Ope
17/sci01/051
Nasir Firdaus Opeyemi
CSC 312

1. **GRAMMAR**:

A grammar in compiler construction usually consists of at least two parts and sometimes three, less often one. I will start with the most common case first. The grammar has two parts, a lexical (lexer) specification, and a syntactic (parser) specification.

A formal grammar is a set of rules for rewriting strings, along with a "start symbol" from which rewriting starts. Therefore, a grammar is usually thought of as a

language generator. However, it can also sometimes be used as the basis for a “ recognizer"ma function in computing that determines whether a given string belongs to the language or is grammatically incorrect

I.Derivation:

A derivation is basically a sequence of production rules, in order to get the input string. During parsing, we take two decisions for some sentential form of input:

- Deciding the non-terminal which is to be replaced.
- Deciding the

production rule, by which, the non-terminal will be replaced.

To decide which non-terminal to be replaced with production rule, we can have two options.

Left-most Derivation

If the sentential form of an input is scanned and replaced from left to right, it is called left-most derivation. The sentential form derived by the left-most derivation is called the left-sentential form.

Right-most Derivation

If we scan and replace the input with production rules, from right to left, it is known as right-most derivation. The sentential form derived from the right-most derivation is called the right-

sentential form.

ii. PRODUCTION:

A production or production rule in computer science is a rewrite rule specifying a symbol substitution that can be recursively performed to generate new symbol sequences. A finite set of productions
P
{\displaystyle P}
 is the main component in the specification of a formal grammar (specifically a generative grammar). The other components are a finite set
N
{\displaystyle N}
 of nonterminal symbols, a finite set (known as an alphabet)

Σ
{\displaystyle \Sigma }
 of terminal symbols that is disjoint from
N
{\displaystyle N}
 and a distinguished symbol
S
∈
N
{\displaystyle S\in N}
 that is the start symbol.

iii. SENTENCE:

A sentence is a sentential form consisting only of terminals such as a + a * a. A sentence can be derived using the following algorithm: Algorithm Derive String String := Start Symbol REPEAT Choose any nonterminal in String. Find a production with this

nonterminal on the left-hand side.
A sentence is a sentential form that has only terminal symbols. A sentence form is every string of symbols in the derivation.

iv. NULL SYMBOL:
Null is both a value and a pointer. Null is a built-in constant that has a value of zero. It is the same as the character 0 used to terminate strings in C.

Null can also be the value of a pointer, which is the same as zero unless the CPU supports a special bit pattern for a null pointer.Null is the value of reference variable.

The null or empty string is denoted with ε or

sometimes Λ or λ. The empty string should not be confused with the empty language ∅, which is a formal language (i.e. a set of strings) that contains no strings, not even the empty string. The empty string has several properties: $|\varepsilon| = 0$.

In formal language theory, the empty string, or empty word is the unique string of length zero.