Analysis

The task of analysis has two major components, type and name analysis. Where name analysis determines the scopes as well as the defining and applied occurrences, and the type analysis assures compile time type checking. Together they form the core of compile time checks that can be run on an input. In eli this is done using computational roles; these are common tree computations that are built into libraries that can be included in a specification.

Of these two tasks there is the leaf dependances which are constants and variables, and the constructs that combine them to form statements. From this it is clear that the leaf dependancies must be computed first, and of the two tasks name analysis must come first since type analysis depends on certain aspects of it. Once the scopes of the symbols have been assigned, in this case everything uses Algol scope, we must then look at predefined identifiers. These are identifiers where you specify their parsed names and associate types and their associations to computations; the latter of which is type analysis.

Once this is done, type analysis can then be considered. The basic roles are much like name analyses basic roles in that there is little interdependance, it is just necessary to define the properties of the leaf nodes. This involves setting a set of contexts where types are set in various nodes where the type can be explicitly set. Once this is done we can then check for the appropriate types at parent nodes in the tree.

Next constructs like function calls, arrays, and inheritance must be checked, using a combination of type and name analysis, type to assure proper variables are used, and name to assure the proper variables exist in that scope. This is the most difficult task, since it involves interdependence of variables.

This has been a rough roadmap of how the analysis section of this project was approached. To review what I did; I started with the most basic definitions, ones without dependence that could easily be added sequentially for debugging purposes, name analysis first, then type. Afterwards I worked on the constructs. These steps were not really by choice, rather the latter steps depended on the completeness of the former.

In terms of error reporting, I felt that the built in error messages, as well as the demo messages provided within the tutorials provided enough information that new ones were not required. Also the file im turning in tonight is not completely functional, as there are some pieces of type analysis that are not yet working. Most of it however, appears to be functioning properly.