Go Compiler

Release 2 Plan
April 4th, 2018 - June 5, 2018
Kyle Remmert (Scrum Master), Petar Zaninovich, Trevor Ching, Vincent Kim (Product Owner)

High level goals

The way that language creators would create new programming languages would be by translating code to an intermediate representation(IR) that looks similar to machine code. Another way of creating a language would be the use of an Abstract Syntax Tree(AST) which is an IR that provides a high level overview of what code looks like. But the AST approach was usually too slow and time consuming. Oracle has come out with a new Java framework called Truffle which speeds up the process of creating a language by using an Abstract Syntax Tree. The Truffle framework not only helps language developers create languages in a more streamlined fashion, but it also works on top of a compiler called Graal that will optimize the language being developed. Our goal is to translate a subset of the Go Language into Java using the Truffle framework to create an AST interpreter of the language. The subset of the Go Language which we are aiming for will include: the basic Go Datatypes including primitive types, Arrays, Slices, Maps, Pointers, and Structs. Go functions that are able to handle, multiple return types, recursion, closures, function pointers, method receivers. Go type checking and exceptions.

Sprint 1

(21)User Story 1: As a user, I would like to use pointers in my Go source code.

Assigned: Trevor

Tasks:

- **(2)Trevor:** Add the missing Golang ast tree nodes into the parser and collect the child nodes involved.
- **(5)Trevor:** Figure out a way to either model the memory for Go or mimic memory without implementing the Unsafe api
- **(5)Trevor:** Star nodes execute on the expression and returns the value pointed in the frameslot
- **(3)Trevor:** Unary & nodes return the frameslot of the variable
- **(6)Trevor:** Dereferencing pointers can be written into and read from using an index expression

(8)User Story 2: As a user, I would like to be able to import a library so I could use more functions.

Assigned: Kyle

Tasks:

Go Compiler

- **(1)Kyle:** Create the nodes in the IR for import
- **(3)Kyle:** Create the SelectorExpr node in Truffle
- **(3)Kyle:** Update invoke to allow for SelectorExpr or Ident nodes
- **(1)Kyle:** Install the package if it is not installed.

(5)User Story 3: As a user, I would like to be able to call a function.

Assigned: Trevor

Tasks:

- **(2)Trevor:** Call a function without parameters
- **(2)Trevor:**Create/Declare a function with 0 parameters
- **(1)Trevor:** Setup the IR for a fieldlist

(5)User Story 4: As a user, I would like to be able to use all of Go's types.

Assigned: Vince

Tasks:

- **(1)Vince:** Add float(32), double(64) type nodes to Truffle
- **(1)Vince:** Add float, double type IR nodes
- **(2)Vince:** Create visit methods for each type of node
- **(1)Vince:** BasicLit GoIRFloatNode, GoIRCharNode accounted for in the visit methods

(7)User Story 5: As a user, I would like functions to be able to return a value

Assigned: Petar

Tasks:

- **(4)Petar:** Add the corresponding IR nodes to the IR tree and switch statement object factory.
  - Create field, fieldlist, and return class files
  - Add associated ir to truffle node transformer methods.
- **(2)Petar:** Create Truffle node return stmt
  - On execute this will throw the value to where it was called.
- **(1)Petar:** Assign variables with function return value

(13)User Story 6: As a developer, I would like Arrays and Slices to work in a consistent manner. Arrays and Slices do not work in the same fashion when being executed during reading and writing.

Assigned: Trevor

- **(1)Trevor:** Create super class type for arrays and slices, will be able to read, insert, len, cap.
- **(3)Trevor:** Create new write ir to truffle node transformer to distinguish between a read and a write to an index node.
- **(1)Trevor:** Figure out what should be returned when a array type or a slice type is executed.

Go Compiler

- **(3)Trevor:** Calculate the offset for slices when reading and writing to a slice.
- **(5)Trevor:** Handle slice operators on arrays and slices, returning a new slice.

Sprint 2

(21)User Story 1: As a user, I would like to call and create functions with parameters

Assigned: Kyle, Trevor

Tasks:

- **(2)Kyle:** Read in the Functype nodes and Fieldlist nodes from the AST file
- **(10)Kyle:** Add the field nodes into the function frame descriptor and the lexical scope
- **(7)Trevor:** Create write nodes out of each field listed so the invoke will recognize the arguments.
- **(3)Kyle:** Execute the write nodes at the beginning of the function

(21)User Story 2: As a user, I would like to be able to use a struct and the fields within a struct

Assigned: Vince, Trevor

Tasks:

- **(5)Trevor:**Change the call target of the program from the main function to a File node, so that execution of global variables and type declarations can be made
- **(1)Vince:**Read in the struct type node from the AST parser.
- **(5)Vince:**Create the struct type node with a symbol table.
- **(2)Trevor:**Allow composite lits to either assign variables by name or by order
- **(3)Trevor:**Selector expressions read from the frameslot and access the struct variable in its symbol table.
- **(5)Vince:**Allow structs to define themselves in the lexicalscope at the top level so that they can add themselves as a field

(3)User Story 3: As a user, I would like to diff my program output with Go output

Assigned: Petar

Tasks:

- **(3)Petar:** Change gt - this runs our entire program sequence
  - Run and output go program to file
  - Output our program to a file
  - Diff the 2 files

(18)User Story 4: As a user, I would like my functions to be able to return multiple things

Assigned: Petar

Tasks:

- **(3)Petar:** Be able to assign the values from the multiple returns
  - Change how assignment works, in parser and in the ir to truffle transformation
- **(1)Petar:** Change the Truffle Nodes to allow the return of multiple values

Go Compiler

    - Returnstmt node
- **(7)Petar:** Type Check the return with the assignment variable
- **(7)Petar:** Check Number assignments corresponds with number of things returned

Sprint 3

(18)User Story 1: As a user, I would like to be able to use a struct and the fields within a struct.

Assigned: Vince, Trevor

Tasks:

- **(5)Vince:** Create the struct type node with a symbol table.
- **(3)Trevor:**Allow composite lits to either assign variables by name or by order
- **(5)Trevor:**Selector expressions read from the frameslot and access the struct variable in its symbol table.
- **(5)Vince:**Allow structs to define themselves in the lexical scope at the top level so that they can add themselves as a field

(22)User Story 2: As a user, I would like my Go program to typecheck correctly to keep language consistency and provide with meaningful error messages.

Assigned: Kyle, Petar

Tasks:

- **(8)Kyle:** Single assignments check the type of both the variable and the value before assigning
    - Throw GoException if the types do not match
    - Modify before assignment to check types
        - In the parser, get the identifier node for the left and right hand side then check types there.
- **(9)Petar:** Function type checking checks parameters and values.
    - Multiple assignments: check the length of every variable before assigning the values
    - Function calls are type checked in the parameters with the function being called.
    - Return types and length match function signature checked inside a function
- **(5)Petar:** Non primitive types check before accessing and writing to the type

(11)User Story 3: As a user, I would like the float32 and float64 types for various math operations.

Assigned: Petar

Tasks:

- (3)**Petar:** Make floats work with the composite lit node

    - Case 1: It is initially assigned an int value, but declared as float
    - Case 2: Declared and initialized as float
- (1)**Petar:** Add unit tests for following types
    - Float32, Float64
        - Various math operations using the two values
- (1)**Petar:** Add specializations when floats are used
    - Binary/Unary expressions
        - Add corresponding functions
- (5)**Petar:** Type check
    - Left and right hand side match in type and size
    - Throw corresponding GoException with dialogues similar to Go's error
- (1)**Petar:** Prints floats in correct notation
    - Modified println builtin

Sprint 4

(17)User Story 1: As a user, I would like lexical scoping so my variables are unique across scopes.

Assigned: Kyle

Tasks:

- **(4)Kyle:** Add unit tests to account for all scoping problems
    - Global variables remain the same in recursion/ across multiple functions
    - Using the same parameter inside recursion
    - Variables with same name across different scopes
    - Check if you can't see variables outside for loops and if statements
- **(13)Kyle:** Function scoping for calling a function that is not yet created in the IR
    - Change how scoping works for forward declared functions so these functions have their own scope
    - Arrange function scoping
    - Test that functions work correctly

(33)User Story 2: As a user, I would like my Go program to typecheck correctly to keep language consistency and provide with meaningful error messages.

Assigned: Petar

Tasks:

- (10)**Petar:** Multiple assignments: check the type of every variable before assigning the values
- (8)**Petar:** Function calls are type checked in the parameters with the function being called.

- (8)**Petar:** Return types and length match function signature checked inside a function
- (7)**Petar:** Non primitive types check before accessing and writing to the type

(19)User Story 3: As a user, I would like to be able to use a map

Assigned: Vince

Tasks:

- **(1)Vince:** Implement Map type IR Nodes from the GoLang ast.
- **(3)Vince:** Create the Map type node with a Hashmap accessed similar to Struct nodes
- **(8)Vince:** Allow composite lits to be constructed for a Map type node
- **(2)Vince:** Read and writing to a map using the index node
- **(5)Vince:** Add read and writes specialization inside the frameslot write nodes and the corresponding read/writes for Map nodes.

(15)User Story 4: As a user, I would like to append values to the end of slices

Assigned: Trevor

Tasks:

- **(1)Trevor:** Create a new Rootnode for the append builtin in GoContext, then add the function to the function registry.
- **(1)Trevor:** Create Append file which will read in the frame arguments and append it to a slice.
- **(3)Trevor:** Allow a whole slice to be passed in with multiple single objects to be appended at the end of the slice.
- **(5)Trevor:** Allow a slice to be appended to another slice.
- **(5)Trevor:** Allow a portion of a slice to be passed in and append another slice on the end of it.

(11)User Story 5: As a user, I would like to be able to print multiple arguments in a single call

Assigned: Trevor

Tasks:

- **(1)Trevor:** Add the fmt println root node to the function registry.
- **(2)Trevor:** Redo the fmt println file to handle taking in variable amounts of arguments.
- **(3)Trevor:** Loop through the arguments and append it to a string builder, then print the result.
- **(5)Trevor:** Fix selector expressions to handle both imports and structs when selecting a method or a field.

Sprint 5

Go Compiler

(28)User Story 1: As a user, I would like global variables to be changed at the global scope level when updating them inside a function's scope

Assigned: Kyle

Tasks:

- **(1)Kyle:** Add unit test to check for changing a global variable
- **(15)Kyle:** Research how to modify global variables at Truffle time
- **(3)Kyle:** Create the global materialized frame in the GoContext
- **(3)Kyle:** Create the Truffle read and write nodes for the global variables.
- **(3)Kyle:** Create a writer visitor file following the visitor pattern for global variables
- **(3)Kyle:** Identify if a variable is global at assignment time

(34)User Story 2: As a user, I would like my Go program to typecheck correctly to keep language consistency and provide with meaningful error messages.

Assigned: Petar

Tasks:

- (8)**Petar:** Function calls are type checked in the parameters with the function being called.
- (7)**Petar:** Non primitive types check before accessing and writing to the type
- (7)**Petar:** Type check binary/Unary operations and expressions
- (12)**Petar:** Restructure Type Checking for visitor design pattern for easy extension of adding types and type/error checking.

(25)User Story 3: As a developer, I would like structs to be a subclass of the Truffle Object model.

Assigned: Trevor

Tasks:

- **(3)Trevor:** Read how to implement a Truffle Dynamic Object, how the Layout works, and what the Shape property corresponds to.
- **(2)Trevor:** Structs assigns new struct objects during assignments
- **(20)Trevor:** Implement Struct Dynamic Object
    - **(10)**Allow struct fields to be added in as properties during initial type declaration, Create write property nodes
    - **(10)**Assign new struct objects to variables which can only read and write from properties, not create new properties, create write and read from property nodes

(22)User Story 4: As a user, I would like to declare a inline function without a name (Anonymous function / Function literal).

Assigned: Vince

Tasks:

Go Compiler

- **(5)Vince:** Create and change necessary files for the func lit node (Update truffle, IRVisitor, create IR funclit, truffle funclit)
- **(7)Vince:** Make sure the func lit node can be assigned correctly to a variable and called from it. Additionally, make sure the func lit node can be called straight from callexpr without any assignment.
- **(10)Vince:** Make sure the funclit can behave like closures (Access variables if in scope).

(3)User Story 5: As a developer, I would like struct objects to be printed entirely in fmt.Println function calls

Assigned: Trevor

Tasks:

- **(3)Trevor:** Either create a case inside the println builtin to print the keys and values or find a way to create a toString function for the DynamicObject Struct

(11)User Story: As a user, I would like to be able to write methods for structs

Tasks: Trevor

- **(2)Trevor:**Edit function definition to handle the case of a struct receiver inside GoTruffle
- **(1)Trevor:**Possibly create a new visitor for the function handler
- **(5)Trevor:**Read from the struct object and create write variables into the frame descriptor for each field the struct has.
- **(3)Trevor:**Create specialization for the selector expression node for method calls.

Sprint 6

(4)User Story 1: As a user, I would like global variables to be changed at the global scope level when updating them inside a function's scope

Assigned: Kyle

Tasks:

- **(2)Kyle:** Create the Truffle read and write nodes for the global variables.
- **(2)Kyle:** Create a writer visitor file following the visitor pattern for global variables of various object types

(14)User Story 2: As a user, I would like extensive documentation about what type of subset is available to use

Assigned: Kyle

Tasks:

- **(2)Kyle:** Create small github site for the GoLang project
- **(12)Kyle:** Add information about each functionality
    - What type of assignments
    - What object types are available
    - What type of expressions

Go Compiler

    - Make a small code segment for each functionality
    - Show output of each code segment

(28)User Story 3: As a user, I would like my compiler to give comprehensive error messages and typecheck Nonprimitive types

Assigned: Petar

Tasks:

- **(5) Petar:** Create more compare type functions for each situation that requires a different error message
- **(8) Petar:** Create a systemic way of identifying objects.
- **(15) Petar:** Update type checking visitor for the nonprimitive types to return their types and/or throw errors when appropriate

(10)User Story 4: As a user, I would like my compiler to give a list of compile time errors.

Assigned: Trevor

Tasks:

- **(3)Trevor:** Figure out the best way of handling compile time errors using either exceptions or using some other method to output all errors found.
- **(1)Trevor:** Stop the java stack trace being printed
- **(3)Trevor:** Add error catching in Gotruffle compilation and a list of caught errors. If the list is empty, then there are no errors. If the list is not empty then compilation has failed, print errors.
- **(3)Trevor:** Add source sections for better error checking results.

(1)User Story 5: As a user, I would like the compiler to throw an error when a variable that is not defined is used, not have it print a String.

Assigned: Trevor

Tasks:

- **(1):** Upon printing a variable, check that it exists in the frame and print an error if it does not exist, not the toString() of the variable.

(2)User Story 6: As a user, I would like the compiler to complain if I have unused variables.

Assigned: Trevor

Tasks:

- **(1):** Create a set to keep track of all user defined variables
- **(1):** Remove variables from the set everytime they are used. At the end of truffle compilation, check if the set is empty. If it is not empty, then there are unused variables.

(3)User Story 7: As a user, I would like test files to understand what part of the Go subset is available to use

Assigned: Vince

Tasks:

- **(3):** Add several unit tests showcasing our compiler

Go Compiler

(2)User Story 8: As a user, I would like arrays and slices to get copied over properly

Assigned: Trevor

Tasks:

- **Trevor(1):** Fix append built-in bug not creating a new slice copy.
- **Trevor(1):** Check if arrays and slices can consistently be assigned to each other.

Product Backlog

(20)User Story 1: As a user, I would like my compiler to give comprehensive error messages and typecheck Nonprimitive types

Assigned: Petar

Tasks:

- **(5) Petar:** Create more compare type functions for each situation that requires a different error message
- **(15) Petar:** Type check builtin functions

(4)User Story 2: As a user, I would like global variables to be changed at the global scope level when updating them inside a function's scope

Assigned: Kyle

Tasks:

- **(2)Kyle:** Create the Truffle read and write nodes for the global variables.
- **(2)Kyle:** Create a writer visitor file following the visitor pattern for global variables of various object types

()User Story: As a user, I would like the compiler to pass objects to functions by value

Assigned: Trevor

Tasks:

- **(1):** Type checking needs to occur on the value contained inside the pointer
- Check if pointers can properly be assigned to objects and dereference them. Basically act like the object using either indices or selector, but is a pointer.
- Pointers as parameters
- Star and unary

()User Story: As a user, I would like maps to be a dynamic object

Assigned:

Tasks:

- **(1):**

(12)User Story 1: As a user, I would like to use function closures so I could close variables in functions and define a function inline without having to name it

Assigned:

Tasks:

- **(1):** Add unit test to check for correctness

Go Compiler

- **(1):** Create the FuncLit intermediate representation class file
- **(2):** Make sure lexical scoping works correctly for closures
- **(4):** Update assignments so that they can correctly point to a function
- **(4):** Update CallExpr to make sure it uses the correct values that the closure "closes"

(1)User Story: As a user, I would like the const keyword.

Tasks:

- Add a if statement case inside of write visitor for every read from the lexical scope, check if the const flag is true or false.