Milestone 3

Zoe Guan, Samuel Laferriere, Emil Rose

April 1, 2015

Our target language is C. We chose C so that we could avoid writing low-level code and take advantage of the optimizations provided by GCC. Also, GoLite's scoping rules are similar to C's. So far, we have implemented code generation for the following subset of GoLite:

- types: int, float64, rune, bool, array (we have not implemented runtime bounds checking for arrays yet)
- variable declarations and short variable declarations
- type declarations involving the types stated above
- function declarations
- assignment statements (not op-assignments)
- block statements
- infinite loops, while loops and for loops
- if statements without the optional init statement
- break/continue statements
- increment/decrement statements
- return statements
- print/println statements
- all expressions except for append, field selection and some cases involving types we have not implemented yet (ex: binary expressions with strings)

This subset is sufficient to generate code for the benchmarks dice.go and evolve.go.

Since the code generation for some constructs requires type information, the code generator obtains a copy of the typemap generated by the type checker. A GoLite declaration, assignment, or print statement containing a list of identifiers and/or a list of expressions gets expanded into multiple C statements (ex: "var x, y = 1, 2" in GoLite $\rightarrow$ "int x =1; int y = 2;" in C) and blank identifiers are ignored. To emulate Go behaviour, a GoLite variable declaration without initialization gets converted to a C variable declaration that explicitly initializes the variable to the zero value for its type. For each variable in a GoLite short variable declaration, we generate either a C variable declaration or a C assignment depending on whether the variable is newly declared or not (ex: "x := 1; x, y := 2, 3;" $\rightarrow$ "int x = 1; x = 2; int y = 3;"). To keep track of which variables are newly declared, we added a hashmap to the type checker where each key is a AShortDeclStmt node and the corresponding value is a list of the newly declared variables in the statement; the code generator obtains a copy of this hashmap.