P2: Semantic Analysis

Compiler Construction
2013-2014

In this second assignment you will implement semantic analysis for the SPL language. This implementation is based on the abstract syntax tree produced by the parser from the first assignment. The semantic analysis should at least contain:

Binding time analysis Here you should check whether all applied occurrences of identifiers are associated with a proper definition. There is no formal definition of SPL, so you have to make your own decisions. The given examples and grammar provide hints for the decisions to be made.

About the order of function definition and use: should all functions be defined before use (like in C), or is any order allowed? When you require definition before use, you probably need some additional syntax to allow mutual recursive functions.

You should decide whether you allow higher order functions, i.e. can arguments be used as functions, can function results be functions, can functions be partially parameterized. It is sufficient to use SPL as a first order language (no functions as arguments or result of other functions).

There are three kinds of variables in SPL: **1)** the global variables defined in the program; **2)** the function arguments that can be used as variables within the body of that function (like in any other imperative language); **3)** the local variables that can be defined at the beginning of each function body.

Type check In the lectures we provided typing rules for expressions. Design and implement typing rules for SPL.

Producing fancy error messages is not required, just the indication that there is a problem and hence compilation is stopped is sufficient.

When submitting your solution, please include the following:

- A document containing:
  - The grammar used to parse SPL, if it is different from the given grammar.
  - The scoping rules you use and check in your compiler.
  - The typing rules used for SPL.
  - A brief guide telling what each of the examples tests.
  - A concise but precise description of how the work was divided amongst the members of the team.
- At least 10 nontrivial example programs that sufficiently validate your implementation. Give these programs in separate plain-text `.spl` files, as well as the compilation results for those programs. For correct programs this is a pretty printed version of the program augmented with type information. These results should indicate that the binding power and direction of infix-operators is correct, and augments type information. The examples should also contain examples of incorrect SPL programs showing that binding time and type problems are indicated.
- We will provide a set of test programs based on the programs handed in for exercise 1. Your document should contain the compilation results for those programs.

On April 15 you have to give a brief presentation about your semantic analysis, before April 18 at 24:00 you have submit your solution on Blackboard.