# CSC330 Summer 2004 Assignment 3

George Tzanetakis

June 18, 2004

## 1 Overview

This assignment will be done in groups of two. You are free to choose your own group. In case you do, I will need an email from both partners to confirm that they are in group. Otherwise you will be assigned randomly a partner. YOU NEED TO EMAIL ME IN EITHER CASE. FAILURE TO DO SO MIGHT RESULT IN NOT GETTING GRADE FOR THE ASSIGNMENT.

A random 25% sample of the students will be orally interviewed by me regarding their submission. Make sure you have written everything you submit. Solve the parts of the assignment in the order they are given as they get progressively harder. If you are stuck somewhere move to the next part and come back later.

The careful design and documentation (i.e comments) of your code will be important factors in your grade. If there is a bug it's better to report it than hide it. Be honest, precise and clear and you shall be rewarded. NEVER (at least in this class) sacrifice clarity for efficiency of execution unless explicitely asked to do so. PROVIDE A SET OF WELL-DESIGNED TEST CASES FOR EVERY PART OF THE ASSIGNMENT.

The assignment is worth 10 points (10% of the final grade) and will be graded in using half-point intervals. The overall documentation and packaging of your submission according to the instructions provided in the web page (READ THEM CAREFULLY) will be worth 1 point.

The main goal of the assignment is to familiarize with how the semantics of a programming language are expressed in an actual working interpreter. It is also a good exercise for learning more about C, Scheme and ML programming. I hope you find the assignment interesting and helpful. The assignment is intentionally not fully specified so you get practice in how programming is in the real world. Part of your task is to interpret each part of the assignment and provide a reasonable answer and implementation.

IMPORTANT: A major goal of this assignment is to understand how the provided interpreters work. You should study their source code in detail and be ready to answer questions about them during the interview (not only the parts you modify). The assignment is easy but requires understanding and the training wheels are off this time.

## 2 Part 1 (3 pt)

Write 1,2,3 of assignment 2 in Scheme (each part is worth 1 point). Make sure your programs run with the provided interpreters for Scheme implemented in C and ML. Make the necessary adjustments within reason to allow implementation in Scheme (for example ignore the infix part).

## 3 Part 2 (2 pt)

Add a new primitive *list* which should accept any number of arguments and return a list consisting of these arguments. (we have already shown list1, list2 for specific number of arguments). For example the result of evaluating (list 1 2 3) is (1 2 3). Add the new primitive to the C version of the interpreter (1 pt) and to the ML version of the interpreter (1 pt).

## 4 Part 3 (2 pt)

In this part you will extend assignment 1 to include binding of names to circuits. Your language will have two types of expressions: bindings and print statements. To illustrate this consider the following input file:

```
foo := - 0.5 0.8
bar := - 0.5 0.8
print - foo bar
```

This file should produce the same output as the original assignment input file:

```
- - 0.5 0.8 - 0.5 0.8
```

You will have to extend your lexical analysis and parsing to handle identifiers and include a symbol table to lookup circuits. Output is produced only with print expressions and is the same as assignment 1.

You can choose to either extend your own implementation of assignment 1 or use the provided SML solution (or both but no extra credit for doing that other than your increased knowledge)

## 5 Part 4 (3pt)

Add a trace facility to the interpreter (both for the C and ML version). Whenever a trace function is called, it's name (if any) and arguments should be printed to the terminal. (If the name of the function is not known, print a representation of it's abstract syntax). When a traced function returns the function and its result should be printed. The trace output should be clearly labelled and should include identation to indicate calls and returns.

Provide primitives to turn tracing of individual functions on and off. Trace *length*, *sieve* and *remove multiples*. Trace part 1 of this assignment.

When calling a closure print it in closure form instead of printing its name. Which of the two methods is better ? Explain in your README file.

## 6 Extra Credit

- (1pt) Extend your circuit interpreter (part 3) to include simple functions (basically parametrized circuits). Part of the extra credit is understanding what this means and coming up with the right syntax and primitives.
- (2pt) Do exercise 32 (page 142) of the Ramsey Kamin Book
- (3pt) Do exercise 36 (page 143) of the Ramsey Kamin Book

## 7 Submission

Please follow the submission guidelines from the course webpage. In summary tar and gzip everything into one file. Include a README explaining how things are structured. Include all the code necessary to compile the assignment not just the parts you extended/modified.

HAVE FUN AND I HOPE YOU ENJOY THIS ASSIGNMENT AS MUCH AS I ENJOYED PREPARING IT