A Comparitive Exploration of Three Unusual Languages

J Bolton

May 24, 2011

Contents

Abstract

This paper provides an introductory tutorial to three contrasting and unusual programming languages: Smalltalk, SETL, and Agda. Some prior experience in programming is assumed. The languages are highly abstract yet represent very different perspectives: the object oriented paradigm, functional programming with dependent types, and set-based imperative programming. Working with each of them at the same time will be an invigorating brain exercise and will hopefully provide some unique insights into the strengths and weaknesses of each style.

1 Introduction

Agda represents the extreme end of functional type theory based on the elegant syntax of Haskell. The language will introduce the very interesting world of dependent types and all the expressiveness that comes with them. Smalltalk is a much older language known for its pure implementation of the object oriented type system, made famous by such extremely popular languages as C++ and Java. Finally, SETL is the oldest of the three languages whose main feature is its flexible, built-in support for creating and manipulating sets and tuples.

We will start out with a very quick overview of each language, starting with the least difficult, SETL, and progressing to the most difficult, Agda. This paper will emphasize a discussion of the typing model of each language.

We will then examine three programs chosen to showcase the advantages of each of the languages. We will follow this with a larger, single program

(an LL parser generator) written in all three languages to provide a direct, contrasting example among all three styles. The larger program, a top down parser, was chosen in the hope that none of the three paradigms would have a very big advantage in its implementation over the others

2 SETL

2.1 Squishing Trees of Numbers

SETL uses an imperative style of programming similar to the classic language Ada. The language has assignments, loops, conditionals, and arithmetic that shouldn't look all that foreign to those familiar with other imperative languages.

We will examine the language by walking through a couple problems that seem to particularly suit its style. We will end by examining a general problem that we will also implement in Smalltalk and Agda later on.

The first of our problems comes from the puzzle website called Project Euler. It involves finding a maximal path through a large tree of random numbers. These aren't quite binary trees, but more like lattices, where every child node is shared to the left and right. This is the statement of the problem from the website:

> By starting at the top of the triangle below and moving to adjacent numbers on the row below, the maximum total from top to bottom is 23.
>
> 3 7 4 2 4 6 8 5 9 3
>
> That is, 3 + 7 + 4 + 9 = 23.
>
> Find the maximum total from top to bottom in triangle.txt

2.2 Set Magic

2.3 SETL's Type Model

2.4 Case Study: Climbing Trees

2.5 Case Study: A Top-Down Parser

3 Smalltalk

3.1 The Type Model

3.2 Control Flow

3.3 Case Study: A Simple Text Adventure Interpreter

3.4 The Parser Revisited

4 Agda

4.1 Ordinary Types

4.2 Dependent Types

4.3 Pattern Matching Heaven

4.4 Case Study: A Lambda Expression Evaluator

4.5 The Parser a Final Time

5 Summary and Final Comparison