Sample Report for Simple ML Example

Shiu-Kai Chin

September 10, 2017

Abstract

The project brings together elements of functional programming in ML and documentation ins LaTeX. The purpose of this project is to lay the groundwork for credibility: results that are thoroughly documented and easily reproducible by independent third parties. We establish the documentation and programming infrastructure where each chapter documents a problem or exercise. Within each chapter are sections stating or showing:

- Problem statement
- Relevant code
- Test results

For each problem or exercise-oriented chapter in the main body of the report is a corresponding chapter in the Appendix containing the source code in ML. This source code is not pasted into the Appendix. Rather, it is input directly from the source code file itself. This means changes in source code are easily captured in the report by recompiling the report in LaTeX.

We introduce the use of style files and packages. Specifically, we use:

- a style file for the course, *634format.sty*,
- the *listings* package for displaying and inputting ML source code, and
- HOL style files and commands to display interactive ML/HOL sessions.

Finally, we show how to:

- easily generate a table of contents for the report, and
- refer to chapter and section labels in our report.

There are numerous LaTeX tutorials on the web, for example, `https://www.latex-tutorial.com`, is very accessible for beginners.

Acknowledgments: We gratefully acknowledge the hard work, trust, and dedication of our past students in the Syracuse University Cyber Engineering Semester and the Air Force Research Laboratory's Advanced Course (ACE) in Engineering Cybersecurity Boot Camp. They bridged dreams and reality.

Contents

Chapter 1

Executive Summary

All requirements for this project are satisfied. Specifically,

Report Contents
Our report has the following content:

Reproducibility in ML and LaTeX
Our ML and LaTeX source files compile with no errors.

Chapter 2

Sample Exercise

2.1 Problem Statement

2.1.1 Functions to implement

In this exercise we are to define in ML the following functions:

$$plus\ x\ y = x + y \tag{2.1}$$
$$times\ x\ y = x \times y \tag{2.2}$$
$$plus\ x\ y = x + y \tag{2.3}$$
$$times\ x\ y = x \times y \tag{2.4}$$

2.1.2 Test cases for plus

The required tests for *plus* are as follows.

```
(****************************************************************************)
(* Test Cases Specified in the requirements                                *)
(****************************************************************************)
plus 0 0;
plus 0 1;
plus 0 2;
plus 0 3;

plus 0 0;
plus 1 0;
plus 2 0;
plus 3 0;

plus 1 0;
plus 1 1;
plus 1 2;
plus 1 3;

plus 0 1;
plus 1 1;
plus 2 1;
plus 3 1;
```

2.1.3 Test cases for times

The required test cases for *times* are as follows.

```
(****************************************************************************)
```

```
(* Test Cases Specified in the requirements                              *)
(***************************************************************************)
times 1 0;
times 1 1;
times 1 2;
times 1 3;

times 0 1;
times 1 1;
times 2 1;
times 3 1;

times 4 0;
times 4 1;
times 4 2;
times 4 3;

times 0 4;
times 1 4;
times 2 4;
times 3 4;
```

2.2 Relevant Code

The following code takes advantage of function definition using *fun* in ML, and *currying*, i.e., defining functions with multiple arguments as a sequence of functions. This supports partial evaluation.

```
fun plus x y = x + y;

fun times x y = x*y;
```

2.3 Test Results

Notice in what appears below, we split up the tests for printing clarity.

2.3.1 Test results for plus

1

```
 > plus 0 0;
plus 0 1;
plus 0 2;
plus 0 3;
val it = 0: int
> val it = 1: int
> val it = 2: int
> val it = 3: int
```

2

```
>
plus 0 0;
plus 1 0;
plus 2 0;
plus 3 0;
> val it = 0: int
> val it = 1: int
> val it = 2: int
> val it = 3: int
```

3
```
>
plus 1 0;
plus 1 1;
plus 1 2;
plus 1 3;
> val it = 1: int
> val it = 2: int
> val it = 3: int
> val it = 4: int
```

4
```
>
plus 1 0;
plus 1 1;
plus 1 2;
plus 1 3;
> val it = 1: int
> val it = 2: int
> val it = 3: int
> val it = 4: int
```

2.3.2 Tests results for times

5
```
> > > > val plus = fn: int -> int -> int
> > val times = fn: int -> int -> int
> > > > > plus 0 0;
plus 0 1;
plus 0 2;
plus 0 3;
val it = 0: int
> val it = 1: int
> val it = 2: int
> val it = 3: int
```

6
```
> times 1 0;
times 1 1;
times 1 2;
times 1 3;
val it = 0: int
> val it = 1: int
> val it = 2: int
> val it = 3: int
```

7
```
>
times 0 1;
times 1 1;
times 2 1;
times 3 1;
> val it = 0: int
> val it = 1: int
> val it = 2: int
> val it = 3: int
```

8
```
>
times 4 0;
times 4 1;
times 4 2;
times 4 3;
> val it = 0: int
> val it = 4: int
> val it = 8: int
> val it = 12: int
```

9
```
>
times 0 4;
times 1 4;
times 2 4;
times 3 4;
> val it = 0: int
> val it = 4: int
> val it = 8: int
> val it = 12: int
```

Appendix A

Source Code for Sample Exercise

The following code is from *simple.sml*

```
(**************************************************************************)
(* Author: Shiu-Kai Chin                                                  *)
(* Date: 20 January 2017                                                  *)
(* email: skchin@syr.edu                                                  *)
(**************************************************************************)
fun plus x y = x + y;

fun times x y = x*y;



(**************************************************************************)
(* Test Cases Specified in the requirements                               *)
(**************************************************************************)
plus 0 0;
plus 0 1;
plus 0 2;
plus 0 3;

plus 0 0;
plus 1 0;
plus 2 0;
plus 3 0;

plus 1 0;
plus 1 1;
plus 1 2;
plus 1 3;

plus 0 1;
plus 1 1;
plus 2 1;
plus 3 1;

(**************************************************************************)
(* Test Cases Specified in the requirements                               *)
(**************************************************************************)
times 1 0;
times 1 1;
times 1 2;
times 1 3;
times 0 1;
times 1 1;
times 2 1;
times 3 1;

times 4 0;
times 4 1;
times 4 2;
times 4 3;

times 0 4;
times 1 4;
times 2 4;
times 3 4;
```