

Golang Compiler

Milestone 2 Report

Group 2

March 22, 2016

Members: Shabbir Hussain
Ossama Ahmed
Michael Ho

1 Invalid Programs

All invalid programs pertaining to type checking can be found in the folder /TEST_PROGRAMS/INVALID_TYPECHECK. All tests are commented to describe the type check rule under test. The type checking test cases were based on the specification located at: <http://www.sable.mcgill.ca/hendren/520/2016/assignments/typechecker.pdf>. To test invalid programs we run the program and expect an error. To test for valid programs we look run the program and expect to see no error. For test programs which are valid we force a situation that will cause an error to occur if our test case fails. For example, after a binary expression, we put the result on the right hand side of an assignment to check of the correct type is resolved by the binary expression. If the expression is not well typed then the assignment will fail. We've also first tested our assignment type checking in order to validate binary expressions.

2 Features

2.1 Weeding Phases

The first feature implemented in this milestone was the weeder phase. In the weeder phase we've implemented the following checks:

- break statements inside loops
- continue statements inside loops
- checking all function paths have a return value

- checking that switch cases have only one default case
- checking that assignments have LHS matching RHS

We decided to place these syntactic checks in the weeding phase as opposed to the parser because it is easier to check. For example it is easier to check if a switch statement has only one default case after all the cases have been defined. The parser builds the AST bottom up and our weeder checks the syntax from the top down. It searches through every node until it hits a node that needs to propagate information to its leaf node. For example, a loop node needs to pass on information to its children that it is in a loop such that we can catch dangling break and continue statements.

2.2 Type checking Phase

The biggest feature for this milestone was the type checking. We've implemented the type checking based on the specification on the sable course page as well as the reference compiler.

3 Team Contribution

3.1 Shabbir Hussain

In the last milestone we lost a considerable amount of points due to our program failing a few test cases. In this milestone, I worked on creating as many test cases as possible both for invalid type check (to catch programs we shouldn't allow) and valid type check (to make sure we allow valid programs). I also added a some functionality to the weeder to check for breaks/continue inside loops as well as all paths containing a return statement.

3.2 Ossama Ahmed

For this milestone I worked on the weeder and the type checking. I designed the overall architecture of the weeder and type checking tree search. I also worked on several weeder rules, as well as all the type check rules other than the ones Michael did.

3.3 Michael Ho

In this milestone I worked on the type checking phase. I implemented type checking for statements and expressions. I implemented all the recursive search type checking rules and the list of type check rules for expressions as defined in the specification.