Type Checking in Compiler Design

Type Systems

- A type system is a collection of rules that assign types to program constructs (more constraints added to checking the validity of the programs, violation of such constraints indicate errors).
- A languages type system specifies which operations are valid for which types.
- Type systems provide a concise formalization of the semantic checking rules.
- Type rules are defined on the structure of expressions.
- Type rules are language specific.

Type Expressions

- A type expression is either a basic type or is formed by applying an operator called a type constructor to a type expression. The sets of basic types and constructors depend on the language to be checked.

The following are some of type expressions:

- A basic type is a type expression. Typical basic types for a language include boolean, char, integer, float, and void(the absence of a value). type_error is a special basic type.
- A type constructor applied to type expressions. Constructors include:
    - **Arrays :** If T is a type expression, then array(I, T) is a type expression denoting the type of an array with elements of type T and index set I. I is often a range of integers. Ex. int a[25] ;

- **Products :** If T1 and T2 are type expressions, then their Cartesian product T1 x T2 is a type expression. x associates to the left and that it has higher precedence. Products are introduced for completeness; they can be used to represent a list or tuple of types (e.g., for function parameters).
- **Records :** A record is a data structure with named fields. A type expression can be formed by applying the record type constructor to the field names and their types.
- **Pointers :** If T is a type expression, then pointer (T) is a type expression denoting the type "pointer to an object of type T". For example: int a; int *p=&a;
- **Functions :** Mathematically, a function maps depends on one set (domain) to another set(range). Function F : D -> R.A type expression can be formed by using the type constructor -> for function types. We write s -> t for "function from types to type t".

For More Details Click Here:

https://www.wikitechy.com/tutorials/compiler-design/type-checking-in-compiler-design

