Project 6 Report

Kyle Peppe

February 20, 2020

Abstract

This project was to demonstrate my command of proving several different theories using the tacticals, types, and in Chapter 13 doing slightly simpler goal oriented proofs. I have completed the exercises 13.10.1, 13.10.2, and 14.4.1. 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

Contents

Acknowledgments: I would like to acknowledge the 2 professors, Professor Chin and Professor Hamner, that have helped me begin to understand this new ML programming language. Also to Syracuse University for accepting me to this Masters program in Cybersecurity.

Chapter 1

Executive Summary

All requirements for this project are satisfied. Specifically,

Report Contents

Our report has the following content:

Reproducibility in ML and LaTeX

The ML and LaTeX source files compile with no errors.

Chapter 2

Excercise 13.10.1

2.1 Problem statement

Do a goal-oriented proof for parts A, B, and C:

Alice says go Bob says go Alice and Bob says go

For each part we use first the inference rules, then for part B just prove_tac and for part C using specific ACL tactics

2.2 Forward proof of theorem aclExercise1

```
(* Exercise 1            *)
val aclExercise1 =
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;
```

Solution/Theorem

```
# # # # # # # # # # # val aclExercise1 =                                   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
```

2.3 USE PROVE TAC only to prove theorem aclExercise1B

```
(* Exercise 1A            *)
val aclExercise1A =
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]
```

```
            )
```

Solution/Theorem

```
# # # # # # # Meson search level: .....                                                                  2
val aclExercise1A =
   |- ((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 using ACL tactics

```
(* Exercise 1B           *)
val aclExercise1B =
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
          PROVE_TAC[]
         )
```

Solution/Theorem

```
# # # # # # # # # # Meson search level: ..                                                               3
Meson search level: ..
val aclExercise1B =
   |- ((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
```

Chapter 3

Excercise 13.10.2

3.1 Problem statement

Do a goal-oriented proof for parts A, B, and C:

Alice says go Alice Controls go go implies launch Bob says launch

For each part we use first the inference rules, then for part B just prove_tac and for part C using specific ACL tactics

3.2 Forward proof of theorem aclExercise2

```
val aclExercise2 =
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 launch)):
        (commands,staff,'d,'e)Form‘‘
        val th4 = CONTROLS th2 th1
        val th5 = ACL_MP th4 th3
        val th6 = SAYS ‘‘(Name Bob):staff Princ‘‘ th5
        val th7 = DISCH(hd(hyp th3)) th6
        val th8 = DISCH(hd(hyp th2)) th7
in
        DISCH(hd(hyp th1)) th8
end;
```

```
 # # # # # # # # # # # # # # # val aclExercise2 =
    |- ((M :(commands, 'b, staff, 'd, 'e) Kripke),(Oi :'d po),                4
     (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
```

3.3 Use PROVE TAC only to prove theorem aclExercise2B

```
val aclExercise2A =
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[Controls, Modus_Ponens, Says]
           );
```

```
# # # # # # # # Meson search level: .......
val aclExercise2A =                                                      5
   |- ((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
```

3.4 Goal Oriented proof using ACL tactics

```
val aclExercise2B =
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 says (prop go)``
          (fn th1 => (PAT_ASSUM ``(M,Oi,Os) sat Name Alice controls
          (prop go)`` (fn th2 => ASSUME_TAC(CONTROLS th2 th1)))) THEN
         PAT_ASSUM ``(M,Oi,Os) sat (prop go)``
          (fn th1 => (PAT_ASSUM ``(M,Oi,Os) sat (prop go) impf (prop launch)``
          (fn th2 => PROVE_TAC[(ACL_MP th1 th2)])))
        )
```

```
# # # # # # # # # # # # # # # <<HOL message: inventing new type variable names: 'a, 'b, 'c>>
<<HOL message: inventing new type variable names: 'a, 'b, 'c, 'd>>       6
Meson search level: ..
val aclExercise2B =
   |- ((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
```

Chapter 4

Excercise 14.4.1

4.1 Problem statement

Since the actual HOL Theorems are too long to individually lay out here I am just giving the 4 theorem names that we were given to solve: OpRuleLaunch_thm OpRuleAbort_thm ApRuleActivate_thm ApRuleStandDown_thm

4.2 Definition of datatypes

```
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 people | Role roles | Ap num‘
val _ = Datatype ‘principals = PR keyPrinc | Key keyPrinc‘
```

```
<<HOL message: Defined type: "commands">>
> <<HOL message: Defined type: "people">>                                          7
> <<HOL message: Defined type: "roles">>
> <<HOL message: Defined type: "keyPrinc">>
> <<HOL message: Defined type: "principals">>
```

4.3 Proof of OpRuleLaunch thm

```
val OpRuleLaunch_thm =
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 th8 = SPEAKS_FOR th5 th6
        val th9 = CONTROLS th7 th8
        val th10 = IDEMP_SPEAKS_FOR ``Name (PR (Role Commander))
        : principals Princ``
        val th11 = INST_TYPE [``:'a`` |-> ``:commands``] th10
        val th12 = MONO_SPEAKS_FOR th9 th11
        val th13 = SPEAKS_FOR th12 th3
        val th14 = REPS th2 th13 th1
        val th15 = ACL_MP th14 th4
        val th16 = SAYS ``(Name (Key (Staff Bob))) quoting
        (Name (PR (Role Operator))) : principals Princ`` 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 OpRuleLaunch_thm =
   |- ((M :(commands, 'b, principals, 'd, 'e) Kripke),(Oi :'d po),                    8
    (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 of OpRuleAbort thm

```
val OpRuleAbort_thm =
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 th8 = SPEAKS_FOR th5 th6
        val th9 = CONTROLS th7 th8
        val th10 = IDEMP_SPEAKS_FOR ``Name (PR (Role Commander))
        : principals Princ``
        val th11 = INST_TYPE [``:'a`` |-> ``:commands``] th10
        val th12 = MONO_SPEAKS_FOR th9 th11
        val th13 = SPEAKS_FOR th12 th3
        val th14 = REPS th2 th13 th1
        val th15 = ACL_MP th14 th4
        val th16 = SAYS ``(Name (Key (Staff Bob))) quoting
        (Name (PR (Role Operator))) : principals Princ`` 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 OpRuleAbort_thm =
    |- ((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
```

9

4.5 Proof of ApRuleActivate thm

```
val ApRuleActivate_thm =
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 th8 = SPEAKS_FOR th5 th6
        val th9 = CONTROLS th7 th8
        val th10 = IDEMP_SPEAKS_FOR ``Name (PR (Role Operator))
        : principals Princ``
        val th11 = INST_TYPE [``:'a`` |-> `` :commands``] th10
        val th12 = MONO_SPEAKS_FOR th9 th11
        val th13 = SPEAKS_FOR th12 th3
        val th14 = REPS th2 th13 th1
        val th15 = ACL_MP th14 th4
        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 ApRuleActivate_thm =
    |- ((M :(commands, 'b, principals, 'd, 'e) Kripke),(Oi :'d po),           10
     (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
```

4.6 Proof of ApRuleStandDown thm

```
val ApRuleStandDown_thm =
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 th8 = SPEAKS_FOR th5 th6
        val th9 = CONTROLS th7 th8
        val th10 = IDEMP_SPEAKS_FOR ``Name (PR (Role Operator)):
        principals Princ``
        val th11 = INST_TYPE [``:'a`` |-> ``:commands``] th10
        val th12 = MONO_SPEAKS_FOR th9 th11
        val th13 = SPEAKS_FOR th12 th3
        val th14 = REPS th2 th13 th1
        val th15 = ACL_MP th14 th4
        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 ApRuleStandDown_thm =
```

11

```
   |- ((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
```

Appendix A

Source code for Example1Script

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

Appendix B

Source code for 13.10.1 and 13.10.2

```
(**************************************************************************)
(* Author: Kyle Peppe                                                     *)
(* Exercises 13.10.1 and 13.10.2                                          *)
(* Date: 2/11/20                                                          *)
(**************************************************************************)

(* Beginning commands                            *)
(* Including opens and setting theory name       *)
structure solutions1Script = struct

open HolKernel Parse boolLib bossLib
open acl_infRules aclrulesTheory aclDrulesTheory example1Theory

val _ = new_theory "solutions1";

(* Exercise 1            *)
val aclExercise1 =
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;

(* Save the Theory       *)
val _ = save_thm("aclExercise1",aclExercise1)

(* Exercise 1A           *)
val aclExercise1A =
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]
         )

(* Save the Theory       *)
```

```
val _ = save_thm("aclExercise1A",aclExercise1A)

(* Exercise 1B          *)
val aclExercise1B =
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
          PROVE_TAC[]
         )

(* Save the Theory      *)
val _ = save_thm("aclExercise1B",aclExercise1B)

(* Exercise 2           *)
val aclExercise2 =
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 launch)):
        (commands,staff,'d,'e)Form``
        val th4 = CONTROLS th2 th1
        val th5 = ACL_MP th4 th3
        val th6 = SAYS ``(Name Bob):staff Princ`` th5
        val th7 = DISCH(hd(hyp th3)) th6
        val th8 = DISCH(hd(hyp th2)) th7
in
        DISCH(hd(hyp th1)) th8
end;

(* Save the Theory      *)
val _ = save_thm("aclExercise2",aclExercise2)

(* Exercise 2A          *)
val aclExercise2A =
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[Controls, Modus_Ponens, Says]
          );

(* Save the Theory      *)
val _ = save_thm("aclExercise2A",aclExercise2A)
```

```
(* Exercise 2B          *)
val aclExercise2B =
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 says (prop go)``
          (fn th1 => (PAT_ASSUM ``(M,Oi,Os) sat Name Alice controls
          (prop go)`` (fn th2 => ASSUME_TAC(CONTROLS th2 th1)))) THEN
         PAT_ASSUM ``(M,Oi,Os) sat (prop go)``
          (fn th1 => (PAT_ASSUM ``(M,Oi,Os) sat (prop go) impf (prop launch)``
          (fn th2 => PROVE_TAC[(ACL_MP th1 th2)])))
        )

(* Save the Theory      *)
val _ = save_thm("aclExercise2B",aclExercise2B)

(* Export, Print, End   *)
val _ = export_theory();
val _ = print_theory "-";

end
```

Appendix C

Source code for 14.4.1

```
(****************************************************************************)
(* Author: Kyle Peppe                                                       *)
(* Exercise 14.4.1                                                          *)
(* Date: 2/11/20                                                            *)
(****************************************************************************)

(* Beginning commands, open calls, and setting theory name        *)
structure conops0SolutionScript = struct
open HolKernel Parse boolLib bossLib
open acl_infRules aclrulesTheory aclDrulesTheory

val _ = new_theory ”conops0Solution”;

(* Setting datatypes                                              *)
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 people | Role roles | Ap num‘
val _ = Datatype ‘principals = PR keyPrinc | Key keyPrinc‘

val OpRuleLaunch_thm =
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 Commande
                  says (prop go)) : (commands,principals,'d,'e)Form‘‘
        val th4 = ACL_ASSUM ‘‘((prop go) impf (prop launch)) : (commands,principals,'d,'e)F
        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 th8 = SPEAKS_FOR th5 th6
        val th9 = CONTROLS th7 th8
        val th10 = IDEMP_SPEAKS_FOR ‘‘Name (PR (Role Commander)) : principals Princ‘‘
        val th11 = INST_TYPE [‘‘:'a‘‘ |-> ‘‘:commands‘‘] th10
        val th12 = MONO_SPEAKS_FOR th9 th11
        val th13 = SPEAKS_FOR th12 th3
        val th14 = REPS th2 th13 th1
        val th15 = ACL_MP th14 th4
        val th16 = SAYS ``(Name (Key (Staff Bob))) quoting (Name (PR (Role Operator))) :
                   principals Princ`` 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;

(* Save the Theory                                    *)
val _ = save_thm("OpRuleLaunch_thm", OpRuleLaunch_thm)

val OpRuleAbort_thm =
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 Commande
                  says (prop nogo)) : (commands,principals,'d,'e)Form``
        val th4 = ACL_ASSUM ``((prop nogo) impf (prop abort)) : (commands,principals,'d,'e
        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 th8 = SPEAKS_FOR th5 th6
        val th9 = CONTROLS th7 th8
        val th10 = IDEMP_SPEAKS_FOR ``Name (PR (Role Commander)): principals Princ``
        val th11 = INST_TYPE [``:'a`` |-> ``:commands``] th10
        val th12 = MONO_SPEAKS_FOR th9 th11
        val th13 = SPEAKS_FOR th12 th3
        val th14 = REPS th2 th13 th1
        val th15 = ACL_MP th14 th4
        val th16 = SAYS ``(Name (Key (Staff Bob))) quoting (Name (PR (Role Operator)))
                   : principals Princ`` 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;

(* Save the Theory                                    *)
val _ = save_thm("OpRuleAbort_thm",OpRuleAbort_thm)
val ApRuleActivate_thm =
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))) (pro
                  : (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,
        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 th8 = SPEAKS_FOR th5 th6
        val th9 = CONTROLS th7 th8
        val th10 = IDEMP_SPEAKS_FOR ``Name (PR (Role Operator)) : principals Princ``
        val th11 = INST_TYPE [``:'a`` |-> `` :commands``] th10
        val th12 = MONO_SPEAKS_FOR th9 th11
        val th13 = SPEAKS_FOR th12 th3
        val th14 = REPS th2 th13 th1
        val th15 = ACL_MP th14 th4
        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;

(* Save the Theory                                                    *)
val _ = save_thm("ApRuleActivate_thm",ApRuleActivate_thm)


val ApRuleStandDown_thm =
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
        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 th8 = SPEAKS_FOR th5 th6
        val th9 = CONTROLS th7 th8
        val th10 = IDEMP_SPEAKS_FOR ``Name (PR (Role Operator)): principals Princ``
        val th11 = INST_TYPE [``:'a`` |-> ``:commands``] th10
        val th12 = MONO_SPEAKS_FOR th9 th11
        val th13 = SPEAKS_FOR th12 th3
        val th14 = REPS th2 th13 th1
        val th15 = ACL_MP th14 th4
        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;

(* Save the Theory                                          *)
val _ = save_thm("ApRuleStandDown_thm",ApRuleStandDown_thm)

(* Exporting all the Theories into one file                 *)
val _ = export_theory();
val _ = print_theory "-";

end
```