Project 8

Jinhao Wei

21 March 2019

Abstract

This report is basically a summary on my attempts on Project 8, which includes datatype definiton, forward and goal-oriented proofs. This report provides my solution on *13.10.1*, *13.10.2* and *14.4.1*. In addition, I had fine printed the corresponding datatypes and proofs and put the reports in *../HOL/HOLReports/solutions1Report.pdf* and *../HOL/HOLReports/conops0SolutionReport.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 one of my previous projects, which is project 5, and project 5 followed the sturcture of Professor Shiu-Kai Chin's *sampleTheory* project.

Besides, this report relies partly on *example1Theory*, without which we will not be able to create *solutions1Theory*

I will also need to thank Bowei Qu, my friend, who pointed out my mistakes in time. I thought the "Form" was not important while he told me it was more than important. If it wasn't him, I might be late for even more time.

Contents

Chapter 1

Executive Summary

This is a late submission by two days.
All requirements for this project are satisfied. In particular, we defined all the datatypes and 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 datatypes are defined in this report

```
commands = go | nogo | launch | abort | activate | stand_down

keyPrinc = Staff people | Role roles | Ap num

people = Alice | Bob

principals = PR keyPrinc | Key keyPrinc

roles = Commander | Operator | CA
```

and the following theorems are proved

[aclExerciseTheorem1]

```
⊢ (M,Oi,Os) sat Name Alice says prop go ⇒
  (M,Oi,Os) sat Name Bob says prop go ⇒
  (M,Oi,Os) sat Name Alice meet Name Bob says prop go
```

[aclExerciseTheorem1A]

```
⊢ (M,Oi,Os) sat Name Alice says prop go ⇒
  (M,Oi,Os) sat Name Bob says prop go ⇒
  (M,Oi,Os) sat Name Alice meet Name Bob says prop go
```

[aclExerciseTheorem1B]

```
⊢ (M,Oi,Os) sat Name Alice says prop go ⇒
  (M,Oi,Os) sat Name Bob says prop go ⇒
  (M,Oi,Os) sat Name Alice meet Name Bob says prop go
```

[aclExerciseTheorem2]

```
⊢ (M,Oi,Os) sat Name Alice says prop go ⇒
  (M,Oi,Os) sat Name Alice controls prop go ⇒
  (M,Oi,Os) sat prop go impf prop val ⇒
  (M,Oi,Os) sat Name Bob says prop go
```

[aclExerciseTheorem2A]

```
⊢ (M,Oi,Os) sat Name Alice says prop go ⇒
  (M,Oi,Os) sat Name Alice controls prop go ⇒
  (M,Oi,Os) sat prop go impf prop launch ⇒
  (M,Oi,Os) sat Name Bob says prop launch
```

[aclExerciseTheorem2B]

```
⊢ (M,Oi,Os) sat Name Alice says prop go ⇒
  (M,Oi,Os) sat Name Alice controls prop go ⇒
  (M,Oi,Os) sat prop go impf prop launch ⇒
  (M,Oi,Os) sat Name Bob says prop launch
```

[ApRuleActive_thm]

```
⊢ (M,Oi,Os) sat
  Name (PR (Role Operator)) controls prop launch ⇒
  (M,Oi,Os) sat
  reps (Name (PR (Staff Bob))) (Name (PR (Role Operator)))
    (prop launch) ⇒
  (M,Oi,Os) sat
  Name (Key (Staff Bob)) quoting Name (PR (Role Operator)) says
  prop launch ⇒
  (M,Oi,Os) sat prop launch impf prop activate ⇒
  (M,Oi,Os) sat
  Name (Key (Role CA)) speaks_for Name (PR (Role CA)) ⇒
  (M,Oi,Os) sat
  Name (Key (Role CA)) says
  Name (Key (Staff Bob)) speaks_for Name (PR (Staff Bob)) ⇒
  (M,Oi,Os) sat
  Name (PR (Role CA)) controls
  Name (Key (Staff Bob)) speaks_for Name (PR (Staff Bob)) ⇒
  (M,Oi,Os) sat prop activate
```

[ApRuleStandDown_thm]

```
⊢ (M,Oi,Os) sat Name (PR (Role Operator)) controls prop abort ⇒
  (M,Oi,Os) sat
  reps (Name (PR (Staff Bob))) (Name (PR (Role Operator)))
    (prop abort) ⇒
  (M,Oi,Os) sat
  Name (Key (Staff Bob)) quoting Name (PR (Role Operator)) says
  prop abort ⇒
  (M,Oi,Os) sat prop abort impf prop stand_down ⇒
  (M,Oi,Os) sat
  Name (Key (Role CA)) speaks_for Name (PR (Role CA)) ⇒
  (M,Oi,Os) sat
  Name (Key (Role CA)) says
  Name (Key (Staff Bob)) speaks_for Name (PR (Staff Bob)) ⇒
  (M,Oi,Os) sat
  Name (PR (Role CA)) controls
  Name (Key (Staff Bob)) speaks_for Name (PR (Staff Bob)) ⇒
  (M,Oi,Os) sat prop stand_down
```

[OpRuleAbort_thm]

```
⊢ (M,Oi,Os) sat Name (PR (Role Commander)) controls prop nogo ⇒
  (M,Oi,Os) sat
  reps (Name (PR (Staff Alice))) (Name (PR (Role Commander)))
    (prop nogo) ⇒
  (M,Oi,Os) sat
  Name (Key (Staff Alice)) quoting
  Name (PR (Role Commander)) says prop nogo ⇒
  (M,Oi,Os) sat prop nogo impf prop abort ⇒
  (M,Oi,Os) sat
  Name (Key (Role CA)) speaks_for Name (PR (Role CA)) ⇒
  (M,Oi,Os) sat
```

```
    Name (Key (Role CA)) says
    Name (Key (Staff Alice)) speaks_for Name (PR (Staff Alice)) ⇒
    (M,Oi,Os) sat
    Name (PR (Role CA)) controls
    Name (Key (Staff Alice)) speaks_for Name (PR (Staff Alice)) ⇒
    (M,Oi,Os) sat
    Name (Key (Staff Bob)) quoting Name (PR (Role Operator)) says
    prop abort
```

[OpRuleLaunch_thm]

```
  ⊢ (M,Oi,Os) sat Name (PR (Role Commander)) controls prop go ⇒
    (M,Oi,Os) sat
    reps (Name (PR (Staff Alice))) (Name (PR (Role Commander)))
      (prop go) ⇒
    (M,Oi,Os) sat
    Name (Key (Staff Alice)) quoting
    Name (PR (Role Commander)) says prop go ⇒
    (M,Oi,Os) sat prop go impf prop launch ⇒
    (M,Oi,Os) sat
    Name (Key (Role CA)) speaks_for Name (PR (Role CA)) ⇒
    (M,Oi,Os) sat
    Name (Key (Role CA)) says
    Name (Key (Staff Alice)) speaks_for Name (PR (Staff Alice)) ⇒
    (M,Oi,Os) sat
    Name (PR (Role CA)) controls
    Name (Key (Staff Alice)) speaks_for Name (PR (Staff Alice)) ⇒
    (M,Oi,Os) sat
    Name (Key (Staff Bob)) quoting Name (PR (Role Operator)) says
    prop launch
```

Reproducibility in ML and LaTeX

All ML and LaTeX source files compile well on the environment provided by this course.

Chapter 2

Exercise 10.13.1

2.1 Problem Statement

In this exercise, we imported a previous defined theory *example1Theory* and used the datatypes in it.we will give our proofs on the following theorem

```
⊢ (M,Oi,Os) sat Name Alice says prop go ⇒
  (M,Oi,Os) sat Name Bob says prop go ⇒
  (M,Oi,Os) sat Name Alice meet Name Bob says prop go
```

with three different methods.

Before we go through the following sections, please kindly not that in the source file I provided in *../HOL/solutions1Script.sml*, at the very begining I had imported the following package by doing

```
open HolKernel boolLib Parse bossLib;
open acl_infRules aclrulesTheory aclDrulesTheory ;
open example1Theory;
```

2.2 Forward Proofs

In this section, we will give a forward proof on theorem

```
⊢ (M,Oi,Os) sat Name Alice says prop go ⇒
  (M,Oi,Os) sat Name Bob says prop go ⇒
  (M,Oi,Os) sat Name Alice meet Name Bob says prop go
```

2.2.1 Relevant Codes

```
val aclExerciseTheorem1 =
let
 val th1 = ACL_ASSUM``((Name Alice) says (prop go)):(commands, staff, 'd, 'e)
    Form``
 val th2 = ACL_ASSUM``((Name Bob) says (prop go)):(commands, staff, 'd, 'e)
    Form``
 val th3 = ACL_CONJ th1 th2
 val th4 = AND_SAYS_RL th3
 val th5 = DISCH(hd(hyp th2)) th4
in
 DISCH (hd(hyp th1)) th5
end;
```

2.2.2 Session Transcript

If we send the above code to HOL, we will see the transcript as below:

```
 > # # # # # # # # # val aclExerciseTheorem1 =                                   1
    |- ((M :(commands, 'b, staff, 'd, 'e) Kripke),(Oi :'d po),
     (Os :'e po)) sat
    Name Alice says (prop go :(commands, staff, 'd, 'e) Form) ==>
    (M,Oi,Os) sat
    Name Bob says (prop go :(commands, staff, 'd, 'e) Form) ==>
    (M,Oi,Os) sat
    Name Alice meet Name Bob says
    (prop go :(commands, staff, 'd, 'e) Form):
    thm
>
*** Emacs/HOL command completed ***
```

2.3 Goal-oriented Proof with PROVE_TAC

In this section, we will give a goal-oriented proof on theorem

$\vdash$ (M, Oi, Os) `sat Name Alice says prop go` $\Rightarrow$
(M, Oi, Os) `sat Name Bob says prop go` $\Rightarrow$
(M, Oi, Os) `sat Name Alice meet Name Bob says prop go`

2.3.1 Relevant Codes

```
val aclExerciseTheorem1A =
TAC_PROOF(
([],
``((M :(commands, 'b, staff, 'd, 'e) Kripke),(Oi :'d po),(Os :'e po)) sat
   ((Name Alice) says (prop go)) ==>
  (M,Oi,Os) sat ((Name Bob) says (prop go)) ==>
  (M,Oi,Os) sat (((Name Alice) meet (Name Bob)) says (prop go))``
),
PROVE_TAC[Conjunction, And_Says_Eq]
);
```

2.3.2 Session Transcript

If we send the above code to HOL, we will see the transcript as below:

```
> # # # # # # # # # Meson search level: .....                                    2

val aclExerciseTheorem1A =
   |- ((M :(commands, 'b, staff, 'd, 'e) Kripke),(Oi :'d po),
    (Os :'e po)) sat
   Name Alice says (prop go :(commands, staff, 'd, 'e) Form) ==>
   (M,Oi,Os) sat
   Name Bob says (prop go :(commands, staff, 'd, 'e) Form) ==>
   (M,Oi,Os) sat
   Name Alice meet Name Bob says
   (prop go :(commands, staff, 'd, 'e) Form):
   thm
>
```

2.4 Goal-oriented Proof without PROVE_TAC

2.4.1 Relevant Code

```
val aclExerciseTheorem1B =
TAC_PROOF(
([],
``((M :(commands, 'b, staff, 'd, 'e) Kripke),(Oi :'d po),(Os :'e po)) sat
   ((Name Alice) says (prop go)) ==>
  (M,Oi,Os) sat ((Name Bob) says (prop go)) ==>
  (M,Oi,Os) sat (((Name Alice) meet (Name Bob)) says (prop go))``
),
REPEAT STRIP_TAC THEN
ACL_AND_SAYS_RL_TAC THEN
ACL_CONJ_TAC THEN
REWRITE_TAC [] THEN
ASM_REWRITE_TAC []
);
```

2.4.2 Session Transcript

```
> # # # # # # # # # # # # # # val aclExerciseTheorem1B =                          1
   |- ((M :(commands, 'b, staff, 'd, 'e) Kripke),(Oi :'d po),
    (Os :'e po)) sat
   Name Alice says (prop go :(commands, staff, 'd, 'e) Form) ==>
   (M,Oi,Os) sat
   Name Bob says (prop go :(commands, staff, 'd, 'e) Form) ==>
   (M,Oi,Os) sat
   Name Alice meet Name Bob says
   (prop go :(commands, staff, 'd, 'e) Form):
   thm
>
*** Emacs/HOL command completed ***
```

Chapter 3

Exercise 13.10.2

3.1 Problem Statement

In this exercise, we will prove the theorems

$\vdash$ `(`M`,`Oi`,`Os`) sat Name Alice says prop go` $\Rightarrow$
`(`M`,`Oi`,`Os`) sat Name Alice controls prop go` $\Rightarrow$
`(`M`,`Oi`,`Os`) sat prop go impf prop` *val* $\Rightarrow$
`(`M`,`Oi`,`Os`) sat Name Bob says prop go`

Just like in the previous chapter. Before we go through the following sections, please kindly not that in the source file I provided in *../HOL/solutions1Script.sml*, at the very begining I had imported the following package by doing

```
open HolKernel Parse boolLib bossLib;
open TypeBase boolTheory arithmeticTheory
```

3.2 Forward Proof

3.2.1 Relevant Code

```
val aclExerciseTheorem2 =
let
 val th1 = ACL_ASSUM``((Name Alice) says (prop go)):(commands, staff, 'd, 'e)
     Form``
 val th2 = ACL_ASSUM``((Name Alice) controls (prop go)):(commands, staff, 'd,
     'e)Form``
 val th3 = ACL_ASSUM``((prop go) impf (prop val))``
 val th4 = CONTROLS th2 th1
 val th5 = SAYS ``(Name Bob)`` th4
 val th6 = DISCH (hd(hyp th3)) th5
 val th7 = DISCH (hd(hyp th2)) th6
in
 DISCH (hd(hyp th1)) th7
end;
```

3.2.2 Session Transcript

```
> # # # # # # # # # # # <<HOL message: inventing new type variable names: 'a, 'b, 'c>>
val aclExerciseTheorem2 =
   |- ((M :(commands, 'b, staff, 'd, 'e) Kripke),(Oi :'d po),
    (Os :'e po)) sat
   Name Alice says (prop go :(commands, staff, 'd, 'e) Form) ==>
   (M,Oi,Os) sat
   Name Alice controls (prop go :(commands, staff, 'd, 'e) Form) ==>
   ((M :(commands, 'b, 'a, 'b, 'c) Kripke),(Oi :'b po),(Os :'c po)) sat
   (prop go :(commands, 'a, 'b, 'c) Form) impf
   (prop (val :commands) :(commands, 'a, 'b, 'c) Form) ==>
   (M,Oi,Os) sat Name Bob says (prop go :(commands, staff, 'd, 'e) Form):
   thm
>
*** Emacs/HOL command completed ***
```

3.3 Goal-oriented Proof with PROVE_TAC

3.3.1 Relevant Code

```
val aclExerciseTheorem2A =
TAC_PROOF(
([],
``((M:(commands, 'b, staff, 'd, 'e) Kripke),(Oi: 'd po),(Os: 'e po)) sat
    (Name Alice says (prop go)) ==>
    (M, Oi, Os) sat ((Name Alice) controls (prop go)) ==>
    (M, Oi, Os) sat ((prop go) impf (prop launch)) ==>
    (M, Oi, Os) sat ((Name Bob) says (prop launch))``
),
PROVE_TAC[Modus_Ponens, Controls, Says]);
```

3.3.2 Session Transcript

```
> # # # # # # # # # # Meson search level: .......
val aclExerciseTheorem2A =
   |- ((M :(commands, 'b, staff, 'd, 'e) Kripke),(Oi :'d po),
    (Os :'e po)) sat
   Name Alice says (prop go :(commands, staff, 'd, 'e) Form) ==>
   (M,Oi,Os) sat
   Name Alice controls (prop go :(commands, staff, 'd, 'e) Form) ==>
   (M,Oi,Os) sat
   (prop go :(commands, staff, 'd, 'e) Form) impf
   (prop launch :(commands, staff, 'd, 'e) Form) ==>
   (M,Oi,Os) sat
   Name Bob says (prop launch :(commands, staff, 'd, 'e) Form):
   thm
>
*** Emacs/HOL command completed ***
```

3.4 Goal-oriented Proof without PROVE_TAC

3.4.1 Relevant Code

```
val aclExerciseTheorem2B =
TAC_PROOF(([],
``((M:(commands, 'b, staff, 'd, 'e) Kripke),(Oi: 'd po),(Os: 'e po)) sat
    ((Name Alice) says (prop go)) ==>
    (M, Oi, Os) sat ((Name Alice) controls (prop go)) ==>
    (M, Oi, Os) sat ((prop go) impf (prop launch)) ==>
    (M, Oi, Os) sat ((Name Bob) says (prop launch))``
),
REPEAT STRIP_TAC THEN
ACL_SAYS_TAC THEN
PAT_ASSUM ``(M, Oi, Os) sat (Name Alice controls prop go)``
          (fn th1 =>(PAT_ASSUM ``(M,Oi, Os) sat (Name Alice says prop go)``
                     (fn th2 => ASSUME_TAC (CONTROLS th1 th2)))) THEN

PAT_ASSUM ``(M, Oi, Os) sat ((prop go) impf (prop launch))``
          (fn th1 =>(PAT_ASSUM ``(M,Oi,Os) sat (Name Alice controls prop go)``
              (fn th2 =>(PAT_ASSUM ``(M,Oi,Os) sat (Name Alice says prop go)``
                  (fn th3 => ASSUME_TAC (ACL_MP (CONTROLS th2 th3) th1))))))
                     THEN

PROVE_TAC []
);
```

3.4.2 Session Transcript

```
 <<HOL message: inventing new type variable names: 'a, 'b, 'c>>                         1
<<HOL message: inventing new type variable names: 'a, 'b, 'c, 'd>>
Meson search level: ..
val aclExerciseTheorem2B =
   |- ((M :(commands, 'b, staff, 'd, 'e) Kripke),(Oi :'d po),
    (Os :'e po)) sat
   Name Alice says (prop go :(commands, staff, 'd, 'e) Form) ==>
   (M,Oi,Os) sat
   Name Alice controls (prop go :(commands, staff, 'd, 'e) Form) ==>
   (M,Oi,Os) sat
   (prop go :(commands, staff, 'd, 'e) Form) impf
   (prop launch :(commands, staff, 'd, 'e) Form) ==>
   (M,Oi,Os) sat
   Name Bob says (prop launch :(commands, staff, 'd, 'e) Form):
   thm
val it = (): unit
>
*** Emacs/HOL command completed ***
```

Chapter 4

Exercise 14.4.1

4.1 Problem Statement

In this question, we will give some definitions on our own datatypes, then we will use those datatypes to prove four theorems which are essentially the same.

Please kindly note, in the corresponding source file *../HOL/Conops0SolutionsScript.sml*, I had imported the following packages

```
open HolKernel Parse boolLib bossLib;
open acl_infRules aclrulesTheory aclDrulesTheory ;
```

to guarantee that all the following procedures work as expected.

4.2 Definitions on Datatypes

We will need to define several datatypes before we dive into the proofs, and they are respectively

```
val _ = Datatype `commands = go | nogo | launch|abort|activate|stand_down`;

val _ = Datatype `people = Alice | Bob`;

val _ = Datatype `roles = Commander | Operator | CA`;

val _ = Datatype `keyPrinc = Staff conops0Solution$people | Role
    conops0Solution$roles | Ap num`;

val _ = Datatype `principals = PR keyPrinc | Key keyPrinc`;
```

4.3 Proof on OpRuleLaunch

4.3.1 Relevant Code

```

val OpRuleLaunch =
let
  val th1 = ACL_ASSUM ``(Name (PR (Role Commander)) controls (prop go)): (
      commands, principals, 'd, 'e)Form``
  val th2 = ACL_ASSUM ``(reps (Name (PR (Staff Alice))) (Name (PR (Role
      Commander))) (prop go)): (commands, principals, 'd, 'e)Form``
  val th3 = ACL_ASSUM ``(Name (Key (Staff Alice)) quoting Name (PR (Role
      Commander)) says prop go): (commands, principals, 'd, 'e)Form``
  val th4 = ACL_ASSUM ``(prop go impf prop launch):(commands, principals, 'd,
      'e)Form``
  val th5 = ACL_ASSUM ``(Name (Key (Role CA)) speaks_for Name (PR (Role CA))):
       (commands,principals , 'd, 'e)Form``
  val th6 = ACL_ASSUM ``(Name (Key (Role CA)) says Name (Key (Staff Alice))
      speaks_for Name (PR (Staff Alice))): (commands, principals , 'd, 'e)Form
      ``
  val th7 = ACL_ASSUM ``(Name (PR (Role CA)) controls Name (Key (Staff Alice))
       speaks_for Name (PR (Staff Alice))): (commands, principals , 'd, 'e)Form
      ``



  val th9 = SPEAKS_FOR th5 th6;
  val th10 = CONTROLS th7 th9;
  val th11 = QUOTING_LR th3;
  val th12 = SPEAKS_FOR th10 th11
  val th13 = QUOTING_RL th12
  val th14 = REPS th2 th13 th1;
  val th15 = ACL_MP th14 th4;
  val th16 = SAYS ``Name(Key (Staff Bob)) quoting Name (PR (Role Operator))``
      th15
  val th17 = DISCH(hd (hyp th7)) th16
  val th18 = DISCH(hd (hyp th6)) th17
  val th19 = DISCH(hd (hyp th5)) th18
  val th20 = DISCH(hd (hyp th4)) th19
  val th21 = DISCH(hd (hyp th3)) th20
  val th22 = DISCH(hd (hyp th2)) th21
in
  DISCH (hd (hyp th1)) th22
end;
```

4.3.2 Session Transcript

```
# # # # # # # # # # # # # # # # # # # # # # # # # # # # # val OpRuleLaunch =                1
    |- ((M :(commands, 'b, principals, 'd, 'e) Kripke),(Oi :'d po),
     (Os :'e po)) sat
    Name (PR (Role Commander)) controls
    (prop go :(commands, principals, 'd, 'e) Form) ==>
    (M,Oi,Os) sat
    reps (Name (PR (Staff Alice))) (Name (PR (Role Commander)))
      (prop go :(commands, principals, 'd, 'e) Form) ==>
    (M,Oi,Os) sat
    Name (Key (Staff Alice)) quoting Name (PR (Role Commander)) says
    (prop go :(commands, principals, 'd, 'e) Form) ==>
    (M,Oi,Os) sat
    (prop go :(commands, principals, 'd, 'e) Form) impf
    (prop launch :(commands, principals, 'd, 'e) Form) ==>
    (M,Oi,Os) sat
    ((Name (Key (Role CA)) speaks_for Name (PR (Role CA)))
       :(commands, principals, 'd, 'e) Form) ==>
    (M,Oi,Os) sat
    Name (Key (Role CA)) says
    ((Name (Key (Staff Alice)) speaks_for Name (PR (Staff Alice)))
       :(commands, principals, 'd, 'e) Form) ==>
    (M,Oi,Os) sat
    Name (PR (Role CA)) controls
    ((Name (Key (Staff Alice)) speaks_for Name (PR (Staff Alice)))
       :(commands, principals, 'd, 'e) Form) ==>
    (M,Oi,Os) sat
    Name (Key (Staff Bob)) quoting Name (PR (Role Operator)) says
    (prop launch :(commands, principals, 'd, 'e) Form):
    thm
>
```

4.4 Proof on ApRuleActive

4.4.1 Relevant Code

```
val ApRuleActive =
let
  val th1 = ACL_ASSUM ``(Name (PR (Role Operator)) controls (prop launch)): (
      commands, principals, 'd, 'e)Form``
  val th2 = ACL_ASSUM ``(reps (Name (PR (Staff Bob))) (Name (PR (Role Operator
      ))) (prop launch)): (commands, principals, 'd, 'e)Form``
  val th3 = ACL_ASSUM ``(Name (Key (Staff Bob)) quoting Name (PR (Role
      Operator)) says prop launch): (commands, principals, 'd, 'e)Form``
  val th4 = ACL_ASSUM ``(prop launch impf prop activate):(commands, principals
      , 'd, 'e)Form``
  val th5 = ACL_ASSUM ``(Name (Key (Role CA)) speaks_for Name (PR (Role CA))):
       (commands,principals, 'd, 'e)Form``
  val th6 = ACL_ASSUM ``(Name (Key (Role CA)) says Name (Key (Staff Bob))
      speaks_for Name (PR (Staff Bob))): (commands, principals, 'd, 'e)Form``
  val th7 = ACL_ASSUM ``(Name (PR (Role CA)) controls Name (Key (Staff Bob))
      speaks_for Name (PR (Staff Bob))): (commands, principals, 'd, 'e)Form``


  val th9 = SPEAKS_FOR th5 th6;
  val th10 = CONTROLS th7 th9;
  val th11 = QUOTING_LR th3;
  val th12 = SPEAKS_FOR th10 th11
  val th13 = QUOTING_RL th12
  val th14 = REPS th2 th13 th1;
  val th15 = ACL_MP th14 th4;
  (*val th16 = SAYS ``Name(Key (Staff Bob)) quoting Name (PR (Role Operator))
      `` th15*)
  val th16 = DISCH(hd (hyp th7)) th15
  val th17 = DISCH(hd (hyp th6)) th16
  val th18 = DISCH(hd (hyp th5)) th17
  val th19 = DISCH(hd (hyp th4)) th18
  val th20 = DISCH(hd (hyp th3)) th19
  val th21 = DISCH(hd (hyp th2)) th20
in
  DISCH (hd (hyp th1)) th21
end;
```

4.4.2 Session Transcript

```
# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # val ApRuleActive =                                    1
   |- ((M :(commands, 'b, principals, 'd, 'e) Kripke),(Oi :'d po),
    (Os :'e po)) sat
   Name (PR (Role Operator)) controls
   (prop launch :(commands, principals, 'd, 'e) Form) ==>
   (M,Oi,Os) sat
   reps (Name (PR (Staff Bob))) (Name (PR (Role Operator)))
     (prop launch :(commands, principals, 'd, 'e) Form) ==>
   (M,Oi,Os) sat
   Name (Key (Staff Bob)) quoting Name (PR (Role Operator)) says
   (prop launch :(commands, principals, 'd, 'e) Form) ==>
   (M,Oi,Os) sat
   (prop launch :(commands, principals, 'd, 'e) Form) impf
   (prop activate :(commands, principals, 'd, 'e) Form) ==>
   (M,Oi,Os) sat
   ((Name (Key (Role CA)) speaks_for Name (PR (Role CA)))
      :(commands, principals, 'd, 'e) Form) ==>
   (M,Oi,Os) sat
   Name (Key (Role CA)) says
   ((Name (Key (Staff Bob)) speaks_for Name (PR (Staff Bob)))
      :(commands, principals, 'd, 'e) Form) ==>
   (M,Oi,Os) sat
   Name (PR (Role CA)) controls
   ((Name (Key (Staff Bob)) speaks_for Name (PR (Staff Bob)))
      :(commands, principals, 'd, 'e) Form) ==>
   (M,Oi,Os) sat (prop activate :(commands, principals, 'd, 'e) Form):
   thm
```

4.5 Proof on OpRuleAbort

4.5.1 Relevant Code

```
val OpRuleAbort =
let
  val th1 = ACL_ASSUM ``(Name (PR (Role Commander)) controls (prop nogo)): (
      commands, principals, 'd, 'e)Form``
  val th2 = ACL_ASSUM ``(reps (Name (PR (Staff Alice))) (Name (PR (Role
      Commander))) (prop nogo)): (commands, principals, 'd, 'e)Form``
  val th3 = ACL_ASSUM ``(Name (Key (Staff Alice)) quoting Name (PR (Role
      Commander)) says prop nogo): (commands, principals, 'd, 'e)Form``
  val th4 = ACL_ASSUM ``(prop nogo impf prop abort):(commands, principals, 'd,
       'e)Form``
  val th5 = ACL_ASSUM ``(Name (Key (Role CA)) speaks_for Name (PR (Role CA))):
       (commands,principals, 'd, 'e)Form``
  val th6 = ACL_ASSUM ``(Name (Key (Role CA)) says Name (Key (Staff Alice))
      speaks_for Name (PR (Staff Alice))): (commands, principals, 'd, 'e)Form
      ``
  val th7 = ACL_ASSUM ``(Name (PR (Role CA)) controls Name (Key (Staff Alice))
       speaks_for Name (PR (Staff Alice))): (commands, principals, 'd, 'e)Form
      ``



  val th9 = SPEAKS_FOR th5 th6;
  val th10 = CONTROLS th7 th9;
  val th11 = QUOTING_LR th3;
  val th12 = SPEAKS_FOR th10 th11
  val th13 = QUOTING_RL th12
  val th14 = REPS th2 th13 th1;
  val th15 = ACL_MP th14 th4;
  val th16 = SAYS ``Name(Key (Staff Bob)) quoting Name (PR (Role Operator))``
      th15
  val th17 = DISCH(hd (hyp th7)) th16
  val th18 = DISCH(hd (hyp th6)) th17
  val th19 = DISCH(hd (hyp th5)) th18
  val th20 = DISCH(hd (hyp th4)) th19
  val th21 = DISCH(hd (hyp th3)) th20
  val th22 = DISCH(hd (hyp th2)) th21
in
  DISCH (hd (hyp th1)) th22
end;
```

4.5.2 Session Transcript

```
> val OpRuleAbort =
   |- ((M :(commands, 'b, principals, 'd, 'e) Kripke),(Oi :'d po),
    (Os :'e po)) sat
   Name (PR (Role Commander)) controls
   (prop nogo :(commands, principals, 'd, 'e) Form) ==>
   (M,Oi,Os) sat
   reps (Name (PR (Staff Alice))) (Name (PR (Role Commander)))
     (prop nogo :(commands, principals, 'd, 'e) Form) ==>
   (M,Oi,Os) sat
   Name (Key (Staff Alice)) quoting Name (PR (Role Commander)) says
   (prop nogo :(commands, principals, 'd, 'e) Form) ==>
   (M,Oi,Os) sat
   (prop nogo :(commands, principals, 'd, 'e) Form) impf
   (prop abort :(commands, principals, 'd, 'e) Form) ==>
   (M,Oi,Os) sat
   ((Name (Key (Role CA)) speaks_for Name (PR (Role CA)))
      :(commands, principals, 'd, 'e) Form) ==>
   (M,Oi,Os) sat
   Name (Key (Role CA)) says
   ((Name (Key (Staff Alice)) speaks_for Name (PR (Staff Alice)))
      :(commands, principals, 'd, 'e) Form) ==>
   (M,Oi,Os) sat
   Name (PR (Role CA)) controls
   ((Name (Key (Staff Alice)) speaks_for Name (PR (Staff Alice)))
      :(commands, principals, 'd, 'e) Form) ==>
   (M,Oi,Os) sat
   Name (Key (Staff Bob)) quoting Name (PR (Role Operator)) says
   (prop abort :(commands, principals, 'd, 'e) Form):
   thm
val it = (): unit
>
*** Emacs/HOL command completed ***
```

4.6 Proof on ApRuleStandDown

4.6.1 Relevant Code

```
val ApRuleStandDown =
let
  val th1 = ACL_ASSUM ``(Name (PR (Role Operator)) controls (prop abort)): (
      commands, principals, 'd, 'e)Form``
  val th2 = ACL_ASSUM ``(reps (Name (PR (Staff Bob))) (Name (PR (Role Operator
      ))) (prop abort)): (commands, principals, 'd, 'e)Form``
  val th3 = ACL_ASSUM ``(Name (Key (Staff Bob)) quoting Name (PR (Role
      Operator)) says prop abort): (commands, principals, 'd, 'e)Form``
  val th4 = ACL_ASSUM ``(prop abort impf prop stand_down):(commands,
      principals, 'd, 'e)Form``
  val th5 = ACL_ASSUM ``(Name (Key (Role CA)) speaks_for Name (PR (Role CA))):
       (commands,principals, 'd, 'e)Form``
  val th6 = ACL_ASSUM ``(Name (Key (Role CA)) says Name (Key (Staff Bob))
      speaks_for Name (PR (Staff Bob))): (commands, principals, 'd, 'e)Form``
  val th7 = ACL_ASSUM ``(Name (PR (Role CA)) controls Name (Key (Staff Bob))
      speaks_for Name (PR (Staff Bob))): (commands, principals, 'd, 'e)Form``


  val th9 = SPEAKS_FOR th5 th6;
  val th10 = CONTROLS th7 th9;
  val th11 = QUOTING_LR th3;
  val th12 = SPEAKS_FOR th10 th11
  val th13 = QUOTING_RL th12
  val th14 = REPS th2 th13 th1;
  val th15 = ACL_MP th14 th4;
  (*val th16 = SAYS ``Name(Key (Staff Bob)) quoting Name (PR (Role Operator))
      `` th15*)
  val th16 = DISCH(hd (hyp th7)) th15
  val th17 = DISCH(hd (hyp th6)) th16
  val th18 = DISCH(hd (hyp th5)) th17
  val th19 = DISCH(hd (hyp th4)) th18
  val th20 = DISCH(hd (hyp th3)) th19
  val th21 = DISCH(hd (hyp th2)) th20
in
  DISCH (hd (hyp th1)) th21
end;
```

4.6.2 Session Transcript

```
> val ApRuleStandDown =                                                     1
   |- ((M :(commands, 'b, principals, 'd, 'e) Kripke),(Oi :'d po),
    (Os :'e po)) sat
   Name (PR (Role Operator)) controls
   (prop abort :(commands, principals, 'd, 'e) Form) ==>
   (M,Oi,Os) sat
   reps (Name (PR (Staff Bob))) (Name (PR (Role Operator)))
     (prop abort :(commands, principals, 'd, 'e) Form) ==>
   (M,Oi,Os) sat
   Name (Key (Staff Bob)) quoting Name (PR (Role Operator)) says
   (prop abort :(commands, principals, 'd, 'e) Form) ==>
   (M,Oi,Os) sat
   (prop abort :(commands, principals, 'd, 'e) Form) impf
   (prop stand_down :(commands, principals, 'd, 'e) Form) ==>
   (M,Oi,Os) sat
   ((Name (Key (Role CA)) speaks_for Name (PR (Role CA)))
      :(commands, principals, 'd, 'e) Form) ==>
   (M,Oi,Os) sat
   Name (Key (Role CA)) says
   ((Name (Key (Staff Bob)) speaks_for Name (PR (Staff Bob)))
      :(commands, principals, 'd, 'e) Form) ==>
   (M,Oi,Os) sat
   Name (PR (Role CA)) controls
   ((Name (Key (Staff Bob)) speaks_for Name (PR (Staff Bob)))
      :(commands, principals, 'd, 'e) Form) ==>
   (M,Oi,Os) sat (prop stand_down :(commands, principals, 'd, 'e) Form):
   thm
val it = (): unit
>
*** Emacs/HOL command completed ***
```

Appendix A

Source Code for example1Script.sml

The following code is from *emample1Script.sml*, which is located in directory "../HOL/"

```
(**************************************************)
(* Engineering Assurance Lab: example1Script.sml   *)
(* Shiu-Kai Chin                                   *)
(* Date: 23 September 2013                         *)
(**************************************************)

(* Interactive mode: these are theories that are in the ACL  *)
(* subdirectory pointed to in the Holmakefile file. The file *)
(* acl_infRules contains the ML functions that are the       *)
(* inference rules in the access control logic.              *)

(* only necessary when working interactively
app load ["acl_infRules","aclrulesTheory","aclDrulesTheory","example1Theory"];
open acl_infRules aclrulesTheory aclDrulesTheory example1Theory
*)

(* The following structure is similar to the module command in Haskell *)
structure example1Script = struct

open HolKernel boolLib Parse bossLib (* used by Holmake, not in interactive
    *)
open acl_infRules aclrulesTheory aclDrulesTheory (* used by Holmake and
    interactive mode *)

(**********
* create a new theory
**********)
val _ = new_theory "example1";

(* Example 1: Practice with ACL syntax in HOL *)
(********************************************************)
(* let's define a concrete example of a set of instructions *)
(********************************************************)
val _ =
Datatype
`commands = go | nogo | launch | abort`

(****************************************************)
(* Define some names of people who will be principals *)
(****************************************************)
val _ =
Datatype
```

```
`staff = Alice | Bob | Carol | Dan`

(* The simplest access-control logic formula is a proposition *)
val commandProp = ``(prop go):(commands,staff,'d,'e)Form``;

(* We can still use type variables for propositions *)
val xProposition = ``(prop x):('a,'c,'d,'e)Form``

(* We can be completely general *)
val x = ``x:('a,'c,'d,'e)Form``

(* Mapping type :staff to type :staff Princ *)
val princTerm = ``Name Alice``;
(* Principals make statements *)
val term1 = ``((Name Alice) says (prop go)):(commands,staff,'d,'e)Form``;

(* Principals have jurisdiction *)
val term2 = ``((Name Alice) controls (prop go)):(commands,staff,'d,'e)Form``;

(* Alice with Bob says <go> *)
val term3 =
 ``((Name Alice) meet (Name Bob) says (prop launch)):(commands,staff,'d,'e)
    Form``;

(* Carol | Dan says <nogo> *)
val term4 =
 ``((Name Carol) quoting (Name Dan) says (prop nogo)):(commands,staff,'d,'e)
    Form``;

(* Dan => Carol *)
val term5 =
 ``((Name Dan) speaks_for (Name Carol)):(commands,staff,'d,'e)Form``;

(******************)
(* Our first proof *)
(******************)
(* Develop the proof line by line *)


val th1 = ACL_ASSUM``((Name Alice) says (prop go)):(commands,staff,'d,'e)Form
   ``;
val th2 = ACL_ASSUM``((Name Alice) controls (prop go)):(commands,staff,'d,'e)
   Form``;
val th3 = CONTROLS th2 th1;
val th4 = DISCH(hd(hyp th2)) th3;
val th5 = DISCH(hd(hyp th1)) th4;


(* Package up the proof into a single function *)
val example1Theorem =
let
 val th1 = ACL_ASSUM``((Name Alice) says (prop go)):(commands,staff,'d,'e)Form
    ``
 val th2 = ACL_ASSUM``((Name Alice) controls (prop go)):(commands,staff,'d,'e)
     Form``
 val th3 = CONTROLS th2 th1
 val th4 = DISCH(hd(hyp th2)) th3
in
 DISCH(hd(hyp th1)) th4
end;

(* We save the theorem by using save_thm *)
val _ = save_thm("example1Theorem",example1Theorem)



(***************************************************************************
    *)
(* A goal-oriented proof
    *)
(***************************************************************************
    *)

val example1TheoremA =
TAC_PROOF(([],
``((M :(commands, 'b, staff, 'd, 'e) Kripke),(Oi :'d po),(Os :'e po)) sat
   Name Alice says (prop go) ==>
  (M,Oi,Os) sat Name Alice controls (prop go) ==>
  (M,Oi,Os) sat (prop go)``),
PROVE_TAC[Controls])

val _ = save_thm("example1TheoremA",example1TheoremA)



(***************************************************************************
    *)
(* A proof using ACL_CONTROLS_TAC
    *)
(***************************************************************************
    *)
val example1TheoremB =
TAC_PROOF(([],
``((M :(commands, 'b, staff, 'd, 'e) Kripke),(Oi :'d po),(Os :'e po)) sat
   Name Alice says (prop go) ==>
  (M,Oi,Os) sat Name Alice controls (prop go) ==>
  (M,Oi,Os) sat (prop go)``),
REPEAT STRIP_TAC THEN
ACL_CONTROLS_TAC ``Name Alice`` THEN
ASM_REWRITE_TAC[])

val _ = save_thm("example1TheoremB",example1TheoremB)

(* Example 2 *)
(* develop the proof line by line *)
```

```
 val th1 = ACL_ASSUM``((Name Alice) says (prop go)):(commands,staff,'d,'e)Form
     ``;
 val th2 = ACL_ASSUM``((Name Alice) speaks_for (Name Bob)):(commands,staff,'d
     ,'e)Form``;
 val th3 = ACL_ASSUM``((Name Bob) controls (prop go)):(commands,staff,'d,'e)
     Form``;
 val th4 = SPEAKS_FOR th2 th1;
 val th5 = CONTROLS th3 th4;
 val th6 = DISCH(hd(hyp th3)) th5;
 val th7 = DISCH(hd(hyp th2)) th6;
 val th8 = DISCH(hd(hyp th1)) th7;

(* Package up the proof into a single function *)
val example2Theorem =
let
 val th1 = ACL_ASSUM``((Name Alice) says (prop go)):(commands,staff,'d,'e)Form
     ``
 val th2 = ACL_ASSUM``((Name Alice) speaks_for (Name Bob)):(commands,staff,'d
     ,'e)Form``
 val th3 = ACL_ASSUM``((Name Bob) controls (prop go)):(commands,staff,'d,'e)
     Form``
 val th4 = SPEAKS_FOR th2 th1
 val th5 = CONTROLS th3 th4
 val th6 = DISCH(hd(hyp th3)) th5
 val th7 = DISCH(hd(hyp th2)) th6
in
 DISCH(hd(hyp th1)) th7
end;

(* We save the theorem by using save_thm *)
val _ = save_thm("example2Theorem",example2Theorem)

(***************************************************************************
    *)
(* A goal-oriented proof
    *)
(***************************************************************************
    *)
val example2TheoremA =
TAC_PROOF(([],
``((M :(commands, 'b, staff, 'd, 'e) Kripke),(Oi :'d po), (Os :'e po)) sat
   Name Alice says (prop go) ==>
  (M,Oi,Os) sat (Name Alice speaks_for Name Bob) ==>
  (M,Oi,Os) sat Name Bob controls (prop go) ==>
  (M,Oi,Os) sat (prop go)``),
PROVE_TAC[Derived_Speaks_For, Controls])

val _ = save_thm("example2TheoremA",example2TheoremA)

(***************************************************************************
    *)
(* A goal-oriented proof using tactics
    *)
```

```
(****************************************************************************
    *)
val example2TheoremB =
TAC_PROOF(([],
``((M :(commands, 'b, staff, 'd, 'e) Kripke),(Oi :'d po), (Os :'e po)) sat
   Name Alice says (prop go) ==>
  (M,Oi,Os) sat (Name Alice speaks_for Name Bob) ==>
  (M,Oi,Os) sat Name Bob controls (prop go) ==>
  (M,Oi,Os) sat (prop go)``),
REPEAT STRIP_TAC THEN
ACL_CONTROLS_TAC ``Name Bob`` THEN
ASM_REWRITE_TAC[] THEN
PAT_ASSUM
``(M,Oi,Os) sat (Name Alice speaks_for Name Bob)``
(fn th1 =>
 (PAT_ASSUM
  ``(M,Oi,Os) sat (Name Alice says (prop go))``
  (fn th2 => ASSUME_TAC(SPEAKS_FOR th1 th2)))) THEN
ASM_REWRITE_TAC[])

val _ = save_thm("example2TheoremB",example2TheoremB)

(* Example 3 *)
(* develop the proof line by line *)
val th1 = ACL_ASSUM``((prop go) impf (prop launch)):(commands,staff,'d,'e)Form
    ``;
val th2 = ACL_ASSUM``(prop go):(commands,staff,'d,'e)Form``;
val th3 = ACL_MP th2 th1;
val th4 = SAYS ``(Name Carol):staff Princ`` th3;
val th5 = DISCH(hd(hyp th2)) th4;
val th6 = DISCH(hd(hyp th1)) th5;

(* Package up the proof into a single function *)
val example3Theorem =
let
 val th1 = ACL_ASSUM``((prop go) impf (prop launch)):(commands,staff,'d,'e)
     Form``
 val th2 = ACL_ASSUM``(prop go):(commands,staff,'d,'e)Form``
 val th3 = ACL_MP th2 th1
 val th4 = SAYS ``(Name Carol):staff Princ`` th3
 val th5 = DISCH(hd(hyp th2)) th4
in
 DISCH(hd(hyp th1)) th5
end;

(* We save the theorem by using save_thm *)
val _ = save_thm("example3Theorem",example3Theorem)



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

```
(* A goal-oriented proof
    *)
(**************************************************************************
    *)
val example3TheoremA =
TAC_PROOF(([],concl example3Theorem),
PROVE_TAC[Modus_Ponens,Says])

val _ = save_thm("example3TheoremA",example3TheoremA)

(**************************************************************************
    *)
(* Mono_Reps_Theorem
    *)
(**************************************************************************
    *)
val Mono_Reps_Theorem =
TAC_PROOF(([],
``(M,Oi,Os) sat ((Q controls f):('a,'c,'d,'e)Form) ==>
  (M,Oi,Os) sat ((reps P Q f):('a,'c,'d,'e)Form) ==>
  (M,Oi,Os) sat ((P' quoting Q' says f):('a,'c,'d,'e)Form) ==>
  (M,Oi,Os) sat ((P' speaks_for P):('a,'c,'d,'e)Form) ==>
  (M,Oi,Os) sat ((Q' speaks_for Q):('a,'c,'d,'e)Form) ==>
  (M,Oi,Os) sat (f:('a,'c,'d,'e)Form)``),
PROVE_TAC[Controls,Reps,Mono_speaks_for,Derived_Speaks_For])

val _ = save_thm("Mono_Reps_Theorem",Mono_Reps_Theorem)

(* ==== start here ====

==== end here ==== *)

(*****************************)
(* Print and export the theory *)
(*****************************)
val _ = print_theory "-";

val _ = export_theory();

end (* structure *)
```

Appendix B

Source Code for solutions1Script.sml

The following code is from *solutions1Script.sml*, which is located in directory "../HOL/"

```
structure solutions1Script = struct

open HolKernel boolLib Parse bossLib;
open acl_infRules aclrulesTheory aclDrulesTheory ;
open example1Theory;

val _ = new_theory "solutions1";



val aclExerciseTheorem1 =
let
 val th1 = ACL_ASSUM``((Name Alice) says (prop go)):(commands, staff, 'd, 'e)
     Form``
 val th2 = ACL_ASSUM``((Name Bob) says (prop go)):(commands, staff, 'd, 'e)
     Form``
 val th3 = ACL_CONJ th1 th2
 val th4 = AND_SAYS_RL th3
 val th5 = DISCH(hd(hyp th2)) th4
in
 DISCH (hd(hyp th1)) th5
end;
val _ = save_thm("aclExerciseTheorem1",aclExerciseTheorem1);

val aclExerciseTheorem1A =
TAC_PROOF(
([],
``((M :(commands, 'b, staff, 'd, 'e) Kripke),(Oi :'d po),(Os :'e po)) sat
   ((Name Alice) says (prop go)) ==>
  (M,Oi,Os) sat ((Name Bob) says (prop go)) ==>
  (M,Oi,Os) sat (((Name Alice) meet (Name Bob)) says (prop go))``
),
PROVE_TAC[Conjunction, And_Says_Eq]
);
val _ = save_thm("aclExerciseTheorem1A",aclExerciseTheorem1A);



val aclExerciseTheorem1B =
TAC_PROOF(
([],
``((M :(commands, 'b, staff, 'd, 'e) Kripke),(Oi :'d po),(Os :'e po)) sat
    ((Name Alice) says (prop go)) ==>
   (M,Oi,Os) sat ((Name Bob) says (prop go)) ==>
   (M,Oi,Os) sat (((Name Alice) meet (Name Bob)) says (prop go))``
),
REPEAT STRIP_TAC THEN
ACL_AND_SAYS_RL_TAC THEN
ACL_CONJ_TAC THEN
REWRITE_TAC [] THEN
ASM_REWRITE_TAC []
);

val _ = save_thm("aclExerciseTheorem1B",aclExerciseTheorem1B);

val aclExerciseTheorem2 =
let
 val th1 = ACL_ASSUM``((Name Alice) says (prop go)):(commands, staff, 'd, 'e)
     Form``
 val th2 = ACL_ASSUM``((Name Alice) controls (prop go)):(commands, staff, 'd,
     'e)Form``
 val th3 = ACL_ASSUM``((prop go) impf (prop val))``
 val th4 = CONTROLS th2 th1
 val th5 = SAYS ``(Name Bob)`` 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("aclExerciseTheorem2",aclExerciseTheorem2);

val aclExerciseTheorem2A =
TAC_PROOF(
([],
``((M:(commands, 'b, staff, 'd, 'e) Kripke),(Oi: 'd po),(Os: 'e po)) sat
     (Name Alice says (prop go)) ==>
     (M, Oi, Os) sat ((Name Alice) controls (prop go)) ==>
     (M, Oi, Os) sat ((prop go) impf (prop launch)) ==>
     (M, Oi, Os) sat ((Name Bob) says (prop launch))``
),
PROVE_TAC[Modus_Ponens, Controls, Says]);
val _ = save_thm("aclExerciseTheorem2A",aclExerciseTheorem2A);


val aclExerciseTheorem2B =
TAC_PROOF(([],
``((M:(commands, 'b, staff, 'd, 'e) Kripke),(Oi: 'd po),(Os: 'e po)) sat
     ((Name Alice) says (prop go)) ==>
     (M, Oi, Os) sat ((Name Alice) controls (prop go)) ==>
     (M, Oi, Os) sat ((prop go) impf (prop launch)) ==>
     (M, Oi, Os) sat ((Name Bob) says (prop launch))``
),
REPEAT STRIP_TAC THEN
ACL_SAYS_TAC THEN
```

```
PAT_ASSUM ``(M, Oi, Os) sat (Name Alice controls prop go)``
          (fn th1 =>(PAT_ASSUM ``(M,Oi, Os) sat (Name Alice says prop go)``
                     (fn th2 => ASSUME_TAC (CONTROLS th1 th2)))) THEN

PAT_ASSUM ``(M, Oi, Os) sat ((prop go) impf (prop launch))``
          (fn th1 =>(PAT_ASSUM ``(M,Oi,Os) sat (Name Alice controls prop go)``
              (fn th2 =>(PAT_ASSUM ``(M,Oi,Os) sat (Name Alice says prop go)``
                  (fn th3 => ASSUME_TAC (ACL_MP (CONTROLS th2 th3) th1))))))
                     THEN

PROVE_TAC []
);

val _ = save_thm("aclExerciseTheorem2B",aclExerciseTheorem2B);

val _ = export_theory();
end
```

Appendix C

Source Code for conops0SolutionScript.sml

The following code is from *conops0SolutionScript.sml*, which is located in directory "../HOL/"

```
structure conops0SolutionScript = struct

open HolKernel Parse boolLib bossLib;
open acl_infRules aclrulesTheory aclDrulesTheory ;

val _ = new_theory "conops0Solution";

val _ = Datatype `commands = go | nogo | launch | abort | activate |
    stand_down`


val _ = Datatype `people = Alice | Bob`;

val _ = Datatype `roles = Commander | Operator | CA`;

val _ = Datatype `keyPrinc = Staff conops0Solution$people | Role
    conops0Solution$roles | Ap num`;

val _ = Datatype `principals = PR keyPrinc | Key keyPrinc`;


val OpRuleLaunch =
let
  val th1 = ACL_ASSUM ``(Name (PR (Role Commander)) controls (prop go)): (
      commands, principals, 'd, 'e)Form``
  val th2 = ACL_ASSUM ``(reps (Name (PR (Staff Alice))) (Name (PR (Role
      Commander))) (prop go)): (commands, principals, 'd, 'e)Form``
  val th3 = ACL_ASSUM ``(Name (Key (Staff Alice)) quoting Name (PR (Role
      Commander)) says prop go): (commands, principals, 'd, 'e)Form``
  val th4 = ACL_ASSUM ``(prop go impf prop launch):(commands, principals, 'd,
      'e)Form``
  val th5 = ACL_ASSUM ``(Name (Key (Role CA)) speaks_for Name (PR (Role CA))):
       (commands,principals, 'd, 'e)Form``
  val th6 = ACL_ASSUM ``(Name (Key (Role CA)) says Name (Key (Staff Alice))
      speaks_for Name (PR (Staff Alice))): (commands, principals, 'd, 'e)Form
      ``
  val th7 = ACL_ASSUM ``(Name (PR (Role CA)) controls Name (Key (Staff Alice))
       speaks_for Name (PR (Staff Alice))): (commands, principals, 'd, 'e)Form
      ``


  val th9 = SPEAKS_FOR th5 th6;
  val th10 = CONTROLS th7 th9;
  val th11 = QUOTING_LR th3;
  val th12 = SPEAKS_FOR th10 th11
  val th13 = QUOTING_RL th12
  val th14 = REPS th2 th13 th1;
  val th15 = ACL_MP th14 th4;
  val th16 = SAYS ``Name(Key (Staff Bob)) quoting Name (PR (Role Operator))``
      th15
  val th17 = DISCH(hd (hyp th7)) th16
  val th18 = DISCH(hd (hyp th6)) th17
  val th19 = DISCH(hd (hyp th5)) th18
  val th20 = DISCH(hd (hyp th4)) th19
  val th21 = DISCH(hd (hyp th3)) th20
  val th22 = DISCH(hd (hyp th2)) th21
in
  DISCH (hd (hyp th1)) th22
end;


val ApRuleActive =
let
  val th1 = ACL_ASSUM ``(Name (PR (Role Operator)) controls (prop launch)): (
      commands, principals, 'd, 'e)Form``
  val th2 = ACL_ASSUM ``(reps (Name (PR (Staff Bob))) (Name (PR (Role Operator
      ))) (prop launch)): (commands, principals, 'd, 'e)Form``
  val th3 = ACL_ASSUM ``(Name (Key (Staff Bob)) quoting Name (PR (Role
      Operator)) says prop launch): (commands, principals, 'd, 'e)Form``
  val th4 = ACL_ASSUM ``(prop launch impf prop activate):(commands, principals
      , 'd, 'e)Form``
  val th5 = ACL_ASSUM ``(Name (Key (Role CA)) speaks_for Name (PR (Role CA))):
       (commands,principals, 'd, 'e)Form``
  val th6 = ACL_ASSUM ``(Name (Key (Role CA)) says Name (Key (Staff Bob))
      speaks_for Name (PR (Staff Bob))): (commands, principals, 'd, 'e)Form``
  val th7 = ACL_ASSUM ``(Name (PR (Role CA)) controls Name (Key (Staff Bob))
      speaks_for Name (PR (Staff Bob))): (commands, principals, 'd, 'e)Form``


  val th9 = SPEAKS_FOR th5 th6;
  val th10 = CONTROLS th7 th9;
  val th11 = QUOTING_LR th3;
  val th12 = SPEAKS_FOR th10 th11
  val th13 = QUOTING_RL th12
  val th14 = REPS th2 th13 th1;
  val th15 = ACL_MP th14 th4;
  (*val th16 = SAYS ``Name(Key (Staff Bob)) quoting Name (PR (Role Operator))
      `` th15*)
  val th16 = DISCH(hd (hyp th7)) th15
  val th17 = DISCH(hd (hyp th6)) th16
  val th18 = DISCH(hd (hyp th5)) th17
  val th19 = DISCH(hd (hyp th4)) th18
  val th20 = DISCH(hd (hyp th3)) th19
  val th21 = DISCH(hd (hyp th2)) th20
in
```

```
  DISCH (hd (hyp th1)) th21
end;



val OpRuleAbort =
let
  val th1 = ACL_ASSUM ``(Name (PR (Role Commander)) controls (prop nogo)): (
     commands, principals, 'd, 'e)Form``
  val th2 = ACL_ASSUM ``(reps (Name (PR (Staff Alice))) (Name (PR (Role
     Commander))) (prop nogo)): (commands, principals, 'd, 'e)Form``
  val th3 = ACL_ASSUM ``(Name (Key (Staff Alice)) quoting Name (PR (Role
     Commander)) says prop nogo): (commands, principals, 'd, 'e)Form``
  val th4 = ACL_ASSUM ``(prop nogo impf prop abort):(commands, principals, 'd,
      'e)Form``
  val th5 = ACL_ASSUM ``(Name (Key (Role CA)) speaks_for Name (PR (Role CA))):
      (commands,principals, 'd, 'e)Form``
  val th6 = ACL_ASSUM ``(Name (Key (Role CA)) says Name (Key (Staff Alice))
     speaks_for Name (PR (Staff Alice))): (commands, principals, 'd, 'e)Form
     ``
  val th7 = ACL_ASSUM ``(Name (PR (Role CA)) controls Name (Key (Staff Alice))
      speaks_for Name (PR (Staff Alice))): (commands, principals, 'd, 'e)Form
     ``


  val th9 = SPEAKS_FOR th5 th6;
  val th10 = CONTROLS th7 th9;
  val th11 = QUOTING_LR th3;
  val th12 = SPEAKS_FOR th10 th11
  val th13 = QUOTING_RL th12
  val th14 = REPS th2 th13 th1;
  val th15 = ACL_MP th14 th4;
  val th16 = SAYS ``Name(Key (Staff Bob)) quoting Name (PR (Role Operator))``
     th15
  val th17 = DISCH(hd (hyp th7)) th16
  val th18 = DISCH(hd (hyp th6)) th17
  val th19 = DISCH(hd (hyp th5)) th18
  val th20 = DISCH(hd (hyp th4)) th19
  val th21 = DISCH(hd (hyp th3)) th20
  val th22 = DISCH(hd (hyp th2)) th21
in
  DISCH (hd (hyp th1)) th22
end;


val ApRuleStandDown =
let
  val th1 = ACL_ASSUM ``(Name (PR (Role Operator)) controls (prop abort)): (
     commands, principals, 'd, 'e)Form``
  val th2 = ACL_ASSUM ``(reps (Name (PR (Staff Bob))) (Name (PR (Role Operator
     ))) (prop abort)): (commands, principals, 'd, 'e)Form``
  val th3 = ACL_ASSUM ``(Name (Key (Staff Bob)) quoting Name (PR (Role
     Operator)) says prop abort): (commands, principals, 'd, 'e)Form``
  val th4 = ACL_ASSUM ``(prop abort impf prop stand_down):(commands,
      principals, 'd, 'e)Form``
  val th5 = ACL_ASSUM ``(Name (Key (Role CA)) speaks_for Name (PR (Role CA))):
       (commands,principals, 'd, 'e)Form``
  val th6 = ACL_ASSUM ``(Name (Key (Role CA)) says Name (Key (Staff Bob))
      speaks_for Name (PR (Staff Bob))): (commands, principals, 'd, 'e)Form``
  val th7 = ACL_ASSUM ``(Name (PR (Role CA)) controls Name (Key (Staff Bob))
      speaks_for Name (PR (Staff Bob))): (commands, principals, 'd, 'e)Form``


  val th9 = SPEAKS_FOR th5 th6;
  val th10 = CONTROLS th7 th9;
  val th11 = QUOTING_LR th3;
  val th12 = SPEAKS_FOR th10 th11
  val th13 = QUOTING_RL th12
  val th14 = REPS th2 th13 th1;
  val th15 = ACL_MP th14 th4;
  (*val th16 = SAYS ``Name(Key (Staff Bob)) quoting Name (PR (Role Operator))
      `` th15*)
  val th16 = DISCH(hd (hyp th7)) th15
  val th17 = DISCH(hd (hyp th6)) th16
  val th18 = DISCH(hd (hyp th5)) th17
  val th19 = DISCH(hd (hyp th4)) th18
  val th20 = DISCH(hd (hyp th3)) th19
  val th21 = DISCH(hd (hyp th2)) th20
in
  DISCH (hd (hyp th1)) th21
end;

val _ = save_thm("OpRuleLaunch_thm", OpRuleLaunch);
val _ = save_thm("ApRuleActive_thm", ApRuleActive);
val _ = save_thm("OpRuleAbort_thm", OpRuleAbort);
val _ = save_thm("ApRuleStandDown_thm", ApRuleStandDown);

val _ = export_theory();


end
```