NUMLANG

Dan Aprahamian
Damien Fenske-Corbiere
Sahil Yakmi
Siddhi Mittal

Introduction

- Designed to make numerical computation easy.
- Key Features of this language:
  - Allows mathematical functions to be entered as literals, manipulated, called, and composed.
  - Allows computation with matrices and other common mathematical operations.
  - Arithmetic without loss of precision.
  - An innovative way of implementing a for loop using match statements.
- The language is intended to be suitable for compilation as well as interpreting. The reference implementation is, however, a compiler.

Tutorial Introduction to Language

```
y = [[0, 0, 0], [0, 3, 5]];
new_y = y[1];
new_y2 = y[1][1];
pop<<y>>;y[1][1] = 4;
sub callMe(num x, string list y) {y[1];}
sub call2() {34;}
str2 = "hello";println<<"str2">>;
strList = ["hello", "str"];rm<<strList>>;
str2 = callMe<<4, strList>>;
w = call2<<>>;
w = 43;
x = "hello";
print<<x>>;
match(w) {
  cont: w - (w % 10) ? {x = 1;}
  loop: > 22 ? {x =  - 1;}
  <= 12 % 4 ? pass;
  done: true ? pass; }
```

Language Implementation

- Series of Modules
  - Scanner/Parser and AST
  - Static Semantic Checker
  - Java Source Code Generation (Compiler)
  - Compilation to Java Byte Code

- Tools Used
  - GitHub
  - OcamlDebug
  - Google Docs (Collaborative Code)
  - Google

Our files

File	Lines	Role
Scanner.mll	124 lines	Token rules
Parser.mly	220 lines	CFG
Ast.ml	77 lines	Types
Sast.ml	46 lines	Checked types
ssc.ml	666 lines	Semantic Validity
numlangc.ml	276 lines	Compiler
Total	1409 lines	

Summary/Lessons Learned

- Matrix Like Language capable of
  - Manipulating functions
  - Retaining precision of numbers
  - Making mathematical computation easier with simple syntax

- Lessons Learnt
  - Value of ocamldebug
  - Ocaml's value
  - Resolving reduce/reduce and shift/reduce errors for ambiguous grammar
  - Importance of resolving scoping issues
  - How pieces of Compiler fit together

- Time Value
  - Start early