = { b, ε }**.

Note the following at this point :

(i) We have encountered the fact that First(A) includes ε which led to First(S) = First(A) U First(B)

(ii) We now noted that First(B) also includes ε.

(iii) Hence First(S) must include First(C) :

**First(S) = First(A) U First(B) U First(C) = { a } U { b } U First(C)**

4. First(C) is easily worked out to c.

**Hence First(C) = { c }**

Now note that because First(C) does not include ε, First(S) will not carry on to include First(D) nor First(E). Hence the buck stops here.

5. Working out based on the above findings :

First(S) =  { a } U { b } U { c } = { a, b, c  }

First(A) = { a, ε }

First(B) = { b, ε }.

First(C) = { c }

6. Next we work out the First sets of the rest of the non-terminals D and E.

First(D) = { d, ε }

First(E) = { e, ε }

7. Altogether, we have the following eventual First Sets :

**First(S) =  { a, b, c  }**

**First(A) = { a, ε }**

**First(B) = { b, ε }.**

**First(C) = { c }**

**First(D) = { d, ε }**

**First(E) = { e, ε }**

Computing the Follow Sets.

1. Let’s work out the Follow Set of S.

In parsing, $ is known as the end-of-input marker, which represents the end of the input string, hence used while parsing to indicate that the input string has been completely processed. The Follow Set of the Start Symbol always includes $. It is a special case. Hence Follow(S) must include { $ }.

To determine whether there be any other terminals that are part of the Follow Set of S, we examine the Grammar and see any occurrence of S at the RHS of any of the Rules. If S appeared at the RHS of any Rule, it will be possible for S to hold other terminals in its Follow Set. We see that S does not appear at the RHS of any Rule.

Hence **Follow(S) = { $ }**.

2. Follow(A)

When computing the Follow Set of a non-terminal other than the Start Symbol, we generally look at the RHS of the Rules of the Grammar and make observations of what can appear on the right side of the non-terminal.

We observe that S -> ABCDE and postulate that Follow(A) must at least include First(B).

Hence Follow(A) includes First(B) = { b, ε }

Now, since First(B) includes ε, it means that B can expand to ε and so Follow(A) will also include First(C) = { c }

Now, if First(C) included ε, we would have to further allow that both B and C can expand to ε and we will thus have to examine First(D). However, C does not include ε and so we do not go down this path.

Hence **Follow(A) = { b, c }**

3. Follow(B)

We again use the Rule S -> ABCDE and observe that Follow(B) = First(C)

Hence **Follow(B) = { c }**

4. Follow(C)

We again use the Rule S -> ABCDE and observe that Follow(C) = First(D) = { d, ε }

Since First(D) includes ε, First(D) includes First(E) = { e, ε }

Hence Follow(C) = { d, e, ε }.

Since Follow(C) includes ε, it means that it is possible that C is followed by an ε.

This implies that S can derive a sentence like ABC with the D and E both replaced by ε. Hence Follow(C) includes Follow(S).

Therefore Follow(C) = { d, e } U Follow(S)

Hence **Follow(C) = { d, e, $ }**

5. Follow(D)

We again use the Rule S -> ABCDE and observe that Follow(D) includes First(E) = { e, ε }

Hence Follow(D) = { e, ε }

We see that Follow(D) includes ε and so it is possible for D to be followed immediately by ε.

This implies that S can derive a sentence like ABCD. Hence Follow(D) includes Follow(S)

Hence **Follow(D) = { e, $ }**

6. Follow(E)

We again use the Rule S -> ABCDE and observe that Follow(E) includes Follow(S) = { $ }

Hence **Follow(E) = { $ }**

7. Two tips can be used when computing the Follow Sets :

7.1 Observe the occurrences of the non-terminals in the RHS of Rules and work from there.

7.2 Whenever a non-terminal can be caused to appear at the extreme right end of the RHS of a Rule, the Follow Set of that non-terminal includes the Follow Set of the LHS of the Rule. E.g. :

Since S -> ABCDE, Follow(E) includes Follow(S)

And since it is possible to have S -> ABCD (since E -> ε) , Follow(D) includes Follow(S).

8. Altogether, we have the following eventual Follow Sets :

**Follow(S) = { $ }**.

**Follow(A) = { b, c }**

**Follow(B) = { c }**

**Follow(C) = { d, e, $ }**

**Follow(D) = { e, $ }**

**Follow(E) = { $ }**

|  |  |  |
| --- | --- | --- |
| **Non-terminals** | **First** | **Follow** |
| **S** | **{ a, b, c  }** | **{ $ }** |
| **A** | **{ a, ε }** | **{ b, c }** |
| **B** | **{ b, ε }** | **{ c }** |
| **C** | **{ c }** | **{ d, e, $ }** |
| **D** | **{ d, ε }** | **{ e, $ }** |
| **E** | **{ e, ε }** | **{ $ }** |

**Construct the predictive parsing table**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **N\T** | **a** | **b** | **c** | **d** | **e** | **$** |
| **S** | **S -> ABCDE** | **S -> ABCDE** | **S -> ABCDE** |  |  |  |
| **A** | **A -> a** | **A -> ε** | **A -> ε** |  |  | **A -> ε** |
| **B** |  | **B -> b** | **B -> ε** |  |  | **B -> ε** |
| **C** |  |  | **C -> c** |  |  |  |
| **D** |  |  |  | **D -> d** | **D -> ε** | **D -> ε** |
| **E** |  |  |  |  | **E -> e** | **E -> ε** |