Project 6

Jinhao Wei

27 February 2019

Abstract

This report is a summary on my solutions of problem *9.5.1*, *9.5.2*, *9.5.3*, *10.4.1*, *10.4.2* and *10.4.3*. In these problems, we used *HOL* to prove theorems and used *EmitTex* to do pretty prints. Our works are contained in two folders respectively named *LaTeX* and *HOL*. In folder *LaTeX*, we provided files that are used to produce this project report. In folder *HOL*, we put all the proofs into two seperate files named *exercise9Script.sml* and *exercise10Script.sml*. We also used some other helper files to generate HOL reports named *exercise9Report.pdf* and *exercise10Report.pdf*.

Acknowledgments: This project follows the format and structure of *sampleTheory* provided by Professor Shiu-Kai Chin. To make it more accurate, this project mostly followed the format of my last project, which is project 5, and project 5 followed the sturcture of Professor Shiu-Kai Chin's *sampleTheory* project. Besides, my friend Yihao Sun helped me in solving problem *10.4.2*. It was by his hint that I realized how this problem should be solved.

Contents

Chapter 1

Executive Summary

This is a late submission Due to my incompentence on HOL's tactic-based proof techniques, this is a late submission.

All requirements for this project are satisfied. In particular, we proved all the theorems in this project, pretty printed the HOL theories, and made use of the *EmitTeX* structure to typeset HOL theorems in this report.

The following theorems are proved and their corresponding LaTeX macros used in this report.

[absorptionRule]

$\vdash \forall p\ q.\ (p \Rightarrow q) \Rightarrow p \Rightarrow p \land q$

[absorptionRule2]

$\vdash \forall p\ q.\ (p \Rightarrow q) \Rightarrow p \Rightarrow p \land q$

[constructiveDilemmaRule]

$\vdash \forall p\ q\ r\ s.\ (p \Rightarrow q) \land (r \Rightarrow s) \Rightarrow p \lor r \Rightarrow q \lor s$

[constructiveDilemmaRule2]

$\vdash \forall p\ q\ r\ s.\ (p \Rightarrow q) \land (r \Rightarrow s) \Rightarrow p \lor r \Rightarrow q \lor s$

[problem1_thm]

$\vdash M\ s$

[problem2_thm]

$\vdash p \Rightarrow \neg q$

[problem3_thm]

$\vdash r \lor s$

Chapter 2

Proof of absorptionRule

2.1 Problem Statement

Our task is to prove the theorem

$\vdash \forall p\ q.\ (p \Rightarrow q) \Rightarrow p \Rightarrow p \wedge q$

without using `PROVE_TAC`.

2.2 HOL Code Proving absorptionRule

```
val absorptionRule =
TAC_PROOF(
([], ``!p q.(p ==> q) ==> p ==> (p /\ q)``),
( REPEAT STRIP_TAC THEN
  RES_TAC THEN
  ASM_REWRITE_TAC [])
);
```

2.3 Session Transcript

```
> # # # # # # val absorptionRule =                                   1
     [] |- !(p :bool) (q :bool). (p ==> q) ==> p ==> p /\ q:
   thm
```

Chapter 3

Proof of constructiveDilemmaRule

3.1 Problem Statement

Our task is to prove the theorem

$\vdash \forall p\ q\ r\ s.\ (p \Rightarrow q) \wedge (r \Rightarrow s) \Rightarrow p \vee r \Rightarrow q \vee s$

without using `PROVE_TAC`.

3.2 HOL Code Proving constructiveDilemmaRule

```
val constructiveDilemmaRule =
TAC_PROOF(
([], ``!p q r s. (p==>q)/\(r==>s) ==> p\/r ==> q\/s``),
REPEAT STRIP_TAC THENL
[(DISJ1_TAC THEN RES_TAC) , (DISJ2_TAC THEN RES_TAC)]
);
```

3.3 Session Transcript

```
> # # # # # val constructiveDilemmaRule =
    []
|- !(p :bool) (q :bool) (r :bool) (s :bool).
     (p ==> q) /\ (r ==> s) ==> p \/ r ==> q \/ s:
   thm
```

1

Chapter 4

Proof of absorptionRule2

4.1 Problem Statement

Our task is to prove the theorem

$\vdash \forall p\ q.\ (p \Rightarrow q) \Rightarrow p \Rightarrow p \wedge q$

using PROVE_TAC.

4.2 HOL Code Proving absorptionRule2

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

4.3 Session Transcript

```
> # Meson search level: ........
val absorptionRule2 =
    [] |- !(p :bool) (q :bool). (p ==> q) ==> p ==> p /\ q:
   thm
>
```
1

Chapter 5

Proof of constructiveDilemmaRule2

5.1 Problem Statement

Our task is to prove the following theorem using `PROVE_TAC`.

$\vdash \forall p\ q\ r\ s.\ (p \Rightarrow q) \wedge (r \Rightarrow s) \Rightarrow p \vee r \Rightarrow q \vee s$

5.2 HOL Code Proving constructiveDilemmaRule2

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

5.3 Session Transcript

```
> # # # # Meson search level: ................
val constructiveDilemmaRule2 =
    []
|- !(p :bool) (q :bool) (r :bool) (s :bool).
     (p ==> q) /\ (r ==> s) ==> p \/ r ==> q \/ s:
   thm
>
```
1

Chapter 6

Proof of problem1_thm

6.1 Problem Statement

Our task is to prove the following theorem without using `PROVE_TAC`.

$\vdash M\ s$

6.2 HOL Code Proving problem1_thm

```
val problem1_thm =
TAC_PROOF(
([``!x:'a. P(x)==>M(x)``, ``(P:'a->bool)(s:'a)``], ``(M:'a->bool)(s:'a)``),
RES_TAC
);
```

6.3 Session Transcript

```
> # # # # val problem1_thm =                                                    1

[(P :'a -> bool) (s :'a),
 !(x :'a). (P :'a -> bool) x ==> (M :'a -> bool) x]
|- (M :'a -> bool) (s :'a):
   thm
>
```

Chapter 7

Proof of problem2_thm

7.1 Problem Statement

Our task is to prove the following theorem without using PROVE_TAC.

$\vdash p \Rightarrow \neg q$

7.2 HOL Code Proving problem1_thm

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

7.3 Session Transcript

```
> # # # # # val problem2_thm =                                         1

[~(s :bool), (r :bool) ==> (s :bool),
 (p :bool) /\ (q :bool) ==> (r :bool)] |- (p :bool) ==> ~(q :bool):
   thm
```

Chapter 8

Proof of problem3_thm

8.1 Problem Statement

Our task is to prove the following theorem without using `PROVE_TAC`.

$\vdash r \lor s$

8.2 HOL Code Proving problem3_thm

```
val problem3_thm =
TAC_PROOF(
([``~(p/\q)``, ``~p ==> r``, ``~q ==> s``],``r\/s``),

PAT_ASSUM ``~(p/\q)`` (fn th => (ASSUME_TAC (REWRITE_RULE [DE_MORGAN_THM] th))
    ) THEN

PAT_ASSUM ``~p\/~q`` (fn th => ASSUME_TAC (REWRITE_RULE [] (DISJ_IMP th)))
    THEN

ASSUME_TAC(IMP_TRANS(ASSUME ``p==>~q``)(ASSUME ``~q==>s``)) THEN

PAT_ASSUM ``~p==>r`` (fn th => ASSUME_TAC (REWRITE_RULE [] (DISJ_IMP (
    ONCE_REWRITE_RULE [DISJ_SYM] (IMP_ELIM th))))) THEN

ASSUME_TAC(IMP_TRANS(ASSUME ``~r==>p``)(ASSUME ``p==>s``)) THEN

ASSUME_TAC (REWRITE_RULE [] (IMP_ELIM (ASSUME ``~r==>s``))) THEN

ASM_REWRITE_TAC []
);
```

8.3 Session Transcript

1

```
> val problem3_thm =

[~((p :bool) /\ (q :bool)), ~(p :bool) ==> (r :bool),
 ~(q :bool) ==> (s :bool)] |- (r :bool) \/ (s :bool):
   thm
val it = (): unit
>
```

Appendix A

Source Code for exercise9Script

The following code is from *exercise9Script.sml*, which is located in directory "../HOL/"

```
structure exercise9Script = struct

open HolKernel Parse boolLib bossLib;

val _ = new_theory "exercise9";

val absorptionRule =
TAC_PROOF(
([], ``!p q.(p ==> q) ==> p ==> (p /\ q)``),
( REPEAT STRIP_TAC THEN
  RES_TAC THEN
  ASM_REWRITE_TAC [])
);


val constructiveDilemmaRule =
TAC_PROOF(
([], ``!p q r s. (p==>q)/\(r==>s) ==> p\/r ==> q\/s``),
REPEAT STRIP_TAC THENL
[(DISJ1_TAC THEN RES_TAC) , (DISJ2_TAC THEN RES_TAC)]
);

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


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

val _ = save_thm("absorptionRule", absorptionRule);
val _ = save_thm("constructiveDilemmaRule", constructiveDilemmaRule);
val _ = save_thm("absorptionRule2", absorptionRule2);
val _ = save_thm("constructiveDilemmaRule2", constructiveDilemmaRule2);

val _ = export_theory();

end
```

Appendix B

Source Code for exercise10Script

The following code is from *exercise10Script.sml*, which is located in directory "../HOL/"

```
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)``),
RES_TAC
);

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

val problem3_thm =
TAC_PROOF(
([``~(p/\q)``, ``~p ==> r``, ``~q ==> s``],``r\/s``),

PAT_ASSUM ``~(p/\q)`` (fn th => (ASSUME_TAC (REWRITE_RULE [DE_MORGAN_THM] th))
    ) THEN

PAT_ASSUM ``~p\/~q`` (fn th => ASSUME_TAC (REWRITE_RULE [] (DISJ_IMP th)))
    THEN

ASSUME_TAC(IMP_TRANS(ASSUME ``p==>~q``)(ASSUME ``~q==>s``)) THEN

PAT_ASSUM ``~p==>r`` (fn th => ASSUME_TAC (REWRITE_RULE [] (DISJ_IMP (
    ONCE_REWRITE_RULE [DISJ_SYM] (IMP_ELIM th))))) THEN

ASSUME_TAC(IMP_TRANS(ASSUME ``~r==>p``)(ASSUME ``p==>s``)) THEN

ASSUME_TAC (REWRITE_RULE [] (IMP_ELIM (ASSUME ``~r==>s``))) THEN

ASM_REWRITE_TAC []
);

val _ = save_thm("problem1_thm", problem1_thm);
val _ = save_thm("problem2_thm", problem2_thm);
```

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

val _ = export_theory();

end
```