Reading:
Chapter 6 from Principles of Compiler Design, Alfred V. Aho & Jeffrey D Ullman

Objectives:
1. To understand in detail about Recursive Evaluators, Space for attribute values at compile time, Assigning spaces at compiler-construction time.
2. To understand concepts of Type Checking.

Concepts:
1. Recursive Evaluators
2. Space for attribute values at compile time
3. Assigning spaces at compiler-construction time
4. Analysis of syntax directed analysis
5. Type systems
6. Specification of a simple type checker

Outlines:
1. Syntax directed Translation

   a. Recursive Evaluators
      Explanation of recursive functions that evaluate attributes as they traverse a parse tree which are constructed from syntax directed definition.
      
         o Left to right traversals – Explains how the grammar can be implemented in a recursive manner and the traversing is done in left to right side.
         o Other traversals – Illustrating the power of using manually recursive functions for evaluating the attributes at the node of a parse tree.
b. Space for attribute values at compile time

Consider the compile-time assignment of space for attribute values.

- Assigning space for attributes at compile time – Explains how the space is allocated for attributes at the compile time.
- Avoiding copies – Discussing the different copy rules.

c. Assigning spaces at compiler-construction time

Making copies by using multiple stacks which allows holding all attributes in a single stack.

- Predicting lifetime for grammar – Predicting the lifetime of attributes at compiler time construction.
- Non overlapping lifetimes – Explaining how the lifetime of two attributes do not overlap if their values can be held in a single register.

d. Analysis of syntax directed analysis

- Recursive evaluation of attributes – Explains the overview of how attributes are evaluated recursively.
- Strongly noncircular syntax-directed definitions – Discussing the construction of recursive evaluators for a class of syntax directed definitions.
- A circularity test – Computing whether any cycles are there in the directed graph for syntax-directed definitions.

2. Type checking

Compiler checking the source program for both syntactic and semantic conventions of the source program.
a. Type systems

- Type expressions – Representing the type of language construct using type expressions. Discussing the different definitions of type expressions.
- Type systems – Explaining type systems which is a rule for assigning type expressions to various parts of a program.
- Static and dynamic checking of types – Overview of different types of checking which are static and dynamic.
- Error recovery – Discussing various error recovery procedures done by compiler.

b. Specification of a simple type checker

- A simple language – Discussing how type checker works by taking a simple example for a simple language construct.
- Type checking expressions – How type checking is performed in the expressions.
- Type checking of statements – Statements considered are assignment, conditional and while statements. Here we will discuss how type checking is done in these statements.
- Type checking of functions – Discussing how type checking is done in functions