Milestone 3: Go Parser

Xavier Denis

March 31, 2015

I've chosen to generate working LLVM IR code for this project. While there is good support for LLVM in Haskell, there are a few issues specifically with dynamically sized types (slices, strings) and heap allocated types (struct, array, slice, string). Most of these issues are resolved by writing the necessary implementations directly in IR and linking the modules together.

1. Codegen monads: I implemented the groundwork for codegen in the form of a series of monads which wrap up all generation instructions nicely. Additionally, I wrote wrappers for all instructions required for golite.
2. Bug fixes: Fixed many issues brought up during the milestone 2 review.
3. Change annotation type: The annotation type used to carry type information in the tree was changed to a record type. This allows for raw type information to be passed along. Additionally, it makes the code more flexible and extensible.
4. Quasiquoter: To aid in testing, a quasiquoter was implemented for both Top Level Declarations and Packages.
5. Implemented Codegen Rules:
   (a) Package
   (b) Global Type Defs: Currently there is still an issue with types that share the same name
   (c) Statements:
      i. Return: Bugs for void returns
      ii. Expression Statements

iii. If Statements

iv. Empty

v. Inc / Dec: Only works for int types and aliases

vi. Block

vii. While-For

viii. For-Clause

ix. For-Empty

x. Variable Declarations

xi. Short Declarations

xii. Assignments

xiii. Type Declarations

(d) Expressions:

i. Binary Op: All but bitclear

ii. UnaryOp

iii. Integers

iv. Floats

v. Bools

vi. Func. Calls

vii. Variable reference

viii. Selector

ix. Index: Not bounds checked yet

6. Notable Issues:

(a) Allocations are all made to stack. Need to move them all to the heap.

(b) Variable sized types. Those will require special support in the form of custom IR implementations

(c) Builtin Functions: Similarily to variable sized types those will require special support.

(d) Exceptions: LLVM supports zero cost exceptions through the Itanium ABI however, I need to do more research on how to actually implement / use that.

1 Work Split

I, Xavier Denis, did this entire milestone.