Project 9 Report

Alfred Murabito

March 20, 2020

Abstract

This report details my results for Exercises 17.4.1 and 17.4.3 from *Certified Security by Design Using Higher Order Logic*. In these exercises, we use HOL to define secure state machines using configurations defined in SMOTHeory. These state machines are completely mediated, commands are always checked for authenticity and authorization before they are executed.

Acknowledgments: I received no assistance with this exercise.

Contents

1 Executive Summary **5**

2 Exercise 17.4.1 **6**
2.1 Problem Statement 6
2.2 Relevant Code 7
2.3 Part A: Alice npriv lemma 7
2.4 Part B: Alice exec npriv justified thm 7
2.5 Part C: Alice nrpiv verified thm 8
2.6 Part D: Alice justified npriv exec thm 9
2.7 Execution Transcript 10

3 Exercise 17.4.3 **12**
3.1 Problem Statement 12
3.2 Relevant Code 12
3.3 Part A: Carols's Next State/Output 12
3.4 Part B: Execute Carols's Nonprivileged Command 13
3.5 Part C: Trap Carol's Privileged Command 16
3.6 Execution Transcript 19

4 Appendix A: SM0Script.sml **24**

5 Appendix B: SM0SoluitionsScript.sml **31**

1 Executive Summary

All requirements for this assignment have been satisfied. The results of each exercise are detailed below. The state machines are defined and the SML code used produces theorems that exactly match those described in the course textbook. Additionally, Holmake completes successfully and the theorems proved appear in the pretty-printed report created by emitTeX.

2 Exercise 17.4.1

2.1 Problem Statement

In this exercise, we formalize theorems involving Alice executing nonprivileged commands. In particular we prove theorems that if Alice makes a non-privileged command that is accepted by *inputOK* and verified for authorization by *certs* that it will be executed. The following datatypes and definitons were used from *SM0Theory*

```
command = NP npriv | PR privcmd

npriv = status

output = on | off

privcmd = launch | reset

staff = Alice | Bob | Carol

state = STBY | ACTIVE
```

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

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

2.2 Relevant Code

2.3 Part A: Alice npriv lemma

Theorem:

```
⊢ 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))
```

Proof:

```
val Alice_npriv_lemma =
TAC_PROOF(
([], ``CFGInterpret ((M:(command inst,'b,staff,'d,'e)Kripke),Oi,Os)
  (CFG inputOK SMOStateInterp (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,SMOStateInterp_def,satList_CONS,satList_nil,
sat_TT] THEN
PROVE_TAC[Controls])
```

2.4 Part B: Alice exec npriv justified thm

Theorem:

```
⊢ ∀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))
```

Proof:

```
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``,
   ``SMOStateInterp: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;
```

2.5 Part C: Alice nrpiv verified thm

Theorem:

$\vdash \forall NS\ Out\ M\ Oi\ Os.$
  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 $npriv$))::$outs$)) $\Rightarrow$
  (M,Oi,Os) sat prop (SOME (NP $npriv$))

Proof:

```
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)
             (SMOStateInterp :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)
             (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)``),
PROVE_TAC[Alice_exec_npriv_justified_thm])
```

2.6 Part D: Alice justified npriv exec thm

Theorem:

```
 ⊢ ∀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))
```

Proof:

```
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])
```

2.7 Execution Transcript

```
        HOL-4 [Kananaskis 11 (stdknl, built Sat Aug 19 09:30:06 2017)]

        For introductory HOL help, type: help "hol";
        To exit type <Control>-D

[extending loadPath with Holmakefile INCLUDES variable]
> > > > > Loading ssm1Theory
> Loading SM0Theory
> Loading acl_infRules

> # # # # # # # # # # Meson search level: ....
val Alice_npriv_lemma =
   |- 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)):
   thm
>
*** Emacs/HOL command completed ***

> Meson search level: .....................................
val Alice_exec_npriv_justified_thm =
   |-    N S  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)):
   thm
val it = (): unit
>
*** Emacs/HOL command completed ***

> Meson search level: ...
val Alice_npriv_verified_thm =
   |-    N S  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))
      (M,Oi,Os) sat prop (SOME (NP npriv)):
   thm
val it = (): unit
>
*** Emacs/HOL command completed ***

> Meson search level: ......
val Alice_justified_npriv_exec_thm =
   |-    N S  Out M Oi  Os cmd npriv privcmd ins s 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)
      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)):
   thm
val it = (): unit
>
*** Emacs/HOL command completed ***

>
```

3 Exercise 17.4.3

3.1 Problem Statement

Part A of the exercises involves defining new functions that only validate Carol and that permit her to execute nonprivileged commands but trap privileged commands.The following parts B, and C involve proving theorems for Carol on privileged and nonprivileged commands.

- *inputOK2*
- *certs2*

3.2 Relevant Code

3.3 Part A: Carols's Next State/Output

Definitions:

```
⊢ (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)

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

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):(command inst,staff,’d,’e)Form list =
 [Name Carol controls prop (SOME (NP npriv));
  ((Name Carol) says (prop (SOME (PR privcmd)))) impf (prop NONE)]‘
```

3.4 Part B: Execute Carols’s Nonprivileged Command

Theorems:

```
 ⊢ 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))

 ⊢ ∀ 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))

 ⊢ ∀ 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))

 ⊢ ∀ 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))
```

Proofs:

```
val Carol_npriv_lemma =
TAC_PROOF(
([], ``CFGInterpret ((M:(command inst,'b,staff,'d,'e)Kripke),Oi,Os)
  (CFG inputOK2 SMOStateInterp (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,SMOStateInterp_def,satList_CONS,satList_nil,
sat_TT] THEN
PROVE_TAC[Controls])


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``,
   ``SMOStateInterp: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)
             (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 (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)``),
  PROVE_TAC[th1,Carol_npriv_lemma])
end;


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 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)
            (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 (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))``),
PROVE_TAC[Carol_exec_npriv_justified_thm,inputOK2_def,Carol_npriv_lemma])
```

3.5 Part C: Trap Carol's Privileged Command

Theorems:

```
⊢ 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

⊢ ∀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

⊢ ∀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
```

```
⊢ ∀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))
```

Proofs:

```
val Carol_privcmd_trap_lemma =
TAC_PROOF(
([], ``CFGInterpret ((M:(command inst,'b,staff,'d,'e)Kripke),Oi,Os)
  (CFG inputOK2 SMOStateInterp (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,SMOStateInterp_def,satList_CONS,satList_nil,
sat_TT] THEN
PROVE_TAC[Modus_Ponens])


val Carol_trap_privcmd_justified_thm =
let
  val th1 =
  ISPECL
  [``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``,``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)
             (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, staff, 'd, 'e) Form``),

  PROVE_TAC[th1,Carol_privcmd_trap_lemma])
end;


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)
             (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, staff, 'd, 'e) Form)``),
PROVE_TAC[Carol_trap_privcmd_justified_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])
```

3.6 Execution Transcript

```
---------------------------------------------------------------------
       HOL-4 [Kananaskis 11 (stdknl, built Sat Aug 19 09:30:06 2017)]

       For introductory HOL help, type: help "hol";
       To exit type <Control>-D
---------------------------------------------------------------------
[extending loadPath with Holmakefile INCLUDES variable]
> > > > Loading ssm1Theory
> Loading SM0Theory
> Loading acl_infRules
> # # # # # # <<HOL message: mk_functional:
  pattern completion has added 40 clauses to the original specification.>>
Equations stored under "inputOK2_def".
Induction stored under "inputOK2_ind".
```

```
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
> #### Definition has been stored under "certs2_def"
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
> ########## Meson search level: ....
val Carol_npriv_lemma =
   |- CFGInterpret (M,Oi,Os)
     (CFG inputOK2 SM0StateInterp (certs2 cmd npriv privcmd)
        (Name Carol says prop (SOME (NP npriv))::ins) s outs)
   (M,Oi,Os) sat prop (SOME (NP npriv)):
```

```
   thm
>
*** Emacs/HOL command completed ***

> Meson search level: .....................................
val Carol_exec_npriv_justified_thm =
   |-   N S  Out M Oi Os.
      TR (M,Oi,Os) (exec (NP npriv))
         (CFG inputOK2 SM0StateInterp (certs2 cmd npriv privcmd)
            (Name Carol says prop (SOME (NP npriv))::ins) s outs)
         (CFG inputOK2 SM0StateInterp (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 SM0StateInterp (certs2 cmd npriv privcmd)
            (Name Carol says prop (SOME (NP npriv))::ins) s outs)
      (M,Oi,Os) sat prop (SOME (NP npriv)):
   thm
val it = (): unit
>
*** Emacs/HOL command completed ***

> Meson search level: ...
val Carol_npriv_verified_thm =
   |-   N S  Out M Oi Os.
      TR (M,Oi,Os) (exec (NP npriv))
         (CFG inputOK2 SM0StateInterp (certs2 cmd npriv privcmd)
            (Name Carol says prop (SOME (NP npriv))::ins) s outs)
         (CFG inputOK2 SM0StateInterp (certs2 cmd npriv privcmd) ins
            (NS s (exec (NP npriv))) (Out s (exec (NP npriv))::outs))
      (M,Oi,Os) sat prop (SOME (NP npriv)):
   thm
val it = (): unit
>
*** Emacs/HOL command completed ***

> Meson search level: ......
val Carol_justified_npriv_exec_thm =
   |-   N S  Out M Oi Os cmd npriv privcmd ins s outs.
      inputOK2 (Name Carol says prop (SOME (NP npriv)))
      CFGInterpret (M,Oi,Os)
         (CFG inputOK2 SM0StateInterp (certs2 cmd npriv privcmd)
            (Name Carol says prop (SOME (NP npriv))::ins) s outs)
      TR (M,Oi,Os) (exec (NP npriv))
         (CFG inputOK2 SM0StateInterp (certs2 cmd npriv privcmd)
            (Name Carol says prop (SOME (NP npriv))::ins) s outs)
         (CFG inputOK2 SM0StateInterp (certs2 cmd npriv privcmd) ins
            (NS s (exec (NP npriv))) (Out s (exec (NP npriv))::outs)):
   thm
val it = (): unit
>
*** Emacs/HOL command completed ***
```

```
> # # # # # # # # # # Meson search level: ....
val Carol_privcmd_trap_lemma =
   |- CFGInterpret (M,Oi,Os)
      (CFG inputOK2 SM0StateInterp (certs2 cmd npriv privcmd)
         (Name Carol says prop (SOME (PR privcmd))::ins) s outs)
   (M,Oi,Os) sat prop NONE:
   thm
>
*** Emacs/HOL command completed ***

> Meson search level: .....................................
val Carol_trap_privcmd_justified_thm =
   |-   NS Out M Oi Os.
      TR (M,Oi,Os) (trap (PR privcmd))
        (CFG inputOK2 SM0StateInterp (certs2 cmd npriv privcmd)
           (Name Carol says prop (SOME (PR privcmd))::ins) s outs)
        (CFG inputOK2 SM0StateInterp (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 SM0StateInterp (certs2 cmd npriv privcmd)
           (Name Carol says prop (SOME (PR privcmd))::ins) s outs)
      (M,Oi,Os) sat prop NONE:
   thm
val it = (): unit
>
*** Emacs/HOL command completed ***

> Meson search level: ...
val Carol_privcmd_trapped_thm =
   |-   NS Out M Oi Os.
      TR (M,Oi,Os) (trap (PR privcmd))
        (CFG inputOK2 SM0StateInterp (certs2 cmd npriv privcmd)
           (Name Carol says prop (SOME (PR privcmd))::ins) s outs)
        (CFG inputOK2 SM0StateInterp (certs2 cmd npriv privcmd) ins
           (NS s (trap (PR privcmd)))
           (Out s (trap (PR privcmd))::outs))
      (M,Oi,Os) sat prop NONE:
   thm
val it = (): unit
>
*** Emacs/HOL command completed ***

> Meson search level: ......
val 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 SM0StateInterp (certs2 cmd npriv privcmd)
           (Name Carol says prop (SOME (PR privcmd))::ins) s outs)
      TR (M,Oi,Os) (trap (PR privcmd))
        (CFG inputOK2 SM0StateInterp (certs2 cmd npriv privcmd)
           (Name Carol says prop (SOME (PR privcmd)):: ins) s outs)
        (CFG inputOK2 SM0StateInterp (certs2 cmd npriv privcmd) ins
           (NS s (trap (PR privcmd))) (Out s (trap (PR privcmd)):: outs)):
   thm
val it = (): unit
>
*** Emacs/HOL command completed ***

>
```

4 Appendix A: SM0Script.sml

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

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

5 Appendix B: SM0SoluitionsScript.sml

```
(**********************************************************************)
(* Solutions for Exercises 17.4.1 and 17.4.3                          *)
(* Author: Alfred Murabito                                            *)
(* Date: 15 March 2020                                                *)
(**********************************************************************)

structure SM0Solutions = struct

(* Interactive mode
app load ["ssm1Theory","SM0Theory","acl_infRules","aclrulesTheory",
    "aclDrulesTheory","satListTheory","SM0SolutionsTheory"]
open ssm1Theory SM0Theory acl_infRules aclrulesTheory
     aclDrulesTheory satListTheory SM0SolutionsTheory
*)
open HolKernel Parse boolLib bossLib;
open ssm1Theory SM0Theory acl_infRules aclrulesTheory
     aclDrulesTheory satListTheory

val _ = new_theory "SM0Solutions";


(* ------------------------------------------------------------------ *)
(* Exercise 17.4.1                                                    *)
(* Alice's non-privileged commands are executed and justified         *)
(* ------------------------------------------------------------------ *)

(* Alice_npriv_lemma

set_goal
([], ``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 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)

(* Alice_exec_npriv_justified_thm
set_goal
([], ``!(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)``)
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
PROVE_TAC[th1,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)

(*Alice_npriv_verified_thm

set_goal
([], ``!(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 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)
```

```
(* Alice_justified_npriv_exec_thm

set_goal
([], ``!(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 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.                *)
(* ---------------------------------------------------------------------------- *)
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):(command inst,staff,'d,'e)Form list =
 [Name Carol controls prop (SOME (NP npriv));
  ((Name Carol) says (prop (SOME (PR privcmd)))) impf (prop NONE)]`

(* ---------------------------------------------------------------------------- *)
(* Exercise 17.4.3 B                                                            *)
(* Carol can execute non-privileged commands using inputOK2 and certs2         *)
(* ---------------------------------------------------------------------------- *)

(* Carol_npriv_lemma

set_goal
([], ``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 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)

(* Carol_exec_npriv_justified_thm
set_goal
([], ``!(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)``)
val th1 =
ISPECL
[``inputOK2 :(command inst, staff, 'd, 'e)Form -> bool``,
 ``(certs2 cmd npriv privcmd):(command inst, staff, 'd, 'e)Form list``,
 ``SM0StateInterp: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
PROVE_TAC[th1,Carol_priv_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``,
   ``SM0StateInterp: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)

(* Carol_npriv_verified_thm
set_goal
([], ``!(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 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)

(* Carol_justified_npriv_exec_thm

set_goal
([], ``!(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 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                 *)
(* -------------------------------------------------------------------------- *)
(* Carol_privcmd_trap_lemma

set_goal
([], ``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[Modus_Ponens]
*)

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[Modus_Ponens])

val _ = save_thm("Carol_privcmd_trap_lemma",Carol_privcmd_trap_lemma)

(* Carol_trap_privcmd_justified_thm

set_goal
([], ``!(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``)
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
PROVE_TAC[th1,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)

(* Carol_privcmd_trapped_thm

set_goal
([], ``!(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 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)

(* Carol_justified_privcmd_trap_thm
set_goal
([], ``!(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 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 *)
```