Implementation of Scheme Numeric System for JavaScript

Deniz A. Gürsel*        Uğur Çekmez†
İstanbul Bilgi University İstanbul Bilgi University

R. Emre Başar‡
İstanbul Bilgi University

December 15, 2009

Abstract

By the rise of JavaScript and AJAX based web applications, there is a new trend towards programming the web using unorthodox methods. One of those approaches is writing compilers which generate JavaScript applications from the native programming language of the programmer. While Scheme is one of the languages that is compiled to JavaScript, writing a compiler from Scheme to JavaScript is harder than most other languages because of the extraordinary feature-set of Scheme. One of the main missing parts is the implementation of the numeric tower of Scheme in JavaScript.

This article presents the design and implementation of a JavaScript library which implements a Scheme-like numeric system which can be used by the aforementioned compilers and also by JavaScript programmers.

1 Introduction

There has been an increasing tendency to develop compiler from Scheme to JavaScript such as Moby, WorldWithWeb, Scheme2Js. All the projects provide a web interface to compile Scheme applications in the web by re-changing the code from Scheme to JavaScript and display the outputs in the web browser and all of these need an new infrastructure for number systems to keep. In Scheme, there exists a library called world.ss and this provides many useful models to create animations. It is done by processing different algorithms and thinking functionally but this is a bit tricky to share these visuals with other people. Because, to open the animations, it is needed to compile it with an environment such as Dr.Scheme. Implementation of Scheme Numeric System for JavaScript is one of the footsteps to complete World-
WithWeb project. To keep the numeric system of the Scheme, it is needed to remove memory limit of the numbers in JavaScript. Since the number models have their own limited access memory such as all numbers in JavaScript are 53 bits floating point numbers (?), if it is needed to work on bigger numbers, there were no chance to do this but in Scheme, there is no limit for numbers. The only limit is the memory of the computer. This article presents how the numeric tower of Scheme is in JavaScript and the implementation of the big numbers on JavaScript such that the number model is kept.

2 Numeric Tower of Scheme

Scheme provides an important classification which does not exist in JavaScript. Numbers are classified as exact or inexact. An exact number is a model that represents the number in mathematics. Calculations are always accurate with exact numbers. Contrary to exact numbers inexact numbers may cause accuracy and precision problems because inexact numbers are floating point numbers and floating point numbers have poor accuracy. When exact numbers are required, Scheme catches the use of inexact numbers and distinguish between them. Those numbers are modelled in a hierarchy called the numeric tower which organizes numbers as integer numbers, rational numbers, real numbers and complex numbers which are derived from exact/inexact numbers. This numeric tower is based on the hierarchy of number in mathematics. The reason it is called a tower, every type is subset of the type which follows it. Every integer number is also a rational number, every rational number is real number and every real number is a complex number. (2)

3 JavaScript Implementation

JavaScript has a little bit different object system unlike the other Object-Oriented programming languages. Everything can be called as an object like variables, numbers, Strings etc. In general, JavaScript does not use directly OOP principles such as class definition. For instance, there is neither special structure to defining classes nor inheritance method. Every object can be assigned to another object to be used as a class and inheritance is generally done by using some external objects like prototype. Prototype is also preferred to be able to extend the JavaScript objects model directly. In this case, there exist two types of objects such as primitives and the objects. Every object can be defined as a class because all types such as string, number can be written as ”new” parameter and the operations such as comparing items is processed on this event.

When the numeric tower is implemented in JavaScript, Scheme’s number hierarchy has been taken as a base. The uppermost structure of the tower is chosen as ”Number” and every number is separated by exact and inexact. The other sub-classes of the tower inherited the basic operations from them. The other important factor is that every operation results return more
general type between two element in the process. For example, let one of the given numbers is complex and the other one is rational. Then the result value of the operation between those two number must be a complex number again. (1)

3.1 Exact Numbers

Exact numbers does not exist in JavaScript and they are implemented by using the “BigNum” library which is written by Matthew Crumley. It allows to do arbitrary precision arithmetic and it is exactly same as Scheme.

3.2 Inexact Numbers

Default JavaScript numbers are 53bit floating point numbers and they refer to inexact numbers in Scheme. Because inexact numbers can not be represented as exactly a mathematical number, sometimes it causes accuracy problems. For instance, 0.1*3 gives 0.30000000000000004 in JavaScript. (3)

3.3 Complex Numbers

Complex number is the uppermost class of the tower. Every integer, real and rational number are considered as complex number. A complex number with an exact zero imaginary part is a real number. A complex number has two parts such as real and imaginary.

3.4 Rational Numbers

They are implemented by using the field ”numerator”, ”denominator”. Every rational number is a complex number with zero imaginary part.

3.5 Integers

Integer numbers are the bottom class of the tower. A rational number with denominator ”1” is an integer.

4 Basic Numeric Functions

In the top of the tower, the basic numeric functions are defined as an generalized functions and they are inherited by the subclasses in every layer of the tower. As it is said in the JavaScript implementation part, every single operation returns the type which is the upper one. Addition, subtraction, multiplication and division functions are basically implemented. With usage of those functions, the other functions in Dr.Scheme’s Beginning Student Language (?) such as computing the power of numbers, finding the greatest common divisor, finding max and min and taking the modulo can be simply implemented.

5 Conclusion

In this article we explained the motivation behind creating a numeric library for JavaScript. In the second section we explained the Scheme numeric system in detail and provided a base for the details of our implementation. Lastly we detailed our design of the numeric tower in JavaScript’s prototype based object system and showed
how these base classes can be used more detailed arithmetic operations.

References

