Report for Project3

Xiaozhi Li

19 September 2017

Abstract

Project 3 using LaTeX. This project demonstrate some equations using ML.

- Problem statement
- Relevant code
- Test results

For each problem there will be a source code included in the Appendix. They are generated directly using the original source file, hence the changes in the file are represent on the report.

The style files 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.

Acknowledgments: This report follows the hand book Certified Security by Design Using Higer Order Logic, and course instructions from CIS400-CSBD.

Contents

Chapter 1

Executive Summary

All requirements for this project are satisfied. Specifically,

Report Contents
Our report has the following content: Our report has the following content:

Reproducibility in ML and LaTeX

Our ML and LaTeX source files compile with no errors.

Chapter 2

Exercise 4.6.3

2.1 Problem Statement

In this exercise we declared 10 functions to evaluate 5 problems from Exercise 4.6.2 in the Text book, each of the problems has two ML functions, the first using *fn* and *val* to define and name the function, the second using *fun* to define and name the function.

2.1.1 4.6.3A

In problem A, we have A function that takes a 3-tuple of integers (x, y, z) as input and returns the value corresponding to the sum x + y + z. Though we will demonstrate in two different forms later, the original function is still:

$$funA\ (x,\ y,\ z)\ =\ (x\ +\ y\ +\ z)$$

2.1.2 4.6.3A relevent code

The following code are used in ML, notice there are two functions, funA1 and funA2, and they are supposed to have the same result.

```
val funA1 = (fn (x,y,z) => x*y*z);
fun funA2 (x,y,z) = x*y*z;
```

2.1.3 4.6.3B

In problem B, we have A function that takes two integer inputs x and y (where x is supplied first followed by y) and returns the boolean value corresponding to /emphx larger than y.Though we will demonstrate in two different forms later, the original function is still:

$$funB\ \ x\,y\ =\ x\ >\ y$$

2.1.4 4.6.3B relevent code

The following code are used in ML, notice there are two functions, funB1 and funB2, and they are supposed to have the same result.

```
val funB1 = (fn x=> (fn y => x<y));
fun funB2 x y = x<y;
```

2.1.5 4.6.3C

In problem C, we hgave A function that takes two strings s1 and s2 (where s1 is supplied first followed by s2) and concatentates them, where denotes string concatenation:

$$funC\ \ s1\, s2\ =\ s1 \wedge s2$$

2.1.6 4.6.3C relevent code

The following code are used in ML, notice there are two functions, funC1 and funC2, and they are supposed to have the same result.

```
val funC1 = (fn s1=> (fn s2 => s1^s2));
fun funC2 s1 s2=s1^s2;
```

2.1.7 4.6.3D

In proiblem D, we have:

```
 A function that takes two lists list 1 and list 2 (where list 1
 comes first) and appends them, where '@' denotes list append. For
 example [true,false] @ [false, false, false] results in the list
 [true,false,false,false,false].
```

hence the function should be:

$$funD\ \ list1\, list2\ =\ list1\ @\ list2$$

2.1.8 4.6.3D relevent code

The following code are used in ML, notice there are two functions, funD1 and funD2, and they are supposed to have the same result.

```
val funD1= ( fn list1 => (fn list2 => list1@list2));
fun funD2 list1 list2 = list1@list2;
```

2.1.9 4.6.3E

```
 function that takes a pair of integers (x, y) and returns the larger
 of the two values. You note that the conditional statement if
 condition then a else b returns a if condition is true, otherwise
 it returns b.
```

2.1.10 4.6.3E relevent code

The following code are used in ML, notice there are two functions, funE1 and funE2, and they are supposed to have the same result.

```
val funE1= (fn (x,y) => if (x>y) then x else y);
fun funE2 (x,y) = if (x>y) then x else y;
```

2.1.11 Test Cases

The following code are for testing our results:

```
  fun test463A f1 f2 inList =
let
  val list1 = map f1 inList
  val list2 = map f2 inList
in
 foldr
  (fn (x,y)  => (x andalso y))
 true
  (ListPair.map (fn (x,y) => x = y) (list1 ,list2))
end;

fun f2P f (x,y) = f x y

fun test463B f1 f2 inList =
let
 val list1 = map (f2P f1) inList
 val list2 = map (f2P f2) inList
in
 foldr
 (fn (x,y)  => (x andalso y))
 true
 (ListPair.map (fn (x,y) => x = y) (list1 ,list2))
end
```

The following is the test case functions for each problem, notice the final results will generate a boolean equation, if that equation evaluates to be true, then the test is passed:

```
(* Part A *)
(*********)

(* ========================================================== *)
val testListA = [(1,2,3),(4,5,6),(7,8,9)]

val outputsA = map funA2 testListA

val testResultA = test463A funA1 funA2 testListA


(*********)
(* Part B *)
(*********)

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

```
val testListB = [(0,0),(1,2),(4,3)]

val outputsB = map (f2P funB1) testListB

val testResultB = test463B funB1 funB2 testListB


(**********)
(* Part C *)
(**********)

(* ================================================================== *)

val testListC = [("Hi","␣there!"),("Oh␣","no!"),("What","␣the␣...")]

val outputsC = map (f2P funC1) testListC

val testResultC = test463B funC1 funC2 testListC


(**********)
(* Part D *)
(**********)

(* ================================================================== *)

val testListD1 = [([0,1],[2,3,4]),([],[0,1])]
val testListD2 = [([true,true],[])]

val outputsD1 = map (f2P funD1) testListD1
val outputsD2 = map (f2P funD2) testListD2

val testResultD1 = test463B funD1 funD2 testListD1
val testResultD2 = test463B funD1 funD2 testListD2


(**********)
(* Part E *)
(**********)

(* ================================================================== *)

val testListE = [(2,1),(5,5),(5,10)]

val sampleResultE = map funE1 testListE

val testResultE = test463A funE1 funE2 testListE

val finalResult= [testResultA,testResultB,testResultC,testResultD1,testResultD2,testResultE
```

2.2 Execution Transcripts

We sent these code to HOL, and here is the output from HOL:

1

```
---------------------------------------------------------------------
       HOL-4 [Kananaskis 11 (stdknl, built Sat Aug 19 09:30:06 2017)]

       For introductory HOL help, type: help "hol";
       To exit type <Control>-D
---------------------------------------------------------------------
> > > > > > val test463A = fn: (’a -> ’’b) -> (’a -> ’’b) -> ’a list -> bool
val f2P = fn: (’a -> ’b -> ’c) -> ’a * ’b -> ’c
val test463B = fn:
   (’a -> ’b -> ’’c) -> (’a -> ’b -> ’’c) -> (’a * ’b) list -> bool
val funA1 = fn: int * int * int -> int
val funA2 = fn: int * int * int -> int
val funB1 = fn: int -> int -> bool
val outputsA = [6, 120, 504]: int list
val testListA = [(1, 2, 3), (4, 5, 6), (7, 8, 9)]: (int * int * int) list
val testResultA = true: bool
val funB2 = fn: int -> int -> bool
val funC1 = fn: string -> string -> string
val outputsB = [false, true, false]: bool list
val testListB = [(0, 0), (1, 2), (4, 3)]: (int * int) list
val testResultB = true: bool
val funC2 = fn: string -> string -> string
val funD1 = fn: ’a list -> ’a list -> ’a list
val outputsC = ["Hi there!", "Oh no!", "What the ..."]: string list
val testListC = [("Hi", " there!"), ("Oh ", "no!"), ("What", " the ...")]:
   (string * string) list
val testResultC = true: bool
val funD2 = fn: ’a list -> ’a list -> ’a list
val funE1 = fn: int * int -> int
val outputsD1 = [[0, 1, 2, 3, 4], [0, 1]]: int list list
val outputsD2 = [[true, true]]: bool list list
val testListD1 = [([0, 1], [2, 3, 4]), ([], [0, 1])]:
   (int list * int list) list
val testListD2 = [([true, true], [])]: (bool list * ’a list) list
val testResultD1 = true: bool
val testResultD2 = true: bool
val funE2 = fn: int * int -> int
val sampleResultE = [2, 5, 10]: int list
val testListE = [(2, 1), (5, 5), (5, 10)]: (int * int) list
val testResultE = true: bool
val it = (): unit
>
*** Emacs/HOL command completed ***

> val finalResult = [true, true, true, true, true, true]: bool list


>
```

2.2.1 explain of result

The overall result script is messy, but we can look at *testResult* by utilizing the *finalResult* function’s result, this is a list of all the results we have, since it is all true, that means our evaluations all passed HOL. 4.6.3 A-E all evaluated to be true.

Chapter 3

Exercise 4.6.4

3.1 Problem Statement

In 4.6.4, we were given a problem:

```
In ML, define a function listSquares that when applied to the empty
list of integers returns the empty list , and when applied to a
non-empty list of integers returns a list where each element is
squared. For example, listSquares [2,3,4] returns [4,9,16]. Define
the function using a let expression in ML. A function that takes two
lists list 1 and list 2 (where list 1 comes first) and appends them,
where '@' denotes list append. For example [true,false] @ [false,
false, false] results in the list [true,false,false,false,false].
```

3.1.1 Relevent Code

The problem asks for a function that has ability to square a list of integers and give results as a list, and this function has to be able to accept an empty list and return an empty list as a result. Hence the followiong function was used:

```
fun listSquares list=
let
 fun square x= x*x
in
 map square list
end;val funE1= (fn (x,y) => if (x>y) then x else y);
fun funE2 (x,y) = if (x>y) then x else y;
```

3.1.2 Test cases

we used the following code for testing:

```
val testList = [1,2,3,4,5]

val testResults = listSquares testList
```

3.1.3 Execution Transcripts

Here is the transcripit for 4.6.4:

```
# # # # # val listSquares = fn: int list -> int list                                              1
> # # # # # val testList = [1, 2, 3, 4, 5]: int list
val testResults = [1, 4, 9, 16, 25]: int list
>
```

3.1.4 Test Result

The transcript result shows our tests has been passed.

Chapter 4

Exercise 5.3.4

4.1 Problem Statement

In 5.3.4, we are asked to define a function that behaves as a filter.

```
Define a function Filter in ML, whose behavior is identical to
filter. Note: you cannot use filter in the definition of
Filter. However, you can adapt the definition of filter and use it
in your definition.  Show test cases of your function returning the
expected results by comparing the outputs of both Filter and
filter. Your examples should include the cases in Exercise 5.3.3.
```

The problem asks for a different version of filter, and the original filter function is defined as:

$$\begin{aligned} \mathit{filter}\ P\ [\,] &= [\,] \\ \mathit{filter}\ P\ (x :: xs) &= \mathit{if}\ P\ x\ \mathit{then}\ x\ ::\ (\mathit{filter}\ P\ xs)\ \mathit{else}\ (\mathit{filter}\ P\ xs) \end{aligned}$$

4.1.1 Relevent Code

```
fun Filter p list=
let

(*Helper A change x into a list of bools *)

fun helperA p []=[]
  | helperA p xs=map p xs

fun helperB [] anything=[]
  | helperB anything []=[]
  | helperB (b::bs) (x::xs)=if b then x::(helperB bs xs) else helperB bs xs
in
helperB (helperA p list) list
end;fun listSquares list=
let
 fun square x= x*x
in
 map square list
end;val funE1= (fn (x,y) => if (x>y) then x else y);
fun funE2 (x,y) = if (x>y) then x else y;
```

4.1.2 Test Cases

we used the following code for testing:

```
val testResults = Filter (fn x => x < 5) [1,2,3,4,5,6,7,8,9]

val testResults2 =Filter (fn x=> x<5)[4,6]Nothing in 5.3.4
```

4.1.3 Execution Transcripts

```
val Filter = fn: ('a -> bool) -> 'a list -> 'a list
val testResults = [1, 2, 3, 4]: int list
val testResults2 = [4]: int list
val it = (): unit
>
*** Emacs/HOL command completed ***

>
```

4.1.4 Test result

The test result shows that all of our tests have been passed.

Chapter 5

Exercise 5.3.5

5.1 Problem Statement

In 5.3.5 we are asked to define an ML function, addPairsGreaterThan n list, which takes an interger n, a list of pairs of integers list, then the function will return a list of integers where each element is the sum of integer pairs in list, while both elements in the pair are greater than n.

5.1.1 Solution

This problem can be break into two parts, part one takes a list of pairs and filter out the ones that both of the pair is greater than n. part two gives the sum of those pairs and return them as a list.

5.2 Relevent Code

The following code was then created:

```
fun addPairsGreaterThan n list=
let
fun sumList []=[]
  | sumList ((x,y)::xs)=(x+y):: (sumList xs)
fun helper n (x,y)= (x>n andalso y>n)
in
sumList( filter (helper n) list)
end;
```

5.3 Test Cases

We used the following code to test the function:

```
addPairsGreaterThan 0 [(0,1),(2,0),(2,3),(4,5)];
```

5.4 Execution Transcripts

1

```
# # # # # # # # val addPairsGreaterThan = fn: int -> (int * int) list -> int list
> # # # val it = [5, 9]: int list
>
*** Emacs/HOL command completed ***
```

5.5 Test Result

By the result, it is proven that our function has passed the test.

Chapter 6

Appendix A: Exercise 4.6.3 Source Code

The following code is from *ex-4-6-3Tests.sml*

```
(*******************************************************************************)
(* Exercise 4.6.3*)
(* Modified by Xiaozhi Li                                                      *)
(* Date: 17 September 2017                                                     *)
(*******************************************************************************)



(*******************************************************************************)
(* Test functions you will need.                                               *)
(*                                                                             *)
(*                                                                             *)
(*******************************************************************************)

fun test463A f1 f2 inList =
let
 val list1 = map f1 inList
 val list2 = map f2 inList
in
 foldr
 (fn (x,y)  => (x andalso y))
 true
 (ListPair.map (fn (x,y) => x = y) (list1,list2))
end;

fun f2P f (x,y) = f x y

fun test463B f1 f2 inList =
let
 val list1 = map (f2P f1) inList
 val list2 = map (f2P f2) inList
in
 foldr
 (fn (x,y)  => (x andalso y))
 true
 (ListPair.map (fn (x,y) => x = y) (list1,list2))
end;


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

```
(* Part A *)
(**********)

(* ================================================================ *)
(*function A funA1 funA2*)
val funA1 = (fn (x,y,z) => x*y*z);
fun funA2 (x,y,z) = x*y*z;

(* ================================================================ *)

val testListA = [(1,2,3),(4,5,6),(7,8,9)]

val outputsA = map funA2 testListA

val testResultA = test463A funA1 funA2 testListA

(**********)
(* Part B *)
(**********)

(* ================================================================ *)
(*  code for funB1 and funB2 here.                                *)
val funB1 = (fn x=> (fn y => x<y));
fun funB2 x y = x<y;
(* ================================================================ *)

val testListB = [(0,0),(1,2),(4,3)]

val outputsB = map (f2P funB1) testListB

val testResultB = test463B funB1 funB2 testListB

(**********)
(* Part C *)
(**********)

(* ================================================================ *)
(*                                                                  *)
(* code for funC1 and funC2 here.                                 *)
val funC1 = (fn s1=> (fn s2 => s1^s2));
fun funC2 s1 s2=s1^s2;
(* ================================================================ *)

val testListC = [("Hi","␣there!"),("Oh␣","no!"),("What","␣the␣...")]

val outputsC = map (f2P funC1) testListC

val testResultC = test463B funC1 funC2 testListC


(**********)
(* Part D *)
(**********)
```

```
(* ======================================================================== *)
(*                                                                          *)
(* code for funD1 and funD2 here.                                  *)

val funD1= ( fn list1 => (fn list2 => list1@list2));
fun funD2 list1 list2 = list1@list2;
(* ======================================================================== *)

val testListD1 = [([0,1],[2,3,4]),([],[0,1])]
val testListD2 = [([true,true],[])]

val outputsD1 = map (f2P funD1) testListD1
val outputsD2 = map (f2P funD2) testListD2

val testResultD1 = test463B funD1 funD2 testListD1
val testResultD2 = test463B funD1 funD2 testListD2

(**********)
(* Part E *)
(**********)

(* ======================================================================== *)
(*                                                                          *)
(* code for funE1 and funE2 here.                                  *)

val funE1= (fn (x,y) => if (x>y) then x else y);
fun funE2 (x,y) = if (x>y) then x else y;
(* ======================================================================== *)

val testListE = [(2,1),(5,5),(5,10)]

val sampleResultE = map funE1 testListE

val testResultE = test463A funE1 funE2 testListE

val finalResult= [testResultA,testResultB,testResultC,testResultD1,testResultD2,testResultE
```

Chapter 7

Appendix B: Exercise 4.6.4 Source Code

The following code is from *ex-4-6-4Tests.sml*

```
(************************************************************************)
(* Exercise 4.6.4                                                      *)
(* Author: Shiu-Kai Chin*)
(* Date: 19 September 2017*)
(* Modified by Xiaozhi Li *)
(************************************************************************)

(* ================================================================== *)
(*                                                                     *)
(*code for listSquares here                                    *)
fun listSquares list=
let
 fun square x= x*x
in
 map square list
end;
(* ================================================================== *)


val testList = [1,2,3,4,5]

val testResults = listSquares testList
```

Chapter 8

Appendix C: Exercise 5.3.4 Source Code

The following code is from *ex-5-3-4Tests.sml*

```
(**************************************************************************)
(* Exercise 5.3.4                                                         *)
(*Modified by Xiaozhi LI                                                  *)
(* Date: 19 September 2017                                                *)
(**************************************************************************)

(* ==================================================================== *)
(*code of filter                                                          *)
fun Filter p list=
let

(*Helper A change x into a list of bools *)

fun helperA p []=[]
  | helperA p xs=map p xs

fun helperB [] anything=[]
  | helperB anything []=[]
  | helperB (b::bs) (x::xs)=if b then x::(helperB bs xs) else helperB bs xs
in
helperB (helperA p list) list
end;
(* ==================================================================== *)


val testResults = Filter (fn x => x < 5) [1,2,3,4,5,6,7,8,9]

(*specified tests*)
val testResults2 =Filter (fn x=> x<5)[4,6]
```

Chapter 9

Appendix D: Exercise 5.3.5 Source Code

The following code is from *ex-5-3-5Tests.sml*

```
(**************************************************************************)
(* Exercise 5.3.5                                                         *)
(* Author: Shiu-Kai Chin*)
(* Modified by Xiaozhi Li*)
(* Date: 20 September 2017                                                *)
(**************************************************************************)

(* ====================================================================== *)
(*                                                                        *)

fun addPairsGreaterThan n list=
let
fun sumList []=[]
  | sumList ((x,y)::xs)=(x+y):: (sumList xs)
fun helper n (x,y)= (x>n andalso y>n)
in
sumList( filter (helper n) list)
end;
(*                                                                        *)
(* ====================================================================== *)

addPairsGreaterThan 0 [(0,1),(2,0),(2,3),(4,5)];
```