Graphical Calculation of First and Follow Sets of a Grammar

Dr. ASR Murty, Maria Navin J R

Abstract— This paper describes a graphical method for computation of First and Follow sets of non terminals of a grammar. First and Follow sets play an important role during syntax analysis with LL and LR parsers. Calculation of First and Follow sets by traditional methods can be tedious and error prone. Though First and Follow calculation is not new, the way it is represented graphically is very much useful to a beginner for understanding the concept. In order to solve this problem a graphical method to obtain First and Follow sets is proposed in this paper. We also perform a check on the correctness of First and Follow with the representation of derivations and a study on the role of associativity for operators is presented.

Index Terms— graphical calculation, first, follow, context free grammar, parse trees, syntax analysis, precedence, associativity.

1 Introduction

The types of parsers available are Universal parsers which can parse any type of grammar, top down and bottom up parsers. During the computation of parsing tables for top down and bottom up parsing First and Follow sets are needed. The subclasses LL and LR of top down and bottom up methods can express most of the constructs in programming languages [1]. These parsers need the calculation of First and Follow in the initial step.

LR parsers are preferred over LL parsers as LL parsers transform original grammar to new form for eliminating left recursion and left factoring. This distorts the original grammar and causes problems for syntax directed translation as it introduces nulls. However, in deciding when to shift, when to reduce and to decide what to reduce, the use of First & Follow is made use of in LR, SLR, LR(k) item sets and LALR parsers.

2 Calculation of First and Follow

To compute First(X) for all grammar symbols, apply the Following rules until no more terminals or ε can be added to any First/Follow set [2][3].

1. If X is terminal, then First(X) is X.
2. If X→ ε is a production, then add ε to First(X).
3. If X is non-terminal and $X \rightarrow Y_1 Y_2 \ldots Y_K$ is a production, then place a in First(X) if for some i, a is in First(Y_i), and ε is in all of First (Y_1)…First (Y_{i-1}); that is, $Y_1 \ldots Y_{i-1} *\Rightarrow \varepsilon$. If ε is in First ($Y_j$) for all j=1, ..., k, then add ε to First(X).

Now define First(α) for any string $\alpha = X_1 X_2 \ldots X_n$ as follows. First(α) contains First(X_1)-{ε}.For each i=2,…,n, if First(X_k) contains ε for all k=1,…,i-1, then First(α) contains First(X_i)-{ε}.Finally, if for all i=1,…,n, First(X_i) contains ε, then First(α) contains ε.

Given a non-terminal A, the set Follow (A), consisting of terminals, and possibly $, is defined as Follows.

1. If A is the start symbol, then $ is in Follow (A).
2. If there is a production B→ αAγ, then First (γ)-{ε} is in Follow (A).
3. If there is a production B→ αAγ such that ε is in First (γ), then Follow (A) contains Follow (B).

3 Graphical Representation

3.1 First and Follow applied to 3 subgrammars

Consider a grammar A → BDE where A, B, D and E are non terminals.

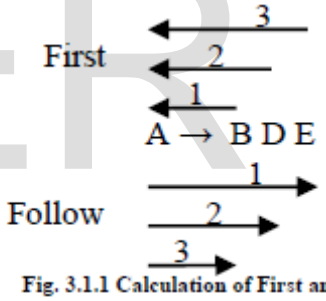


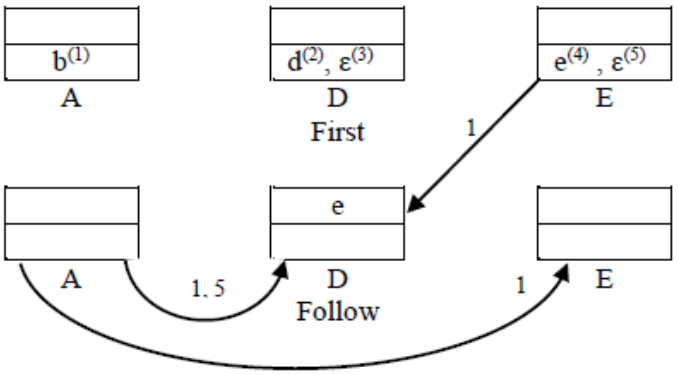
Fig. 3.1.1 Calculation of First and Follow.

Here First goes from right to left and Follow goes from left to right as shown at the top and bottom of the of production rule respectively as shown in Fig. 3.1.1. First(A) ← First (B), First(A) ← First(D) if B is nullable (← may be read as goes to), First(A) ← First(E) if B & C are nullable, Follow (E) ← Follow(A), Follow(D) ← Follow(A) if E is nullable and Follow(B) ← Follow(A) if D & E are nullable.

We now consider 3 cases where either B or D or E become single terminal instead of non terminal.

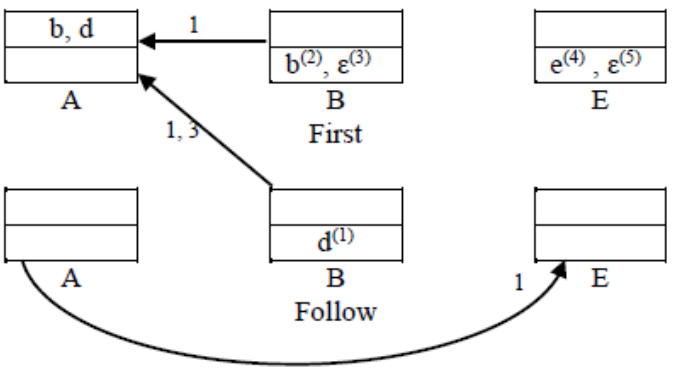
Case 1: B is a single terminal b.

$$A \rightarrow bDE^{(1)}$$
$$D \rightarrow d^{(2)} \mid \varepsilon^{(3)}$$
$$E \rightarrow e^{(4)} \mid \varepsilon^{(5)}$$



	First	Follow
A	b	-
D	d, ε	e
E	e, ε	-

Fig. 3.1.2 Nonterminal B is a single terminal b.



	First	Follow
A	b, d	-
B	b, ε	d
E	e, ε	-

Fig. 3.1.3 Nonterminal D is a single terminal d.

As there are 3 non terminals A, D and E, we draw 3 rectangular boxes for First and 3 rectangular boxes for Follow one below another. The box divided into 2 parts. In the bottom part we are writing the First/Follow of the non terminal that can be directly inferred from the production rules. The corresponding production rule is indicated as a super script at the corresponding terminal.

Some terminals may have to be included due to nullability of non terminals in First/Follow calculations. When a terminal is to be included in First/Follow set, it is included/written at the top and the rules responsible/applicable for it are indicated on the arrow connecting the related non terminals.

Follow(E) ← Follow(A) (rule 1)
Follow(D) ← Follow(A) (rule 1 & 5)
Follow(D) ← First(E) (rule 1)

These 3 inferences are indicated by connecting First and Follow boxes of appropriate non terminals in the directions indicated by ←. After the boxes are connected, the top portions of the boxes are filled using the connectivity of the boxes excluding ε. Now the corresponding First & Follow sets can be filled by looking at Fig. 3.1.2.

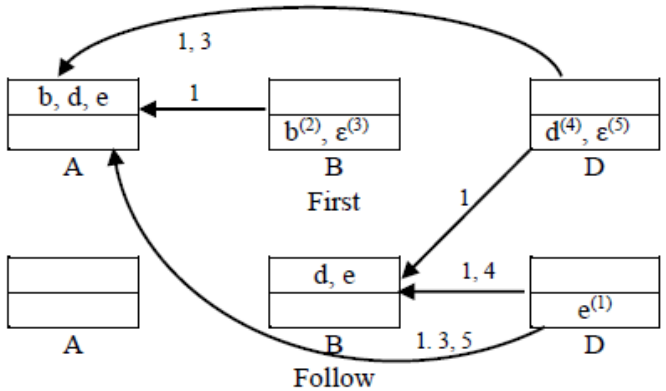
Case 2: D is a single terminal d.

$A \rightarrow BdE^{(1)}$
$B \rightarrow b^{(2)} \mid \varepsilon^{(3)}$
$E \rightarrow e^{(4)} \mid \varepsilon^{(5)}$

First(A) ← First(B) (rule 1)
Follow(E) ← Follow(A) (rule 1)
Follow(B)=d (rule 1)
First(A)=d (rule 1 & 3)
Hence First(A)=d=Follow(B)
This is indicated by connecting Follow(B) to First(A). Now the corresponding First & Follow sets can be filled by looking at the Fig. 3.1.3.

Case 3: E is a single terminal e.

$A \rightarrow BDe^{(1)}$
$B \rightarrow b^{(2)} \mid \varepsilon^{(3)}$
$D \rightarrow d^{(4)} \mid \varepsilon^{(5)}$



	First	Follow
A	b, d, e	-
B	b, ε	d, e
D	d, ε	e

Fig. 3.1.4 Nonterminal E is a single terminal e.

First(A)=e=Follow(D) (rule 1, 3 & 5)
First(A) ← First(B) (rule 1)
First(A) ← First(D) (rule 1 & 3)

Follow(D)=e (rule 1) (bottom of Follow of D)
Hence Follow(D) is connected to First(A).
Follow(B) ← First(D) (rule 1, connect First(D) to Follow(B))
Follow(B)=e (rule 1 & 4)
Now the corresponding First & Follow sets can be filled by looking at the Fig. 3.1.4.

3.2 Applying the graphical method to Expression Grammar

For a grammar to be in LL(1) initially Left Recursion and Left Factoring must be removed.
Consider a grammar A → Aα | β this grammar produces the language $\beta\alpha^n$ (n≥0). Hence to remove Left Recursion the above grammar can be written as

$$A \rightarrow \beta A'$$
$$A' \rightarrow \alpha A' \mid \varepsilon$$

Now consider the expression grammar with + and * operators.

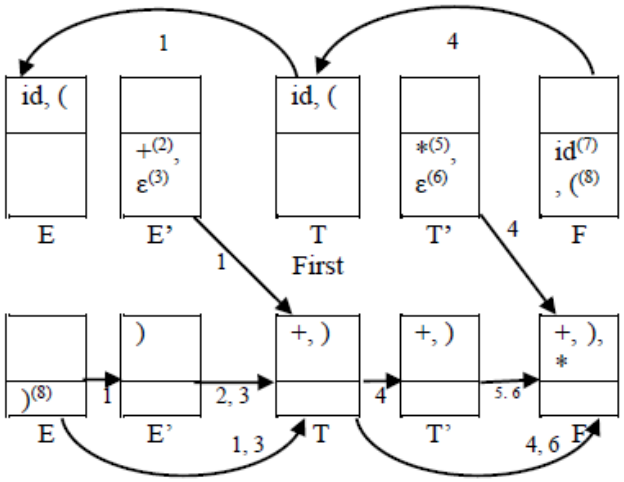
$$E \rightarrow E+T \mid T$$
$$T \rightarrow T*F \mid F$$
$$F \rightarrow id \mid (E)$$

After the elimination of left recursion from the above expression grammar we arrive at

$$E \rightarrow TE'^{(1)}$$
$$E' \rightarrow +TE'^{(2)} \mid \varepsilon^{(3)}$$
$$T \rightarrow FT'^{(4)}$$
$$T' \rightarrow *FT'^{(5)} \mid \varepsilon^{(6)}$$
$$F' \rightarrow id^{(7)} \mid (E)^{(8)}$$

Directly by looking at the rules of the grammar we can calculate the First and Follow sets as
First(E′) ← {+, ε} (rule 2 & 3)
First(T′) ← {*, ε} (rule 5 & 6)
First(F) ← {id, (} (rule 7 & 8)
Follow(E) ← {)} (rule 8)

These are written in the bottom of the boxes of First of E′, T′, F and Follow box of E. Now using the First(E′), First(T′), First(F) and Follow(E) we can calculate the following as indicated by the arrows and graphical method given in section 3.1 can be applied to expression grammar in a similar manner. The obtained results are described in Fig. 3.2.
Follow(E′) ← Follow(E) (rule 1)
Follow(T) ← Follow(E) (rule 1 & 3)
First(E) ← First(T) (rule 1)
Follow(T′) ← Follow(T) (rule 4)
Follow(F) ← Follow(T) (rule 4 & 6)
First(T) ← First(F) (rule 4)
Follow(F) ← Follow(T′) (rule 5 & 6)
Follow(T) ← Follow(E′) (rule 2 & 3)
Follow(T) ← First(E′) (rule 1)
Follow(F) ← First(T′) (rule 4)



	First	Follow
E	id, (	)
E'	+, ε	)
T	id, (	+,)
T'	*, ε	+,)
F	id, (	+,), *

Fig. 3.2 Graphical Calculation of First and Follow for Expression Grammar.

3.3 Understanding/correctness of First and Follows for the Expression Grammar.

First and Follow can be calculated using the rules of the grammar and as per the definition of First and Follow. This is called as Brute Force approach and it is used to verify the proposed graphical method.

Calculation of First

E → TE′ (rule 1)
→ FT′E′ (rule 4)
→ idT′E′ (rule 7)
First(E)={id}

E → TE′ (rule 1)
→ FT′E′ (rule 4)
→ (E)T′E′ (rule 8)
First(E)={(}

E′ →+TE′ (rule 2)
First(E′)={+}

E′ → ε (rule3)
First(E′)={ε}

T → FT′ (rule 4)
→ (E)T′ (rule 8)
First(T)={(}

T → FT′ (rule 4)

→ idT′ (rule 7)
First(T)={id}

T′ → *FT′ (rule 5)
First(T′)={*}

T′ → ε (rule 6)
First(T′)={ε}

F → (E) (rule 8)
First(F)={(}

F → id (rule 7)
First(F)={id}

Calculation of Follow

E → TE′ (rule 1)
→ FT′E′ (rule 4)
→ (E)T′E′ (rule 8)
Follow(E)={)}

E → TE′ (rule 1)
→ FT′E′ (rule 4)
→ (E)T′E′ (rule 8)
→ (TE′)T′E′ (rule 1)
Follow(E′)={)}

E → TE′ (rule 1)
→ T+TE′ (rule 2)
Follow(T)={+}

E → TE′ (rule 1)
→ FT′E′ (rule 4)
→ (E)T′E′ (rule 8)
→ (TE′)T′E′ (rule 1)
→ (T)T′E′ (rule 3)
Follow(T)={)} as E′ is nullable.

E → TE′ (rule 1)
→ FT′E′ (rule 4)
→ FT′+TE′ (rule 2)
Follow(T′)={+}

E → TE′ (rule 1)
→ FT′E′ (rule 4)
→ (E)T′E′ (rule 8)
→ (TE′)T′E′ (rule 1)
→ (FT′E′)T′E′ (rule 4)
→ (FT′)T′E′ (rule 3)
Follow(T′)={)} as E′ is nullable.

E → TE′ (rule 1)
→ FT′E′ (rule 4)
→ (E)T′E′ (rule 8)
→ (TE′)T′E′ (rule 1)
→ (FT′E′)T′E′ (rule 4)
→ (F)T′E′ (rule 3 & 6)
Follow(F)={)} as both E′ and T′ are nullable.

E → TE′ (rule 1)
→ FT′E′ (rule 4)
→ (E)T′E′ (rule 8)
→ (TE′)T′E′ (rule 1)
→ (FT′E′)T′E′ (rule 4)
→ ((E)T′E′)T′E′ (rule 8)
→ ((TE′)T′E′)T′E′ (rule 1)
→ ((FT′E′)T′E′)T′E′ (rule 4)
→ ((FE′)T′E′)T′E′ (rule 6)
→ ((F+TE′)T′E′)T′E′ (rule 2)
Follow(F)={+}.

E → TE′ (rule 1)
→ FT′E′ (rule 4)
→ (E)T′E′ (rule 8)
→ (TE′)T′E′ (rule 1)
→ (FT′E′)T′E′ (rule 4)
→ (F*FT′E′)T′E′ (rule 5)
Follow(F)={*}.

3.4 Extension/study of Expression Grammar for Precendence and Associativity.

Consider the Expression grammar with +, -, *, / operators.

E → E+T | E-T | T
T → T*F | T/F | F
F → id | num | (E) (3.4.1)

Precendence and Associativity of operators play an important role during evaluation of expression.

Case of Precedence: Consider the expression 2+3*4. Using the grammar 3.4.1

2+3*4 = num + num * num
= F + F * F
= F + T * F
= F + T
= T + T
= E + T
= E

So 2+3*4 becomes 2+12=14 same as normal arithmetic.

Now consider the expression grammar as

E → E*T | T
T → T+F | F
F → id | num | (E) (3.4.2)

Consider the expression 2+3*4. Using the grammar 3.4.2

2+3*4 = num + num * num
= F + F * F
= T + F * T
= T * T
= E * T
= E

So 2+3*4 becomes 5*4=20 which is not correct. Hence least precedence operator must be at the starting point of the grammar.

Case of Associativity: Consider the expression 3-5-8. Using the grammar 3.4.1

3-5-8 = num - num - num
= F - F - F
= T - T - T
= E - T - T
= E - T
= E

So 3-5-8 evaluates to -2-8=-10 same as normal arithmetic.

Now consider the expression grammar as

E → T+E | T-E | T
T → T*F | T/F | F
F → id | num | (E) (3.4.3)

Consider the expression 3-5-8. Using the grammar 3.4.3

3-5-8 = num - num - num
= F - F - F
= T - T - T
= T - T - E
= T - E
= E

So 3-5-8 evaluates to 3-(-3)=6 which is not correct. The same applies to division operator. These points are well explained in [4]. We have given here for completeness.

4 CONCLUSION

A graphical method has been given to calculate First and Follow sets of a grammar which is easier to apply for a learner or a student. The method presented in this paper also gives a clear picture on understanding meanings of First and Follow by following the derivation of First and Follow from production rules of the grammars. This graphical method is applied to the Expression Grammar to find the First and Follow sets and we also check its correctness with the help of derivations. This paper also describes the role of precedence and associativity during the evaluation of expressions.

References

[1] Alfred V.Aho, Monica S.Lam, Ravi Sethi, Jeffrey D.Ullman, Compilers: Principles, Techniques, and Tools (2nd Edition), New Jersey: Addison Wesley, 2007.

[2] Yuqiang Sun. Design of Parallel Algorithm for FIRST and FOLLOW Sets. Computer Engineering, 2004: pp.71-73.

[3] Arturo Trujillo, Computing First and Follow functions for feature-theoretic grammars, The 15th International Conference on Computa tional Linguistics,1994: pp.875-880.

[4] Kenneth C. Louden , Compiler Construction Principles and Practice, New Jersey :Addison Wesley, 2007.

Authors Profile:

Dr. A S R Murty was born in Andhra Pradesh, India. He received his B. Tech. Degree in Electrical Engineering from JNT University, Hyderabad in the year of 1976. He has received his M.Tech. Degree in Electrical Engineering from IIT, Kanpur in the year of 1978. He has obtained his Ph.D. degree in Electrical Engineering from IIT, Madras in the year of 1997. He has worked from scientist V to Joint Director in Central Power Research Institute, Bengaluru for 21 years and currently working as Professor in Dept. of CSE, SVCE, Bengaluru. His areas of interest include scientific and engineering calculations, Power systems Analysis and simulation, Control systems and stability studies, Programming Languages as a user.
E Mail: asr_murty2001@yahoo.co.in

Maria Navin J R was born in Karnataka, India. He received his B. E. Degree in Information Science & Engineering from VTU, Belgaum in the year 2004. He received his ME degree in Computer Science & Engineering from UVCE, Bangalore University, Bangalore in the year 2011. He is presently pursuing his Ph. D. under VTU, Belgaum and working as Assistant Professor in Dept. of CSE, SVCE, Bengaluru. His research interest includes Mobile Cloud Computing, Cryptography and Theoretical Foundations of Computer Science.
E Mail: marianavin.jr@gmail.com