Project 3

Bharath Karumudi

August 19, 2019

Abstract

This project is to demonstrate the capabilities of implementiong constructing and deconstructing HOL Terms using the tools and techniques - LaTeX, AcuTeX, emacs and ML.

Each chapter documents the given problems with a structure of:

1. Problem Statement
2. Relevant Code
3. Execution Transcripts
4. Explanation of results

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, and by using HOL proved the below theorems:

Contents

Our report has the following content:

Chapter 8:

[conjSymThm]

$\vdash p \wedge q \iff q \wedge p$

[conjSymThmAll]

$\vdash \forall p\ q.\ p \wedge q \iff q \wedge p$

[problem1Thm]

$\vdash p \Rightarrow (p \Rightarrow q) \Rightarrow (q \Rightarrow r) \Rightarrow r$

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

Chapter 2

Exercise 7.3.1

2.1 Problem Statement

In this exercise we need to create a function *andImp2Imp term*, which will take:

$$p \wedge q \subset r$$

and results to:

$$p \subset q \subset r;$$

2.2 Relevant Code

```
fun andImp2Imp term =
let
 val(conjTerm,r)= dest_imp(term)
 val(p,q) = dest_conj(conjTerm)

in
   mk_imp(p,(mk_imp(q,r)))
end;

(**** Test Case *****)
andImp2Imp ``(p/\q) ==> r``
```

2.3 Test Cases

The required test cases are:

```
andImp2Imp ``(p/\q) ==> r``
```

2.4 Execution Transcripts

```
---------------------------------------------------------------------
       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
---------------------------------------------------------------------
> > > > # # # # # # # # # ** types trace now on
> *** Globals.show_assums now true ***
> # # # # # # # # # ** Unicode trace now off
>
> # # # # # # # val andImp2Imp = fn: term -> term
> >
>
> andImp2Imp ``(p/\q) ==> r``;
val it =
   ``(p :bool) ==> (q :bool) ==> (r :bool)``:
   term
>
```
1

2.4.1 Explanation of Results

The above test results shows the test case has been passed.

Chapter 3

Exercise 7.3.2

3.1 Problem Statement

In this exercise, we have to create *andImp2Imp term*, which takes the term

$$p \subset q \subset r;$$

and results to:

$$p \wedge q \subset r$$

and also should act as a reverse function for 7.3.1

3.2 Relevant Code

```
(**** Function andImp2Imp ~ same as from 7.3.1  ****)

fun andImp2Imp term =
let
 val(conjTerm,r)= dest_imp(term)
 val(p,q) = dest_conj(conjTerm)

in
  mk_imp(p,(mk_imp(q,r)))
end;

(**** Function impImpAnd  ****)

fun impImpAnd term =
let

 val(term1,imp) = dest_imp(term)
 val(term2, term3) = dest_imp(imp)
 val new_conj = mk_conj(term1, term2)
in
 mk_imp(new_conj,term3)
end;

(****** Test Cases **********)

andImp2Imp ``(p/\q) ==> r``
impImpAnd ``p ==> q ==> r``;

impImpAnd(andImp2Imp ``(p/\q) ==> r``);
andImp2Imp(impImpAnd ``p==>q==>r``);
```

3.3 Test Cases

The required test cases are:

```
andImp2Imp ``(p/\q) ==> r``
impImpAnd ``p ==> q ==> r``;
impImpAnd(andImp2Imp ``(p/\q) ==> r``);
andImp2Imp(impImpAnd ``p==>q==>r``);
```

3.4 Execution Transcripts

```
---------------------------------------------------------------------
       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
---------------------------------------------------------------------
> > > >
> # # # # # # # # # ** types trace now on
> *** Globals.show_assums now true ***
> # # # # # # # # # ** Unicode trace now off
>
> # # # # # # # val andImp2Imp = fn: term -> term
> # # # # # # # # val impImpAnd = fn: term -> term
> >
> val it =
   ``(p :bool) ==> (q :bool) ==> (r :bool)``:
   term
> val it =
   ``(p :bool) /\ (q :bool) ==> (r :bool)``:
   term
> val it =
   ``(p :bool) /\ (q :bool) ==> (r :bool)``:
   term
> val it =
   ``(p :bool) ==> (q :bool) ==> (r :bool)``:
   term
>
```
1

3.4.1 Explanation of Results

The above transcript shows the given test cases has been passed.

Chapter 4

Exercise 7.3.3

4.1 Problem Statement

In this exercise we have to create a function *notExists term* which takes the term $\neg\exists x.P(x)$ and returns $\forall x.\neg P(x)$.

4.2 Relevant Code

```
fun notExists term =
let
 val (t1, t2) = dest_exists(dest_neg(term))
in
 mk_forall(t1,t2)
end;


(****** Test Cases **********)
notExists ``~?z.Q z``;
```

4.3 Test Cases

The required test cases are:

```
notExists ``~?z.Q z``;
```

4.4 Execution Transcripts

```
---------------------------------------------------------------------
       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
---------------------------------------------------------------------
> > > >
>
> # # # # # # # # # ** types trace now on
> *** Globals.show_assums now true ***
> # # # # # # # # # ** Unicode trace now off
>
> # # # # # val notExists = fn: term -> term
> >
> <<HOL message: inventing new type variable names: 'a>>
val it =
   ``!(z :'a). (Q :'a -> bool) z``:
   term
>
```
1

4.4.1 Explanation of Results

The above transcript shows the given tests has been passed.

Chapter 5

Exercise 8.4.1

5.1 Problem Statement

In this exercise we need to prove the theorem: $\vdash p \Rightarrow (p \Rightarrow q) \Rightarrow (q \Rightarrow r) \Rightarrow r$

5.2 Relevant Code

```
val problem1Thm =
let
 val th1 = ASSUME ``p:bool``
 val th2 = ASSUME ``p ==> q``
 val th3 = ASSUME ``q ==> r``
 val th4 = MP th2 th1
 val th5 = MP th3 th4
 val th6 = DISCH (hd(hyp th3)) th5
 val th7 = DISCH (hd(hyp th2)) th6
in
  DISCH (hd(hyp th1)) th7
end;

val _ = save_thm("problem1Thm", problem1Thm);
```

5.3 Execution Transcripts

```
---------------------------------------------------------------------
       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
---------------------------------------------------------------------
> > > > # # # # # # # # # ** types trace now on
> *** Globals.show_assums now true ***
> # # # # # # # # # ** Unicode trace now off
>
>
> # # # # # # # # # # # val problem1Thm =
    [] |- (p :bool) ==> (p ==> (q :bool)) ==> (q ==> (r :bool)) ==> r:
   thm
>
> >
```
1

5.3.1 Explanation of Results

The above results shows that theorem is proved.

Chapter 6

Exercise 8.4.2

6.1 Problem Statement

In this exercise we need to prove the theorem: $\vdash \forall p\ q.\ p \wedge q \iff q \wedge p$

6.2 Relevant Code

```
val conj1Thm =
let
 val th1 = ASSUME ``p /\ q``
 val th2 = CONJUNCT1 th1
 val th3 = CONJUNCT2 th1
 val th4 = CONJ th3 th2
in
 DISCH (hd(hyp th1)) th4
end;

val conj2Thm =
let
 val th1 = ASSUME ``q /\ p``
 val th2 = CONJUNCT1 th1
 val th3 = CONJUNCT2 th1
 val th4 = CONJ th3 th2
in
 DISCH (hd(hyp th1)) th4
end;

val conjSymThm =
IMP_ANTISYM_RULE conj1Thm conj2Thm;

val _ = save_thm("conjSymThm", conjSymThm);
```

6.3 Execution Transcripts

```
---------------------------------------------------------------------
       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
---------------------------------------------------------------------
> > > > # # # # # # # # # ** types trace now on
> *** Globals.show_assums now true ***
> # # # # # # # # # ** Unicode trace now off
>
> # # # # # # # # val conj1Thm =
    [] |- (p :bool) /\ (q :bool) ==> q /\ p:
   thm
> > # # # # # # # # val conj2Thm =
    [] |- (q :bool) /\ (p :bool) ==> p /\ q:
   thm
> > # val conjSymThm =
    [] |- (p :bool) /\ (q :bool) <=> q /\ p:
   thm
> > > >
*** Emacs/HOL command completed ***

>
```

1

6.3.1 Explanation of Results

The above results shows that theorem is proved.

Chapter 7

Exercise 8.4.3

7.1 Problem Statement

In this exercise we need to prove the theorem: $\vdash p \Rightarrow (p \Rightarrow q) \Rightarrow (q \Rightarrow r) \Rightarrow r$

7.2 Relevant Code

```
val conj1Thm =
let
val th1 = ASSUME ``p /\ q``
val th2 = CONJUNCT1 th1
val th3 = CONJUNCT2 th1
val th4 = CONJ th3 th2
in
DISCH (hd(hyp th1)) th4
end;

val conj2Thm =
let
val th1 = ASSUME ``q /\ p``
val th2 = CONJUNCT1 th1
val th3 = CONJUNCT2 th1
val th4 = CONJ th3 th2
in
DISCH (hd(hyp th1)) th4
end;

val conjSymThm =
IMP_ANTISYM_RULE conj1Thm conj2Thm;

val conjSymThmAll = GENL[``p:bool``, ``q:bool``] conjSymThm;

val _ = save_thm("conjSymThmAll", conjSymThmAll)
```

7.3 Execution Transcripts

```
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
---------------------------------------------------------------------
> > > > # # # # # # # # # ** types trace now on
> *** Globals.show_assums now true ***
> # # # # # # # # # ** Unicode trace now off
>
> # # # # # # # # val conj1Thm =
    [] |- (p :bool) /\ (q :bool) ==> q /\ p:
   thm
> > # # # # # # # # val conj2Thm =
    [] |- (q :bool) /\ (p :bool) ==> p /\ q:
   thm
> > # val conjSymThm =
    [] |- (p :bool) /\ (q :bool) <=> q /\ p:
   thm
> > val conjSymThmAll =
    [] |- !(p :bool) (q :bool). p /\ q <=> q /\ p:
   thm
> > # >
*** Emacs/HOL command completed ***

>
```

7.3.1 Explanation of Results

The above results shows that theorem is proved.

Chapter 8

Appendix A: Chapter 7

The following code is from the file chapter7Answers.sml

```
(*************************************************************************)
(* Exercise 7..3                                                        *)
(* Author: Bharath Karumudi                                              *)
(* Date: Jul 25, 2019                                                    *)
(************************************************************************


(*************************************************************************)
(* Exercise 7.3.1                                                        *)
(*************************************************************************)

fun andImp2Imp term =
let
 val(conjTerm,r)= dest_imp(term)
 val(p,q) = dest_conj(conjTerm)

in
   mk_imp(p,(mk_imp(q,r)))
end;

(**** Test Case *****)
andImp2Imp ``(p/\q) ==> r``


(*************************************************************************)
(* Exercise 7.3.2                                                        *)
(*************************************************************************)

(**** Function andImp2Imp ~ same as from 7.3.1  ****)

fun andImp2Imp term =
let
 val(conjTerm,r)= dest_imp(term)
 val(p,q) = dest_conj(conjTerm)

in
   mk_imp(p,(mk_imp(q,r)))
end;

(**** Function impImpAnd  ****)

fun impImpAnd term =
```

```
let

  val(term1,imp) = dest_imp(term)
  val(term2, term3) = dest_imp(imp)
  val new_conj = mk_conj(term1, term2)
in
 mk_imp(new_conj,term3)
end;

(****** Test Cases **********)

andImp2Imp ``(p/\q) ==> r``
impImpAnd ``p ==> q ==> r``;

impImpAnd(andImp2Imp ``(p/\q) ==> r``);
andImp2Imp(impImpAnd ``p==>q==>r``);


(*************************************************************************)
(* Exercise 7.3.3                                                        *)
(*************************************************************************)


fun notExists term =
let
 val (t1, t2) = dest_exists(dest_neg(term))
in
 mk_forall(t1,t2)
end;


(****** Test Cases **********)
notExists ``~?z.Q z``;
```

Chapter 9

Appendix B: Chapter 8

The following code is from the file project3bScript.sml

```
(*************************************************************************)
(*   Exercise: Chapter 8                                                 *)
(*   Author: Bharath Karumudi                                            *)
(*   Date: Jul 26, 2019                                                  *)
(*************************************************************************)

structure chapter8Script = struct
open HolKernel Parse boolLib bossLib;

val _ = new_theory "chapter8";

(*************************************************************************)
(*   Exercise: 8.4.1                                                     *)
(*   val problem1Thm =                                                   *)
(*       [] |- p ==> (p ==> q) ==> (q ==> r) ==> r                       *)
(*       : thm                                                           *)
(*                                                                       *)
(*************************************************************************)

val problem1Thm =
let
 val th1 = ASSUME ``p:bool``
 val th2 = ASSUME ``p ==> q``
 val th3 = ASSUME ``q ==> r``
 val th4 = MP th2 th1
 val th5 = MP th3 th4
 val th6 = DISCH (hd(hyp th3)) th5
 val th7 = DISCH (hd(hyp th2)) th6
in
  DISCH (hd(hyp th1)) th7
end;

val _ = save_thm("problem1Thm", problem1Thm);


(*************************************************************************)
(* Exercise: 8.4.2                                                       *)
(* val conjSymThm =                                                      *)
(*     [] |- p /\ q <=> q /\ p                                           *)
(*     : thm
*)
(*                                                                       *)
```

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

val conj1Thm =
let
 val th1 = ASSUME ``p /\ q``
 val th2 = CONJUNCT1 th1
 val th3 = CONJUNCT2 th1
 val th4 = CONJ th3 th2
in
 DISCH (hd(hyp th1)) th4
end;

val conj2Thm =
let
 val th1 = ASSUME ``q /\ p``
 val th2 = CONJUNCT1 th1
 val th3 = CONJUNCT2 th1
 val th4 = CONJ th3 th2
in
 DISCH (hd(hyp th1)) th4
end;

val conjSymThm =
IMP_ANTISYM_RULE conj1Thm conj2Thm;

val _ = save_thm("conjSymThm", conjSymThm);


(**************************************************************************)
(* Exercise: 8.4.3                                                        *)
(* val conjSymThmAll =                                                    *)
(*     [] |- !p q. p /\ q <=> q /\ p                                      *)
(*     : thm                                                              *)
(*                                                                        *)
(**************************************************************************)

val conj1Thm =
let
val th1 = ASSUME ``p /\ q``
val th2 = CONJUNCT1 th1
val th3 = CONJUNCT2 th1
val th4 = CONJ th3 th2
in
DISCH (hd(hyp th1)) th4
end;

val conj2Thm =
let
val th1 = ASSUME ``q /\ p``
val th2 = CONJUNCT1 th1
val th3 = CONJUNCT2 th1
val th4 = CONJ th3 th2
in
```

```
DISCH (hd(hyp th1)) th4
end;

val conjSymThm =
IMP_ANTISYM_RULE conj1Thm conj2Thm;

val conjSymThmAll = GENL[``p:bool``, ``q:bool``] conjSymThm;

val _ = save_thm("conjSymThmAll", conjSymThmAll)


(***********************************************************************)
(* Exporting Theory                                                    *)
(***********************************************************************)

val _ = export_theory();

end (* Structure *)
```