Project 11

Jinhao Wei

9 April 2019

Abstract

This report is basically a summary on my attempts on Project 11, which includes definitions propoties and theorems on several types of state machines. This report provides my solutions on *17.4.1* and *17.4.3*. In addition, I had pretty-printed the corresponding datatypes and proofs and put the reports in the corresponding folders */HOL/HOLReports/*.

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.

Contents

Chapter 1

Executive Summary

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.

For question *17.4.1* and *17.4.3*, we gave proofs on the following theorems or propoties:

[Alice_exec_npriv_justified_thm]

```
⊢ ∀NS Out M Oi Os.
    TR (M,Oi,Os) (exec (NP npriv))
      (CFG inputOK SMOStateInterp (certs cmd npriv privcmd)
         (Name Alice says prop (SOME (NP npriv))::ins) s outs)
      (CFG inputOK SMOStateInterp (certs cmd npriv privcmd) ins
         (NS s (exec (NP npriv)))
         (Out s (exec (NP npriv))::outs)) ⟺
    inputOK (Name Alice says prop (SOME (NP npriv))) ∧
    CFGInterpret (M,Oi,Os)
      (CFG inputOK SMOStateInterp (certs cmd npriv privcmd)
         (Name Alice says prop (SOME (NP npriv))::ins) s
         outs) ∧ (M,Oi,Os) sat prop (SOME (NP npriv))
```

[Alice_justified_npriv_exec_thm]

```
⊢ ∀NS Out M Oi Os cmd npriv privcmd ins s outs.
    inputOK (Name Alice says prop (SOME (NP npriv))) ∧
    CFGInterpret (M,Oi,Os)
      (CFG inputOK SMOStateInterp (certs cmd npriv privcmd)
         (Name Alice says prop (SOME (NP npriv))::ins) s
         outs) ⇒
    TR (M,Oi,Os) (exec (NP npriv))
      (CFG inputOK SMOStateInterp (certs cmd npriv privcmd)
         (Name Alice says prop (SOME (NP npriv))::ins) s outs)
      (CFG inputOK SMOStateInterp (certs cmd npriv privcmd) ins
         (NS s (exec (NP npriv)))
         (Out s (exec (NP npriv))::outs))
```

[Alice_npriv_lemma]

```
⊢ CFGInterpret (M,Oi,Os)
    (CFG inputOK SMOStateInterp (certs cmd npriv privcmd)
       (Name Alice says prop (SOME (NP npriv))::ins) s outs) ⇒
  (M,Oi,Os) sat prop (SOME (NP npriv))
```

[Alice_npriv_verified_thm]

```
⊢ ∀NS Out M Oi Os.
    TR (M,Oi,Os) (exec (NP npriv))
      (CFG inputOK SMOStateInterp (certs cmd npriv privcmd)
         (Name Alice says prop (SOME (NP npriv))::ins) s outs)
      (CFG inputOK SMOStateInterp (certs cmd npriv privcmd) ins
         (NS s (exec (NP npriv))))
```

```
        (Out s (exec (NP npriv))::outs)) ⇒
    (M,Oi,Os) sat prop (SOME (NP npriv))
```

[Carol_exec_npriv_justified_thm]

```
⊢ ∀NS Out M Oi Os.
    TR (M,Oi,Os) (exec (NP npriv))
      (CFG inputOK2 SMOStateInterp (certs2 cmd npriv privcmd)
         (Name Carol says prop (SOME (NP npriv))::ins) s outs)
      (CFG inputOK2 SMOStateInterp (certs2 cmd npriv privcmd)
         ins (NS s (exec (NP npriv)))
         (Out s (exec (NP npriv))::outs)) ⟺
    inputOK2 (Name Carol says prop (SOME (NP npriv))) ∧
    CFGInterpret (M,Oi,Os)
      (CFG inputOK2 SMOStateInterp (certs2 cmd npriv privcmd)
         (Name Carol says prop (SOME (NP npriv))::ins) s
         outs) ∧ (M,Oi,Os) sat prop (SOME (NP npriv))
```

[Carol_justified_npriv_exec_thm]

```
⊢ ∀NS Out M Oi Os cmd npriv privcmd ins s outs.
    inputOK2 (Name Carol says prop (SOME (NP npriv))) ∧
    CFGInterpret (M,Oi,Os)
      (CFG inputOK2 SMOStateInterp (certs2 cmd npriv privcmd)
         (Name Carol says prop (SOME (NP npriv))::ins) s
         outs) ⇒
    TR (M,Oi,Os) (exec (NP npriv))
      (CFG inputOK2 SMOStateInterp (certs2 cmd npriv privcmd)
         (Name Carol says prop (SOME (NP npriv))::ins) s outs)
      (CFG inputOK2 SMOStateInterp (certs2 cmd npriv privcmd)
         ins (NS s (exec (NP npriv)))
         (Out s (exec (NP npriv))::outs))
```

[Carol_justified_privcmd_trap_thm]

```
⊢ ∀NS Out M Oi Os cmd npriv privcmd ins s outs.
    inputOK2 (Name Carol says prop (SOME (PR privcmd))) ∧
    CFGInterpret (M,Oi,Os)
      (CFG inputOK2 SMOStateInterp (certs2 cmd npriv privcmd)
         (Name Carol says prop (SOME (PR privcmd))::ins) s
         outs) ⇒
    TR (M,Oi,Os) (trap (PR privcmd))
      (CFG inputOK2 SMOStateInterp (certs2 cmd npriv privcmd)
         (Name Carol says prop (SOME (PR privcmd))::ins) s
         outs)
      (CFG inputOK2 SMOStateInterp (certs2 cmd npriv privcmd)
         ins (NS s (trap (PR privcmd)))
         (Out s (trap (PR privcmd))::outs))
```

[Carol_npriv_lemma]

```
⊢ CFGInterpret (M,Oi,Os)
    (CFG inputOK2 SMOStateInterp (certs2 cmd npriv privcmd)
       (Name Carol says prop (SOME (NP npriv))::ins) s outs) ⇒
  (M,Oi,Os) sat prop (SOME (NP npriv))
```

[Carol_npriv_verified_thm]

```
⊢ ∀NS Out M Oi Os.
    TR (M,Oi,Os) (exec (NP npriv))
      (CFG inputOK2 SMOStateInterp (certs2 cmd npriv privcmd)
         (Name Carol says prop (SOME (NP npriv))::ins) s outs)
```

```
      (CFG inputOK2 SMOStateInterp (certs2 cmd npriv privcmd)
         ins (NS s (exec (NP npriv)))
         (Out s (exec (NP npriv))::outs)) ⇒
    (M,Oi,Os) sat prop (SOME (NP npriv))
```

[Carol_privcmd_trap_lemma]

```
⊢ CFGInterpret (M,Oi,Os)
    (CFG inputOK2 SMOStateInterp (certs2 cmd npriv privcmd)
       (Name Carol says prop (SOME (PR privcmd))::ins) s
       outs) ⇒
  (M,Oi,Os) sat prop NONE
```

[Carol_privcmd_trapped_thm]

```
⊢ ∀NS Out M Oi Os.
    TR (M,Oi,Os) (trap (PR privcmd))
      (CFG inputOK2 SMOStateInterp (certs2 cmd npriv privcmd)
         (Name Carol says prop (SOME (PR privcmd))::ins) s
         outs)
      (CFG inputOK2 SMOStateInterp (certs2 cmd npriv privcmd)
         ins (NS s (trap (PR privcmd)))
         (Out s (trap (PR privcmd))::outs)) ⇒
    (M,Oi,Os) sat prop NONE
```

[Carol_trap_privcmd_justified_thm]

```
⊢ ∀NS Out M Oi Os.
    TR (M,Oi,Os) (trap (PR privcmd))
      (CFG inputOK2 SMOStateInterp (certs2 cmd npriv privcmd)
         (Name Carol says prop (SOME (PR privcmd))::ins) s
         outs)
      (CFG inputOK2 SMOStateInterp (certs2 cmd npriv privcmd)
         ins (NS s (trap (PR privcmd)))
         (Out s (trap (PR privcmd))::outs)) ⟺
    inputOK2 (Name Carol says prop (SOME (PR privcmd))) ∧
    CFGInterpret (M,Oi,Os)
      (CFG inputOK2 SMOStateInterp (certs2 cmd npriv privcmd)
         (Name Carol says prop (SOME (PR privcmd))::ins) s
         outs) ∧ (M,Oi,Os) sat prop NONE
```

Reproducibility in ML and LaTeX

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

Chapter 2

Exercise 17.4.1

In this exercise, we will prove the following theorems

```
⊢ CFGInterpret (M,Oi,Os)
    (CFG inputOK SMOStateInterp (certs cmd npriv privcmd)
       (Name Alice says prop (SOME (NP npriv))::ins) s outs) ⇒
  (M,Oi,Os) sat prop (SOME (NP npriv))

⊢ ∀NS Out M Oi Os.
    TR (M,Oi,Os) (exec (NP npriv))
      (CFG inputOK SMOStateInterp (certs cmd npriv privcmd)
         (Name Alice says prop (SOME (NP npriv))::ins) s outs)
      (CFG inputOK SMOStateInterp (certs cmd npriv privcmd) ins
         (NS s (exec (NP npriv)))
         (Out s (exec (NP npriv))::outs)) ⟺
    inputOK (Name Alice says prop (SOME (NP npriv))) ∧
    CFGInterpret (M,Oi,Os)
      (CFG inputOK SMOStateInterp (certs cmd npriv privcmd)
         (Name Alice says prop (SOME (NP npriv))::ins) s
         outs) ∧ (M,Oi,Os) sat prop (SOME (NP npriv))

⊢ ∀NS Out M Oi Os.
    TR (M,Oi,Os) (exec (NP npriv))
      (CFG inputOK SMOStateInterp (certs cmd npriv privcmd)
         (Name Alice says prop (SOME (NP npriv))::ins) s outs)
      (CFG inputOK SMOStateInterp (certs cmd npriv privcmd) ins
         (NS s (exec (NP npriv)))
         (Out s (exec (NP npriv))::outs)) ⇒
    (M,Oi,Os) sat prop (SOME (NP npriv))

⊢ ∀NS Out M Oi Os cmd npriv privcmd ins s outs.
    inputOK (Name Alice says prop (SOME (NP npriv))) ∧
    CFGInterpret (M,Oi,Os)
      (CFG inputOK SMOStateInterp (certs cmd npriv privcmd)
         (Name Alice says prop (SOME (NP npriv))::ins) s
         outs) ⇒
    TR (M,Oi,Os) (exec (NP npriv))
      (CFG inputOK SMOStateInterp (certs cmd npriv privcmd)
         (Name Alice says prop (SOME (NP npriv))::ins) s outs)
      (CFG inputOK SMOStateInterp (certs cmd npriv privcmd) ins
         (NS s (exec (NP npriv)))
         (Out s (exec (NP npriv))::outs))
```

2.1 Exercise 17.4.1.A

In this section, we will prove the following theorems *Alice_npriv_lemma*

```
⊢ CFGInterpret (M,Oi,Os)
    (CFG inputOK SMOStateInterp (certs cmd npriv privcmd)
```

```
        (Name Alice says prop (SOME (NP npriv))::ins) s outs) ⇒
    (M,Oi,Os) sat prop (SOME (NP npriv))
```

2.1.1 Relevant Code

```
val Alice_npriv_lemma =
TAC_PROOF(
([], ``CFGInterpret ((M:(command inst,'b,staff,'d,'e)Kripke),Oi,Os)
  (CFG inputOK SM0StateInterp (certs cmd npriv privcmd)
   (((Name Alice) says (prop (SOME (NP (npriv:npriv)))))::ins)
   s (outs:output list)) ==>
  ((M,Oi,Os) sat (prop (SOME(NP npriv)))``),
  REWRITE_TAC[CFGInterpret_def, certs_def, SM0StateInterp_def, satList_CONS,
      satList_nil, sat_TT] THEN
  PROVE_TAC [Controls])
```

2.1.2 Session Transcript

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

1

```
val Alice_npriv_lemma =
   |- CFGInterpret
     ((M :(command inst, 'b, staff, 'd, 'e) Kripke),(Oi :'d po),
      (Os :'e po))
     (CFG (inputOK :(command inst, staff, 'd, 'e) Form -> bool)
        (SM0StateInterp :state -> (command inst, staff, 'd, 'e) Form)
        (certs (cmd :command) (npriv :npriv) (privcmd :privcmd) :
           (command inst, staff, 'd, 'e) Form list)
        (Name Alice says
         (prop (SOME (NP npriv) :command inst) :
            (command inst, staff, 'd, 'e) Form)::
             (ins :(command inst, staff, 'd, 'e) Form list)) (s :state)
        (outs :output list)) ==>
   (M,Oi,Os) sat
   (prop (SOME (NP npriv) :command inst) :
      (command inst, staff, 'd, 'e) Form):
   thm
```

2.2 Exercise 17.4.1.B

In this section, we will prove the following theorems *Alice_exec_npriv_justified_thm*

```
⊢ ∀NS Out M Oi Os.
    TR (M,Oi,Os) (exec (NP npriv))
      (CFG inputOK SM0StateInterp (certs cmd npriv privcmd)
         (Name Alice says prop (SOME (NP npriv))::ins) s outs)
      (CFG inputOK SM0StateInterp (certs cmd npriv privcmd) ins
         (NS s (exec (NP npriv)))
         (Out s (exec (NP npriv))::outs)) ⟺
    inputOK (Name Alice says prop (SOME (NP npriv))) ∧
    CFGInterpret (M,Oi,Os)
      (CFG inputOK SM0StateInterp (certs cmd npriv privcmd)
         (Name Alice says prop (SOME (NP npriv))::ins) s
         outs) ∧ (M,Oi,Os) sat prop (SOME (NP npriv))
```

2.2.1 Relevant Code

```
val Alice_exec_npriv_justified_thm
=
let
val th1 =
    ISPECL
    [``inputOK:(command inst, staff, 'd, 'e)Form->bool``,
    ``(certs cmd npriv privcmd):(command inst, staff, 'd, 'e)Form list``,
    ``SM0StateInterp: state->(command inst, staff, 'd, 'e)Form``,
    ``Name Alice``,``NP npriv``, ``ins:(command inst, staff, 'd, 'e)Form list
       ``,
    ``s:state``,``outs:output list``]
    TR_exec_cmd_rule
in
 TAC_PROOF(([],
    ``!(NS :state -> command trType -> state)
         (Out :state -> command trType -> output)
         (M :(command inst, 'b, staff, 'd, 'e) Kripke) (Oi :'d po)
         (Os :'e po).
       TR (M,Oi,Os) (exec (NP (npriv :npriv)))
         (CFG (inputOK :(command inst, staff, 'd, 'e) Form -> bool)
            (SM0StateInterp :state -> (command inst, staff, 'd, 'e) Form)
            (certs (cmd :command) (npriv :npriv) privcmd :
               (command inst, staff, 'd, 'e) Form list)
            (Name Alice says
             (prop (SOME (NP npriv) :command inst) :
                (command inst, staff, 'd, 'e) Form)::
                 (ins :(command inst, staff, 'd, 'e) Form list)) (s :state)
            (outs :output list))
         (CFG (inputOK :(command inst, staff, 'd, 'e) Form -> bool)
            (SM0StateInterp :state -> (command inst, staff, 'd, 'e) Form)
            (certs cmd npriv privcmd :
               (command inst, staff, 'd, 'e) Form list) ins
            (NS s (exec (NP npriv)))
            (Out s (exec (NP npriv))::outs)) <=>
       inputOK
         (Name Alice says
          (prop (SOME (NP npriv) :command inst) :
             (command inst, staff, 'd, 'e) Form)) /\
       CFGInterpret (M,Oi,Os)
         (CFG (inputOK :(command inst, staff, 'd, 'e) Form -> bool)
            (SM0StateInterp :state -> (command inst, staff, 'd, 'e) Form)
            (certs cmd npriv privcmd :
               (command inst, staff, 'd, 'e) Form list)
            (Name Alice says
             (prop (SOME (NP npriv) :command inst) :
                (command inst, staff, 'd, 'e) Form)::ins) s outs) /\
       (M,Oi,Os) sat
       (prop (SOME (NP npriv) :command inst) :
          (command inst, staff, 'd, 'e) Form)``),
       PROVE_TAC[th1, Alice_npriv_lemma])
end
```

2.2.2 Session Transcript

The code above will give us the transcript as below:

2

```
val Alice_exec_npriv_justified_thm =
   |- !(NS :state -> command trType -> state)
      (Out :state -> command trType -> output)
      (M :(command inst, 'b, staff, 'd, 'e) Kripke) (Oi :'d po)
      (Os :'e po).
     TR (M,Oi,Os) (exec (NP (npriv :npriv)))
       (CFG (inputOK :(command inst, staff, 'd, 'e) Form -> bool)
          (SMOStateInterp :state -> (command inst, staff, 'd, 'e) Form)
          (certs (cmd :command) npriv (privcmd :privcmd) :
             (command inst, staff, 'd, 'e) Form list)
          (Name Alice says
           (prop (SOME (NP npriv) :command inst) :
              (command inst, staff, 'd, 'e) Form)::
            (ins :(command inst, staff, 'd, 'e) Form list))
          (s :state) (outs :output list))
       (CFG (inputOK :(command inst, staff, 'd, 'e) Form -> bool)
          (SMOStateInterp :state -> (command inst, staff, 'd, 'e) Form)
          (certs cmd npriv privcmd :
             (command inst, staff, 'd, 'e) Form list) ins
          (NS s (exec (NP npriv))) (Out s (exec (NP npriv))::outs)) <=>
     inputOK
       (Name Alice says
        (prop (SOME (NP npriv) :command inst) :
           (command inst, staff, 'd, 'e) Form)) /\
     CFGInterpret (M,Oi,Os)
       (CFG (inputOK :(command inst, staff, 'd, 'e) Form -> bool)
          (SMOStateInterp :state -> (command inst, staff, 'd, 'e) Form)
          (certs cmd npriv privcmd :
             (command inst, staff, 'd, 'e) Form list)
          (Name Alice says
           (prop (SOME (NP npriv) :command inst) :
              (command inst, staff, 'd, 'e) Form)::ins) s outs) /\
     (M,Oi,Os) sat
     (prop (SOME (NP npriv) :command inst) :
        (command inst, staff, 'd, 'e) Form):
   thm
val it = (): unit
```

2.3 Exercise 17.4.1.C

In this section, we will prove the following theorems *Alice_npriv_verified_thm*

$$\vdash \forall NS\ Out\ M\ Oi\ Os.$$
$$\texttt{TR}\ (M, Oi, Os)\ (\texttt{exec}\ (\texttt{NP}\ npriv))$$
$$(\texttt{CFG inputOK SMOStateInterp (certs}\ cmd\ npriv\ privcmd)$$
$$(\texttt{Name Alice says prop (SOME (NP}\ npriv))::ins)\ s\ outs)$$
$$(\texttt{CFG inputOK SMOStateInterp (certs}\ cmd\ npriv\ privcmd)\ ins$$
$$(NS\ s\ (\texttt{exec (NP}\ npriv)))$$
$$(Out\ s\ (\texttt{exec (NP}\ npriv))::outs)) \Rightarrow$$
$$(M, Oi, Os)\ \texttt{sat prop (SOME (NP}\ npriv))$$

2.3.1 Relevant Code

```
val Alice_npriv_verified_thm =
TAC_PROOF(([],
      ``!(NS :state -> command trType -> state)
        (Out :state -> command trType -> output)
        (M :(command inst, 'b, staff, 'd, 'e) Kripke) (Oi :'d po)
        (Os :'e po).
       TR (M,Oi,Os) (exec (NP (npriv :npriv)))
         (CFG (inputOK :(command inst, staff, 'd, 'e) Form -> bool)
            (SM0StateInterp :state -> (command inst, staff, 'd, 'e) Form)
            (certs (cmd :command) (npriv :npriv) privcmd :
```

```
            (command inst, staff, 'd, 'e) Form list)
         (Name Alice says
          (prop (SOME (NP npriv) :command inst) :
             (command inst, staff, 'd, 'e) Form)::
              (ins :(command inst, staff, 'd, 'e) Form list)) (s :state)
         (outs :output list))
      (CFG (inputOK :(command inst, staff, 'd, 'e) Form -> bool)
         (SM0StateInterp :state -> (command inst, staff, 'd, 'e) Form)
         (certs cmd npriv privcmd :
            (command inst, staff, 'd, 'e) Form list) ins
         (NS s (exec (NP npriv)))
         (Out s (exec (NP npriv))::outs)) ==>
    (M,Oi,Os) sat
    (prop (SOME (NP npriv) :command inst) :
       (command inst, staff, 'd, 'e) Form)``),
       PROVE_TAC[Alice_exec_npriv_justified_thm])
```

2.3.2 Session Transcript

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

3

```
val Alice_npriv_verified_thm =
   |- !(NS :state -> command trType -> state)
      (Out :state -> command trType -> output)
      (M :(command inst, 'b, staff, 'd, 'e) Kripke) (Oi :'d po)
      (Os :'e po).
     TR (M,Oi,Os) (exec (NP (npriv :npriv)))
       (CFG (inputOK :(command inst, staff, 'd, 'e) Form -> bool)
          (SMOStateInterp :state -> (command inst, staff, 'd, 'e) Form)
          (certs (cmd :command) npriv (privcmd :privcmd) :
             (command inst, staff, 'd, 'e) Form list)
          (Name Alice says
           (prop (SOME (NP npriv) :command inst) :
              (command inst, staff, 'd, 'e) Form)::
               (ins :(command inst, staff, 'd, 'e) Form list))
          (s :state) (outs :output list))
       (CFG (inputOK :(command inst, staff, 'd, 'e) Form -> bool)
          (SMOStateInterp :state -> (command inst, staff, 'd, 'e) Form)
          (certs cmd npriv privcmd :
             (command inst, staff, 'd, 'e) Form list) ins
          (NS s (exec (NP npriv))) (Out s (exec (NP npriv))::outs)) ==>
     (M,Oi,Os) sat
     (prop (SOME (NP npriv) :command inst) :
        (command inst, staff, 'd, 'e) Form):
   thm
```

2.4 Exercise 17.4.1.D

In this section, we will prove the following theorems *Alice_justified_npriv_exec_thm*

$$
\begin{aligned}
\vdash\ & \forall NS\ Out\ M\ Oi\ Os\ cmd\ npriv\ privcmd\ ins\ s\ outs.\\
& \texttt{inputOK (Name Alice says prop (SOME (NP } npriv\texttt{)))} \wedge\\
& \texttt{CFGInterpret (}M\texttt{,}Oi\texttt{,}Os\texttt{)}\\
& \quad \texttt{(CFG inputOK SMOStateInterp (certs } cmd\ npriv\ privcmd\texttt{)}\\
& \quad\quad \texttt{(Name Alice says prop (SOME (NP } npriv\texttt{))::}ins\texttt{)}\ s\\
& \quad\quad outs\texttt{)} \Rightarrow\\
& \texttt{TR (}M\texttt{,}Oi\texttt{,}Os\texttt{) (exec (NP } npriv\texttt{))}\\
& \quad \texttt{(CFG inputOK SMOStateInterp (certs } cmd\ npriv\ privcmd\texttt{)}\\
& \quad\quad \texttt{(Name Alice says prop (SOME (NP } npriv\texttt{))::}ins\texttt{)}\ s\ outs\texttt{)}\\
& \quad \texttt{(CFG inputOK SMOStateInterp (certs } cmd\ npriv\ privcmd\texttt{)}\ ins\\
& \quad\quad \texttt{(}NS\ s\ \texttt{(exec (NP } npriv\texttt{)))}\\
& \quad\quad \texttt{(}Out\ s\ \texttt{(exec (NP } npriv\texttt{))::}outs\texttt{))}
\end{aligned}
$$

2.4.1 Relevant Code

```
val Alice_justified_npriv_exec_thm =
TAC_PROOF(([],
``!(NS :state -> command trType -> state)
         (Out :state -> command trType -> output)
         (M :(command inst, 'b, staff, 'd, 'e) Kripke) (Oi :'d po)
         (Os :'e po) cmd npriv privcmd ins s outs.
         inputOK
          (Name Alice says
           (prop (SOME (NP npriv) :command inst) :
              (command inst, staff, 'd, 'e) Form)) /\
          CFGInterpret (M,Oi,Os)
          (CFG (inputOK :(command inst, staff, 'd, 'e) Form -> bool)
             (SM0StateInterp :state -> (command inst, staff, 'd, 'e) Form)
             (certs cmd npriv privcmd :
                (command inst, staff, 'd, 'e) Form list)
             (Name Alice says
              (prop (SOME (NP npriv) :command inst) :
                 (command inst, staff, 'd, 'e) Form)::ins) s outs) ==>
        TR (M,Oi,Os) (exec (NP (npriv :npriv)))
          (CFG (inputOK :(command inst, staff, 'd, 'e) Form -> bool)
             (SM0StateInterp :state -> (command inst, staff, 'd, 'e) Form)
             (certs (cmd :command) (npriv :npriv) privcmd :
                (command inst, staff, 'd, 'e) Form list)
             (Name Alice says
              (prop (SOME (NP npriv) :command inst) :
                 (command inst, staff, 'd, 'e) Form)::
                  (ins :(command inst, staff, 'd, 'e) Form list)) (s :state)
             (outs :output list))
          (CFG (inputOK :(command inst, staff, 'd, 'e) Form -> bool)
             (SM0StateInterp :state -> (command inst, staff, 'd, 'e) Form)
             (certs cmd npriv privcmd :
                (command inst, staff, 'd, 'e) Form list) ins
             (NS s (exec (NP npriv)))
             (Out s (exec (NP npriv))::outs))``),
PROVE_TAC[Alice_exec_npriv_justified_thm, inputOK_def, Alice_npriv_lemma])
```

2.4.2 Session Transcript

The code above will let us see the transcript as below:

4

```
val Alice_justified_npriv_exec_thm =
   |- !(NS :state -> command trType -> state)
      (Out :state -> command trType -> output)
      (M :(command inst, 'b, staff, 'd, 'e) Kripke) (Oi :'d po)
      (Os :'e po) (cmd :command) (npriv :npriv) (privcmd :privcmd)
      (ins :(command inst, staff, 'd, 'e) Form list) (s :state)
      (outs :output list).
     inputOK
       (Name Alice says
        (prop (SOME (NP npriv) :command inst) :
           (command inst, staff, 'd, 'e) Form)) /\
     CFGInterpret (M,Oi,Os)
       (CFG (inputOK :(command inst, staff, 'd, 'e) Form -> bool)
          (SMOStateInterp :state -> (command inst, staff, 'd, 'e) Form)
          (certs cmd npriv privcmd :
             (command inst, staff, 'd, 'e) Form list)
          (Name Alice says
           (prop (SOME (NP npriv) :command inst) :
              (command inst, staff, 'd, 'e) Form)::ins) s outs) ==>
     TR (M,Oi,Os) (exec (NP npriv))
       (CFG (inputOK :(command inst, staff, 'd, 'e) Form -> bool)
          (SMOStateInterp :state -> (command inst, staff, 'd, 'e) Form)
          (certs cmd npriv privcmd :
             (command inst, staff, 'd, 'e) Form list)
          (Name Alice says
           (prop (SOME (NP npriv) :command inst) :
              (command inst, staff, 'd, 'e) Form)::ins) s outs)
       (CFG (inputOK :(command inst, staff, 'd, 'e) Form -> bool)
          (SMOStateInterp :state -> (command inst, staff, 'd, 'e) Form)
          (certs cmd npriv privcmd :
             (command inst, staff, 'd, 'e) Form list) ins
          (NS s (exec (NP npriv))) (Out s (exec (NP npriv))::outs)):
   thm
```

Chapter 3

Exercise 17.4.3

In this exercise, we will prove the following theorems

Carol_npriv_lemma

```
⊢ CFGInterpret (M,Oi,Os)
    (CFG inputOK2 SMOStateInterp (certs2 cmd npriv privcmd)
       (Name Carol says prop (SOME (NP npriv))::ins) s outs) ⇒
  (M,Oi,Os) sat prop (SOME (NP npriv))
```

Carol_exec_npriv_justified_thm

```
⊢ ∀NS Out M Oi Os.
    TR (M,Oi,Os) (exec (NP npriv))
      (CFG inputOK2 SMOStateInterp (certs2 cmd npriv privcmd)
         (Name Carol says prop (SOME (NP npriv))::ins) s outs)
      (CFG inputOK2 SMOStateInterp (certs2 cmd npriv privcmd)
         ins (NS s (exec (NP npriv)))
         (Out s (exec (NP npriv))::outs)) ⟺
    inputOK2 (Name Carol says prop (SOME (NP npriv))) ∧
    CFGInterpret (M,Oi,Os)
      (CFG inputOK2 SMOStateInterp (certs2 cmd npriv privcmd)
         (Name Carol says prop (SOME (NP npriv))::ins) s
         outs) ∧ (M,Oi,Os) sat prop (SOME (NP npriv))
```

Carol_npriv_verified_thm

```
⊢ ∀NS Out M Oi Os.
    TR (M,Oi,Os) (exec (NP npriv))
      (CFG inputOK2 SMOStateInterp (certs2 cmd npriv privcmd)
         (Name Carol says prop (SOME (NP npriv))::ins) s outs)
      (CFG inputOK2 SMOStateInterp (certs2 cmd npriv privcmd)
         ins (NS s (exec (NP npriv)))
         (Out s (exec (NP npriv))::outs)) ⇒
    (M,Oi,Os) sat prop (SOME (NP npriv))
```

Carol_justified_npriv_exec_thm

```
⊢ ∀NS Out M Oi Os cmd npriv privcmd ins s outs.
    inputOK2 (Name Carol says prop (SOME (NP npriv))) ∧
    CFGInterpret (M,Oi,Os)
      (CFG inputOK2 SMOStateInterp (certs2 cmd npriv privcmd)
         (Name Carol says prop (SOME (NP npriv))::ins) s
         outs) ⇒
    TR (M,Oi,Os) (exec (NP npriv))
      (CFG inputOK2 SMOStateInterp (certs2 cmd npriv privcmd)
         (Name Carol says prop (SOME (NP npriv))::ins) s outs)
      (CFG inputOK2 SMOStateInterp (certs2 cmd npriv privcmd)
         ins (NS s (exec (NP npriv)))
         (Out s (exec (NP npriv))::outs))
```

Carol_privcmd_trap_lemma

```
⊢ CFGInterpret (M,Oi,Os)
    (CFG inputOK2 SMOStateInterp (certs2 cmd npriv privcmd)
       (Name Carol says prop (SOME (PR privcmd))::ins) s
       outs) ⇒
  (M,Oi,Os) sat prop NONE
```

Carol_justified_privcmd_trap_thm

```
⊢ ∀NS Out M Oi Os cmd npriv privcmd ins s outs.
    inputOK2 (Name Carol says prop (SOME (PR privcmd))) ∧
    CFGInterpret (M,Oi,Os)
      (CFG inputOK2 SMOStateInterp (certs2 cmd npriv privcmd)
         (Name Carol says prop (SOME (PR privcmd))::ins) s
         outs) ⇒
    TR (M,Oi,Os) (trap (PR privcmd))
      (CFG inputOK2 SMOStateInterp (certs2 cmd npriv privcmd)
         (Name Carol says prop (SOME (PR privcmd))::ins) s
         outs)
      (CFG inputOK2 SMOStateInterp (certs2 cmd npriv privcmd)
         ins (NS s (trap (PR privcmd)))
         (Out s (trap (PR privcmd))::outs))
```

Carol_npriv_verified_thm

```
⊢ ∀NS Out M Oi Os.
    TR (M,Oi,Os) (exec (NP npriv))
      (CFG inputOK2 SMOStateInterp (certs2 cmd npriv privcmd)
         (Name Carol says prop (SOME (NP npriv))::ins) s outs)
      (CFG inputOK2 SMOStateInterp (certs2 cmd npriv privcmd)
         ins (NS s (exec (NP npriv)))
         (Out s (exec (NP npriv))::outs)) ⇒
    (M,Oi,Os) sat prop (SOME (NP npriv))
```

Carol_privcmd_trapped_thm

```
⊢ ∀NS Out M Oi Os.
    TR (M,Oi,Os) (trap (PR privcmd))
      (CFG inputOK2 SMOStateInterp (certs2 cmd npriv privcmd)
         (Name Carol says prop (SOME (PR privcmd))::ins) s
         outs)
      (CFG inputOK2 SMOStateInterp (certs2 cmd npriv privcmd)
         ins (NS s (trap (PR privcmd)))
         (Out s (trap (PR privcmd))::outs)) ⇒
    (M,Oi,Os) sat prop NONE
```

3.1 Exercise 17.4.3.A

In this section, we will provide the following definitions *inputOK2_def* and *certs2_def*. Please kindly note that I had put the definitions in file *SM0Scripts.sml*

3.1.1 Relevant Code

We will use the follwing code to define *inputOK2_def*

```
val inputOK2_def =
Define
`(inputOK2
  (((Name Carol) says
```

```
   (prop (SOME (cmd:command)))):(command inst,staff,'d,'e)Form) = T) /\
 (inputOK2 _ = F)‘


val certs2_def =
Define
‘certs2 (cmd:command)(npriv:npriv)(privcmd:privcmd) =
 [(Name Carol controls ((prop (SOME (NP npriv))):(command inst, staff,'d,'e)
    Form));
  ((Name Carol) says (prop (SOME (PR privcmd)))) impf (prop NONE)]‘
```

3.1.2 Session Transcript

If we send the code snippet that gives the definitions to HOL session, we will see transcript as below:

5

```
val inputOK2_def =
   |- v98 v97 v96 v95 v94 v93 v92 v91 v90 v9 v89 v88 v87 v86 v85 v84 v83
      v82 v81 v80 v8 v79 v78 v77 v76 v75 v74 v73 v72 v71 v70 v7 v69 v68
      v67 v66 v6 v5 v4 v32 v31 v30 v3 v29 v28 v27 v26 v25 v24 v23 v22
      v21 v20 v2 v19 v18 v17 v16 v15 v142 v14 v136 v135 v134 v133 v132
      v13 v12 v10 v1 v cmd.
     (inputOK2 (Name Carol says prop (SOME cmd))  T)
     (inputOK2 TT  F)  (inputOK2 FF  F)  (inputOK2 (prop v)  F)
     (inputOK2 (notf v1)  F)  (inputOK2 (v2 andf v3)  F)
     (inputOK2 (v4 orf v5)  F)  (inputOK2 (v6 impf v7)  F)
     (inputOK2 (v8 eqf v9)  F)  (inputOK2 (v10 says TT)  F)
     (inputOK2 (v10 says FF)  F)
     (inputOK2 (Name Alice says prop (SOME v142))  F)
     (inputOK2 (Name Bob says prop (SOME v142))  F)
     (inputOK2 (Name v132 says prop NONE)  F)
     (inputOK2 (v133 meet v134 says prop v66)  F)
     (inputOK2 (v135 quoting v136 says prop v66)  F)
     (inputOK2 (v10 says notf v67)  F)
     (inputOK2 (v10 says (v68 andf v69))  F)
     (inputOK2 (v10 says (v70 orf v71))  F)
     (inputOK2 (v10 says (v72 impf v73))  F)
     (inputOK2 (v10 says (v74 eqf v75))  F)
     (inputOK2 (v10 says v76 says v77)  F)
     (inputOK2 (v10 says v78 speaks_for v79)  F)
     (inputOK2 (v10 says v80 controls v81)  F)
     (inputOK2 (v10 says reps v82 v83 v84)  F)
     (inputOK2 (v10 says v85 domi v86)  F)
     (inputOK2 (v10 says v87 eqi v88)  F)
     (inputOK2 (v10 says v89 doms v90)  F)
     (inputOK2 (v10 says v91 eqs v92)  F)
     (inputOK2 (v10 says v93 eqn v94)  F)
     (inputOK2 (v10 says v95 lte v96)  F)
     (inputOK2 (v10 says v97 lt v98)  F)
     (inputOK2 (v12 speaks_for v13)  F)
     (inputOK2 (v14 controls v15)  F)
     (inputOK2 (reps v16 v17 v18)  F)  (inputOK2 (v19 domi v20)  F)
     (inputOK2 (v21 eqi v22)  F)  (inputOK2 (v23 doms v24)  F)
     (inputOK2 (v25 eqs v26)  F)  (inputOK2 (v27 eqn v28)  F)
     (inputOK2 (v29 lte v30)  F)  (inputOK2 (v31 lt v32)  F):
   thm


val certs2_def =
   |- cmd npriv privcmd.
     certs2 cmd npriv privcmd =
     [Name Carol controls prop (SOME (NP npriv));
      Name Carol says prop (SOME (PR privcmd)) impf prop NONE]:
   thm
```

and the fine-printed definitions should look like this, please check it at file */HOLReports/SM0Report.pdf*:

inputOK2_def

$$\vdash \texttt{(inputOK2 (Name Carol says prop (SOME } cmd\texttt{))} \iff \texttt{T)} \wedge \texttt{(inputOK2 TT} \iff \texttt{F)} \wedge \texttt{(inputOK2 FF} \iff \texttt{F)} \wedge$$

```
(inputOK2 (prop v) ⟺ F) ∧ (inputOK2 (notf v1) ⟺ F) ∧
(inputOK2 (v2 andf v3) ⟺ F) ∧ (inputOK2 (v4 orf v5) ⟺ F) ∧
(inputOK2 (v6 impf v7) ⟺ F) ∧ (inputOK2 (v8 eqf v9) ⟺ F) ∧
(inputOK2 (v10 says TT) ⟺ F) ∧
(inputOK2 (v10 says FF) ⟺ F) ∧
(inputOK2 (Name Alice says prop (SOME v142)) ⟺ F) ∧
(inputOK2 (Name Bob says prop (SOME v142)) ⟺ F) ∧
(inputOK2 (Name v132 says prop NONE) ⟺ F) ∧
(inputOK2 (v133 meet v134 says prop v66) ⟺ F) ∧
(inputOK2 (v135 quoting v136 says prop v66) ⟺ F) ∧
(inputOK2 (v10 says notf v67) ⟺ F) ∧
(inputOK2 (v10 says (v68 andf v69)) ⟺ F) ∧
(inputOK2 (v10 says (v70 orf v71)) ⟺ F) ∧
(inputOK2 (v10 says (v72 impf v73)) ⟺ F) ∧
(inputOK2 (v10 says (v74 eqf v75)) ⟺ F) ∧
(inputOK2 (v10 says v76 says v77) ⟺ F) ∧
(inputOK2 (v10 says v78 speaks_for v79) ⟺ F) ∧
(inputOK2 (v10 says v80 controls v81) ⟺ F) ∧
(inputOK2 (v10 says reps v82 v83 v84) ⟺ F) ∧
(inputOK2 (v10 says v85 domi v86) ⟺ F) ∧
(inputOK2 (v10 says v87 eqi v88) ⟺ F) ∧
(inputOK2 (v10 says v89 doms v90) ⟺ F) ∧
(inputOK2 (v10 says v91 eqs v92) ⟺ F) ∧
(inputOK2 (v10 says v93 eqn v94) ⟺ F) ∧
(inputOK2 (v10 says v95 lte v96) ⟺ F) ∧
(inputOK2 (v10 says v97 lt v98) ⟺ F) ∧
(inputOK2 (v12 speaks_for v13) ⟺ F) ∧
(inputOK2 (v14 controls v15) ⟺ F) ∧
(inputOK2 (reps v16 v17 v18) ⟺ F) ∧
(inputOK2 (v19 domi v20) ⟺ F) ∧
(inputOK2 (v21 eqi v22) ⟺ F) ∧
(inputOK2 (v23 doms v24) ⟺ F) ∧
(inputOK2 (v25 eqs v26) ⟺ F) ∧
(inputOK2 (v27 eqn v28) ⟺ F) ∧
(inputOK2 (v29 lte v30) ⟺ F) ∧ (inputOK2 (v31 lt v32) ⟺ F)
```

certs2_def

```
⊢ ∀ cmd npriv privcmd.
    certs2 cmd npriv privcmd =
    [Name Carol controls prop (SOME (NP npriv));
     Name Carol says prop (SOME (PR privcmd)) impf prop NONE]
```

3.2 Exercise 17.4.3.B

In this section, we will prove the following theorems *Carol_npriv_lemma*

```
⊢ CFGInterpret (M,Oi,Os)
    (CFG inputOK2 SMOStateInterp (certs2 cmd npriv privcmd)
       (Name Carol says prop (SOME (NP npriv))::ins) s outs) ⇒
  (M,Oi,Os) sat prop (SOME (NP npriv))
```

3.2.1 Relevant Code

For theorem *Carol_npriv_lemma*, we will use the following code to prove it:

```
val Carol_npriv_lemma
=
```

```
TAC_PROOF(
([],
``CFGInterpret ((M:(command inst,'b,staff,'d,'e)Kripke),Oi,Os)
  (CFG inputOK2 SM0StateInterp (certs2 cmd npriv privcmd)
   (((Name Carol) says (prop (SOME (NP (npriv:npriv)))))::ins)
   s (outs:output list)) ==>
  ((M,Oi,Os) sat (prop (SOME(NP npriv))))``),
  REWRITE_TAC [CFGInterpret_def, certs2_def, SM0StateInterp_def, satList_CONS,
      satList_nil, sat_TT] THEN
  PROVE_TAC[Controls])
```

For theorem *Carol_exec_npriv_justified_thm*, we will use the following code:

```
val Carol_exec_npriv_justified_thm
=
    let val th1 =
    ISPECL
    [``inputOK2:(command inst, staff, 'd, 'e)Form -> bool``,
    ``(certs2 cmd npriv privcmd):(command inst, staff, 'd, 'e)Form list``,
    ``SM0StateIntern: state->(command inst, staff, 'd, 'e)Form``,
    ``Name Carol``, ``NP npriv``, ``ins:(command inst, staff, 'd, 'e)Form list
       ``,
    ``s: state``, ``outs:output list``]
    TR_exec_cmd_rule;
in
TAC_PROOF(
([],
      ``!(NS :state -> command trType -> state)
        (Out :state -> command trType -> output)
        (M :(command inst, 'b, staff, 'd, 'e) Kripke) (Oi :'d po)
        (Os :'e po).
       TR (M,Oi,Os) (exec (NP (npriv :npriv)))
         (CFG (inputOK2 :(command inst, staff, 'd, 'e) Form -> bool)
            (SM0StateInterp :state -> (command inst, staff, 'd, 'e) Form)
            (certs2 (cmd :command) (npriv :npriv) privcmd :
               (command inst, staff, 'd, 'e) Form list)
            (Name Carol says
             (prop (SOME (NP npriv) :command inst) :
                (command inst, staff, 'd, 'e) Form)::
                 (ins :(command inst, staff, 'd, 'e) Form list)) (s :state)
            (outs :output list))
         (CFG (inputOK2 :(command inst, staff, 'd, 'e) Form -> bool)
            (SM0StateInterp :state -> (command inst, staff, 'd, 'e) Form)
            (certs2 cmd npriv privcmd :
               (command inst, staff, 'd, 'e) Form list) ins
            (NS s (exec (NP npriv)))
            (Out s (exec (NP npriv))::outs)) <=>
       inputOK2
         (Name Carol says
          (prop (SOME (NP npriv) :command inst) :
             (command inst, staff, 'd, 'e) Form)) /\
       CFGInterpret (M,Oi,Os)
         (CFG (inputOK2 :(command inst, staff, 'd, 'e) Form -> bool)
            (SM0StateInterp :state -> (command inst, staff, 'd, 'e) Form)
```

```
            (certs2 cmd npriv privcmd :
               (command inst, staff, 'd, 'e) Form list)
            (Name Carol says
             (prop (SOME (NP npriv) :command inst) :
                (command inst, staff, 'd, 'e) Form)::ins) s outs) /\
        (M,Oi,Os) sat
        (prop (SOME (NP npriv) :command inst) :
           (command inst, staff, 'd, 'e) Form)``),
          PROVE_TAC[th1, Carol_npriv_lemma])
end
```

For theorem *Carol_npriv_verified_thm*, we will use the following code to prove it:

```
val Carol_npriv_verified_thm
= TAC_PROOF(
([],
      ``!(NS :state -> command trType -> state)
        (Out :state -> command trType -> output)
        (M :(command inst, 'b, staff, 'd, 'e) Kripke) (Oi :'d po)
        (Os :'e po).
       TR (M,Oi,Os) (exec (NP (npriv :npriv)))
         (CFG (inputOK2 :(command inst, staff, 'd, 'e) Form -> bool)
            (SM0StateInterp :state -> (command inst, staff, 'd, 'e) Form)
            (certs2 (cmd :command) (npriv :npriv) privcmd :
               (command inst, staff, 'd, 'e) Form list)
            (Name Carol says
             (prop (SOME (NP npriv) :command inst) :
                (command inst, staff, 'd, 'e) Form)::
                 (ins :(command inst, staff, 'd, 'e) Form list)) (s :state)
            (outs :output list))
         (CFG (inputOK2 :(command inst, staff, 'd, 'e) Form -> bool)
            (SM0StateInterp :state -> (command inst, staff, 'd, 'e) Form)
            (certs2 cmd npriv privcmd :
               (command inst, staff, 'd, 'e) Form list) ins
            (NS s (exec (NP npriv)))
            (Out s (exec (NP npriv))::outs)) ==>
        (M,Oi,Os) sat
        (prop (SOME (NP npriv) :command inst) :
           (command inst, staff, 'd, 'e) Form)``),
          PROVE_TAC [Carol_exec_npriv_justified_thm])
```

For theorem *Carol_justified_npriv_exec_thm*, we will use the following code to prove it:

```
val Carol_justified_npriv_exec_thm =
TAC_PROOF(
([],
      ``!(NS :state -> command trType -> state)
        (Out :state -> command trType -> output)
        (M :(command inst, 'b, staff, 'd, 'e) Kripke) (Oi :'d po)
        (Os :'e po) cmd npriv privcmd ins s outs.
        inputOK2
         (Name Carol says
          (prop (SOME (NP npriv) :command inst) :
             (command inst, staff, 'd, 'e) Form)) /\
```

```
          CFGInterpret (M,Oi,Os)
          (CFG (inputOK2 :(command inst, staff, 'd, 'e) Form -> bool)
             (SM0StateInterp :state -> (command inst, staff, 'd, 'e) Form)
             (certs2 cmd npriv privcmd :
                (command inst, staff, 'd, 'e) Form list)
             (Name Carol says
              (prop (SOME (NP npriv) :command inst) :
                 (command inst, staff, 'd, 'e) Form)::ins) s outs) ==>
        TR (M,Oi,Os) (exec (NP (npriv :npriv)))
          (CFG (inputOK2 :(command inst, staff, 'd, 'e) Form -> bool)
             (SM0StateInterp :state -> (command inst, staff, 'd, 'e) Form)
             (certs2 (cmd :command) (npriv :npriv) privcmd :
                (command inst, staff, 'd, 'e) Form list)
             (Name Carol says
              (prop (SOME (NP npriv) :command inst) :
                 (command inst, staff, 'd, 'e) Form)::
                  (ins :(command inst, staff, 'd, 'e) Form list)) (s :state)
             (outs :output list))
          (CFG (inputOK2 :(command inst, staff, 'd, 'e) Form -> bool)
             (SM0StateInterp :state -> (command inst, staff, 'd, 'e) Form)
             (certs2 cmd npriv privcmd :
                (command inst, staff, 'd, 'e) Form list) ins
             (NS s (exec (NP npriv)))
             (Out s (exec (NP npriv))::outs))``),
PROVE_TAC[Carol_exec_npriv_justified_thm, inputOK2_def, Carol_npriv_lemma])
```

3.2.2 Session Transcript

If we send the code snippets above to HOL session, we will see the following transcripts, respectively:

For *Carol_npriv_lemma:*

6

```
val Carol_npriv_lemma =
   |- CFGInterpret
     ((M :(command inst, 'b, staff, 'd, 'e) Kripke),(Oi :'d po),
      (Os :'e po))
     (CFG (inputOK2 :(command inst, staff, 'd, 'e) Form -> bool)
        (SMOStateInterp :state -> (command inst, staff, 'd, 'e) Form)
        (certs2 (cmd :command) (npriv :npriv) (privcmd :privcmd) :
           (command inst, staff, 'd, 'e) Form list)
        (Name Carol says
         (prop (SOME (NP npriv) :command inst) :
            (command inst, staff, 'd, 'e) Form)::
             (ins :(command inst, staff, 'd, 'e) Form list)) (s :state)
        (outs :output list)) ==>
   (M,Oi,Os) sat
   (prop (SOME (NP npriv) :command inst) :
      (command inst, staff, 'd, 'e) Form):
   thm
```

For *Carol_exec_npriv_justified_thm:*

7

```
val Carol_exec_npriv_justified_thm =
   |- !(NS :state -> command trType -> state)
      (Out :state -> command trType -> output)
      (M :(command inst, 'b, staff, 'd, 'e) Kripke) (Oi :'d po)
      (Os :'e po).
     TR (M,Oi,Os) (exec (NP (npriv :npriv)))
       (CFG (inputOK2 :(command inst, staff, 'd, 'e) Form -> bool)
          (SMOStateInterp :state -> (command inst, staff, 'd, 'e) Form)
          (certs2 (cmd :command) npriv (privcmd :privcmd) :
             (command inst, staff, 'd, 'e) Form list)
          (Name Carol says
           (prop (SOME (NP npriv) :command inst) :
              (command inst, staff, 'd, 'e) Form)::
               (ins :(command inst, staff, 'd, 'e) Form list))
          (s :state) (outs :output list))
       (CFG (inputOK2 :(command inst, staff, 'd, 'e) Form -> bool)
          (SMOStateInterp :state -> (command inst, staff, 'd, 'e) Form)
          (certs2 cmd npriv privcmd :
             (command inst, staff, 'd, 'e) Form list) ins
          (NS s (exec (NP npriv))) (Out s (exec (NP npriv))::outs)) <=>
     inputOK2
       (Name Carol says
        (prop (SOME (NP npriv) :command inst) :
           (command inst, staff, 'd, 'e) Form)) /\
     CFGInterpret (M,Oi,Os)
       (CFG (inputOK2 :(command inst, staff, 'd, 'e) Form -> bool)
          (SMOStateInterp :state -> (command inst, staff, 'd, 'e) Form)
          (certs2 cmd npriv privcmd :
             (command inst, staff, 'd, 'e) Form list)
          (Name Carol says
           (prop (SOME (NP npriv) :command inst) :
              (command inst, staff, 'd, 'e) Form)::ins) s outs) /\
     (M,Oi,Os) sat
     (prop (SOME (NP npriv) :command inst) :
        (command inst, staff, 'd, 'e) Form):
   thm
```

For *Carol_npriv_verified_thm:*

8

```
val Carol_npriv_verified_thm =
   |- !(NS :state -> command trType -> state)
      (Out :state -> command trType -> output)
      (M :(command inst, 'b, staff, 'd, 'e) Kripke) (Oi :'d po)
      (Os :'e po).
     TR (M,Oi,Os) (exec (NP (npriv :npriv)))
       (CFG (inputOK2 :(command inst, staff, 'd, 'e) Form -> bool)
          (SMOStateInterp :state -> (command inst, staff, 'd, 'e) Form)
          (certs2 (cmd :command) npriv (privcmd :privcmd) :
             (command inst, staff, 'd, 'e) Form list)
          (Name Carol says
           (prop (SOME (NP npriv) :command inst) :
              (command inst, staff, 'd, 'e) Form)::
               (ins :(command inst, staff, 'd, 'e) Form list))
          (s :state) (outs :output list))
       (CFG (inputOK2 :(command inst, staff, 'd, 'e) Form -> bool)
          (SMOStateInterp :state -> (command inst, staff, 'd, 'e) Form)
          (certs2 cmd npriv privcmd :
             (command inst, staff, 'd, 'e) Form list) ins
          (NS s (exec (NP npriv))) (Out s (exec (NP npriv))::outs)) ==>
     (M,Oi,Os) sat
     (prop (SOME (NP npriv) :command inst) :
        (command inst, staff, 'd, 'e) Form):
   thm
```

For *Carol_justified_npriv_exec_thm:*

```
val Carol_justified_npriv_exec_thm =                                                    9
   |- !(NS :state -> command trType -> state)
      (Out :state -> command trType -> output)
      (M :(command inst, 'b, staff, 'd, 'e) Kripke) (Oi :'d po)
      (Os :'e po) (cmd :command) (npriv :npriv) (privcmd :privcmd)
      (ins :(command inst, staff, 'd, 'e) Form list) (s :state)
      (outs :output list).
     inputOK2
       (Name Carol says
        (prop (SOME (NP npriv) :command inst) :
           (command inst, staff, 'd, 'e) Form)) /\
     CFGInterpret (M,Oi,Os)
       (CFG (inputOK2 :(command inst, staff, 'd, 'e) Form -> bool)
          (SM0StateInterp :state -> (command inst, staff, 'd, 'e) Form)
          (certs2 cmd npriv privcmd :
             (command inst, staff, 'd, 'e) Form list)
          (Name Carol says
           (prop (SOME (NP npriv) :command inst) :
              (command inst, staff, 'd, 'e) Form)::ins) s outs) ==>
     TR (M,Oi,Os) (exec (NP npriv))
       (CFG (inputOK2 :(command inst, staff, 'd, 'e) Form -> bool)
          (SM0StateInterp :state -> (command inst, staff, 'd, 'e) Form)
          (certs2 cmd npriv privcmd :
             (command inst, staff, 'd, 'e) Form list)
          (Name Carol says
           (prop (SOME (NP npriv) :command inst) :
              (command inst, staff, 'd, 'e) Form)::ins) s outs)
       (CFG (inputOK2 :(command inst, staff, 'd, 'e) Form -> bool)
          (SM0StateInterp :state -> (command inst, staff, 'd, 'e) Form)
          (certs2 cmd npriv privcmd :
             (command inst, staff, 'd, 'e) Form list) ins
          (NS s (exec (NP npriv))) (Out s (exec (NP npriv))::outs)):
   thm
```

3.3 Exercise 17.4.3.C

In this section, we will prove the following theorems

$$\begin{array}{l}\vdash \texttt{CFGInterpret}\ (M,Oi,Os)\\ \quad\quad \texttt{(CFG inputOK2 SM0StateInterp (certs2}\ cmd\ npriv\ privcmd\texttt{)}\\ \quad\quad\quad \texttt{(Name Carol says prop (SOME (NP}\ npriv\texttt{))::}ins\texttt{)}\ s\ outs\texttt{)} \Rightarrow\\ \quad \texttt{(}M\texttt{,}Oi\texttt{,}Os\texttt{)}\ \texttt{sat prop (SOME (NP}\ npriv\texttt{))}\end{array}$$

3.3.1 Relevant Code

For theorem *Carol_privcmd_trap_lemma*, we will use the following code to prove it:

```
val Carol_privcmd_trap_lemma
= TAC_PROOF(
([], ``CFGInterpret ((M:(command inst,'b,staff,'d,'e)Kripke),Oi,Os)
  (CFG inputOK2 SM0StateInterp (certs2 cmd npriv privcmd)
   (((Name Carol) says (prop (SOME (PR (privcmd:privcmd)))))::ins)
   s (outs:output list)) ==>
  ((M,Oi,Os) sat (prop NONE))``),
  REWRITE_TAC[CFGInterpret_def, certs2_def, SM0StateInterp_def, satList_CONS,
      satList_nil, sat_TT] THEN
  PROVE_TAC[Controls, TR_trap_cmd_rule, Modus_Ponens])
```

As for *Carol_trap_privcmd_justified_thm*, please use the following code:

```
val Carol_trap_privcmd_justified_thm
=
let
val th1 =
    ISPECL
    [``inputOK2:(command inst, staff, 'd, 'e)Form->bool``,
```

```
    ``SM0StateInterp: state->(command inst, staff, 'd, 'e)Form``,
    ``(certs2 cmd npriv privcmd):(command inst, staff, 'd, 'e)Form list``,
    ``Name Carol``,``PR privcmd``, ``ins:(command inst, staff, 'd, 'e)Form
       list``,
    ``s:state``,``outs:output list``]
    TR_trap_cmd_rule
in
TAC_PROOF(
(
        [],
      ``!(NS :state -> command trType -> state)
         (Out :state -> command trType -> output)
         (M :(command inst, 'b, staff, 'd, 'e) Kripke) (Oi :'d po)
         (Os :'e po).
       TR (M,Oi,Os) (trap (PR (privcmd :privcmd)))
         (CFG (inputOK2 :(command inst, staff, 'd, 'e) Form -> bool)
            (SM0StateInterp :state -> (command inst, staff, 'd, 'e) Form)
            (certs2 (cmd :command) (npriv :npriv) privcmd :
               (command inst, staff, 'd, 'e) Form list)
            (Name Carol says
             (prop (SOME (PR privcmd) :command inst) :
                (command inst, staff, 'd, 'e) Form)::
                 (ins :(command inst, staff, 'd, 'e) Form list)) (s :state)
            (outs :output list))
         (CFG (inputOK2 :(command inst, staff, 'd, 'e) Form -> bool)
            (SM0StateInterp :state -> (command inst, staff, 'd, 'e) Form)
            (certs2 cmd npriv privcmd :
               (command inst, staff, 'd, 'e) Form list) ins
            (NS s (trap (PR privcmd)))
            (Out s (trap (PR privcmd))::outs)) <=>
       inputOK2
         (Name Carol says
          (prop (SOME (PR privcmd) :command inst) :
             (command inst, staff, 'd, 'e) Form)) /\
       CFGInterpret (M,Oi,Os)
         (CFG (inputOK2 :(command inst, staff, 'd, 'e) Form -> bool)
            (SM0StateInterp :state -> (command inst, staff, 'd, 'e) Form)
            (certs2 cmd npriv privcmd :
               (command inst, staff, 'd, 'e) Form list)
            (Name Carol says
             (prop (SOME (PR privcmd) :command inst) :
                (command inst, staff, 'd, 'e) Form)::ins) s outs) /\
       (M,Oi,Os) sat (prop NONE) : (command inst, staff, 'd, 'e) Form``),
       PROVE_TAC [th1, Carol_privcmd_trap_lemma])
end
```

The following code will prove *Carol_privcmd_trapped_thm*

```
val Carol_privcmd_trapped_thm
= TAC_PROOF(
([],
      ``!(NS :state -> command trType -> state)
         (Out :state -> command trType -> output)
         (M :(command inst, 'b, staff, 'd, 'e) Kripke) (Oi :'d po)
```

```
        (Os :'e po).
      TR (M,Oi,Os) (trap (PR (privcmd :privcmd)))
        (CFG (inputOK2 :(command inst, staff, 'd, 'e) Form -> bool)
           (SM0StateInterp :state -> (command inst, staff, 'd, 'e) Form)
           (certs2 (cmd :command) (npriv :npriv) privcmd :
              (command inst, staff, 'd, 'e) Form list)
           (Name Carol says
            (prop (SOME (PR privcmd) :command inst) :
               (command inst, staff, 'd, 'e) Form)::
                (ins :(command inst, staff, 'd, 'e) Form list)) (s :state)
           (outs :output list))
        (CFG (inputOK2 :(command inst, staff, 'd, 'e) Form -> bool)
           (SM0StateInterp :state -> (command inst, staff, 'd, 'e) Form)
           (certs2 cmd npriv privcmd :
              (command inst, staff, 'd, 'e) Form list) ins
           (NS s (trap (PR privcmd)))
           (Out s (trap (PR privcmd))::outs)) ==>
      (M,Oi,Os) sat
      (prop NONE:
         (command inst, staff, 'd, 'e) Form)``),
        PROVE_TAC [Carol_trap_privcmd_justified_thm]
)
```

Finally, we can use the following code to prove *Carol_justified_privcmd_trap_thm*

```
val Carol_justified_privcmd_trap_thm
=
TAC_PROOF(
([],
     ``!(NS :state -> command trType -> state)
       (Out :state -> command trType -> output)
       (M :(command inst, 'b, staff, 'd, 'e) Kripke) (Oi :'d po)
       (Os :'e po) cmd npriv privcmd ins s outs.
       inputOK2
        (Name Carol says
         (prop (SOME (PR privcmd) :command inst) :
            (command inst, staff, 'd, 'e) Form)) /\
        CFGInterpret (M,Oi,Os)
        (CFG (inputOK2 :(command inst, staff, 'd, 'e) Form -> bool)
           (SM0StateInterp :state -> (command inst, staff, 'd, 'e) Form)
           (certs2 cmd npriv privcmd :
              (command inst, staff, 'd, 'e) Form list)
           (Name Carol says
            (prop (SOME (PR privcmd) :command inst) :
               (command inst, staff, 'd, 'e) Form)::ins) s outs) ==>
      TR (M,Oi,Os) (trap (PR (privcmd :privcmd)))
        (CFG (inputOK2 :(command inst, staff, 'd, 'e) Form -> bool)
           (SM0StateInterp :state -> (command inst, staff, 'd, 'e) Form)
           (certs2 (cmd :command) (npriv :npriv) privcmd :
              (command inst, staff, 'd, 'e) Form list)
           (Name Carol says
            (prop (SOME (PR privcmd) :command inst) :
               (command inst, staff, 'd, 'e) Form)::
                (ins :(command inst, staff, 'd, 'e) Form list)) (s :state)
```

```
        (outs :output list))
      (CFG (inputOK2 :(command inst, staff, 'd, 'e) Form -> bool)
        (SM0StateInterp :state -> (command inst, staff, 'd, 'e) Form)
        (certs2 cmd npriv privcmd :
          (command inst, staff, 'd, 'e) Form list) ins
        (NS s (trap (PR privcmd)))
        (Out s (trap (PR privcmd))::outs))``),
        PROVE_TAC[Carol_trap_privcmd_justified_thm, inputOK2_def,
          Carol_privcmd_trap_lemma])
```

3.3.2 Session Transcript

If we send each of the code snippets to HOL session, we will see transcripts as below:

10

```
val Carol_privcmd_trap_lemma =
   |- CFGInterpret
     ((M :(command inst, 'b, staff, 'd, 'e) Kripke),(Oi :'d po),
      (Os :'e po))
     (CFG (inputOK2 :(command inst, staff, 'd, 'e) Form -> bool)
        (SMOStateInterp :state -> (command inst, staff, 'd, 'e) Form)
        (certs2 (cmd :command) (npriv :npriv) (privcmd :privcmd) :
           (command inst, staff, 'd, 'e) Form list)
        (Name Carol says
         (prop (SOME (PR privcmd) :command inst) :
            (command inst, staff, 'd, 'e) Form)::
             (ins :(command inst, staff, 'd, 'e) Form list)) (s :state)
        (outs :output list)) ==>
   (M,Oi,Os) sat
   (prop (NONE :command inst) :(command inst, staff, 'd, 'e) Form):
   thm
```

11

```
val Carol_trap_privcmd_justified_thm =
   |- !(NS :state -> command trType -> state)
      (Out :state -> command trType -> output)
      (M :(command inst, 'b, staff, 'd, 'e) Kripke) (Oi :'d po)
      (Os :'e po).
     TR (M,Oi,Os) (trap (PR (privcmd :privcmd)))
       (CFG (inputOK2 :(command inst, staff, 'd, 'e) Form -> bool)
          (SMOStateInterp :state -> (command inst, staff, 'd, 'e) Form)
          (certs2 (cmd :command) (npriv :npriv) privcmd :
             (command inst, staff, 'd, 'e) Form list)
          (Name Carol says
           (prop (SOME (PR privcmd) :command inst) :
              (command inst, staff, 'd, 'e) Form)::
               (ins :(command inst, staff, 'd, 'e) Form list))
          (s :state) (outs :output list))
       (CFG (inputOK2 :(command inst, staff, 'd, 'e) Form -> bool)
          (SMOStateInterp :state -> (command inst, staff, 'd, 'e) Form)
          (certs2 cmd npriv privcmd :
             (command inst, staff, 'd, 'e) Form list) ins
          (NS s (trap (PR privcmd)))
          (Out s (trap (PR privcmd))::outs)) <=>
     inputOK2
       (Name Carol says
        (prop (SOME (PR privcmd) :command inst) :
           (command inst, staff, 'd, 'e) Form)) /\
     CFGInterpret (M,Oi,Os)
       (CFG (inputOK2 :(command inst, staff, 'd, 'e) Form -> bool)
          (SMOStateInterp :state -> (command inst, staff, 'd, 'e) Form)
          (certs2 cmd npriv privcmd :
             (command inst, staff, 'd, 'e) Form list)
          (Name Carol says
           (prop (SOME (PR privcmd) :command inst) :
              (command inst, staff, 'd, 'e) Form)::ins) s outs) /\
     (M,Oi,Os) sat
     (prop (NONE :command inst) :(command inst, staff, 'd, 'e) Form):
   thm
```

12

```
val Carol_privcmd_trapped_thm =
   |- !(NS :state -> command trType -> state)
      (Out :state -> command trType -> output)
      (M :(command inst, 'b, staff, 'd, 'e) Kripke) (Oi :'d po)
      (Os :'e po).
     TR (M,Oi,Os) (trap (PR (privcmd :privcmd)))
       (CFG (inputOK2 :(command inst, staff, 'd, 'e) Form -> bool)
          (SMOStateInterp :state -> (command inst, staff, 'd, 'e) Form)
          (certs2 (cmd :command) (npriv :npriv) privcmd :
             (command inst, staff, 'd, 'e) Form list)
          (Name Carol says
           (prop (SOME (PR privcmd) :command inst) :
              (command inst, staff, 'd, 'e) Form)::
               (ins :(command inst, staff, 'd, 'e) Form list))
          (s :state) (outs :output list))
       (CFG (inputOK2 :(command inst, staff, 'd, 'e) Form -> bool)
          (SMOStateInterp :state -> (command inst, staff, 'd, 'e) Form)
          (certs2 cmd npriv privcmd :
             (command inst, staff, 'd, 'e) Form list) ins
          (NS s (trap (PR privcmd)))
          (Out s (trap (PR privcmd))::outs)) ==>
     (M,Oi,Os) sat
     (prop (NONE :command inst) :(command inst, staff, 'd, 'e) Form):
   thm
```

13

```
val Carol_justified_privcmd_trap_thm =
   |- !(NS :state -> command trType -> state)
      (Out :state -> command trType -> output)
      (M :(command inst, 'b, staff, 'd, 'e) Kripke) (Oi :'d po)
      (Os :'e po) (cmd :command) (npriv :npriv) (privcmd :privcmd)
      (ins :(command inst, staff, 'd, 'e) Form list) (s :state)
      (outs :output list).
     inputOK2
       (Name Carol says
        (prop (SOME (PR privcmd) :command inst) :
           (command inst, staff, 'd, 'e) Form)) /\
     CFGInterpret (M,Oi,Os)
       (CFG (inputOK2 :(command inst, staff, 'd, 'e) Form -> bool)
          (SMOStateInterp :state -> (command inst, staff, 'd, 'e) Form)
          (certs2 cmd npriv privcmd :
             (command inst, staff, 'd, 'e) Form list)
          (Name Carol says
           (prop (SOME (PR privcmd) :command inst) :
              (command inst, staff, 'd, 'e) Form)::ins) s outs) ==>
     TR (M,Oi,Os) (trap (PR privcmd))
       (CFG (inputOK2 :(command inst, staff, 'd, 'e) Form -> bool)
          (SMOStateInterp :state -> (command inst, staff, 'd, 'e) Form)
          (certs2 cmd npriv privcmd :
             (command inst, staff, 'd, 'e) Form list)
          (Name Carol says
           (prop (SOME (PR privcmd) :command inst) :
              (command inst, staff, 'd, 'e) Form)::ins) s outs)
       (CFG (inputOK2 :(command inst, staff, 'd, 'e) Form -> bool)
          (SMOStateInterp :state -> (command inst, staff, 'd, 'e) Form)
          (certs2 cmd npriv privcmd :
             (command inst, staff, 'd, 'e) Form list) ins
          (NS s (trap (PR privcmd))) (Out s (trap (PR privcmd))::outs)):
   thm
```

Appendix A

Source Code for ssm1Script.sml

The following code is from *ssm1Script.sml*, which is located in directory "../HOL/"

```
(*****************************************************************************
    *)
(* Secure State Machine Theory: authentication, authorization, and state
    *)
(* interpretation.
    *)
(* Author: Shiu-Kai Chin
    *)
(* Date: 27 November 2015
    *)
(*****************************************************************************
    *)


structure ssm1Script = struct

(* ==== Interactive mode ====
app load ["TypeBase", "ssminfRules","listTheory","optionTheory","acl_infRules
    ",
          "satListTheory","ssm1Theory"];
open TypeBase listTheory ssminfRules optionTheory acl_infRules satListTheory
     ssm1Theory
 ==== end interactive mode ==== *)

open HolKernel boolLib Parse bossLib
open TypeBase listTheory optionTheory ssminfRules acl_infRules satListTheory
(************************)
(* create a new theory *)
(************************)
val _ = new_theory "ssm1";

(* ---------------------------------------------------------------------------
    *)
(* Define the type of transition: discard, execute, or trap. We discard from
    *)
(* the input stream those inputs that are not of the form P says command. We
    *)
(* execute commands that users and supervisors are authorized for. We trap
    *)
(* commands that users are not authorized to execute.
    *)
```

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

(* ------------------------------------------------------------------------
   *)
(* In keeping with virtual machine design principles as described by Popek
   *)
(* and Goldberg, we add a TRAP instruction to the commands by users.
   *)
(* In effect, we are LIFTING the commands available to users to include the
   *)
(* TRAP instruction used by the state machine to handle authorization errors.
   *)
(* ------------------------------------------------------------------------
   *)

val _ =
Datatype
`inst = SOME 'command | NONE`

val inst_distinct_clauses = distinct_of``:'command inst``
val _ = save_thm("inst_distinct_clauses",inst_distinct_clauses)

val inst_one_one = one_one_of``:'command inst``
val _ = save_thm("inst_one_one",inst_one_one)

val _ =
Datatype
`trType =
  discard | trap 'command | exec 'command`

val trType_distinct_clauses = distinct_of``:'command trType``
val _ = save_thm("trType_distinct_clauses",trType_distinct_clauses)

val trType_one_one = one_one_of``:'command trType``
val _ = save_thm("trType_one_one",trType_one_one)

(* ------------------------------------------------------------------------
   *)
(* Define configuration to include the security context within which the
   *)
(* inputs are evaluated. The components are as follows: (1) the authentication
   *)
(* function, (2) the intepretation of the state, (3) the security context,
   *)
(* (4) the input stream, (5) the state, and (6) the output stream.
   *)
(* ------------------------------------------------------------------------
   *)
val _ =
Datatype
`configuration =
 CFG
```

```
  (('command inst,'principal,'d,'e)Form -> bool)
  (('state -> ('command inst,'principal,'d,'e)Form))
  (('command inst,'principal,'d,'e)Form list)
  (('command inst,'principal,'d,'e)Form list)
  ('state)
  ('output list)`


(* ------------------------------------------------------------------------
   *)
(* Prove one-to-one properties of configuration
   *)
(* ------------------------------------------------------------------------
   *)
val configuration_one_one =
    one_one_of``:('command inst,'d,'e,'output,'principal,'state)configuration
      ``

val _ = save_thm("configuration_one_one",configuration_one_one)

(* ------------------------------------------------------------------------
   *)
(* The interpretation of configuration is the conjunction of the formulas in
   *)
(* the context and the first element of a non-empty input stream.
   *)
(* ------------------------------------------------------------------------
   *)
val CFGInterpret_def =
Define
`CFGInterpret
 ((M:('command inst,'b,'principal,'d,'e)Kripke),Oi:'d po,Os:'e po)
 (CFG
  (inputTest:('command inst,'principal,'d,'e)Form -> bool)
  (stateInterp:'state -> ('command inst,'principal,'d,'e)Form)
  (context:('command inst,'principal,'d,'e)Form list)
  ((x:('command inst,'principal,'d,'e)Form)::ins)
  (state:'state)
  (outStream:'output list))
 =
  ((M,Oi,Os) satList context) /\
  ((M,Oi,Os) sat x) /\
  ((M,Oi,Os) sat (stateInterp state))`

(* ------------------------------------------------------------------------
   *)
(* Define transition relation among configurations. This definition is
   *)
(* parameterized in terms of next-state transition function and output
   *)
(* function.
   *)
```

```
(* The first rule is set up with the expectation it is some principal P
   *)
(* ordering a command cmd be executed.
   *)
(* ------------------------------------------------------------------------
   *)
val (TR_rules, TR_ind, TR_cases) =
Hol_reln
`(!(inputTest:('command inst,'principal,'d,'e)Form -> bool) (P:'principal
   Princ)
    (NS: 'state -> 'command trType -> 'state) M Oi Os Out (s:'state)
    (certList:('command inst,'principal,'d,'e)Form list)
    (stateInterp:'state -> ('command inst,'principal,'d,'e)Form)
    (cmd:'command)(ins:('command inst,'principal,'d,'e)Form list)
    (outs:'output list).
 (inputTest ((P says (prop (SOME cmd))):('command inst,'principal,'d,'e)Form)
    /\
 (CFGInterpret (M,Oi,Os)
  (CFG inputTest stateInterp certList
       (((P says (prop (SOME cmd))):('command inst,'principal,'d,'e)Form)::ins
           )
       s outs))) ==>
 (TR
  ((M:('command inst,'b,'principal,'d,'e)Kripke),Oi:'d po,Os:'e po) (exec cmd)
  (CFG inputTest stateInterp certList
       (((P says (prop (SOME cmd))):('command inst,'principal,'d,'e)Form)::ins
           )
      s outs)
  (CFG inputTest stateInterp certList ins (NS s (exec cmd))
       ((Out s (exec cmd))::outs)))) /\
 (!(inputTest:('command inst,'principal,'d,'e)Form -> bool) (P:'principal
    Princ)
   (NS:'state -> 'command trType -> 'state) M Oi Os Out (s:'state)
   (certList:('command inst,'principal,'d,'e)Form list)
   (stateInterp:'state -> ('command inst,'principal,'d,'e)Form)
   (cmd:'command)(ins:('command inst,'principal,'d,'e)Form list)
   (outs:'output list).
 (inputTest ((P says (prop (SOME cmd))):('command inst,'principal,'d,'e)Form)
    /\
 (CFGInterpret (M,Oi,Os)
  (CFG inputTest stateInterp certList
       (((P says (prop (SOME cmd))):('command inst,'principal,'d,'e)Form)::ins
           )
       s outs))) ==>
 (TR
  ((M:('command inst,'b,'principal,'d,'e)Kripke),Oi:'d po,Os:'e po) (trap cmd)
 (CFG inputTest stateInterp certList
      (((P says (prop (SOME cmd))):('command inst,'principal,'d,'e)Form)::ins)
      s outs)
 (CFG inputTest stateInterp certList ins (NS s (trap cmd))
      ((Out s (trap cmd))::outs)))) /\
 (!(inputTest:('command inst,'principal,'d,'e)Form -> bool)
   (NS:'state -> 'command trType -> 'state)
```

```
   M Oi Os (Out: 'state -> 'command trType -> 'output) (s:'state)
    (certList:('command inst,'principal,'d,'e)Form list)
    (stateInterp:'state -> ('command inst,'principal,'d,'e)Form)
    (cmd:'command)(x:('command inst,'principal,'d,'e)Form)
    (ins:('command inst,'principal,'d,'e)Form list)
    (outs:'output list).
 ~inputTest x ==>
 (TR
  ((M:('command inst,'b,'principal,'d,'e)Kripke),Oi:'d po,Os:'e po)
  (discard:'command trType)
 (CFG inputTest stateInterp certList
      ((x:('command inst,'principal,'d,'e)Form)::ins) s outs)
 (CFG inputTest stateInterp certList ins (NS s discard)
      ((Out s discard)::outs))))`


(* ------------------------------------------------------------------------
   *)
(* Split up TR_rules into individual clauses
   *)
(* ------------------------------------------------------------------------
   *)
val [rule0,rule1,rule2] = CONJUNCTS TR_rules

(*****************************************************************************
   *)
(* Prove the converse of rule0, rule1, and rule2
   *)
(*****************************************************************************
   *)
val TR_lemma0 =
TAC_PROOF(([],flip_TR_rules rule0),
DISCH_TAC THEN
IMP_RES_TAC TR_cases THEN
PAT_ASSUM
  ``exec cmd = y``
 (fn th => ASSUME_TAC(REWRITE_RULE[trType_one_one,trType_distinct_clauses]th))
     THEN
PROVE_TAC[configuration_one_one,list_11,trType_distinct_clauses])

val TR_lemma1 =
TAC_PROOF(([],flip_TR_rules rule1),
DISCH_TAC THEN
IMP_RES_TAC TR_cases THEN
PAT_ASSUM
  ``trap cmd = y``
 (fn th => ASSUME_TAC(REWRITE_RULE[trType_one_one,trType_distinct_clauses]th))
     THEN
 PROVE_TAC[configuration_one_one,list_11,trType_distinct_clauses])

val TR_lemma2 =
TAC_PROOF(([],flip_TR_rules rule2),
```

```
DISCH_TAC THEN
IMP_RES_TAC TR_cases THEN
PAT_ASSUM
 ``discard = y``
 (fn th => ASSUME_TAC(REWRITE_RULE[trType_one_one, trType_distinct_clauses]th))
     THEN
PROVE_TAC[configuration_one_one, list_11, trType_distinct_clauses])

val TR_rules_converse =
TAC_PROOF(([], flip_TR_rules TR_rules),
REWRITE_TAC[TR_lemma0, TR_lemma1, TR_lemma2])

val TR_EQ_rules_thm = TR_EQ_rules TR_rules TR_rules_converse

val _ = save_thm("TR_EQ_rules_thm", TR_EQ_rules_thm)

val [TRrule0, TRrule1, TR_discard_cmd_rule] = CONJUNCTS TR_EQ_rules_thm

val _ = save_thm("TRrule0", TRrule0)
val _ = save_thm("TRrule1", TRrule1)
val _ = save_thm("TR_discard_cmd_rule", TR_discard_cmd_rule)


(* ------------------------------------------------------------------------
   *)
(* If (CFGInterpret
   *)
(*     (M, Oi, Os)
   *)
(*     (CFG inputTest stateInterpret certList
                                          *)
(*          ((P says (prop (CMD cmd))) :: ins) s outs) ==>
   *)
(*    ((M, Oi, Os) sat (prop (CMD cmd))))
   *)
(* is a valid inference rule, then executing cmd the exec(CMD cmd) transition
   *)
(* occurs if and only if prop (CMD cmd), inputTest, and
   *)
(* CFGInterpret (M, Oi, Os)
   *)
(*  (CFG inputTest stateInterpret certList (P says prop (CMD cmd) :: ins) s outs
   ) *)
(* are true.
   *)
(* ------------------------------------------------------------------------
   *)
val TR_exec_cmd_rule =
TAC_PROOF(([],
``!inputTest certList stateInterp P cmd ins s outs.
 (!M Oi Os.
 (CFGInterpret
```

```
  ((M :('command inst, 'b, 'principal, 'd, 'e) Kripke),(Oi :'d po), (Os :'e po
     ))
  (CFG inputTest
       (stateInterp:'state -> ('command inst,'principal,'d,'e)Form) certList
       (P says (prop (SOME cmd) :('command inst, 'principal, 'd, 'e) Form)::
          ins)
       (s:'state) (outs:'output list)) ==>
  (M,Oi,Os) sat (prop (SOME cmd) :('command inst, 'principal, 'd, 'e) Form)))
     ==>
(!NS Out M Oi Os.
 TR
  ((M :('command inst, 'b, 'principal, 'd, 'e) Kripke),(Oi :'d po),
   (Os :'e po)) (exec (cmd :'command))
  (CFG (inputTest :('command inst, 'principal, 'd, 'e) Form -> bool)
           (stateInterp:'state -> ('command inst,'principal,'d,'e)Form)
           (certList :('command inst, 'principal, 'd, 'e) Form list)
           ((P :'principal Princ) says
            (prop (SOME cmd) :('command inst, 'principal, 'd, 'e) Form)::
                (ins :('command inst, 'principal, 'd, 'e) Form list))
           (s :'state) (outs :'output list))
   (CFG inputTest stateInterp certList ins
           ((NS :'state -> 'command trType -> 'state) s (exec cmd))
           ((Out :'state -> 'command trType -> 'output) s (exec cmd)::
               outs)) <=>
   inputTest
    (P says (prop (SOME cmd) :('command inst, 'principal, 'd, 'e) Form)) /\
   (CFGInterpret (M,Oi,Os)
    (CFG
      inputTest stateInterp certList
      (P says (prop (SOME cmd):('command inst, 'principal, 'd, 'e) Form)::ins)
      s outs)) /\
   (M,Oi,Os) sat (prop (SOME cmd) :('command inst, 'principal, 'd, 'e) Form))
       ''),
REWRITE_TAC[TRrule0] THEN
REPEAT STRIP_TAC THEN
EQ_TAC THEN
REPEAT STRIP_TAC THEN
PROVE_TAC[])

val _ = save_thm("TR_exec_cmd_rule",TR_exec_cmd_rule)

(* ------------------------------------------------------------------------
   *)
(* If (CFGInterpret
   *)
(*      (M,Oi,Os)
   *)
(*      (CFG inputTest stateInterpret certList
                                          *)
(*           ((P says (prop (CMD cmd)))::ins) s outs) ==>
   *)
(*    ((M,Oi,Os) sat (prop TRAP)))
   *)
```

```
(* is a valid inference rule, then executing cmd the exec(CMD cmd) transition
   *)
(* occurs if and only if prop TRAP, inputTest, and
   *)
(* CFGInterpret (M,Oi,Os)
   *)
(*  (CFG inputTest stateInterpret certList (P says prop (CMD cmd)::ins)
          *)
(*       s outs) are true.
   *)
(* ------------------------------------------------------------------------
   *)
val TR_trap_cmd_rule =
TAC_PROOF(
([],
``!inputTest (stateInterp:'state -> ('command inst,'principal,'d,'e)Form)
   certList
   P cmd ins (s:'state) (outs:'output list).
  (!M Oi Os.
  CFGInterpret
   ((M :('command inst, 'b, 'principal, 'd, 'e) Kripke),(Oi :'d po), (Os :'e
      po))
   (CFG inputTest stateInterp certList
        (P says (prop (SOME cmd) :('command inst, 'principal, 'd, 'e) Form)::
           ins)
           s outs) ==>
  (M,Oi,Os) sat (prop NONE :('command inst, 'principal, 'd, 'e) Form)) ==>
(!NS Out M Oi Os.
TR
  ((M :('command inst, 'b, 'principal, 'd, 'e) Kripke),(Oi :'d po),
   (Os :'e po)) (trap (cmd :'command))
  (CFG (inputTest :('command inst, 'principal, 'd, 'e) Form -> bool)
     (stateInterp:'state -> ('command inst,'principal,'d,'e)Form)
     (certList :('command inst, 'principal, 'd, 'e) Form list)
     ((P :'principal Princ) says
      (prop (SOME cmd) :('command inst, 'principal, 'd, 'e) Form)::
          (ins :('command inst, 'principal, 'd, 'e) Form list))
     (s :'state) outs)
  (CFG inputTest (stateInterp:'state -> ('command inst,'principal,'d,'e)Form)
      certList ins
     ((NS :'state -> 'command trType -> 'state) s (trap cmd))
     ((Out :'state -> 'command trType -> 'output) s
        (trap cmd)::outs)) <=>
inputTest
  (P says
   (prop (SOME cmd) :('command inst, 'principal, 'd, 'e) Form)) /\
CFGInterpret (M,Oi,Os)
  (CFG inputTest (stateInterp:'state -> ('command inst,'principal,'d,'e)Form)
      certList
     (P says
      (prop (SOME cmd) :('command inst, 'principal, 'd, 'e) Form)::ins)
     s outs) /\
(M,Oi,Os) sat (prop NONE))``),
```

```
REWRITE_TAC[TRrule1] THEN
REPEAT STRIP_TAC THEN
EQ_TAC THEN
REPEAT STRIP_TAC THEN
PROVE_TAC[])

val _ = save_thm("TR_trap_cmd_rule",TR_trap_cmd_rule)
(* ==== start here ====

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

val _ = export_theory ();
val _ = print_theory "-";

end (* structure *)
```

Appendix B

Source Code for SM0Script.sml

The following code is from *SM0Script.sml*, which is located in directory "../HOL/"

```
(*****************************************************************************
    *)
(* Machine SM0 example
    *)
(* Author: Shiu-Kai Chin
    *)
(* Date: 30 November 2015
    *)
(*****************************************************************************
    *)

structure SM0Script = struct


(* interactive mode
app load ["TypeBase","ssm1Theory","SM0Theory","acl_infRules","aclrulesTheory",
          "aclDrulesTheory","SM0Theory"];
open TypeBase ssm1Theory acl_infRules aclrulesTheory
     aclDrulesTheory satListTheory SM0Theory
*)

open HolKernel boolLib Parse bossLib
open TypeBase ssm1Theory acl_infRules aclrulesTheory aclDrulesTheory
     satListTheory

(***********
* create a new theory
***********)

val _ = new_theory "SM0"



(* ------------------------------------------------------------------------
    *)
(* Define datatypes for commands and their properties
    *)
(* ------------------------------------------------------------------------
    *)
val _ =
Datatype `privcmd = launch | reset `
```

```
val privcmd_distinct_clauses = distinct_of ``:privcmd``
val _ = save_thm("privcmd_distinct_clauses",privcmd_distinct_clauses)

val _ =
Datatype `npriv = status`

val _ =
Datatype `command = NP npriv | PR privcmd`

val command_distinct_clauses = distinct_of ``:command``
val _ = save_thm("command_distinct_clauses",command_distinct_clauses)

val command_one_one = one_one_of ``:command``
val _ = save_thm("command_one_one",command_one_one)

(* ------------------------------------------------------------------------
    *)
(* Define the states
    *)
(* ------------------------------------------------------------------------
    *)
val _ =
Datatype`state = STBY | ACTIVE`

val state_distinct_clauses = distinct_of ``:state``
val _ = save_thm("state_distinct_clauses",state_distinct_clauses)


(* ------------------------------------------------------------------------
    *)
(* Define the outputs
    *)
(* ------------------------------------------------------------------------
    *)
val _ =
Datatype`output = on | off`

val output_distinct_clauses = distinct_of ``:output``
val _ = save_thm("output_distinct_clauses",output_distinct_clauses)

(* ------------------------------------------------------------------------
    *)
(* Define next-state function for machine M0
    *)
(* ------------------------------------------------------------------------
    *)
val SM0ns_def =
Define
`(SM0ns STBY (exec (PR reset)) = STBY) /\
 (SM0ns STBY (exec (PR launch)) = ACTIVE) /\
 (SM0ns STBY (exec (NP status)) = STBY) /\
 (SM0ns ACTIVE (exec (PR reset)) = STBY) /\
 (SM0ns ACTIVE (exec (PR launch)) = ACTIVE) /\
```

```
 (SM0ns ACTIVE (exec (NP status)) = ACTIVE) /\
 (SM0ns STBY (trap (PR reset)) = STBY) /\
 (SM0ns STBY (trap (PR launch)) = STBY) /\
 (SM0ns STBY (trap (NP status)) = STBY) /\
 (SM0ns ACTIVE (trap (PR reset)) = ACTIVE) /\
 (SM0ns ACTIVE (trap (PR launch)) = ACTIVE) /\
 (SM0ns ACTIVE (trap (NP status)) = ACTIVE) /\
 (SM0ns STBY discard = STBY) /\
 (SM0ns ACTIVE discard = ACTIVE)‘

(* ------------------------------------------------------------------------
   *)
(* Define next-output function for machine M0
   *)
(* ------------------------------------------------------------------------
   *)
val SM0out_def =
Define
‘(SM0out STBY (exec (PR reset)) = off) /\
 (SM0out STBY (exec (PR launch)) = on) /\
 (SM0out STBY (exec (NP status)) = off) /\
 (SM0out ACTIVE (exec (PR reset)) = off) /\
 (SM0out ACTIVE (exec (PR launch)) = on) /\
 (SM0out ACTIVE (exec (NP status)) = on) /\
 (SM0out STBY (trap (PR reset)) = off) /\
 (SM0out STBY (trap (PR launch)) = off) /\
 (SM0out STBY (trap (NP status)) = off) /\
 (SM0out ACTIVE (trap (PR reset)) = on) /\
 (SM0out ACTIVE (trap (PR launch)) = on) /\
 (SM0out ACTIVE (trap (NP status)) = on) /\
 (SM0out STBY discard = off) /\
 (SM0out ACTIVE discard = on)‘
(* ------------------------------------------------------------------------
   *)
(* Define datatypes for principles and their properties
   *)
(* ------------------------------------------------------------------------
   *)
val _ =
Datatype ‘staff = Alice | Bob | Carol‘

val staff_distinct_clauses = distinct_of ‘‘:staff‘‘
val _ = save_thm("staff_distinct_clauses",staff_distinct_clauses)

(* ------------------------------------------------------------------------
   *)
(* Input Authentication
   *)
(* ------------------------------------------------------------------------
   *)
val inputOK_def =
Define
‘(inputOK
```

```
  (((Name Alice) says
   (prop (SOME (cmd:command)))):(command inst,staff,'d,'e)Form) = T) /\
 (inputOK
  (((Name Bob) says
   (prop (SOME (cmd:command)))):(command inst,staff,'d,'e)Form) = T) /\
 (inputOK _ = F)`


(* ------------------------------------------------------------------------
   *)
(* SM0StateInterp
   *)
(* ------------------------------------------------------------------------
   *)
val SM0StateInterp_def =
Define
`SM0StateInterp (state:state) = (TT:(command inst,staff,'d,'e)Form)`

(* ------------------------------------------------------------------------
   *)
(* certs definition
   *)
(* ------------------------------------------------------------------------
   *)
val certs_def =
Define
`certs (cmd:command)(npriv:npriv)(privcmd:privcmd) =
 [(Name Alice controls ((prop (SOME (NP npriv))):(command inst, staff,'d,'e)
    Form));
  Name Alice controls (prop (SOME (PR privcmd)));
  Name Bob controls prop (SOME (NP npriv));
  ((Name Bob) says (prop (SOME (PR privcmd)))) impf (prop NONE)]`


val inputOK2_def =
Define
`(inputOK2
  (((Name Carol) says
   (prop (SOME (cmd:command)))):(command inst,staff,'d,'e)Form) = T) /\
 (inputOK2 _ = F)`

val certs2_def =
Define
`certs2 (cmd:command)(npriv:npriv)(privcmd:privcmd) =
 [(Name Carol controls ((prop (SOME (NP npriv))):(command inst, staff,'d,'e)
    Form));
  ((Name Carol) says (prop (SOME (PR privcmd)))) impf (prop NONE)]`



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

```
(* Some theorems showing any message from Carol is rejected
   *)
(* ------------------------------------------------------------------------
   *)
val Carol_rejected_lemma =
TAC_PROOF(([],
``~inputOK
   (((Name Carol) says (prop (SOME (cmd:command)))):(command inst,staff,'d,'e)
      Form)``),
PROVE_TAC[inputOK_def])

val _ = save_thm("Carol_rejected_lemma",Carol_rejected_lemma)

val Carol_discard_lemma =
TAC_PROOF(([],
``TR ((M:(command inst,'b,staff,'d,'e)Kripke),Oi,Os) discard
  (CFG inputOK SM0StateInterp (certs cmd npriv privcmd)
   (((Name Carol) says (prop (SOME (cmd:command))))::ins)
   s (outs:output list))
  (CFG inputOK SM0StateInterp (certs cmd npriv privcmd) ins
  (SM0ns s discard) ((SM0out s discard)::outs))``),
PROVE_TAC[Carol_rejected_lemma,TR_discard_cmd_rule])

val _ = save_thm("Carol_discard_lemma",Carol_discard_lemma)


(* ------------------------------------------------------------------------
   *)
(* Alice authorized on any privileged command
   *)
(* ------------------------------------------------------------------------
   *)
val Alice_privcmd_lemma =
TAC_PROOF(([],
``CFGInterpret ((M:(command inst,'b,staff,'d,'e)Kripke),Oi,Os)
  (CFG inputOK SM0StateInterp (certs cmd npriv privcmd)
   (((Name Alice) says (prop (SOME (PR (privcmd:privcmd)))))::ins)
   s (outs:output list)) ==>
  ((M,Oi,Os) sat (prop (SOME(PR privcmd))))``),
REWRITE_TAC[CFGInterpret_def,certs_def,SM0StateInterp_def,satList_CONS,
            satList_nil,sat_TT] THEN
PROVE_TAC[Controls])

val _ = save_thm("Alice_privcmd_lemma",Alice_privcmd_lemma)

(* ------------------------------------------------------------------------
   *)
(* exec privcmd occurs if and only if Alice's command is authenticated and
   *)
(* authorized
   *)
(* ------------------------------------------------------------------------
   *)
```

```
val Alice_exec_privcmd_justified_thm =
let
 val th1 =
 ISPECL
 [``inputOK:(command inst, staff,'d,'e)Form -> bool``,
  ``(certs cmd npriv privcmd):(command inst, staff,'d,'e)Form list``,
  ``SM0StateInterp:state->(command inst, staff,'d,'e)Form``,
  ``Name Alice``,``PR privcmd``,``ins:(command inst,staff,'d,'e)Form list``,
  ``s:state``,``outs:output list``]
 TR_exec_cmd_rule
in
 TAC_PROOF(([],
    ``!(NS :state -> command trType -> state)
         (Out :state -> command trType -> output)
         (M :(command inst, 'b, staff, 'd, 'e) Kripke) (Oi :'d po)
         (Os :'e po).
       TR (M,Oi,Os) (exec (PR (privcmd :privcmd)))
         (CFG (inputOK :(command inst, staff, 'd, 'e) Form -> bool)
            (SM0StateInterp :state -> (command inst, staff, 'd, 'e) Form)
            (certs (cmd :command) (npriv :npriv) privcmd :
               (command inst, staff, 'd, 'e) Form list)
            (Name Alice says
             (prop (SOME (PR privcmd) :command inst) :
                (command inst, staff, 'd, 'e) Form)::
                 (ins :(command inst, staff, 'd, 'e) Form list)) (s :state)
            (outs :output list))
         (CFG (inputOK :(command inst, staff, 'd, 'e) Form -> bool)
            (SM0StateInterp :state -> (command inst, staff, 'd, 'e) Form)
            (certs cmd npriv privcmd :
               (command inst, staff, 'd, 'e) Form list) ins
            (NS s (exec (PR privcmd)))
            (Out s (exec (PR privcmd))::outs)) <=>
       inputOK
         (Name Alice says
          (prop (SOME (PR privcmd) :command inst) :
             (command inst, staff, 'd, 'e) Form)) /\
       CFGInterpret (M,Oi,Os)
         (CFG (inputOK :(command inst, staff, 'd, 'e) Form -> bool)
            (SM0StateInterp :state -> (command inst, staff, 'd, 'e) Form)
            (certs cmd npriv privcmd :
               (command inst, staff, 'd, 'e) Form list)
            (Name Alice says
             (prop (SOME (PR privcmd) :command inst) :
                (command inst, staff, 'd, 'e) Form)::ins) s outs) /\
       (M,Oi,Os) sat
       (prop (SOME (PR privcmd) :command inst) :
          (command inst, staff, 'd, 'e) Form)``),
 PROVE_TAC[th1, Alice_privcmd_lemma])
end

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

```
(* ------------------------------------------------------------------------
   *)
(* If Alice's privileged command was executed, then the request was verified.
   *)
(* ------------------------------------------------------------------------
   *)
val Alice_privcmd_verified_thm =
TAC_PROOF(([], ``!(NS :state -> command trType -> state)
          (Out :state -> command trType -> output)
          (M :(command inst, 'b, staff, 'd, 'e) Kripke) (Oi :'d po)
          (Os :'e po).
         TR (M,Oi,Os) (exec (PR (privcmd :privcmd)))
           (CFG (inputOK :(command inst, staff, 'd, 'e) Form -> bool)
              (SM0StateInterp :state -> (command inst, staff, 'd, 'e) Form)
              (certs (cmd :command) (npriv :npriv) privcmd :
                 (command inst, staff, 'd, 'e) Form list)
              (Name Alice says
               (prop (SOME (PR privcmd) :command inst) :
                  (command inst, staff, 'd, 'e) Form)::
                   (ins :(command inst, staff, 'd, 'e) Form list)) (s :state)
              (outs :output list))
           (CFG (inputOK :(command inst, staff, 'd, 'e) Form -> bool)
              (SM0StateInterp :state -> (command inst, staff, 'd, 'e) Form)
              (certs cmd npriv privcmd :
                 (command inst, staff, 'd, 'e) Form list) ins
              (NS s (exec (PR privcmd)))
              (Out s (exec (PR privcmd))::outs)) ==>
         (M,Oi,Os) sat
         (prop (SOME (PR privcmd) :command inst) :
            (command inst, staff, 'd, 'e) Form)``),
PROVE_TAC[Alice_exec_privcmd_justified_thm])

val _ = save_thm("Alice_privcmd_verified_thm",Alice_privcmd_verified_thm)

(* ------------------------------------------------------------------------
   *)
(* If Alice's privileged command was authorized, then the command is executed
   *)
(* ------------------------------------------------------------------------
   *)
val Alice_justified_privcmd_exec_thm =
TAC_PROOF(([], ``!(NS :state -> command trType -> state)
          (Out :state -> command trType -> output)
          (M :(command inst, 'b, staff, 'd, 'e) Kripke) (Oi :'d po)
          (Os :'e po) cmd npriv privcmd ins s outs.
          inputOK
           (Name Alice says
            (prop (SOME (PR privcmd) :command inst) :
               (command inst, staff, 'd, 'e) Form)) /\
           CFGInterpret (M,Oi,Os)
           (CFG (inputOK :(command inst, staff, 'd, 'e) Form -> bool)
              (SM0StateInterp :state -> (command inst, staff, 'd, 'e) Form)
```

```
            (certs cmd npriv privcmd :
               (command inst, staff, 'd, 'e) Form list)
            (Name Alice says
             (prop (SOME (PR privcmd) :command inst) :
                (command inst, staff, 'd, 'e) Form)::ins) s outs) ==>
      TR (M,Oi,Os) (exec (PR (privcmd :privcmd)))
        (CFG (inputOK :(command inst, staff, 'd, 'e) Form -> bool)
            (SM0StateInterp :state -> (command inst, staff, 'd, 'e) Form)
            (certs (cmd :command) (npriv :npriv) privcmd :
               (command inst, staff, 'd, 'e) Form list)
            (Name Alice says
             (prop (SOME (PR privcmd) :command inst) :
                (command inst, staff, 'd, 'e) Form)::
                 (ins :(command inst, staff, 'd, 'e) Form list)) (s :state)
            (outs :output list))
        (CFG (inputOK :(command inst, staff, 'd, 'e) Form -> bool)
            (SM0StateInterp :state -> (command inst, staff, 'd, 'e) Form)
            (certs cmd npriv privcmd :
               (command inst, staff, 'd, 'e) Form list) ins
            (NS s (exec (PR privcmd)))
            (Out s (exec (PR privcmd))::outs))``),
PROVE_TAC[Alice_exec_privcmd_justified_thm,inputOK_def,Alice_privcmd_lemma])

val _ = save_thm("Alice_justified_privcmd_exec_thm",
    Alice_justified_privcmd_exec_thm)


val _ = export_theory ()
val _ = print_theory "-"

end (* structure *)
```

Appendix C

Source Code for SM0SolutionsScript.sml

The following code is from *SM0SolutionsScript.sml*, which is located in directory "../HOL/"

```
(*****************************************************************************
    *)
(* Solutions for Exercises 17.4.1 and 17.4.3
    *)
(* Author: Shiu-Kai Chin
    *)
(* Date: 1 April 2017
    *)
(*****************************************************************************
    *)

structure SM0SolutionsScript = struct

open HolKernel Parse boolLib bossLib;
open ssm1Theory SM0Theory;
open acl_infRules aclrulesTheory aclDrulesTheory satListTheory;

val _ = new_theory "SM0Solutions";


(* ---------------------------------------------------------------------------
    *)
(* Exercise 17.4.1
    *)
(* Alice's non-privileged commands are executed and justified
    *)
(* ---------------------------------------------------------------------------
    *)
val Alice_npriv_lemma =
TAC_PROOF(
([], ``CFGInterpret ((M:(command inst,'b,staff,'d,'e)Kripke),Oi,Os)
  (CFG inputOK SM0StateInterp (certs cmd npriv privcmd)
   (((Name Alice) says (prop (SOME (NP (npriv:npriv)))))::ins)
   s (outs:output list)) ==>
  ((M,Oi,Os) sat (prop (SOME(NP npriv))))``),
  REWRITE_TAC[CFGInterpret_def, certs_def, SM0StateInterp_def, satList_CONS,
      satList_nil, sat_TT] THEN
  PROVE_TAC [Controls])

val _ = save_thm("Alice_npriv_lemma", Alice_npriv_lemma)

val Alice_exec_npriv_justified_thm
```

```
=
let
val th1 =
    ISPECL
    [``inputOK:(command inst, staff, 'd, 'e)Form->bool``,
    ``(certs cmd npriv privcmd):(command inst, staff, 'd, 'e)Form list``,
    ``SM0StateInterp: state->(command inst, staff, 'd, 'e)Form``,
    ``Name Alice``,``NP npriv``, ``ins:(command inst, staff, 'd, 'e)Form list
       ``,
    ``s:state``,``outs:output list``]
    TR_exec_cmd_rule
in
 TAC_PROOF(([],
    ``!(NS :state -> command trType -> state)
         (Out :state -> command trType -> output)
         (M :(command inst, 'b, staff, 'd, 'e) Kripke) (Oi :'d po)
         (Os :'e po).
       TR (M,Oi,Os) (exec (NP (npriv :npriv)))
         (CFG (inputOK :(command inst, staff, 'd, 'e) Form -> bool)
            (SM0StateInterp :state -> (command inst, staff, 'd, 'e) Form)
            (certs (cmd :command) (npriv :npriv) privcmd :
               (command inst, staff, 'd, 'e) Form list)
            (Name Alice says
             (prop (SOME (NP npriv) :command inst) :
                (command inst, staff, 'd, 'e) Form)::
                 (ins :(command inst, staff, 'd, 'e) Form list)) (s :state)
            (outs :output list))
         (CFG (inputOK :(command inst, staff, 'd, 'e) Form -> bool)
            (SM0StateInterp :state -> (command inst, staff, 'd, 'e) Form)
            (certs cmd npriv privcmd :
               (command inst, staff, 'd, 'e) Form list) ins
            (NS s (exec (NP npriv)))
            (Out s (exec (NP npriv))::outs)) <=>
       inputOK
         (Name Alice says
          (prop (SOME (NP npriv) :command inst) :
             (command inst, staff, 'd, 'e) Form)) /\
       CFGInterpret (M,Oi,Os)
         (CFG (inputOK :(command inst, staff, 'd, 'e) Form -> bool)
            (SM0StateInterp :state -> (command inst, staff, 'd, 'e) Form)
            (certs cmd npriv privcmd :
               (command inst, staff, 'd, 'e) Form list)
            (Name Alice says
             (prop (SOME (NP npriv) :command inst) :
                (command inst, staff, 'd, 'e) Form)::ins) s outs) /\
       (M,Oi,Os) sat
       (prop (SOME (NP npriv) :command inst) :
          (command inst, staff, 'd, 'e) Form)``),
       PROVE_TAC[th1, Alice_npriv_lemma])
end

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

```
val Alice_npriv_verified_thm =
TAC_PROOF(([],
      ``!(NS :state -> command trType -> state)
         (Out :state -> command trType -> output)
         (M :(command inst, 'b, staff, 'd, 'e) Kripke) (Oi :'d po)
         (Os :'e po).
        TR (M,Oi,Os) (exec (NP (npriv :npriv)))
          (CFG (inputOK :(command inst, staff, 'd, 'e) Form -> bool)
             (SM0StateInterp :state -> (command inst, staff, 'd, 'e) Form)
             (certs (cmd :command) (npriv :npriv) privcmd :
                (command inst, staff, 'd, 'e) Form list)
             (Name Alice says
              (prop (SOME (NP npriv) :command inst) :
                 (command inst, staff, 'd, 'e) Form)::
                  (ins :(command inst, staff, 'd, 'e) Form list)) (s :state)
             (outs :output list))
          (CFG (inputOK :(command inst, staff, 'd, 'e) Form -> bool)
             (SM0StateInterp :state -> (command inst, staff, 'd, 'e) Form)
             (certs cmd npriv privcmd :
                (command inst, staff, 'd, 'e) Form list) ins
             (NS s (exec (NP npriv)))
             (Out s (exec (NP npriv))::outs)) ==>
        (M,Oi,Os) sat
        (prop (SOME (NP npriv) :command inst) :
           (command inst, staff, 'd, 'e) Form)``),
          PROVE_TAC[Alice_exec_npriv_justified_thm])

val _ = save_thm("Alice_npriv_verified_thm", Alice_npriv_verified_thm)

val Alice_justified_npriv_exec_thm =
TAC_PROOF(([],
``!(NS :state -> command trType -> state)
         (Out :state -> command trType -> output)
         (M :(command inst, 'b, staff, 'd, 'e) Kripke) (Oi :'d po)
         (Os :'e po) cmd npriv privcmd ins s outs.
         inputOK
          (Name Alice says
           (prop (SOME (NP npriv) :command inst) :
              (command inst, staff, 'd, 'e) Form)) /\
          CFGInterpret (M,Oi,Os)
          (CFG (inputOK :(command inst, staff, 'd, 'e) Form -> bool)
             (SM0StateInterp :state -> (command inst, staff, 'd, 'e) Form)
             (certs cmd npriv privcmd :
                (command inst, staff, 'd, 'e) Form list)
             (Name Alice says
              (prop (SOME (NP npriv) :command inst) :
                 (command inst, staff, 'd, 'e) Form)::ins) s outs) ==>
        TR (M,Oi,Os) (exec (NP (npriv :npriv)))
          (CFG (inputOK :(command inst, staff, 'd, 'e) Form -> bool)
             (SM0StateInterp :state -> (command inst, staff, 'd, 'e) Form)
             (certs (cmd :command) (npriv :npriv) privcmd :
                (command inst, staff, 'd, 'e) Form list)
```

```
            (Name Alice says
             (prop (SOME (NP npriv) :command inst) :
                (command inst, staff, 'd, 'e) Form)::
                 (ins :(command inst, staff, 'd, 'e) Form list)) (s :state)
            (outs :output list))
          (CFG (inputOK :(command inst, staff, 'd, 'e) Form -> bool)
            (SM0StateInterp :state -> (command inst, staff, 'd, 'e) Form)
            (certs cmd npriv privcmd :
               (command inst, staff, 'd, 'e) Form list) ins
            (NS s (exec (NP npriv)))
            (Out s (exec (NP npriv))::outs))``),
PROVE_TAC[Alice_exec_npriv_justified_thm, inputOK_def, Alice_npriv_lemma])

val _ = save_thm("Alice_justified_npriv_exec_thm",
    Alice_justified_npriv_exec_thm)


(* ------------------------------------------------------------------------
    *)
(* Exercise 17.4.3A
    *)
(* inputOK2 and certs2 defined to authenticate Carol only. Carol is
    *)
(* authorized solely on npriv commands, and trapped on privcmd.
    *)
(* ------------------------------------------------------------------------
    *)

(* Not placed here, I had put it in SM0Script.sml*)


(* ------------------------------------------------------------------------
    *)
(* Exercise 17.4.3 B
    *)
(* Carol can execute non-privileged commands using inputOK2 and certs2
    *)
(* ------------------------------------------------------------------------
    *)
val Carol_npriv_lemma
=
TAC_PROOF(
([],
``CFGInterpret ((M:(command inst,'b,staff,'d,'e)Kripke),Oi,Os)
  (CFG inputOK2 SM0StateInterp (certs2 cmd npriv privcmd)
   (((Name Carol) says (prop (SOME (NP (npriv:npriv)))))::ins)
   s (outs:output list)) ==>
  ((M,Oi,Os) sat (prop (SOME(NP npriv))))``),
  REWRITE_TAC [CFGInterpret_def, certs2_def, SM0StateInterp_def, satList_CONS,
       satList_nil, sat_TT] THEN
  PROVE_TAC[Controls])

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

```
val Carol_exec_npriv_justified_thm
=
    let val th1 =
    ISPECL
    [``inputOK2:(command inst, staff, 'd, 'e)Form -> bool``,
    ``(certs2 cmd npriv privcmd):(command inst, staff, 'd, 'e)Form list``,
    ``SM0StateIntern: state->(command inst, staff, 'd, 'e)Form``,
    ``Name Carol``, ``NP npriv``, ``ins:(command inst, staff, 'd, 'e)Form list
       ``,
    ``s: state``, ``outs:output list``]
    TR_exec_cmd_rule;
in
TAC_PROOF(
([],
      ``!(NS :state -> command trType -> state)
        (Out :state -> command trType -> output)
        (M :(command inst, 'b, staff, 'd, 'e) Kripke) (Oi :'d po)
        (Os :'e po).
       TR (M,Oi,Os) (exec (NP (npriv :npriv)))
         (CFG (inputOK2 :(command inst, staff, 'd, 'e) Form -> bool)
            (SM0StateInterp :state -> (command inst, staff, 'd, 'e) Form)
            (certs2 (cmd :command) (npriv :npriv) privcmd :
               (command inst, staff, 'd, 'e) Form list)
            (Name Carol says
             (prop (SOME (NP npriv) :command inst) :
                (command inst, staff, 'd, 'e) Form)::
                 (ins :(command inst, staff, 'd, 'e) Form list)) (s :state)
            (outs :output list))
         (CFG (inputOK2 :(command inst, staff, 'd, 'e) Form -> bool)
            (SM0StateInterp :state -> (command inst, staff, 'd, 'e) Form)
            (certs2 cmd npriv privcmd :
               (command inst, staff, 'd, 'e) Form list) ins
            (NS s (exec (NP npriv)))
            (Out s (exec (NP npriv))::outs)) <=>
       inputOK2
         (Name Carol says
          (prop (SOME (NP npriv) :command inst) :
             (command inst, staff, 'd, 'e) Form)) /\
       CFGInterpret (M,Oi,Os)
         (CFG (inputOK2 :(command inst, staff, 'd, 'e) Form -> bool)
            (SM0StateInterp :state -> (command inst, staff, 'd, 'e) Form)
            (certs2 cmd npriv privcmd :
               (command inst, staff, 'd, 'e) Form list)
            (Name Carol says
             (prop (SOME (NP npriv) :command inst) :
                (command inst, staff, 'd, 'e) Form)::ins) s outs) /\
       (M,Oi,Os) sat
       (prop (SOME (NP npriv) :command inst) :
          (command inst, staff, 'd, 'e) Form)``),
          PROVE_TAC[th1, Carol_npriv_lemma])
end
```

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

val Carol_npriv_verified_thm
= TAC_PROOF(
([],
       ``!(NS :state -> command trType -> state)
          (Out :state -> command trType -> output)
          (M :(command inst, 'b, staff, 'd, 'e) Kripke) (Oi :'d po)
          (Os :'e po).
        TR (M,Oi,Os) (exec (NP (npriv :npriv)))
          (CFG (inputOK2 :(command inst, staff, 'd, 'e) Form -> bool)
             (SM0StateInterp :state -> (command inst, staff, 'd, 'e) Form)
             (certs2 (cmd :command) (npriv :npriv) privcmd :
                (command inst, staff, 'd, 'e) Form list)
             (Name Carol says
              (prop (SOME (NP npriv) :command inst) :
                 (command inst, staff, 'd, 'e) Form)::
                  (ins :(command inst, staff, 'd, 'e) Form list)) (s :state)
             (outs :output list))
          (CFG (inputOK2 :(command inst, staff, 'd, 'e) Form -> bool)
             (SM0StateInterp :state -> (command inst, staff, 'd, 'e) Form)
             (certs2 cmd npriv privcmd :
                (command inst, staff, 'd, 'e) Form list) ins
             (NS s (exec (NP npriv)))
             (Out s (exec (NP npriv))::outs)) ==>
        (M,Oi,Os) sat
        (prop (SOME (NP npriv) :command inst) :
           (command inst, staff, 'd, 'e) Form)``,
         PROVE_TAC [Carol_exec_npriv_justified_thm])

val _ = save_thm("Carol_npriv_verified_thm", Carol_npriv_verified_thm)

val Carol_justified_npriv_exec_thm =
TAC_PROOF(
([],
       ``!(NS :state -> command trType -> state)
          (Out :state -> command trType -> output)
          (M :(command inst, 'b, staff, 'd, 'e) Kripke) (Oi :'d po)
          (Os :'e po) cmd npriv privcmd ins s outs.
          inputOK2
           (Name Carol says
            (prop (SOME (NP npriv) :command inst) :
               (command inst, staff, 'd, 'e) Form)) /\
           CFGInterpret (M,Oi,Os)
           (CFG (inputOK2 :(command inst, staff, 'd, 'e) Form -> bool)
              (SM0StateInterp :state -> (command inst, staff, 'd, 'e) Form)
              (certs2 cmd npriv privcmd :
                 (command inst, staff, 'd, 'e) Form list)
              (Name Carol says
               (prop (SOME (NP npriv) :command inst) :
                  (command inst, staff, 'd, 'e) Form)::ins) s outs) ==>
        TR (M,Oi,Os) (exec (NP (npriv :npriv)))
```

```
          (CFG (inputOK2 :(command inst, staff, 'd, 'e) Form -> bool)
             (SM0StateInterp :state -> (command inst, staff, 'd, 'e) Form)
             (certs2 (cmd :command) (npriv :npriv) privcmd :
                (command inst, staff, 'd, 'e) Form list)
             (Name Carol says
              (prop (SOME (NP npriv) :command inst) :
                 (command inst, staff, 'd, 'e) Form)::
                  (ins :(command inst, staff, 'd, 'e) Form list)) (s :state)
             (outs :output list))
          (CFG (inputOK2 :(command inst, staff, 'd, 'e) Form -> bool)
             (SM0StateInterp :state -> (command inst, staff, 'd, 'e) Form)
             (certs2 cmd npriv privcmd :
                (command inst, staff, 'd, 'e) Form list) ins
             (NS s (exec (NP npriv)))
             (Out s (exec (NP npriv))::outs))``),
PROVE_TAC[Carol_exec_npriv_justified_thm, inputOK2_def, Carol_npriv_lemma])

val _ = save_thm("Carol_justified_npriv_exec_thm",
    Carol_justified_npriv_exec_thm)

(* ------------------------------------------------------------------------
    *)
(* Exercise 17.4.3 C
    *)
(* Carol's request to execute a privileged command is trapped
    *)
(* ------------------------------------------------------------------------
    *)
val Carol_privcmd_trap_lemma
= TAC_PROOF(
([], ``CFGInterpret ((M:(command inst,'b,staff,'d,'e)Kripke),Oi,Os)
  (CFG inputOK2 SM0StateInterp (certs2 cmd npriv privcmd)
   (((Name Carol) says (prop (SOME (PR (privcmd:privcmd)))))::ins)
   s (outs:output list)) ==>
  ((M,Oi,Os) sat (prop NONE))``),
  REWRITE_TAC[CFGInterpret_def, certs2_def, SM0StateInterp_def, satList_CONS,
      satList_nil, sat_TT] THEN
  PROVE_TAC[Controls, TR_trap_cmd_rule, Modus_Ponens])

val _ = save_thm("Carol_privcmd_trap_lemma", Carol_privcmd_trap_lemma)

val Carol_trap_privcmd_justified_thm
=
let
val th1 =
    ISPECL
    [``inputOK2:(command inst, staff, 'd, 'e)Form->bool``,
    ``SM0StateInterp: state->(command inst, staff, 'd, 'e)Form``,
    ``(certs2 cmd npriv privcmd):(command inst, staff, 'd, 'e)Form list``,
    ``Name Carol``,``PR privcmd``, ``ins:(command inst, staff, 'd, 'e)Form
       list``,
    ``s:state``,``outs:output list``]
    TR_trap_cmd_rule
```

```
in
TAC_PROOF(
(
        [],
      ``!(NS :state -> command trType -> state)
        (Out :state -> command trType -> output)
        (M :(command inst, 'b, staff, 'd, 'e) Kripke) (Oi :'d po)
        (Os :'e po).
       TR (M,Oi,Os) (trap (PR (privcmd :privcmd)))
         (CFG (inputOK2 :(command inst, staff, 'd, 'e) Form -> bool)
            (SM0StateInterp :state -> (command inst, staff, 'd, 'e) Form)
            (certs2 (cmd :command) (npriv :npriv) privcmd :
               (command inst, staff, 'd, 'e) Form list)
            (Name Carol says
             (prop (SOME (PR privcmd) :command inst) :
                (command inst, staff, 'd, 'e) Form)::
                 (ins :(command inst, staff, 'd, 'e) Form list)) (s :state)
            (outs :output list))
         (CFG (inputOK2 :(command inst, staff, 'd, 'e) Form -> bool)
            (SM0StateInterp :state -> (command inst, staff, 'd, 'e) Form)
            (certs2 cmd npriv privcmd :
               (command inst, staff, 'd, 'e) Form list) ins
            (NS s (trap (PR privcmd)))
            (Out s (trap (PR privcmd))::outs)) <=>
       inputOK2
         (Name Carol says
          (prop (SOME (PR privcmd) :command inst) :
             (command inst, staff, 'd, 'e) Form)) /\
       CFGInterpret (M,Oi,Os)
         (CFG (inputOK2 :(command inst, staff, 'd, 'e) Form -> bool)
            (SM0StateInterp :state -> (command inst, staff, 'd, 'e) Form)
            (certs2 cmd npriv privcmd :
               (command inst, staff, 'd, 'e) Form list)
            (Name Carol says
             (prop (SOME (PR privcmd) :command inst) :
                (command inst, staff, 'd, 'e) Form)::ins) s outs) /\
       (M,Oi,Os) sat (prop NONE) : (command inst, staff, 'd, 'e) Form``),
       PROVE_TAC [th1, Carol_privcmd_trap_lemma])
end

val _ = save_thm("Carol_trap_privcmd_justified_thm",
    Carol_trap_privcmd_justified_thm)

val Carol_privcmd_trapped_thm
= TAC_PROOF(
([],
      ``!(NS :state -> command trType -> state)
        (Out :state -> command trType -> output)
        (M :(command inst, 'b, staff, 'd, 'e) Kripke) (Oi :'d po)
        (Os :'e po).
       TR (M,Oi,Os) (trap (PR (privcmd :privcmd)))
         (CFG (inputOK2 :(command inst, staff, 'd, 'e) Form -> bool)
            (SM0StateInterp :state -> (command inst, staff, 'd, 'e) Form)
```

```
            (certs2 (cmd :command) (npriv :npriv) privcmd :
               (command inst, staff, 'd, 'e) Form list)
            (Name Carol says
             (prop (SOME (PR privcmd) :command inst) :
                (command inst, staff, 'd, 'e) Form)::
                 (ins :(command inst, staff, 'd, 'e) Form list)) (s :state)
            (outs :output list))
          (CFG (inputOK2 :(command inst, staff, 'd, 'e) Form -> bool)
            (SM0StateInterp :state -> (command inst, staff, 'd, 'e) Form)
            (certs2 cmd npriv privcmd :
               (command inst, staff, 'd, 'e) Form list) ins
            (NS s (trap (PR privcmd)))
            (Out s (trap (PR privcmd))::outs)) ==>
        (M,Oi,Os) sat
        (prop NONE:
           (command inst, staff, 'd, 'e) Form)``),
          PROVE_TAC [Carol_trap_privcmd_justified_thm]
)

val _ = save_thm("Carol_privcmd_trapped_thm", Carol_privcmd_trapped_thm)

val Carol_justified_privcmd_trap_thm
=
TAC_PROOF(
([],
      ``!(NS :state -> command trType -> state)
        (Out :state -> command trType -> output)
        (M :(command inst, 'b, staff, 'd, 'e) Kripke) (Oi :'d po)
        (Os :'e po) cmd npriv privcmd ins s outs.
        inputOK2
         (Name Carol says
          (prop (SOME (PR privcmd) :command inst) :
             (command inst, staff, 'd, 'e) Form)) /\
         CFGInterpret (M,Oi,Os)
         (CFG (inputOK2 :(command inst, staff, 'd, 'e) Form -> bool)
            (SM0StateInterp :state -> (command inst, staff, 'd, 'e) Form)
            (certs2 cmd npriv privcmd :
               (command inst, staff, 'd, 'e) Form list)
            (Name Carol says
             (prop (SOME (PR privcmd) :command inst) :
                (command inst, staff, 'd, 'e) Form)::ins) s outs) ==>
        TR (M,Oi,Os) (trap (PR (privcmd :privcmd)))
          (CFG (inputOK2 :(command inst, staff, 'd, 'e) Form -> bool)
            (SM0StateInterp :state -> (command inst, staff, 'd, 'e) Form)
            (certs2 (cmd :command) (npriv :npriv) privcmd :
               (command inst, staff, 'd, 'e) Form list)
            (Name Carol says
             (prop (SOME (PR privcmd) :command inst) :
                (command inst, staff, 'd, 'e) Form)::
                 (ins :(command inst, staff, 'd, 'e) Form list)) (s :state)
            (outs :output list))
          (CFG (inputOK2 :(command inst, staff, 'd, 'e) Form -> bool)
            (SM0StateInterp :state -> (command inst, staff, 'd, 'e) Form)
```

```
            (certs2 cmd npriv privcmd :
               (command inst, staff, 'd, 'e) Form list) ins
            (NS s (trap (PR privcmd)))
            (Out s (trap (PR privcmd))::outs))``),
            PROVE_TAC[Carol_trap_privcmd_justified_thm, inputOK2_def,
               Carol_privcmd_trap_lemma])

val _ = save_thm("Carol_justified_privcmd_trap_thm",
   Carol_justified_privcmd_trap_thm)


val _ = export_theory();
val _ = print_theory "-"


end (* structure *)
```