Project 1

Bharath Karumudi

July 12, 2019

Abstract

This project is to demostrate the basic functional programming skills using the tools and techniques - LaTeX, AcuTeX, emacs and ML. Each chapter documents the given problems with a structure of:

- Problem Statement
- Relevant Code
- Test Cases

Acknowledgments: Professor Marvine Hamner and Professor Shiu-Kai Chin who taught the Certified Security By Design.

Contents

Chapter 1

Executive Summary

All the requirements for this project are statisfied specifically,

Contents
Our report has the following content:

1. Chapter 1: Executive Summary
2. Chapter 2: Exercise 2.5.1
   (a) Section 2.1 Problem Statement
   (b) Section 2.2 Relevant Code
   (c) Section 2.3 Test Cases
   (d) Section 2.4 Test Results
3. Chapter 3 Exercise 3.4.1
   (a) Section 3.1 Problem Statement
   (b) Section 3.2 Relevant Code
   (c) Section 3.3 Test Results
4. Chapter 4 Exercise 3.4.2
   (a) Section 4.1 Problem Statement
   (b) Section 4.2 Relevant Code
   (c) Section 4.3 Test Results

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

Chapter 2

Exercise 2.5.1

2.1 Problem Statement

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

$$timesPlus\ x\ y = (x * y,\ x + y)$$

2.2 Relevant Code

```
fun timesPlus x y = (x*y, x+y);
```

2.3 Test Cases

The required test cases are:

```
(***************************************************************************)
(* Test Cases*)
(***************************************************************************)
timesPlus 100 27;
timesPlus 10 26;
timesPlus 1 25;
timesPlus 2 24;
timesPlus 30 23;
timesPlus 50 200;
```

2.4 Test Results

1

```
> > > > val timesPlus = fn: int -> int -> int * int
>
> timesPlus 100 27;
val it = (2700, 127): int * int
> timesPlus 10 26;
val it = (260, 36): int * int
> timesPlus 1 25;
val it = (25, 26): int * int
> timesPlus 2 24;
val it = (48, 26): int * int
> timesPlus 30 23;
val it = (690, 53): int * int
> timesPlus 50 200;
val it = (10000, 250): int * int
>
>
```

Chapter 3

Exercise 3.4.1

3.1 Problem Statement

In this exercise, we will be solving the pattern matching:

$$\begin{aligned}
&val\ listA\ =\ [(0,\text{"}Alice\text{"}),\ (1,\text{"}Bob\text{"}),\ (3,\text{"}Carol\text{"}),\ (4,\text{"}Dan\text{"})];\\
&val\ elB :: listB\ =\ listA;\\
&val\ (e1C1, e1C2)\ =\ elB;\\
&val\ [e1C3, e1C4, e1C5]\ =\ listB;
\end{aligned}$$

3.2 Relevant Code

```
val listA = [(0,"Alice"), (1,"Bob"), (3,"Carol"),(4,"Dan")];

val elB::listB = listA;
val (e1C1,e1C2) = elB;
val [e1C3, e1C4, e1C5] = listB;
```

3.3 Test Results

```
> > > > val listA = [(0, "Alice"), (1, "Bob"), (3, "Carol"), (4, "Dan")]:
   (int * string) list
> val elB = (0, "Alice"): int * string
val listB = [(1, "Bob"), (3, "Carol"), (4, "Dan")]: (int * string) list
> val e1C1 = 0: int
val e1C2 = "Alice": string
> val e1C3 = (1, "Bob"): int * string
val e1C4 = (3, "Carol"): int * string
val e1C5 = (4, "Dan"): int * string
>
```

1

Chapter 4

Exercise 3.4.2

4.1 Problem Statement

In this exercise we will evaluate the following assignment statements and provide the reason in case if there are any errors.

$$\begin{aligned}
val\ (x1, x2, x3) &= (1, true, "Alice");\\
val\ pair1 &= (x1, x3);\\
val\ list1 &= [0, x1, 2];\\
val\ list2 &= [x2, x1];\\
val\ list3 &= (1\ ::\ [x3]);
\end{aligned}$$

4.2 Relevant Code

```
val (x1,x2,x3) = (1,true,"Alice");
val pair1 = (x1,x3);
val list1 = [0,x1,2];
val list2 = [x2,x1];
val list3 = (1 :: [x3]);
```

4.3 Test Results

```
> > > >                                                                    1
> val x1 = 1: int
val x2 = true: bool
val x3 = "Alice": string
> val pair1 = (1, "Alice"): int * string
> val list1 = [0, 1, 2]: int list
> poly: : error: Elements in a list have different types.
   Item 1: x2 : bool
   Item 2: x1 : int
   Reason:
      Can't unify bool (*In Basis*) with int (*In Basis*)
         (Different type constructors)
Found near [x2, x1]
Static Errors
> poly: : error: Type error in function application.
   Function: :: : int * int list -> int list
   Argument: (1, [x3]) : int * string list
   Reason:
      Can't unify int (*In Basis*) with string (*In Basis*)
         (Different type constructors)
Found near (1 :: [x3])
Static Errors
>
```

4.3.1 Explanation for Errors

The errors occured in the statements are due to:

- val list2 = [x2, x1]; is due to creating a list with different types, where x2 is a boolean and x1 is a integer. A list will take similar data types.
- val list3 = (1 :: [x3]); is due to creating a list with different types. 1 is a integer and x3 is a string type. HOL cannot create a list of two different data types.

Chapter 5

Appendix A: Exercise 2.5.1

The following code is from the file ex-2-5-1.sml

```
(* Name: Bharath Karumudi *)
(* Email: bhkarumu@syr.edu *)

fun timesPlus x y = (x*y, x+y);
```

Chapter 6

Appendix B: Exercise 3.4.1

The following code is from the file ex-3-4-1.sml

```
(***************************************************************************)
(* Exercise 3.4.1 *)
(* Author: Bharath Karumudi *)
(* Date: Jul 11, 2019 *)
(***************************************************************************)

val listA = [(0,"Alice"), (1,"Bob"), (3,"Carol"),(4,"Dan")];

val elB::listB = listA;
val (e1C1,e1C2) = elB;
val [e1C3, e1C4, e1C5] = listB;
```

Chapter 7

Appendix C: Exercise 3.4.2

The following code is from the file ex-3-4-2.sml

```
(*************************************************************************)
(* Exercise 3.4.2 *)
(* Author: Bharath Karumudi *)
(* Date: Jul 11, 2019 *)
(*************************************************************************)

val (x1,x2,x3) = (1,true,"Alice");
val pair1 = (x1,x3);
val list1 = [0,x1,2];
val list2 = [x2,x1];
val list3 = (1 :: [x3]);

(************************************ Errors ************************************)
(* val list2 = [x2,x1]; is due to creating a list with different types, where x2 is a *)
(*     boolean and x1 is a integer. A list will take similar data types. *)
(* val list3 = (1 :: [x3]); is due to creating a list with different types. 1 is a int *)
(*     and x3 is a string type. HOL cannot create a list of two different data types. *)
(*********************************************************************************)
```