A Brief Overview of Semantic Analysis

- Semantic analysis helps evaluate language constructs, so that the source code is progressed toward executable action sequences or target code.
- Firstly, semantics help correct 'simple errors', that is, interpret tokens, their types, and their relations with each other.
- Secondly, semantics associate information with recognized constructs for finding the meaning of them.

- **Simple semantic errors**
  - Type mismatch
  - Undeclared variable
  - Reserved word misuse
  - Multiple declaration of variable in a scope
  - Accessing an out-of-scope variable
  - Actual and formal parameter mismatch

    and so on.

- **Correction of simple semantic errors**
  - Most of the above semantic errors are detected and corrected during the earlier phases of compilation, that is, lexical analysis and syntactic analysis.
  - Construction and use of symbol tables during lexical analysis and syntactic analysis are the right points where correction of simple semantic errors may be arranged with a very little effort.

- **Syntax Directed Translation**
  - Parse trees generated by CFGs during syntax analysis lack information of how to evaluate a construct that has been recognized.
  - To have interpreted a parse tree, that is, to find meaning of the construct, semantic rules need to be associated with the productions of the CFGs that generate it.
  - CFG + semantic rules = Syntax Directed Definitions (SDD).
  - Syntax Directed Definitions during parsing thus help do useful things toward code generation. The process is known as *Syntax Directed Translation (SDT)*.

- Two ways to do Syntax Directed Translation:
  - Top Down
  - Bottom UP

➢ Attribute Grammar

- A set of attributes is associated with each grammar symbol.
- Some of the attributes are called *synthesized*, and some are called *inherited*.
- Actions written corresponding to a production manipulate these attributes effectively to do the desired translation.
- The parse tree with attributes is called an *annotated parse tree*.

➢ Synthesized attributes

These attributes get values from the attribute values of their child nodes. In the production

S → ABC

if S is taking values from its child nodes *A, B, C*, then it is said to be a synthesized attribute.

➢ Inherited attributes

In contrast to synthesized attributes, inherited attributes can take values from parent and/or siblings. In the production,

S → ABC

A can get values from S, B and C. B can take values from S, A, and C. Likewise, C can take values from S, A, and B.

Example of annotating a parse tree

Evaluating arithmetic expression using SDT:

Consider the following grammar:

E → E+T | E-T | T

T → num

Let us assume an attribute, val, with E, T and F. Following are the grammar rules with semantic actions.

Production	Semantic rule
$E \rightarrow E_1 + T$	$\{E.val = E_{1.val} \,\|\|\, '+' \,\|\| T.val\}$
$E \rightarrow E_1 - T$	$\{E.val = E_{1.val} \,\|\|\, '-' \,\|\| T.val\}$
$E \rightarrow T$	$\{E.val = T.val\}$
$T \rightarrow num$	$\{T.val = num\}$

The operator '||' represents concatenation, *num* represents any number constant and E_1, T_1 are used to distinguish two Es and Ts.

Consider the problem of generating postfix equivalents of infix expressions with the following grammar.

$E \rightarrow E+T \mid E-T \mid T$

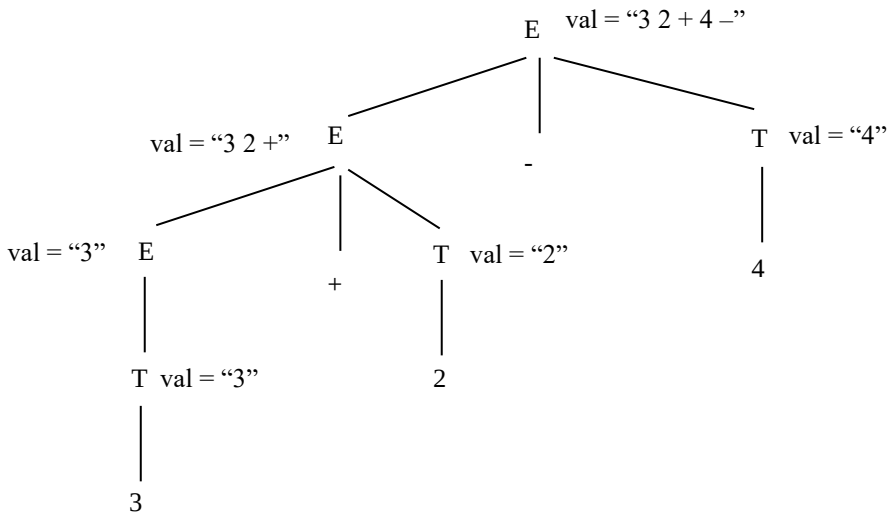
$T \rightarrow num$

Let us assume an attribute, val, with E and T that holds the string corresponding to the postfix expression. The semantic actions can be written as,

Production	**Semantic rule**
$E \rightarrow E_1+T$	$\{E.val = E_{1.val} \,\|\|\, T.val \,\|\| \text{'+'}\}$
$E \rightarrow E_1-T$	$\{E.val = E_{1.val} \,\|\|\, T.val \,\|\| \text{'-'}\}$
$E \rightarrow T$	$\{E.val = T.val\}$
$T \rightarrow num$	$\{T.val = num\}$

The operator '||' represents concatenation, *num* represents any number constant and E_1 is used to distinguish two Es.

The **annotated parse tree** for the expression "3+2-4" is given below, which generates the postfix string "3 2 + 4 –".



The parse trees produced by this stage will be used in later stages for code generation and other related activities.