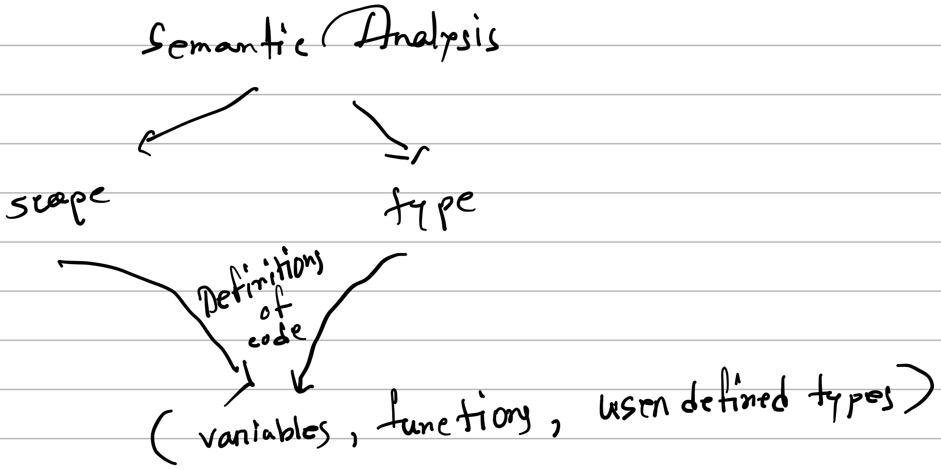

Semantic Analysis

scope

type

Definitions of code

(variables , functions , user defined types)

Thege informations are used for findin meanings of code blocks, predominantly expressions in our program.

* Semantic analysis is done concurrently with syntax analysis and using an intermediate form of the program source code.

intermediate form: abstract syntax Tree (AST) } This is not a direct product of syntax analysis, we have to do it for doing semantic analysis.

A grammer accepts a program

=

There is a derivation from the start symbol to the sequence of tokens representing the program.

=

There is a parse tree representing the program.

AST is an augmented parse tree.

∴ We have to construct a parse tree, when we are doing syntax analysis so that we can augment it later.

~~*~~ Along with the production rules of the grammer, write semantic rules in the form of small code blocks so that a parse tree can be constructed during syntax analysis.

↗ syntax Directed Definitions (SDD)

Production Rules	Semantic Rules

$E \to E_1 + T$	E.node = newNode (+)
	E.node.children = [E₁.node , T.node]

$E \to T$	E.node = T.node

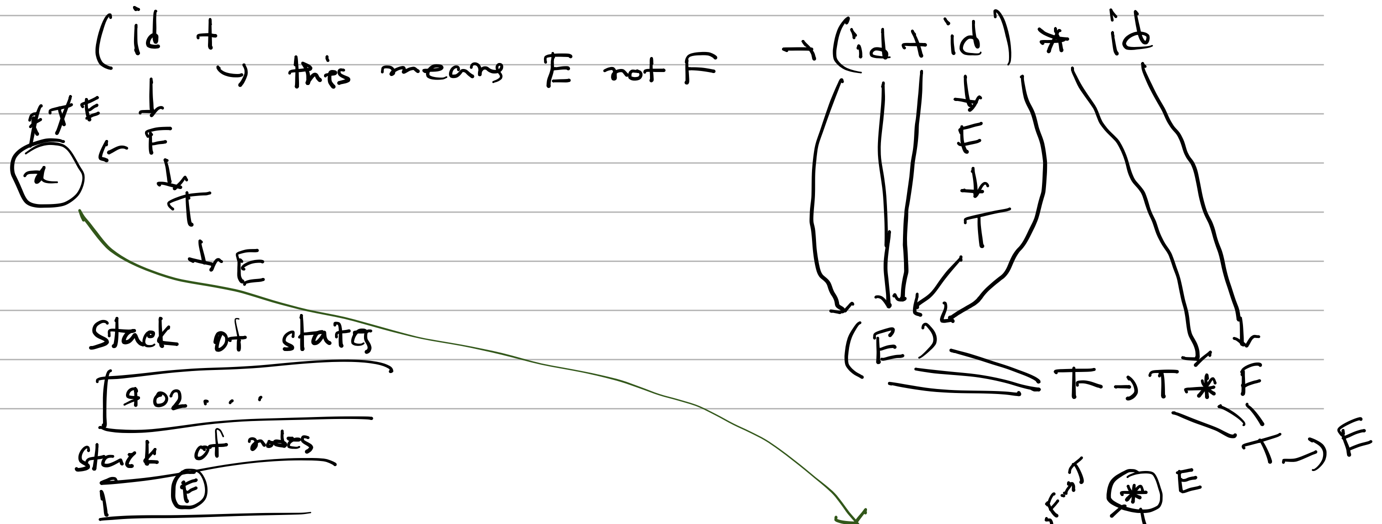
$T \to T_1 * F$	T.node = newNode (*)
	T.node.children = [T₁.node , F.node]

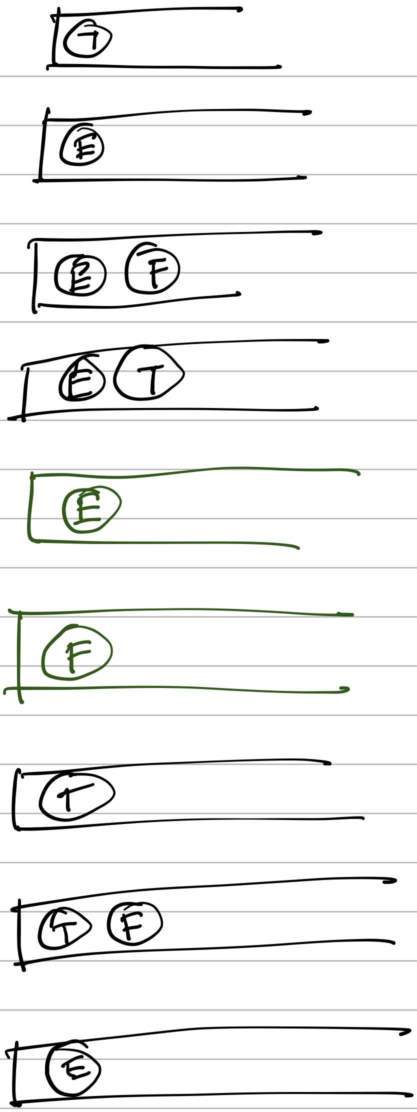
$T \to F$	T.node = F.node
$F \to id$	F.node = newLeafNode (id.lexeme)
$F \to (E)$	F.node = E.node

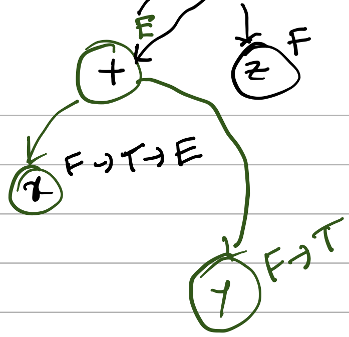
E1 since we have to know which E

↳ we dont need to know there is parenthesis.

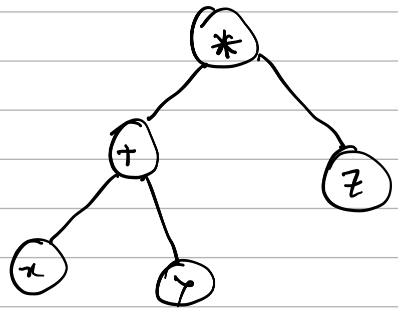
(x+y) * z → (id + id)*id







∴ Syntax tree :



Abstract syntax tree is when we add more informations.

This construction is called Syntax Directed Translation. (SDT)

Byson Format:

$$\begin{array}{cccc} E \rightarrow & E, & + & T \\ \downarrow & \downarrow & \downarrow & \downarrow \\ \$\$ & \$1 & \$2 & \$3 \end{array}$$ ← positional identifiers

```
$$.node = newNode(+)
$$.node.children = [$1.node, $3.node]
```

* attributes of nodes represents terminals and non terminals of the grammar. Since all informations are represented as attributes that is why Production Rules & Semantic Rules are called Attribute Grammar.

* Attributes of syntax tree nodes.

  - → can be inherited from the parent node
  - → can be synthesized from the child node to the parent.

* Inherited attributes are done after the construction of syntax tree since we construct the syntax tree in Bottom-up approach.

  ∴ synthesized attribute can be done before.