FIRST and FOLLOW sets

To compute for all grammar symbols *X*, apply the following rules until no more terminals or can be added to any FIRST set.

1. If *X* is a terminal, then
2. If *X* is a nonterminal, and is a production rule, then
   * everything in is in
   * if contains, then everything in is also in
   * repeat for , and so on…
3. If , then add to

To compute for all nonterminals *A*, apply the following rules until nothing can be added to any FOLLOW set.

1. Place $ in , where *S* is the start symbol, and $ is the input right end-marker (i.e. end of the program code).
2. If there is a production rule , then everything in except is in
3. If there is a production rule , or a production where contains , then everything in is in

Given our LL grammar:

Compute the FIRST sets:

Compute the FOLLOW sets:

Generate the predictive parsing table, , (for LL grammars) from the FIRST/FOLLOW sets.

For each production rule of the grammar, do the following:

1. For each terminal in , add to
2. If is in , then for each terminal *b* in , add to .
3. Every other entry in the parsing table, , implicitly generates an **error**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Non-Terminal | Input Symbol | | | | | | | |
| id | + | - | \* | / | ( | ) | $ |
| *S* |  |  |  |  |  |  |  |  |
| *E* |  |  |  |  |  |  |  |  |
| *E’* |  |  |  |  |  |  |  |  |
| *T* |  |  |  |  |  |  |  |  |
| *T’* |  |  |  |  |  |  |  |  |
| *F* |  |  |  |  |  |  |  |  |