COMS W4115

Programming Languages & Translators

Graphr Project Proposal

09/24/07

Paul Dix – pcd2104@columbia.edu
Joe Kamien – jtk2105@columbia.edu
Michael Cole – mtc2106@columbia.edu
Zhe Chen – zc2118@columbia.edu

Introduction

Graphr is a language for processing data sets and creating charts and graphs from that processed data. Built into the language are functions and types for specifying different kinds of charts, graphs, histograms, and their source data. The language should be useful in cases where important data must be derived from earlier data. The language is designed to be readable, expandable, and dynamic.

Usefulness

The ability to analyze data and draw conclusions is extremely important. It is difficult to work with large datasets in current spreadsheet software. In cases where multiple functions need to be used on one dataset, the list of dependencies may become unmanageable, and when mistakes are made, debugging is lengthy and difficult. Moreover, since functions must be defined based on locations within the dataset, not based on variables which are meant to be represented by the dataset, it is impossible to reuse functions across multiple datasets.

By allowing the programmer to easily define his or her own parser, it will be much simpler to use similar functions across myriad large datasets. By allowing the user to work with functions by variable names, instead of locations within the dataset, debugging will be much simpler.

Portable

We predict that Graphr will be commonly used by scientists and engineers. Therefore, in order to support the principles of multilateral cooperation and peer review, which are hallmarks of the laboratory sciences, the language should be as platform-independent as possible.

Readable

Graphr's syntax should be a human readable language for describing graphs. It will be easy for the programmer to specify different mathematical functions. It is important in many real world applications to clarify the path from source data to conclusions drawn from that data. Making the methods and algorithms for how data files are parsed understandable is another important consideration for readability.

Data Types

We plan to make the language dynamically typed in order to help programmers who are not computer scientists understand our language. therefore, we want to avoid using cryptic types like integer, float, etc which can confuse users who are unfamiliar with these terms.

However, there will be a certain number of first class types in the syntax of the language. These include numbers, regular expressions, functions, and graphs.

Expandable

Graphr includes built in functionality for parsing different source data files like .CSV. However, not all source data file types can be built into the language by default. It will be possible for users of the language to design additional preprocessing modules for parsing file types besides CSV.

The language will also have a number of built in chart and graph types. These will define how a graph is drawn and how the data appears on the graph. Users of Graphr will be able to define their own custom types to change aspects of how a graph is drawn or data is shown.

Scoping

We are currently analyzing the advantages and disadvantages of different scoping paradigms.   So far we have not decided on a specific scoping scheme, but block level scoping is something we are heavily considering.

Example of Syntax

```
// comments will look like this

// the following is specific type of function to populate variables from the input
// additional functions can be defined to perform parsing tasks and expressing algorithms
parser lab_one_parser {
  length = col[1]
  width = col[2]
}

// file will be a built in function to assist in parsing graph data
maindata = file(csv, lab_one_parser, “somefile.csv”)
```

```
// here is a function definition to perform a calculation on data
function area(length, width) {length * width}

// the following function creates a histogram based on the data and calls the function for
// each row in the data
create_histogram(maindata, area)
```

Possible Output

Since the goal of the language is to quickly parse through data and create charts and graphs, output is a consideration. Some of the possibilities are image files of the graph, an interactive window that will allow the user to modify the graph on the fly, or possibly SVG. As a group the initial focus will be to output image files, but with an eye towards expanding to other formats.

We are strongly considering utilizing the JChart package software. The programmer should be able to produce GUI applications with rich support for user interaction.