COMP 325 - Interpreter 2- Functional Boolean Arithmetic

Fall 2015

Abstract

Your second interpreter still involves a single data type but includes Functions and Conditionals. In addition to implementing the language, you're asked to write a small

Due Tuesday, 10/6

For this interpreter we'll be building a functional language for boolean arithmetic and then consider its usage for expressing and simulating boolean logic circuits.

Your language will provide:

- The unary *not* operator
- The n-ary versions of the operators *and*, *or*, *xor*, *nor*, *nand*, *xnor*
- A multi-branch conditional expression a la *if elseif ... else* or Racket/Scheme *cond*
- n-ary functions

Once you've completed the language, use it to write a program to carry out an adder circuit.

0.1 Implementation Notes

You have many issues to deal with and choices to make as the language implementer. We'll discuss some of them in class. Many of them are also addressed in the text.

- Conditional Expression Syntax (cond-like or more if...elseif like?)
- n-ary operators and conditionals as sugar
- Universality of key sets of binary operators: {*xor,and*}, {*and,or,not*}, {*nand*}, and more!
- Short-circuiting operators
- Lazy vs Eager evaluation
- Efficient Substitution via Environments