# ASSIGNMENT NO 4

Title : Implementation of Semantic Analysis Operations.

Problem Statement : Implement Semantic Analysis Operations (like type checking, verification of function parameters, variable declarations & corrections).

Objective :

Semantic analysis is the task of ensuring that the declarations & statements of a program are semantically correct, i.e. that their meaning is clear and consistent with the way in which control structures & data types are supposed to be used.

Outcome :

The students will be able to :

- Construct meaningful sentence in programming & natural language.
- Able to understand how semantic analysis phase of compiler behaves.

S/w package :- 64 bit Open Source Linux.
Eclipse IDE.
Java
i3 & i5 machines.

Theory :

Semantics

Semantics of a language provide meaning to its constructs, like tokens & syntax structure. Semantics help interpret symbols, their types & their relations with each other. Semantic analysis judges whether the syntax structure constructed in the course program derives any meaning or not.

CFG + Semantic rules = Syntax Directed Definitions.

For eg:

int a="value";

should not issue an error in lexical & syntax analysis phase, as it is lexically & structurally correct, but it should generate a semantic error as the type of the assignment differs. These rules are set by grammar of the language & evaluated in semantic analysis.

- Scope resolution.
- Type Checking.
- Array-bound Checking.

Semantic Errors :

We have mentioned some of the semantics error that the semantic analyzer is expected to recognize:

- Type mismatch.
- Undeclared variable.
- Reserved identifier misuse.
- Multiple declaration of variable in a scope.

- Accessing an out of scope variable.
- Actual & formal parameter mismatch.

Attribute Grammar:

Attribute grammar is a special form of context-free grammar where some additional information (attributes) are appended to one or more of its non-terminals in order to provide context-sensitive information. Each attribute has well defined domain of values, such as integer, float, character, string & expressions.

Attribute grammar is a reduction medium to provide semantics to the context-free grammar & it can help specify the syntax & semantics of a programming language. Attribute grammar can pass values or information among the nodes of a tree.

Eg:-

$E \rightarrow E + T \{ E.value = E.value + T.value \}$

Synthesized Attributes:-

These attributes get values from the attribute values of their child nodes. To illustrate assume the following production

$S \rightarrow ABC$.

If S is taking values from its child nodes (A, B, C) then it is said to be synthesized attribute, as the values

of A,B,C then it is said to be a synthesized attribute, as the values of ABC are synthesized to S.

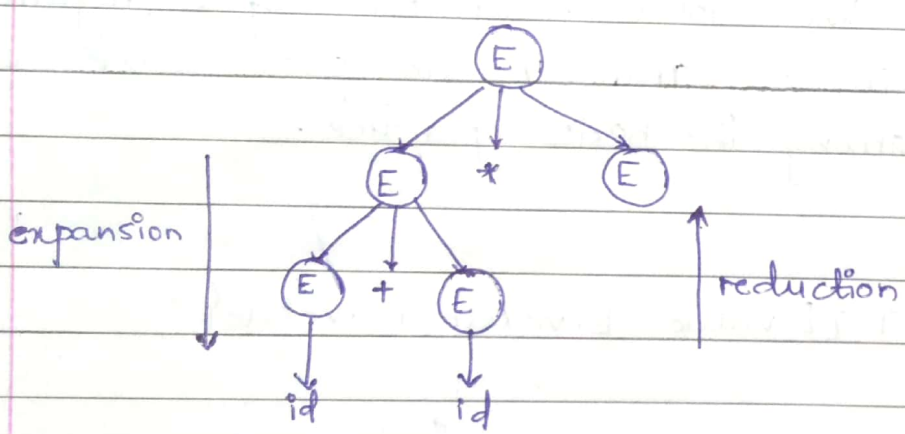
Inherited attributes

In contrast to synthesized attributes, inherited attributes can take values from parent or siblings. As in the foll° production

$S \rightarrow ABC$.

Expansion:

When a non-terminal is expanded to terminals as per a grammatical rule.



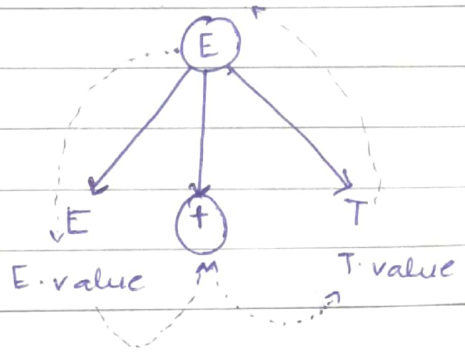

Reduction :

When a terminal is reduced to its corresponding non-terminal according to grammar rules. Syntax trees are parsed top-down & left right. Whenever reduction occurs, we appl its corresponding semantic rules.

S-attributed SDT

If an SDT uses only synthesized attributes, it is called as S-attributed SDT. These attributes are evaluated using S-attributed SDTs that have their semantic actions written after the production.

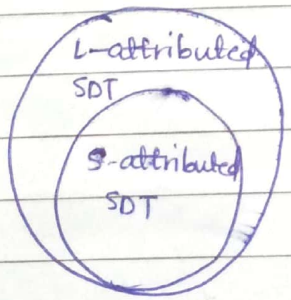
E . value = E. value + T.value .




L-attributed SDT

These form of SDT uses both synthesized & inherited attributes with restriction of not taking values from right sibling.

S→ABC.




Conclusion:

Hence, I have implemented Semantic Analysis Operations.