CS 5540 Project 2

Arjun Biddanda (aab227)
Muhammad Khan (mhk98)

March 12, 2013

1 Module Design

1.1 Language Selection and other Details

For this project, we chose to perform our analyses using OCaml. OCaml, being a functional programming language, offers a simplicity in its syntax so that it is easier to understand. We utilized the capabilities of OCaml to be modular very heavily in our design so that each module performs a given functionality that we require of it. We also chose OCaml because we both enjoyed CS 3110 very much and wanted to pay tribute to it.

1.1.1 Module IO

This is the input/output module that we have designed. It contains the reader type, which is essentially an element which allows us to read a text file. This is essentially the entire function of the IO module, to be able to read any form of textual input that we give it, line by line. In addition to the reader, the large portion of functionality in this module is that of the str_to_float_lst function, which takes in a string of length n consisting of floats that are delimited by spaces. We are able to parse in all of the floats that we require, and output a list of floats which is an easy stream of data that we are able to work with.

1.1.2 Module Parser

The parser module is where we start to make assumptions on the type of data that we are given. We define the parser module as a functor of the IO module that we created above. This means that we are able to use all the functionality of the IO module within the Parser module. There are three things that highlight this module:

1. We have formally defined the type of a signal to be a list of floats, which essentially acts as a time series of readings, where each consecutive index in the list is the next timestep.

2. We define the data as a whole (all of the signals) to be and array A of signals. The array is always of length 40, because that is the number of signals that are given in the data set.

3. Possibly the most important function in the entire project is the parse function. This is what allows us to represent our data from the text file that is given. We use the IO modules string to float list conversion function in order to generate each signal and then input it into our data array. Iteratively this takes $O(n)$ time across all of the inputs. However, the memory requirements are great due to our construction of the data. This helps us along later because of the ease of access that we have to our data.

1.1.3 analysis.ml

The analysis.ml file contains all of our analytic techniques used to investigate this data. These techniques were explained in the writeup and will be further elaborated upon in a later section to discuss the design of these particular algorithms.

1.1.4 graphics.ml

This module is somewhat of a new discovery to us. We initially had no idea that graphics were even possible in OCaml, so we played around with the Graphics module that comes preinstalled. The intent of this module is so that we are able to plug in an operation from our analysis tools into the Graphics module, along with the data, and are able to see the effect that our analysis had on the data from a graphical sense.

2 Analysis of Algorithms Used

2.1 Zero-cross Algorithm

We created a subfunction called compare which basically compares two elements in order to return a boolean value based on the expression given in the assignment page. We then recursively go through the list, picking the first two elements and calling our compare function on them. If