Homework 6

Chirag Sachdev

Week 6

Abstract

This project is a part of HW6 of Assurance Foundations. The homework deals with integration of ML and HOL to LaTeX. The goal of this report is to show reproducibility which is the groundwork for credibility that I have done this on my own without any external help. Every Chapter demonstrates the following sections:

- Problem Statement
- Relevant Code
- Test Results

This project includes the following packages:

634format.sty A format style for this course

listings Package for displaying and inputting ML source code

holtex HOL style files and commands to display in the report

This document also demonstrates my ability to :

- Easily generate a table of contents,
- Refer to chapter and section labels

My skills and my professional details can be found at `https://www.linkedin.in/in/chiragsachdev`.

Acknowledgments

I would gratefully acknowledge Dr. Shiu-Kai Chin and my other professors at Syracuse University and my Professors at Drexel University for being the wonderful mentors they are to guide me through my journey of obtaining a Master's Dregree.

Contents

Chapter 1

Executive Summary

All requirements for this project are satisfied. Specifically,

Report Contents
Our report has the following content:

Chapter 1: Executive Summary

Chapter 2: Exercise 9.5.1

Section 2.1: Problem Statement
Section 2.2: Relevant Code
Section 2.3: Session Transcripts

Chapter 3: Exercise 9.5.2

Section 3.1: Problem Statement
Section 3.2: Relevant Code
Section 3.3: Session Transcripts

Chapter 4: Exercise 9.5.3

Section 4.1: Problem Statement
Section 4.2: Relevant Code
Section 4.3: Session Transcripts

Chapter 5: Exercise 10.4.1

Section 5.1: Problem Statement
Section 5.2: Relevant Code
Section 5.3: Session Transcripts

Chapter 6: Exercise 10.4.2

Section 6.1: Problem Statement
Section 6.2: Relevant Code
Section 6.3: Session Transcripts

Chapter 7: Exercise 10.4.3

Section 7.1: Problem Statement
Section 7.2: Relevant Code
Section 7.3: Session Transcripts

Appendix A: Source Code Ex 9

Appendix B: Source Code Ex 10

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

Chapter 2

Excercise 9.5.1

2.1 Problem statement

Do a tactic-based proof of the absorption rule, and do **not** use *PROVE TAC*: *[absorptionRule]* ∀ p q. (p ==¿ q) ==¿ p ==¿ p ∧ q

2.2 Relevant Code

```
val absorptionRule =
TAC_PROOF
(
 ([],``!(p:bool)(q:bool).(p ==> q) ==> p ==> p /\ q``),
 (REPEAT STRIP_TAC THENL
 [(ACCEPT_TAC (ASSUME ``p: bool``)),
 RES_TAC])
);

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

2.3 Session Transcript

```
> val absorptionRule =
TAC_PROOF
(
 ([],``!(p:bool)(q:bool).(p ==> q) ==> p ==> p /\ q``),
 (REPEAT STRIP_TAC THENL
 [(ACCEPT_TAC (ASSUME ``p: bool``)),
 RES_TAC])
);
# # # # # # # val absorptionRule =
   |- !(p :bool) (q :bool). (p ==> q) ==> p ==> p /\ q:
   thm
```

1

Chapter 3

Excercise 9.5.2

3.1 Problem statement

Do a tactic-based proof of the constructive dilemma rule, and do not use **PROVE TAC**: *[constructiveDilemmaRule]* ∀ p q r s. (p ==¿ q) ∧ (r ==¿ s) ==¿ p ∨ r ==¿ q ∨ s

3.2 Relevant Code

```
> val constructiveDilemmaRule =
TAC_PROOF(
 ([],``!(p:bool)(q:bool)(r:bool)(s:bool).
 (p ==> q) /\ (r ==> s) ==> p \/ r ==> q \/ s``),
 (REPEAT STRIP_TAC THEN
 RES_TAC THEN
 ASM_REWRITE_TAC[] THEN
 RES_TAC THEN
 ASM_REWRITE_TAC[])
);
```

3.3 Session Transcript

```
> val constructiveDilemmaRule =
TAC_PROOF(                                                        2
 ([],``!(p:bool)(q:bool)(r:bool)(s:bool).
 (p ==> q) /\ (r ==> s) ==> p \/ r ==> q \/ s``),
 (REPEAT STRIP_TAC THEN
 RES_TAC THEN
 ASM_REWRITE_TAC[] THEN
 RES_TAC THEN
 ASM_REWRITE_TAC[])
);
# # # # # # # # # val constructiveDilemmaRule =
   |- !(p :bool) (q :bool) (r :bool) (s :bool).
      (p ==> q) /\ (r ==> s) ==> p \/ r ==> q \/ s:
   thm
```

Chapter 4

Excercise 9.5.3

4.1 Problem statement

Repeat the previous exercises using *PROVE_TAC*.

4.2 Relevant Code

```
> val absorptionRule2 =
TAC_PROOF
(
([],``!(p:bool)(q:bool).(p ==> q) ==> p ==> p /\ q``),
(PROVE_TAC[])
);

> val constructiveDilemmaRule2 =
TAC_PROOF
(
([],``!(p:bool)(q:bool)(r:bool)(s:bool).
(p ==> q) /\ (r ==> s) ==> p \/ r ==> q \/ s``),
(PROVE_TAC[])
);
```

4.3 Test Case

3

```
> val absorptionRule2 =
TAC_PROOF
(
([],``!(p:bool)(q:bool).(p ==> q) ==> p ==> p /\ q``),
(PROVE_TAC[])
);
# # # # # Meson search level: ........
val absorptionRule2 =
   |- !(p :bool) (q :bool). (p ==> q) ==> p ==> p /\ q:
   thm

> val constructiveDilemmaRule2 =
TAC_PROOF
(
([],``!(p:bool)(q:bool)(r:bool)(s:bool).
(p ==> q) /\ (r ==> s) ==> p \/ r ==> q \/ s``),
(PROVE_TAC[])
);

# # # # # # Meson search level: ................
val constructiveDilemmaRule2 =
   |- !(p :bool) (q :bool) (r :bool) (s :bool).
     (p ==> q) /\ (r ==> s) ==> p \/ r ==> q \/ s:
   thm
```

Chapter 5

Excercise 10.4.1

5.1 Problem statement

Prove the following goal without using *PROVE TAC*. Your final solution must be exe-cutable in a single step using *TAC PROOF*. set_goal ([!x:a.P(x) ==¿ M(x),(P:a-¿bool)(s:a)], (M:a-¿bool)(s:a));

5.2 Relevant Code

```
> val problem1_thm =
TAC_PROOF(
([``!x:'a.P(x) ==> M(x)``,``(P:'a -> bool)(s:'a)``],``(M:'a->bool)(s:'a)``),
(PAT_ASSUM ``!x.t`` (fn th => (ASSUME_TAC (SPEC ``s`` th))) THEN
RES_TAC)
);
```

5.3 Session Transcript

```
> val problem1_thm =                                                          4
TAC_PROOF(
([``!x:'a.P(x) ==> M(x)``,``(P:'a->bool)(s:'a)``],``(M:'a->bool)(s:'a)``),
(PAT_ASSUM ``!x.t`` (fn th => (ASSUME_TAC (SPEC ``s`` th))) THEN
RES_TAC)
);
# # # # # <<HOL message: inventing new type variable names: 'a>>
val problem1_thm =
    [..] |- (M :'a -> bool) (s :'a):
   thm
```

Chapter 6

Excercise 10.4.2

6.1 Problem statement

Prove the following goal without using PROVE TAC. Your final solution must be exe-cutable in a single step using TAC PROOF. set_goal([p $\wedge$ q ==¿ r,r ==¿ s, s],p ==¿ q);

[problem2_thm] p ==¿ $\neg$q

6.2 Relevant Code

```
> val problem2_thm =
TAC_PROOF(
([``p /\ q ==> r``, ``r ==> s``, ``~s``],``p ==> ~q``),
(REPEAT STRIP_TAC THEN
REPEAT RES_TAC)
);
```

6.3 Session Transcript

5

```
> val problem2_thm =
TAC_PROOF(
([``p /\ q ==> r``, ``r ==> s``, ``~s``],``p ==> ~q``),
(REPEAT STRIP_TAC THEN
REPEAT RES_TAC)
);
# # # # # val problem2_thm =
    [...] |- (p :bool) ==> ~(q :bool):
   thm
```

Chapter 7

Excercise 10.4.3

7.1 Problem statement

Prove the following goal without using *PROVE TAC*. Your final solution must be exe-cutable in a single step using *TAC PROOF*. set_goal([(p ∧ q), p ==¿ r, q ==¿ s],r ∨ s); *[problem3_thm]* r ∨ s

7.2 Relevant Code

```
> val problem3_thm =
TAC_PROOF(
([``~(p /\ q)``, ``~p ==> r``, ``~q ==> s``],``r \/ s``),
(PAT_ASSUM ``A ==> B`` (fn th => ASSUME_TAC(REWRITE_RULE[]
(DISJ_IMP(ONCE_REWRITE_RULE [DISJ_SYM](IMP_ELIM th))))) THEN
PAT_ASSUM ``~(A /\ B)`` (fn th => (ASSUME_TAC(REWRITE_RULE[]
(DISJ_IMP(REWRITE_RULE [DE_MORGAN_THM] th))))) THEN
ASSUME_TAC(IMP_TRANS (ASSUME ``p ==> ~q``) (ASSUME ``~q ==> s``)) THEN
ASSUME_TAC(IMP_TRANS (ASSUME ``~r ==> p``) (ASSUME ``p ==> s``)) THEN
PAT_ASSUM ``A ==> B``
(fn th=> (ASSUME_TAC (REWRITE_RULE[] (IMP_ELIM th)))) THEN
ASM_REWRITE_TAC[])
);
```

7.3 Test Case

```
> val problem3_thm =
TAC_PROOF(
([``~(p /\ q)``, ``~p ==> r``, ``~q ==> s``],``r \/ s``),
(PAT_ASSUM ``A ==> B`` (fn th => ASSUME_TAC(REWRITE_RULE[]
(DISJ_IMP(ONCE_REWRITE_RULE [DISJ_SYM](IMP_ELIM th))))) THEN
PAT_ASSUM ``~(A /\ B)`` (fn th => (ASSUME_TAC(REWRITE_RULE[]
(DISJ_IMP(REWRITE_RULE [DE_MORGAN_THM] th))))) THEN
ASSUME_TAC(IMP_TRANS (ASSUME ``p ==> ~q``) (ASSUME ``~q ==> s``)) THEN
ASSUME_TAC(IMP_TRANS (ASSUME ``~r ==> p``) (ASSUME ``p ==> s``)) THEN
PAT_ASSUM ``A ==> B`` (fn th=> (ASSUME_TAC (REWRITE_RULE[] (IMP_ELIM th))))
THEN ASM_REWRITE_TAC[])
);
# # # # # # # # # # # val problem3_thm =
    [...] |- (r :bool) \/ (s :bool):
   thm
```

6

Appendix A

Source code: Ex 9

```
(*****************************************************************************)
(* Chirag Sachdev                                                            *)
(* 9.5.1, 9.5.2, 9.5.3                                                       *)
(*****************************************************************************)

structure exercise9Script = struct

open HolKernel Parse boolLib bossLib;

val _ = new_theory "exercise9";

val absorptionRule =
TAC_PROOF
(
  ([], ``!(p:bool)(q:bool).(p ==> q) ==> p ==> p /\ q``),
  (REPEAT STRIP_TAC THENL
  [(ACCEPT_TAC (ASSUME ``p: bool``)),
 RES_TAC])
);

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

val constructiveDilemmaRule =
TAC_PROOF(
  ([], ``!(p:bool)(q:bool)(r:bool)(s:bool).
  (p ==> q) /\ (r ==> s) ==> p \/ r ==> q \/ s``),
  (REPEAT STRIP_TAC THEN
 RES_TAC THEN
 ASM_REWRITE_TAC[] THEN
 RES_TAC THEN
 ASM_REWRITE_TAC[])
);

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

val absorptionRule2 =
TAC_PROOF
(
([], ``!(p:bool)(q:bool).(p ==> q) ==> p ==> p /\ q``),
(PROVE_TAC[])
);
```

```
val _ = save_thm("absorptionRule2",absorptionRule2);

val constructiveDilemmaRule2 =
TAC_PROOF
(
([],``!(p:bool)(q:bool)(r:bool)(s:bool).
(p ==> q) /\ (r ==> s) ==> p \/ r ==> q \/ s``),
(PROVE_TAC[])
);

val _ = save_thm("constructiveDilemmaRule2",constructiveDilemmaRule2);

val _ = export_theory();

end (* structure *)
```

Appendix B

Source code: Ex 10

```
(*****************************************************************************)
(*          Chirag Sachdev                                                   *)
(*              10.4.1, 10.4.2, 10.4.3                                       *)
(*****************************************************************************)

structure exercise10Script = struct

open HolKernel Parse boolLib bossLib;

val _ = new_theory "exercise10";


val problem1_thm =
TAC_PROOF(
([``!x:'a.P(x) ==> M(x)``,``(P:'a->bool)(s:'a)``],``(M:'a->bool)(s:'a)``),
(PAT_ASSUM ``!x.t`` (fn th => (ASSUME_TAC (SPEC ``s`` th))) THEN
RES_TAC)
);

val _ = save_thm("problem1_thm",problem1_thm);

val problem2_thm =
TAC_PROOF(
([``p /\ q ==> r``, ``r ==> s``, ``~s``],``p ==> ~q``),
(REPEAT STRIP_TAC THEN
REPEAT RES_TAC)
);

val _ = save_thm("problem2_thm",problem2_thm);

val problem3_thm =
TAC_PROOF(
([``~(p /\ q)``, ``~p ==> r``, ``~q ==> s``],``r \/ s``),
(PAT_ASSUM ``A ==> B`` (fn th => ASSUME_TAC(REWRITE_RULE[]
(DISJ_IMP(ONCE_REWRITE_RULE [DISJ_SYM](IMP_ELIM th))))) THEN
PAT_ASSUM ``~(A /\ B)`` (fn th => (ASSUME_TAC(REWRITE_RULE[]
(DISJ_IMP(REWRITE_RULE [DE_MORGAN_THM] th))))) THEN
ASSUME_TAC(IMP_TRANS (ASSUME ``p ==> ~q``) (ASSUME ``~q ==> s``)) THEN
ASSUME_TAC(IMP_TRANS (ASSUME ``~r ==> p``) (ASSUME ``p ==> s``)) THEN
PAT_ASSUM ``A ==> B`` (fn th=> (ASSUME_TAC (REWRITE_RULE[] (IMP_ELIM th))))
THEN ASM_REWRITE_TAC[])
);
```

```
val _ = save_thm("problem3_thm",problem3_thm);

val _ = export_theory();

end (* structure *)
```