# Report: Calculating First and Follow sets of given Grammar


| Adesh Ramgude | Mayur Jadhav | Pranit Jadhao | Samihan Inamdar |
|---|---|---|---|
| E-57 | E-43 | E-27 | E-24 |
| GR-182059 | GR-182151 | GR-1710283 | GR-1710892 |


---

**Problem Statement:**

To calculate the FIRST and FOLLOW sets of the given context-free grammar.

**Objectives:**

Understanding the parsing of the grammar and also how it helps in syntax analysis.

**Introduction:**

The need for efficient parsing is a constant one in Natural Language Processing. With the advent of feature-theoretic grammars, many of the optimization techniques that are applicable to Context-Free (CF) grammars have required modification. For instance, a number of algorithms used to extract parsing tables from CF grammars have involved discarding information that otherwise would have constrained the parsing process, Briscoe and Carroll (1993). This paper describes an extension to an algorithm that operates rover CF grammar to make it applicable to feature-theoretic ones. One advantage of the extended algorithm is that it preserves as much of the information into grammar as possible.

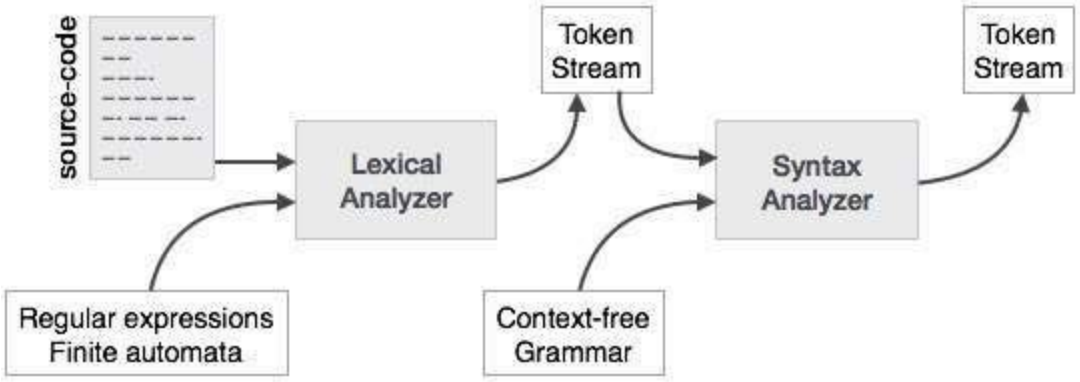
In order to make more efficient parsers, it is sometimes necessary to preprocess (compile) a grammar to extract from top-down information to guide the search during analysis. The first step in the preprocessing stage of several compilation algorithms requires the solution of two functions normally called FIRST and FOLLOW.

**Literature Review:**

The extension to the LR algorithm presented by Nakazawa (1991) uses a similar approach to that described here; the functions involved, However, are those necessary for the construction of an LR parsing table (i.e. the GOTO and ACTION functions). One technical difference between the two approaches is that he uses positive restrictors (Shieber 1985) instead of negative ones. In addition, both of his algorithms also differ in another way from the algorithm described here. The difference is that they add items to a set using simple set addition whereas in the algorithm we add element using the operator +<=. Furthermore, when computing

the closure of a set of items, both of the algorithms there ignore the Effect that unification has on getting categories in the rules.

**Architecture diagram:**




**Parsing Algorithm:**

for each symbol X
    FIRST[X] := { }, FOLLOW[X] := { }, nullable[X] := false
For each terminal symbol t
    FIRST[t] := {t}
repeat
    for each production X → Y1 Y2 … Yk,
        if all Yi is nullable then
            nullable[X]:=true nullable[X] : true
        if Y1..Yi-1 are nullable then
            FIRST[X] := FIRST[X] U FIRST[Yi]
        if Yi+1..Yk are all nullable then
            FOLLOW[Yi] := FOLLOW[Yi] U FOLLOW[X]
        if Yi+1..Yj-1 are all nullable then
            FOLLOW[Yi] := FOLLOW[Yi] U FIRST[Yj]
Until FIRST, FOLLOW, nullable do not change

**Hardware and software requirements:**

**Hardware:**

An Intel 80386 or higher processor.
8 Megabytes of free memory.
At least 80 MB free disk space.

**Software:**

Linux:

1. GNUas, the GNUassembler.
2. GNUld, the GNUlinker.

Windows:

mingw32 or cygwin

**Advantages:**

1. FOLLOW can be applied to a single non-terminal only, and returns a set of terminals.
2. FIRST and FOLLOW help us to pick a rule when we have a choice between two or more r.h.s. by predicting the first symbol that each r.h.s. can derive.
3. Even if there are only one r.h.s. we can still use them to tell us whether or not we have an error - if the current input symbol cannot be derived from the only r.h.s. available, then we know immediately that the sentence does not belong to the grammar, without having to (attempt to) finish the parse.
4. Backtracking is not needed to get the correct syntax tree.

**Applications:**

In compiler design, first and follow sets are needed by the parser to properly apply the needed production.

Backtracking is not needed to get the correct syntax tree.

**Summary and Conclusion:**

The conclusions are, we need to find FIRST and FOLLOW sets for a given grammar so that the parser can properly apply the needed rule at the correct position.

We have extended an algorithm that manipulates CF grammars to allow it to handle feature - theoretic ones. It was shown how most of the information contained in the grammar rules may be preserved using a set of pairs as the value of a function and by using the notion of subsumption to update this set.

**References:**


1. Nakazawa, Tsuneko. "An extended LR parsing algorithm for grammars using feature-based syntactic categories." *Fifth Conference of the European Chapter of the Association for Computational Linguistics*. 1991.
2. Shieber, Stuart M. "Using restriction to extend parsing algorithms for complex-feature-based formalisms." *Proceedings of the 23rd annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 1985.

3. Briscoe, Ted, and John Carroll. "Generalized probabilistic LR parsing of natural language (corpora) with unification-based grammars." *Computational linguistics* 19.1 (1993): 25-59.
4. Trujillo, Arturo. "Computing first and follow functions for feature theoretic grammars." *Proceedings of the 15th conference on Computational linguistics-Volume 2*. Association for Computational Linguistics, 1994.