**A Simple One-Pass Compiler:**

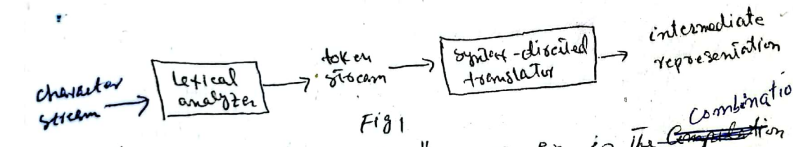
**Programming Language** = Syntax + Semantics

**Context-Free Grammar:**

* It is a notation to specify the syntax of language
* Also known as **Bakus Naur Form**

**Syntax-directed translation:**

* A grammar oriented compiling technique
* Helpful for organizing Compiler frontend



**Syntax Definition:**

* a notation to specify the syntax of language.

Notation for if else statement:

if (expression) statement else statement

It can also be expressed as:

stmt -> if (expr) stmt else stmt

Above rule is also called **production**.

In above production, lexical elements like if keyword and parenthesis are **tokens / terminal** and variables like expr and stmt are called **non-terminals**.

* **Components of Context free grammar:**

1. Set of tokens (terminals)

**E.g.,** keywords

1. Set of non-terminals

**E.g.,** variables

1. Set of Productions

Non-terminals -> left-side of production

Terminals -> right-side of production

1. Designation of one of non-terminal as the start symbol

* List of digits separated by plus or minus signs
* Productions:

List -> list + digit

List -> list – digit

List -> digit

Digit -> 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Above productions can be represented as:

List -> list + digit | list – digit | list

The tokens of the grammars are the symbols:

* + 1 2 3 4 5 6 7 8 9
* **Empty string** of tokens **represented** by **E** abs lawn
* A **grammar** derives strings by replacing non-terminals with the right-side of the production
* The expression can be represented by a parse tree using defined productions of grammar.

Each node in the tree is grammar symbol.

Each interior node and its children correspond to a production. Interior node corresponds to left-side of production and children of node corresponds to right-side of production.

**Parse Tree:**

* Shows how the start symbol of a grammar derives in the language.

**Language generated by grammar:**

* The set of strings that can be generated by same parse tree.

**Parsing:**

* The process of finding a parse tree from a given string of tokens

**Ambiguity:**

* More than one parse trees from a single string of tokens cause ambiguity

**Associativity of operators:**

* Evaluation will be takes place from left to right
* +, -, \*, / is left associative
* Exponents and = is right associative.