Milestone 2

Guillaume Labranche William Bain Si Mei Zhang

March 6, 2015

Contents

1 Design decisions

1.1 Typechecker

We implemented the typechecker as a subclass of the `DepthFirstAdapter` class provided by SableCC. This made it easy to traverse the AST without extra boilerplate. Since we did not have a reliable way of editing the extending the AST classes generated by SableCC, we stored type information for the AST in a hash table mapping from nodes to Go types.[1]

In most cases it was sufficient to apply typechecks after the the typechecker had recursed over the child nodes and typechecked them. There were two cases where more fine-grained control was necessary:

- In short assignment statements, the list of variables is implemented as a list of expressions for reasons having to do with the parser implementation. It was therefore necessary to stop the typechecker from typechecking the variables before they had been added to the symbol table.
- To typecheck function declarations, it was necessary to open a scope after having typechecked some but not all of the child nodes.

In each of these special cases we overrode the `case` method which controls the recursion over the child nodes.

A further implementation detail that is worth noting is our implementation of struct type declarations. To prevent code duplication, we treat struct fields as if they were variable declarations: we open a scope in the symbol table upon entering the struct and enter each field as if it were a variable. Then we pop the scope and use it to build the struct class.

1.2 Type and symbol class hierarchy

TODO (put the UML diagram here)

[1] By extending I mean adding methods and properties to a class used by the parser (here probably `PExp`, the abstract expression production class), not just creating a subclass.

1.3 Symbol table

TODO

1.4 Typed pretty printer

We extended the `PrettyPrinter` class we implemented in Milestone 1 to have it print extra type information. By overriding the `defaultOut` method, it was simple to have it look up and output the type of any node which is an instance of the `PExp` (expression production) class as an inline comment.

2 Scoping rules

TODO

3 Type check operations

TODO

4 Breakdown of major contributions

Guillaume Labranche Type checker architecture, symbol table implementation, type checker implementation (especially toplevel declarations and types)

William Bain Type checker architecture, command-line interface, type checker implementation (especially expressions and types)

Si Mei Zhang Type checker implementation (especially statements), test programs