Report for Project08

Xiaozhi Li

October 29, 2017

Abstract

testing XX Project 08 using LaTeX. In this project we began to prove Access control Logic theories using pre built ACL theories in HOL. In this project the theorems are coded in HOL and generated using EmitTeX. All HOL source files are included in the HOL folder.

Acknowledgments: This report follows the hand book Certified Security by Design Using Higer Order Logic, and course instructions from CIS400-CSBD. In this report, the solution code problem 14.0.2 was a solution given by Dr. Shiu-kai Chin.

Contents

Chapter 1

Executive Summary

Not all requirements for this project are satisfied. Specifically, we utilized HOL to prove the following theorems:

[example1Theorem]

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

[example1TheoremA]

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

[example1TheoremB]

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

[example2Theorem]

```
⊢ (M,Oi,Os) 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
```

[example2TheoremA]

```
⊢ (M,Oi,Os) 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
```

[example2TheoremB]

```
⊢ (M,Oi,Os) 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
```

[example3Theorem]

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

[example3TheoremA]

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

[MonoXXRepsXXTheorem]

$\vdash$ (M, Oi, Os) sat Q controls f $\Rightarrow$
(M, Oi, Os) sat reps P Q f $\Rightarrow$
(M, Oi, Os) sat P' quoting Q' says f $\Rightarrow$
(M, Oi, Os) sat P' speaks_for P $\Rightarrow$
(M, Oi, Os) sat Q' speaks_for Q $\Rightarrow$
(M, Oi, Os) sat f

[aclExercise1]

$\vdash$ (M, Oi, Os) sat Name Alice says prop go $\Rightarrow$
(M, Oi, Os) sat Name Alice controls prop go $\Rightarrow$
(M, Oi, Os) sat
Name Alice says prop go andf Name Alice controls prop go

[aclExercise1A]

$\vdash$ (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

[aclExercise2]

$\vdash$ (M, Oi, Os) sat Name Bob says prop launch

[aclExercise2A]

$\vdash$ (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 launch $\Rightarrow$
(M, Oi, Os) sat Name Bob says prop launch

[aclExercise2B]

$\vdash$ (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

[ApRuleActivateXXthm]

$\vdash$ (M, Oi, Os) sat
Name (PR (Role Operator)) controls prop launch $\Rightarrow$
(M, Oi, Os) sat
reps (Name (PR (Staff Bob))) (Name (PR (Role Operator)))
(prop launch) $\Rightarrow$
(M, Oi, Os) sat
Name (Key (Staff Bob)) quoting Name (PR (Role Operator)) says
prop launch $\Rightarrow$
(M, Oi, Os) sat prop launch impf prop activate $\Rightarrow$
(M, Oi, Os) sat
Name (Key (Role CA)) speaks_for Name (PR (Role CA)) $\Rightarrow$
(M, Oi, Os) sat
Name (Key (Role CA)) says
Name (Key (Staff Bob)) speaks_for Name (PR (Staff Bob)) $\Rightarrow$
(M, Oi, Os) sat
Name (PR (Role CA)) controls
Name (Key (Staff Bob)) speaks_for Name (PR (Staff Bob)) $\Rightarrow$
(M, Oi, Os) sat prop activate

[ApRuleStandDownXXthm]

```
⊢ (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
```

[OpRuleAbortXXthm]

```
⊢ (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 (Key (Staff Bob)) quoting Name (PR (Role Operator)) says
  prop abort
```

[OpRuleLaunchXXthm]

```
⊢ (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
```

Exercise 10.4.3 was not included in this project.

Chapter 2

Exercise 9.5.1

2.1 Problem Statement

2.2 HOL Code

2.3 Session Transcript

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

2.4 Explain Result

Hol is showing our theorem with no type errors, this means our tests have passed.

Chapter 3

Exercise 9.5.2

3.1 Problem Statement

For 9.5.2 we need to prove the therom:

3.2 HOL Code

```
val constructiveDilemmaRule=

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

3.3 Session Transcript

1

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

3.4 Explain Result

In 9.5.2, all of our theorem and theory have passed by HOL.

Chapter 4

Excersice 9.5.3

4.1 Problem Statement

For 9.5.3 we need to prove the therom:
using PROVE_TAC .

4.2 HOL Code

In 9.5.3, our relative HOL code is:

4.3 Session Transcript

1

```
> # # # # Meson search level: ................
val absorptionRule2 =
    []
|- !(p :bool) (q :bool) (r :bool) (s :bool).
     (p ==> q) /\ (r ==> s) ==> p \/ r ==> q \/ s:
   thm
> > > > > # # # # Meson search level: ................
val constructiveDilemmaRule2 =
    []
|- !(p :bool) (q :bool) (r :bool) (s :bool).
     (p ==> q) /\ (r ==> s) ==> p \/ r ==> q \/ s:
   thm
>
```

4.4 Explain Result

All tests from 9.5.3 have been passed in HOL.

Chapter 5

Exercise 9.5.2

5.1 Problem Statement

For 9.5.2 we need to prove the therom:

5.2 HOL Code

```
val constructiveDilemmaRule=

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

5.3 Session Transcript

1

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

5.4 Explain Result

In 9.5.2, all of our theorem and theory have passed by HOL.

Chapter 6

Excersice 10.4.1

6.1 Problem Statement

For 10.4.1 we need to prove the therom:

6.2 HOL Code

In 10.4.1, our relative HOL code is:

```
val problemOnethm=
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 problemOnethm =                                    1
    [..] |- M s: thm
```

6.4 Explain Result

All tests from 10.4.1 have been passed in HOL

Chapter 7

Excersice 10.4.2

7.1 Problem Statement

For 10.4.2 we need to prove the therom:

7.2 HOL Code

In 10.4.2, our relative HOL code is:

```
val problemTwothm=
TAC_PROOF(
([``p /\ q ==> r``, ``r ==> s``, ``~s``], ``p ==> ~q``),
(PAT_ASSUM ``r ==>s``
            (fn th =>
            ASSUME_TAC
              (DISJ_IMP (ONCE_REWRITE_RULE [DISJ_SYM] (IMP_ELIM th) )
              )
            )
) THEN

(PAT_ASSUM ``p /\ q ==> r``
           (fn th2 =>
           ASSUME_TAC
           (DISJ_IMP (ONCE_REWRITE_RULE [DISJ_SYM] (IMP_ELIM th2))))) THEN
REPEAT STRIP_TAC THEN
RES_TAC
)
```

7.3 Session Transcript

```
> > > # # # # # # # # # # # # # # # # # val problemTwothm =
    [...] |- p  q:
  thm
```

1

7.4 Explain Result

All tests from 10.4.2 have been passed in HOL

Chapter 8

Appendix A: source code for 9.5.1, 9.5.2, and 9.5.3