Contents

1 OMNIType Theory

Built: 10 June 2018

Parent Theories: indexedLists, patternMatches

1.1 Datatypes

```
command = ESCc escCommand | SLc 'slCommand

escCommand = returnToBase | changeMission | resupply
           | reactToContact

escOutput = ReturnToBase | ChangeMission | Resupply
          | ReactToContact

escState = RTB | CM | RESUPPLY | RTC

output = ESCo escOutput | SLo 'slOutput

principal = SR 'stateRole

state = ESCs escState | SLs 'slState
```

1.2 Theorems

[command_distinct_clauses]

$\vdash \forall a'\ a.\ \texttt{ESCc}\ a \neq \texttt{SLc}\ a'$

[command_one_one]

$\vdash (\forall a\ a'.\ (\texttt{ESCc}\ a = \texttt{ESCc}\ a') \iff (a = a')) \land$
$\quad \forall a\ a'.\ (\texttt{SLc}\ a = \texttt{SLc}\ a') \iff (a = a')$

[escCommand_distinct_clauses]

$\vdash \texttt{returnToBase} \neq \texttt{changeMission} \land \texttt{returnToBase} \neq \texttt{resupply} \land$
$\quad \texttt{returnToBase} \neq \texttt{reactToContact} \land \texttt{changeMission} \neq \texttt{resupply} \land$
$\quad \texttt{changeMission} \neq \texttt{reactToContact} \land \texttt{resupply} \neq \texttt{reactToContact}$

[escOutput_distinct_clauses]

$\vdash \texttt{ReturnToBase} \neq \texttt{ChangeMission} \land \texttt{ReturnToBase} \neq \texttt{Resupply} \land$
$\quad \texttt{ReturnToBase} \neq \texttt{ReactToContact} \land \texttt{ChangeMission} \neq \texttt{Resupply} \land$
$\quad \texttt{ChangeMission} \neq \texttt{ReactToContact} \land \texttt{Resupply} \neq \texttt{ReactToContact}$

[escState_distinct_clauses]

$\vdash \texttt{RTB} \neq \texttt{CM} \land \texttt{RTB} \neq \texttt{RESUPPLY} \land \texttt{RTB} \neq \texttt{RTC} \land \texttt{CM} \neq \texttt{RESUPPLY} \land$
$\quad \texttt{CM} \neq \texttt{RTC} \land \texttt{RESUPPLY} \neq \texttt{RTC}$

[output_distinct_clauses]

$\vdash \forall a'\ a.\ \texttt{ESCo}\ a \neq \texttt{SLo}\ a'$

[output_one_one]

$\vdash (\forall a\ a'.\ (\texttt{ESCo}\ a = \texttt{ESCo}\ a') \iff (a = a')) \land$
$\quad \forall a\ a'.\ (\texttt{SLo}\ a = \texttt{SLo}\ a') \iff (a = a')$

[principal_one_one]

$\vdash \forall a\ a'.\ (\texttt{SR}\ a = \texttt{SR}\ a') \iff (a = a')$

[state_distinct_clauses]

$\vdash \forall a'\ a.\ \texttt{ESCs}\ a \neq \texttt{SLs}\ a'$

[state_one_one]

$\vdash (\forall a\ a'.\ (\texttt{ESCs}\ a = \texttt{ESCs}\ a') \iff (a = a')) \land$
$\quad \forall a\ a'.\ (\texttt{SLs}\ a = \texttt{SLs}\ a') \iff (a = a')$

2 ssm11 Theory

Built: 10 June 2018
Parent Theories: satList

2.1 Datatypes

```
configuration =
    CFG (('command order, 'principal, 'd, 'e) Form -> bool)
        ('state -> ('command order, 'principal, 'd, 'e) Form)
        (('command order, 'principal, 'd, 'e) Form list)
        (('command order, 'principal, 'd, 'e) Form list) 'state
        ('output list)

order = SOME 'command | NONE

trType = discard 'command | trap 'command | exec 'command
```

2.2 Definitions

[TR_def]

$\vdash \texttt{TR} =$
$\quad (\lambda a_0\ a_1\ a_2\ a_3.$
$\qquad \forall TR'.$
$\qquad\quad (\forall a_0\ a_1\ a_2\ a_3.$
$\qquad\qquad (\exists authenticationTest\ P\ NS\ M\ Oi\ Os\ Out\ s$
$\qquad\qquad\quad securityContext\ stateInterp\ cmd\ ins\ outs.$
$\qquad\qquad\ (a_0 = (M, Oi, Os)) \land (a_1 = \texttt{exec}\ cmd) \land$
$\qquad\qquad\ (a_2 =$

```
          CFG authenticationTest stateInterp
            securityContext (P says prop (SOME cmd)::ins) s
            outs) ∧
         (a3 =
          CFG authenticationTest stateInterp
            securityContext ins (NS s (exec cmd))
            (Out s (exec cmd)::outs)) ∧
         authenticationTest (P says prop (SOME cmd)) ∧
         CFGInterpret (M,Oi,Os)
           (CFG authenticationTest stateInterp
              securityContext (P says prop (SOME cmd)::ins)
              s outs)) ∨
      (∃ authenticationTest P NS M Oi Os Out s
          securityContext stateInterp cmd ins outs.
         (a0 = (M,Oi,Os)) ∧ (a1 = trap cmd) ∧
         (a2 =
          CFG authenticationTest stateInterp
            securityContext (P says prop (SOME cmd)::ins) s
            outs) ∧
         (a3 =
          CFG authenticationTest stateInterp
            securityContext ins (NS s (trap cmd))
            (Out s (trap cmd)::outs)) ∧
         authenticationTest (P says prop (SOME cmd)) ∧
         CFGInterpret (M,Oi,Os)
           (CFG authenticationTest stateInterp
              securityContext (P says prop (SOME cmd)::ins)
              s outs)) ∨
      (∃ authenticationTest NS M Oi Os Out s securityContext
          stateInterp cmd x ins outs.
         (a0 = (M,Oi,Os)) ∧ (a1 = discard cmd) ∧
         (a2 =
          CFG authenticationTest stateInterp
            securityContext (x::ins) s outs) ∧
         (a3 =
          CFG authenticationTest stateInterp
            securityContext ins (NS s (discard cmd))
            (Out s (discard cmd)::outs)) ∧
         ¬authenticationTest x) ⇒
      TR' a0 a1 a2 a3) ⇒
   TR' a0 a1 a2 a3)
```

2.3 Theorems

[CFGInterpret_def]

```
 ⊢ CFGInterpret (M,Oi,Os)
     (CFG authenticationTest stateInterp securityContext
        (input::ins) state outputStream) ⇔
```

```
  (M,Oi,Os) satList securityContext ∧ (M,Oi,Os) sat input ∧
  (M,Oi,Os) sat stateInterp state
```

[CFGInterpret_ind]

```
⊢ ∀ P.
    (∀ M Oi Os authenticationTest stateInterp securityContext
         input ins state outputStream.
       P (M,Oi,Os)
         (CFG authenticationTest stateInterp securityContext
            (input::ins) state outputStream)) ∧
    (∀ v15 v10 v11 v12 v13 v14.
       P v15 (CFG v10 v11 v12 [] v13 v14)) ⇒
    ∀ v v1 v2 v3. P (v,v1,v2) v3
```

[configuration_one_one]

```
⊢ ∀ a0 a1 a2 a3 a4 a5 a0' a1' a2' a3' a4' a5'.
    (CFG a0 a1 a2 a3 a4 a5 = CFG a0' a1' a2' a3' a4' a5') ⇔
    (a0 = a0') ∧ (a1 = a1') ∧ (a2 = a2') ∧ (a3 = a3') ∧
    (a4 = a4') ∧ (a5 = a5')
```

[order_distinct_clauses]

```
⊢ ∀ a. SOME a ≠ NONE
```

[order_one_one]

```
⊢ ∀ a a'. (SOME a = SOME a') ⇔ (a = a')
```

[TR_cases]

```
⊢ ∀ a0 a1 a2 a3.
    TR a0 a1 a2 a3 ⇔
    (∃ authenticationTest P NS M Oi Os Out s securityContext
         stateInterp cmd ins outs.
       (a0 = (M,Oi,Os)) ∧ (a1 = exec cmd) ∧
       (a2 =
        CFG authenticationTest stateInterp securityContext
          (P says prop (SOME cmd)::ins) s outs) ∧
       (a3 =
        CFG authenticationTest stateInterp securityContext ins
          (NS s (exec cmd)) (Out s (exec cmd)::outs)) ∧
       authenticationTest (P says prop (SOME cmd)) ∧
       CFGInterpret (M,Oi,Os)
         (CFG authenticationTest stateInterp securityContext
            (P says prop (SOME cmd)::ins) s outs)) ∨
    (∃ authenticationTest P NS M Oi Os Out s securityContext
         stateInterp cmd ins outs.
       (a0 = (M,Oi,Os)) ∧ (a1 = trap cmd) ∧
       (a2 =
        CFG authenticationTest stateInterp securityContext
          (P says prop (SOME cmd)::ins) s outs) ∧
        (a₃ =
         CFG authenticationTest stateInterp securityContext ins
           (NS s (trap cmd)) (Out s (trap cmd)::outs)) ∧
        authenticationTest (P says prop (SOME cmd)) ∧
        CFGInterpret (M,Oi,Os)
          (CFG authenticationTest stateInterp securityContext
             (P says prop (SOME cmd)::ins) s outs)) ∨
     ∃ authenticationTest NS M Oi Os Out s securityContext
        stateInterp cmd x ins outs.
       (a₀ = (M,Oi,Os)) ∧ (a₁ = discard cmd) ∧
       (a₂ =
        CFG authenticationTest stateInterp securityContext
          (x::ins) s outs) ∧
       (a₃ =
        CFG authenticationTest stateInterp securityContext ins
          (NS s (discard cmd)) (Out s (discard cmd)::outs)) ∧
       ¬authenticationTest x
```

[TR_discard_cmd_rule]

```
 ⊢ TR (M,Oi,Os) (discard cmd)
      (CFG authenticationTest stateInterp securityContext
         (x::ins) s outs)
      (CFG authenticationTest stateInterp securityContext ins
         (NS s (discard cmd)) (Out s (discard cmd)::outs)) ⇔
   ¬authenticationTest x
```

[TR_EQ_rules_thm]

```
 ⊢ (TR (M,Oi,Os) (exec cmd)
       (CFG authenticationTest stateInterp securityContext
          (P says prop (SOME cmd)::ins) s outs)
       (CFG authenticationTest stateInterp securityContext ins
          (NS s (exec cmd)) (Out s (exec cmd)::outs)) ⇔
    authenticationTest (P says prop (SOME cmd)) ∧
    CFGInterpret (M,Oi,Os)
       (CFG authenticationTest stateInterp securityContext
          (P says prop (SOME cmd)::ins) s outs)) ∧
   (TR (M,Oi,Os) (trap cmd)
       (CFG authenticationTest stateInterp securityContext
          (P says prop (SOME cmd)::ins) s outs)
       (CFG authenticationTest stateInterp securityContext ins
          (NS s (trap cmd)) (Out s (trap cmd)::outs)) ⇔
    authenticationTest (P says prop (SOME cmd)) ∧
    CFGInterpret (M,Oi,Os)
       (CFG authenticationTest stateInterp securityContext
          (P says prop (SOME cmd)::ins) s outs)) ∧
   (TR (M,Oi,Os) (discard cmd)
       (CFG authenticationTest stateInterp securityContext
          (x::ins) s outs)
       (CFG authenticationTest stateInterp securityContext ins
      (NS s (discard cmd)) (Out s (discard cmd)::outs)) ⟺
   ¬authenticationTest x)
```

[TR_exec_cmd_rule]

```
 ⊢ ∀ authenticationTest securityContext stateInterp P cmd ins s
     outs.
    (∀ M Oi Os.
       CFGInterpret (M,Oi,Os)
         (CFG authenticationTest stateInterp securityContext
            (P says prop (SOME cmd)::ins) s outs) ⇒
       (M,Oi,Os) sat prop (SOME cmd)) ⇒
    ∀ NS Out M Oi Os.
      TR (M,Oi,Os) (exec cmd)
        (CFG authenticationTest stateInterp securityContext
           (P says prop (SOME cmd)::ins) s outs)
        (CFG authenticationTest stateInterp securityContext ins
           (NS s (exec cmd)) (Out s (exec cmd)::outs)) ⟺
      authenticationTest (P says prop (SOME cmd)) ∧
      CFGInterpret (M,Oi,Os)
        (CFG authenticationTest stateInterp securityContext
           (P says prop (SOME cmd)::ins) s outs) ∧
      (M,Oi,Os) sat prop (SOME cmd)
```

[TR_ind]

```
 ⊢ ∀ TR′.
    (∀ authenticationTest P NS M Oi Os Out s securityContext
        stateInterp cmd ins outs.
       authenticationTest (P says prop (SOME cmd)) ∧
       CFGInterpret (M,Oi,Os)
         (CFG authenticationTest stateInterp securityContext
            (P says prop (SOME cmd)::ins) s outs) ⇒
       TR′ (M,Oi,Os) (exec cmd)
         (CFG authenticationTest stateInterp securityContext
            (P says prop (SOME cmd)::ins) s outs)
         (CFG authenticationTest stateInterp securityContext
            ins (NS s (exec cmd)) (Out s (exec cmd)::outs))) ∧
    (∀ authenticationTest P NS M Oi Os Out s securityContext
        stateInterp cmd ins outs.
       authenticationTest (P says prop (SOME cmd)) ∧
       CFGInterpret (M,Oi,Os)
         (CFG authenticationTest stateInterp securityContext
            (P says prop (SOME cmd)::ins) s outs) ⇒
       TR′ (M,Oi,Os) (trap cmd)
         (CFG authenticationTest stateInterp securityContext
            (P says prop (SOME cmd)::ins) s outs)
         (CFG authenticationTest stateInterp securityContext
            ins (NS s (trap cmd)) (Out s (trap cmd)::outs))) ∧
    (∀ authenticationTest NS M Oi Os Out s securityContext
        stateInterp cmd x ins outs.
        ¬authenticationTest x ⇒
        TR′ (M,Oi,Os) (discard cmd)
          (CFG authenticationTest stateInterp securityContext
             (x::ins) s outs)
          (CFG authenticationTest stateInterp securityContext
             ins (NS s (discard cmd))
             (Out s (discard cmd)::outs))) ⇒
    ∀ a0 a1 a2 a3. TR a0 a1 a2 a3 ⇒ TR′ a0 a1 a2 a3
```

[TR_rules]

```
⊢ (∀ authenticationTest P NS M Oi Os Out s securityContext
       stateInterp cmd ins outs.
      authenticationTest (P says prop (SOME cmd)) ∧
      CFGInterpret (M,Oi,Os)
        (CFG authenticationTest stateInterp securityContext
           (P says prop (SOME cmd)::ins) s outs) ⇒
      TR (M,Oi,Os) (exec cmd)
        (CFG authenticationTest stateInterp securityContext
           (P says prop (SOME cmd)::ins) s outs)
        (CFG authenticationTest stateInterp securityContext ins
           (NS s (exec cmd)) (Out s (exec cmd)::outs))) ∧
  (∀ authenticationTest P NS M Oi Os Out s securityContext
       stateInterp cmd ins outs.
      authenticationTest (P says prop (SOME cmd)) ∧
      CFGInterpret (M,Oi,Os)
        (CFG authenticationTest stateInterp securityContext
           (P says prop (SOME cmd)::ins) s outs) ⇒
      TR (M,Oi,Os) (trap cmd)
        (CFG authenticationTest stateInterp securityContext
           (P says prop (SOME cmd)::ins) s outs)
        (CFG authenticationTest stateInterp securityContext ins
           (NS s (trap cmd)) (Out s (trap cmd)::outs))) ∧
  ∀ authenticationTest NS M Oi Os Out s securityContext
      stateInterp cmd x ins outs.
    ¬authenticationTest x ⇒
    TR (M,Oi,Os) (discard cmd)
      (CFG authenticationTest stateInterp securityContext
         (x::ins) s outs)
      (CFG authenticationTest stateInterp securityContext ins
         (NS s (discard cmd)) (Out s (discard cmd)::outs))
```

[TR_strongind]

```
⊢ ∀ TR′.
    (∀ authenticationTest P NS M Oi Os Out s securityContext
         stateInterp cmd ins outs.
        authenticationTest (P says prop (SOME cmd)) ∧
        CFGInterpret (M,Oi,Os)
          (CFG authenticationTest stateInterp securityContext
             (P says prop (SOME cmd)::ins) s outs) ⇒
```

```
      TR′ (M,Oi,Os) (exec cmd)
        (CFG authenticationTest stateInterp securityContext
           (P says prop (SOME cmd)::ins) s outs)
        (CFG authenticationTest stateInterp securityContext
           ins (NS s (exec cmd)) (Out s (exec cmd)::outs))) ∧
   (∀ authenticationTest P NS M Oi Os Out s securityContext
       stateInterp cmd ins outs.
      authenticationTest (P says prop (SOME cmd)) ∧
      CFGInterpret (M,Oi,Os)
        (CFG authenticationTest stateInterp securityContext
           (P says prop (SOME cmd)::ins) s outs) ⇒
      TR′ (M,Oi,Os) (trap cmd)
        (CFG authenticationTest stateInterp securityContext
           (P says prop (SOME cmd)::ins) s outs)
        (CFG authenticationTest stateInterp securityContext
           ins (NS s (trap cmd)) (Out s (trap cmd)::outs))) ∧
   (∀ authenticationTest NS M Oi Os Out s securityContext
       stateInterp cmd x ins outs.
      ¬authenticationTest x ⇒
      TR′ (M,Oi,Os) (discard cmd)
        (CFG authenticationTest stateInterp securityContext
           (x::ins) s outs)
        (CFG authenticationTest stateInterp securityContext
           ins (NS s (discard cmd))
           (Out s (discard cmd)::outs))) ⇒
   ∀ a0 a1 a2 a3. TR a0 a1 a2 a3 ⇒ TR′ a0 a1 a2 a3
```

[TR_trap_cmd_rule]

```
 ⊢ ∀ authenticationTest stateInterp securityContext P cmd ins s
     outs.
   (∀ M Oi Os.
      CFGInterpret (M,Oi,Os)
        (CFG authenticationTest stateInterp securityContext
           (P says prop (SOME cmd)::ins) s outs) ⇒
      (M,Oi,Os) sat prop NONE) ⇒
   ∀ NS Out M Oi Os.
     TR (M,Oi,Os) (trap cmd)
       (CFG authenticationTest stateInterp securityContext
          (P says prop (SOME cmd)::ins) s outs)
       (CFG authenticationTest stateInterp securityContext ins
          (NS s (trap cmd)) (Out s (trap cmd)::outs)) ⇔
     authenticationTest (P says prop (SOME cmd)) ∧
     CFGInterpret (M,Oi,Os)
       (CFG authenticationTest stateInterp securityContext
          (P says prop (SOME cmd)::ins) s outs) ∧
     (M,Oi,Os) sat prop NONE
```

[TRrule0]

```
 ⊢ TR (M,Oi,Os) (exec cmd)
   (CFG authenticationTest stateInterp securityContext
      (P says prop (SOME cmd)::ins) s outs)
    (CFG authenticationTest stateInterp securityContext ins
      (NS s (exec cmd)) (Out s (exec cmd)::outs)) ⟺
  authenticationTest (P says prop (SOME cmd)) ∧
  CFGInterpret (M,Oi,Os)
    (CFG authenticationTest stateInterp securityContext
      (P says prop (SOME cmd)::ins) s outs)
```

[TRrule1]

```
⊢ TR (M,Oi,Os) (trap cmd)
    (CFG authenticationTest stateInterp securityContext
      (P says prop (SOME cmd)::ins) s outs)
    (CFG authenticationTest stateInterp securityContext ins
      (NS s (trap cmd)) (Out s (trap cmd)::outs)) ⟺
  authenticationTest (P says prop (SOME cmd)) ∧
  CFGInterpret (M,Oi,Os)
    (CFG authenticationTest stateInterp securityContext
      (P says prop (SOME cmd)::ins) s outs)
```

[trType_distinct_clauses]

```
⊢ (∀a' a. discard a ≠ trap a') ∧ (∀a' a. discard a ≠ exec a') ∧
  ∀a' a. trap a ≠ exec a'
```

[trType_one_one]

```
⊢ (∀a a'. (discard a = discard a') ⟺ (a = a')) ∧
  (∀a a'. (trap a = trap a') ⟺ (a = a')) ∧
  ∀a a'. (exec a = exec a') ⟺ (a = a')
```

3 ssm Theory

Built: 10 June 2018
Parent Theories: satList

3.1 Datatypes

```
configuration =
    CFG (('command option, 'principal, 'd, 'e) Form -> bool)
        ('state ->
         ('command option, 'principal, 'd, 'e) Form list ->
         ('command option, 'principal, 'd, 'e) Form list)
        (('command option, 'principal, 'd, 'e) Form list ->
         ('command option, 'principal, 'd, 'e) Form list)
        (('command option, 'principal, 'd, 'e) Form list list)
        'state ('output list)
```

```
trType = discard 'cmdlist | trap 'cmdlist | exec 'cmdlist
```

3.2 Definitions

[authenticationTest_def]

```
⊢ ∀ elementTest x.
    authenticationTest elementTest x ⇐⇒
    FOLDR (λ p q. p ∧ q) T (MAP elementTest x)
```

[commandList_def]

```
⊢ ∀ x. commandList x = MAP extractCommand x
```

[inputList_def]

```
⊢ ∀ xs. inputList xs = MAP extractInput xs
```

[propCommandList_def]

```
⊢ ∀ x. propCommandList x = MAP extractPropCommand x
```

[TR_def]

```
⊢ TR =
  (λ a0 a1 a2 a3.
     ∀ TR′.
       (∀ a0 a1 a2 a3.
          (∃ elementTest NS M Oi Os Out s context stateInterp x
             ins outs.
             (a0 = (M,Oi,Os)) ∧ (a1 = exec (inputList x)) ∧
             (a2 =
              CFG elementTest stateInterp context (x::ins) s
                outs) ∧
             (a3 =
              CFG elementTest stateInterp context ins
                (NS s (exec (inputList x)))
                (Out s (exec (inputList x))::outs)) ∧
             authenticationTest elementTest x ∧
             CFGInterpret (M,Oi,Os)
               (CFG elementTest stateInterp context (x::ins) s
                  outs)) ∨
          (∃ elementTest NS M Oi Os Out s context stateInterp x
             ins outs.
             (a0 = (M,Oi,Os)) ∧ (a1 = trap (inputList x)) ∧
             (a2 =
              CFG elementTest stateInterp context (x::ins) s
                outs) ∧
             (a3 =
              CFG elementTest stateInterp context ins
                (NS s (trap (inputList x)))
                (Out s (trap (inputList x))::outs)) ∧
             authenticationTest elementTest x ∧
             CFGInterpret (M,Oi,Os)
               (CFG elementTest stateInterp context (x::ins) s
```

$$\begin{array}{l}
\qquad\qquad\qquad \mathit{outs})) \ \lor \\
\qquad\quad (\exists\, \mathit{elementTest}\ \mathit{NS}\ M\ \mathit{Oi}\ \mathit{Os}\ \mathit{Out}\ s\ \mathit{context}\ \mathit{stateInterp}\ x \\
\qquad\qquad \mathit{ins}\ \mathit{outs}. \\
\qquad\qquad (a_0 = (M,\mathit{Oi},\mathit{Os})) \land (a_1 = \mathtt{discard}\ (\mathtt{inputList}\ x)) \land \\
\qquad\qquad (a_2 = \\
\qquad\qquad\ \mathtt{CFG}\ \mathit{elementTest}\ \mathit{stateInterp}\ \mathit{context}\ (x::\mathit{ins})\ s \\
\qquad\qquad\quad \mathit{outs}) \land \\
\qquad\qquad (a_3 = \\
\qquad\qquad\ \mathtt{CFG}\ \mathit{elementTest}\ \mathit{stateInterp}\ \mathit{context}\ \mathit{ins} \\
\qquad\qquad\quad (\mathit{NS}\ s\ (\mathtt{discard}\ (\mathtt{inputList}\ x))) \\
\qquad\qquad\quad (\mathit{Out}\ s\ (\mathtt{discard}\ (\mathtt{inputList}\ x))::\mathit{outs})) \land \\
\qquad\qquad \neg\mathtt{authenticationTest}\ \mathit{elementTest}\ x) \Rightarrow \\
\qquad\quad \mathit{TR}'\ a_0\ a_1\ a_2\ a_3) \Rightarrow \\
\qquad \mathit{TR}'\ a_0\ a_1\ a_2\ a_3)
\end{array}$$

3.3 Theorems

[CFGInterpret_def]

$$\begin{array}{l}
\vdash \mathtt{CFGInterpret}\ (M,\mathit{Oi},\mathit{Os}) \\
\qquad (\mathtt{CFG}\ \mathit{elementTest}\ \mathit{stateInterp}\ \mathit{context}\ (x::\mathit{ins})\ \mathit{state} \\
\qquad\quad \mathit{outStream}) \iff \\
\quad (M,\mathit{Oi},\mathit{Os})\ \mathtt{satList}\ \mathit{context}\ x \land (M,\mathit{Oi},\mathit{Os})\ \mathtt{satList}\ x \land \\
\quad (M,\mathit{Oi},\mathit{Os})\ \mathtt{satList}\ \mathit{stateInterp}\ \mathit{state}\ x
\end{array}$$

[CFGInterpret_ind]

$$\begin{array}{l}
\vdash \forall P. \\
\qquad (\forall M\ \mathit{Oi}\ \mathit{Os}\ \mathit{elementTest}\ \mathit{stateInterp}\ \mathit{context}\ x\ \mathit{ins}\ \mathit{state} \\
\qquad\quad \mathit{outStream}. \\
\qquad\quad P\ (M,\mathit{Oi},\mathit{Os}) \\
\qquad\qquad (\mathtt{CFG}\ \mathit{elementTest}\ \mathit{stateInterp}\ \mathit{context}\ (x::\mathit{ins})\ \mathit{state} \\
\qquad\qquad\quad \mathit{outStream})) \land \\
\qquad (\forall v_{15}\ v_{10}\ v_{11}\ v_{12}\ v_{13}\ v_{14}. \\
\qquad\quad P\ v_{15}\ (\mathtt{CFG}\ v_{10}\ v_{11}\ v_{12}\ []\ v_{13}\ v_{14})) \Rightarrow \\
\qquad \forall v\ v_1\ v_2\ v_3.\ P\ (v,v_1,v_2)\ v_3
\end{array}$$

[configuration_one_one]

$$\begin{array}{l}
\vdash \forall a_0\ a_1\ a_2\ a_3\ a_4\ a_5\ a_0'\ a_1'\ a_2'\ a_3'\ a_4'\ a_5'. \\
\qquad (\mathtt{CFG}\ a_0\ a_1\ a_2\ a_3\ a_4\ a_5 = \mathtt{CFG}\ a_0'\ a_1'\ a_2'\ a_3'\ a_4'\ a_5') \iff \\
\qquad (a_0 = a_0') \land (a_1 = a_1') \land (a_2 = a_2') \land (a_3 = a_3') \land \\
\qquad (a_4 = a_4') \land (a_5 = a_5')
\end{array}$$

[extractCommand_def]

$$\vdash \mathtt{extractCommand}\ (P\ \mathtt{says}\ \mathtt{prop}\ (\mathtt{SOME}\ \mathit{cmd})) = \mathit{cmd}$$

[extractCommand_ind]

$$\begin{array}{l}
\vdash \forall P'. \\
\qquad (\forall P\ \mathit{cmd}.\ P'\ (P\ \mathtt{says}\ \mathtt{prop}\ (\mathtt{SOME}\ \mathit{cmd}))) \land P'\ \mathtt{TT} \land P'\ \mathtt{FF} \land \\
\qquad (\forall v_1.\ P'\ (\mathtt{prop}\ v_1)) \land (\forall v_3.\ P'\ (\mathtt{notf}\ v_3)) \land
\end{array}$$

$(\forall v_6\ v_7.\ P'$ (v_6 andf v_7)) $\land$ ($\forall v_{10}\ v_{11}.\ P'$ (v_{10} orf v_{11})) $\land$
($\forall v_{14}\ v_{15}.\ P'$ (v_{14} impf v_{15})) $\land$
($\forall v_{18}\ v_{19}.\ P'$ (v_{18} eqf v_{19})) $\land$ ($\forall$ *v129*. P' (*v129* says TT)) $\land$
($\forall$ *v130*. P' (*v130* says FF)) $\land$
($\forall$ *v132*. P' (*v132* says prop NONE)) $\land$
($\forall$ *v133* v_{66}. P' (*v133* says notf v_{66})) $\land$
($\forall$ *v134* $v_{69}\ v_{70}$. P' (*v134* says (v_{69} andf v_{70}))) $\land$
($\forall$ *v135* $v_{73}\ v_{74}$. P' (*v135* says (v_{73} orf v_{74}))) $\land$
($\forall$ *v136* $v_{77}\ v_{78}$. P' (*v136* says (v_{77} impf v_{78}))) $\land$
($\forall$ *v137* $v_{81}\ v_{82}$. P' (*v137* says (v_{81} eqf v_{82}))) $\land$
($\forall$ *v138* $v_{85}\ v_{86}$. P' (*v138* says v_{85} says v_{86})) $\land$
($\forall$ *v139* $v_{89}\ v_{90}$. P' (*v139* says v_{89} speaks_for v_{90})) $\land$
($\forall$ *v140* $v_{93}\ v_{94}$. P' (*v140* says v_{93} controls v_{94})) $\land$
($\forall$ *v141* $v_{98}\ v_{99}$ *v100*. P' (*v141* says reps $v_{98}\ v_{99}$ *v100*)) $\land$
($\forall$ *v142* *v103* *v104*. P' (*v142* says *v103* domi *v104*)) $\land$
($\forall$ *v143* *v107* *v108*. P' (*v143* says *v107* eqi *v108*)) $\land$
($\forall$ *v144* *v111* *v112*. P' (*v144* says *v111* doms *v112*)) $\land$
($\forall$ *v145* *v115* *v116*. P' (*v145* says *v115* eqs *v116*)) $\land$
($\forall$ *v146* *v119* *v120*. P' (*v146* says *v119* eqn *v120*)) $\land$
($\forall$ *v147* *v123* *v124*. P' (*v147* says *v123* lte *v124*)) $\land$
($\forall$ *v148* *v127* *v128*. P' (*v148* says *v127* lt *v128*)) $\land$
($\forall v_{24}\ v_{25}.\ P'$ (v_{24} speaks_for v_{25})) $\land$
($\forall v_{28}\ v_{29}.\ P'$ (v_{28} controls v_{29})) $\land$
($\forall v_{33}\ v_{34}\ v_{35}.\ P'$ (reps $v_{33}\ v_{34}\ v_{35}$)) $\land$
($\forall v_{38}\ v_{39}.\ P'$ (v_{38} domi v_{39})) $\land$
($\forall v_{42}\ v_{43}.\ P'$ (v_{42} eqi v_{43})) $\land$
($\forall v_{46}\ v_{47}.\ P'$ (v_{46} doms v_{47})) $\land$
($\forall v_{50}\ v_{51}.\ P'$ (v_{50} eqs v_{51})) $\land$
($\forall v_{54}\ v_{55}.\ P'$ (v_{54} eqn v_{55})) $\land$
($\forall v_{58}\ v_{59}.\ P'$ (v_{58} lte v_{59})) $\land$
($\forall v_{62}\ v_{63}.\ P'$ (v_{62} lt v_{63})) $\Rightarrow$
$\forall v.\ P'\ v$

[extractInput_def]

$\vdash$ extractInput (P says prop x) = x

[extractInput_ind]

$\vdash \forall P'$.
($\forall P\ x.\ P'$ (P says prop x)) $\land$ P' TT $\land$ P' FF $\land$
($\forall v_1.\ P'$ (prop v_1)) $\land$ ($\forall v_3.\ P'$ (notf v_3)) $\land$
($\forall v_6\ v_7.\ P'$ (v_6 andf v_7)) $\land$ ($\forall v_{10}\ v_{11}.\ P'$ (v_{10} orf v_{11})) $\land$
($\forall v_{14}\ v_{15}.\ P'$ (v_{14} impf v_{15})) $\land$
($\forall v_{18}\ v_{19}.\ P'$ (v_{18} eqf v_{19})) $\land$ ($\forall$ *v129*. P' (*v129* says TT)) $\land$
($\forall$ *v130*. P' (*v130* says FF)) $\land$
($\forall$ *v131* v_{66}. P' (*v131* says notf v_{66})) $\land$
($\forall$ *v132* $v_{69}\ v_{70}$. P' (*v132* says (v_{69} andf v_{70}))) $\land$
($\forall$ *v133* $v_{73}\ v_{74}$. P' (*v133* says (v_{73} orf v_{74}))) $\land$
($\forall$ *v134* $v_{77}\ v_{78}$. P' (*v134* says (v_{77} impf v_{78}))) $\land$
($\forall$ *v135* $v_{81}\ v_{82}$. P' (*v135* says (v_{81} eqf v_{82}))) $\land$

$(\forall v136\ v_{85}\ v_{86}.\ P'\ (v136\ \mathtt{says}\ v_{85}\ \mathtt{says}\ v_{86}))\ \land$
$(\forall v137\ v_{89}\ v_{90}.\ P'\ (v137\ \mathtt{says}\ v_{89}\ \mathtt{speaks\_for}\ v_{90}))\ \land$
$(\forall v138\ v_{93}\ v_{94}.\ P'\ (v138\ \mathtt{says}\ v_{93}\ \mathtt{controls}\ v_{94}))\ \land$
$(\forall v139\ v_{98}\ v_{99}\ v100.\ P'\ (v139\ \mathtt{says}\ \mathtt{reps}\ v_{98}\ v_{99}\ v100))\ \land$
$(\forall v140\ v103\ v104.\ P'\ (v140\ \mathtt{says}\ v103\ \mathtt{domi}\ v104))\ \land$
$(\forall v141\ v107\ v108.\ P'\ (v141\ \mathtt{says}\ v107\ \mathtt{eqi}\ v108))\ \land$
$(\forall v142\ v111\ v112.\ P'\ (v142\ \mathtt{says}\ v111\ \mathtt{doms}\ v112))\ \land$
$(\forall v143\ v115\ v116.\ P'\ (v143\ \mathtt{says}\ v115\ \mathtt{eqs}\ v116))\ \land$
$(\forall v144\ v119\ v120.\ P'\ (v144\ \mathtt{says}\ v119\ \mathtt{eqn}\ v120))\ \land$
$(\forall v145\ v123\ v124.\ P'\ (v145\ \mathtt{says}\ v123\ \mathtt{lte}\ v124))\ \land$
$(\forall v146\ v127\ v128.\ P'\ (v146\ \mathtt{says}\ v127\ \mathtt{lt}\ v128))\ \land$
$(\forall v_{24}\ v_{25}.\ P'\ (v_{24}\ \mathtt{speaks\_for}\ v_{25}))\ \land$
$(\forall v_{28}\ v_{29}.\ P'\ (v_{28}\ \mathtt{controls}\ v_{29}))\ \land$
$(\forall v_{33}\ v_{34}\ v_{35}.\ P'\ (\mathtt{reps}\ v_{33}\ v_{34}\ v_{35}))\ \land$
$(\forall v_{38}\ v_{39}.\ P'\ (v_{38}\ \mathtt{domi}\ v_{39}))\ \land$
$(\forall v_{42}\ v_{43}.\ P'\ (v_{42}\ \mathtt{eqi}\ v_{43}))\ \land$
$(\forall v_{46}\ v_{47}.\ P'\ (v_{46}\ \mathtt{doms}\ v_{47}))\ \land$
$(\forall v_{50}\ v_{51}.\ P'\ (v_{50}\ \mathtt{eqs}\ v_{51}))\ \land$
$(\forall v_{54}\ v_{55}.\ P'\ (v_{54}\ \mathtt{eqn}\ v_{55}))\ \land$
$(\forall v_{58}\ v_{59}.\ P'\ (v_{58}\ \mathtt{lte}\ v_{59}))\ \land$
$(\forall v_{62}\ v_{63}.\ P'\ (v_{62}\ \mathtt{lt}\ v_{63}))\ \Rightarrow$
$\forall v.\ P'\ v$

[extractPropCommand_def]

$\vdash \mathtt{extractPropCommand}\ (P\ \mathtt{says}\ \mathtt{prop}\ (\mathtt{SOME}\ cmd)) = \mathtt{prop}\ (\mathtt{SOME}\ cmd)$

[extractPropCommand_ind]

$\vdash \forall P'.$
$(\forall P\ cmd.\ P'\ (P\ \mathtt{says}\ \mathtt{prop}\ (\mathtt{SOME}\ cmd)))\ \land\ P'\ \mathtt{TT}\ \land\ P'\ \mathtt{FF}\ \land$
$(\forall v_1.\ P'\ (\mathtt{prop}\ v_1))\ \land\ (\forall v_3.\ P'\ (\mathtt{notf}\ v_3))\ \land$
$(\forall v_6\ v_7.\ P'\ (v_6\ \mathtt{andf}\ v_7))\ \land\ (\forall v_{10}\ v_{11}.\ P'\ (v_{10}\ \mathtt{orf}\ v_{11}))\ \land$
$(\forall v_{14}\ v_{15}.\ P'\ (v_{14}\ \mathtt{impf}\ v_{15}))\ \land$
$(\forall v_{18}\ v_{19}.\ P'\ (v_{18}\ \mathtt{eqf}\ v_{19}))\ \land\ (\forall v129.\ P'\ (v129\ \mathtt{says}\ \mathtt{TT}))\ \land$
$(\forall v130.\ P'\ (v130\ \mathtt{says}\ \mathtt{FF}))\ \land$
$(\forall v132.\ P'\ (v132\ \mathtt{says}\ \mathtt{prop}\ \mathtt{NONE}))\ \land$
$(\forall v133\ v_{66}.\ P'\ (v133\ \mathtt{says}\ \mathtt{notf}\ v_{66}))\ \land$
$(\forall v134\ v_{69}\ v_{70}.\ P'\ (v134\ \mathtt{says}\ (v_{69}\ \mathtt{andf}\ v_{70})))\ \land$
$(\forall v135\ v_{73}\ v_{74}.\ P'\ (v135\ \mathtt{says}\ (v_{73}\ \mathtt{orf}\ v_{74})))\ \land$
$(\forall v136\ v_{77}\ v_{78}.\ P'\ (v136\ \mathtt{says}\ (v_{77}\ \mathtt{impf}\ v_{78})))\ \land$
$(\forall v137\ v_{81}\ v_{82}.\ P'\ (v137\ \mathtt{says}\ (v_{81}\ \mathtt{eqf}\ v_{82})))\ \land$
$(\forall v138\ v_{85}\ v_{86}.\ P'\ (v138\ \mathtt{says}\ v_{85}\ \mathtt{says}\ v_{86}))\ \land$
$(\forall v139\ v_{89}\ v_{90}.\ P'\ (v139\ \mathtt{says}\ v_{89}\ \mathtt{speaks\_for}\ v_{90}))\ \land$
$(\forall v140\ v_{93}\ v_{94}.\ P'\ (v140\ \mathtt{says}\ v_{93}\ \mathtt{controls}\ v_{94}))\ \land$
$(\forall v141\ v_{98}\ v_{99}\ v100.\ P'\ (v141\ \mathtt{says}\ \mathtt{reps}\ v_{98}\ v_{99}\ v100))\ \land$
$(\forall v142\ v103\ v104.\ P'\ (v142\ \mathtt{says}\ v103\ \mathtt{domi}\ v104))\ \land$
$(\forall v143\ v107\ v108.\ P'\ (v143\ \mathtt{says}\ v107\ \mathtt{eqi}\ v108))\ \land$
$(\forall v144\ v111\ v112.\ P'\ (v144\ \mathtt{says}\ v111\ \mathtt{doms}\ v112))\ \land$
$(\forall v145\ v115\ v116.\ P'\ (v145\ \mathtt{says}\ v115\ \mathtt{eqs}\ v116))\ \land$
$(\forall v146\ v119\ v120.\ P'\ (v146\ \mathtt{says}\ v119\ \mathtt{eqn}\ v120))\ \land$

```
    (∀ v147 v123 v124. P′ (v147 says v123 lte v124)) ∧
    (∀ v148 v127 v128. P′ (v148 says v127 lt v128)) ∧
    (∀ v24 v25. P′ (v24 speaks_for v25)) ∧
    (∀ v28 v29. P′ (v28 controls v29)) ∧
    (∀ v33 v34 v35. P′ (reps v33 v34 v35)) ∧
    (∀ v38 v39. P′ (v38 domi v39)) ∧
    (∀ v42 v43. P′ (v42 eqi v43)) ∧
    (∀ v46 v47. P′ (v46 doms v47)) ∧
    (∀ v50 v51. P′ (v50 eqs v51)) ∧
    (∀ v54 v55. P′ (v54 eqn v55)) ∧
    (∀ v58 v59. P′ (v58 lte v59)) ∧
    (∀ v62 v63. P′ (v62 lt v63)) ⇒
    ∀ v. P′ v
```

[TR_cases]

```
 ⊢ ∀ a0 a1 a2 a3.
     TR a0 a1 a2 a3 ⇐⇒
     (∃ elementTest NS M Oi Os Out s context stateInterp x ins
         outs.
        (a0 = (M,Oi,Os)) ∧ (a1 = exec (inputList x)) ∧
        (a2 =
         CFG elementTest stateInterp context (x::ins) s outs) ∧
        (a3 =
         CFG elementTest stateInterp context ins
           (NS s (exec (inputList x)))
           (Out s (exec (inputList x))::outs)) ∧
        authenticationTest elementTest x ∧
        CFGInterpret (M,Oi,Os)
          (CFG elementTest stateInterp context (x::ins) s
             outs)) ∨
     (∃ elementTest NS M Oi Os Out s context stateInterp x ins
         outs.
        (a0 = (M,Oi,Os)) ∧ (a1 = trap (inputList x)) ∧
        (a2 =
         CFG elementTest stateInterp context (x::ins) s outs) ∧
        (a3 =
         CFG elementTest stateInterp context ins
           (NS s (trap (inputList x)))
           (Out s (trap (inputList x))::outs)) ∧
        authenticationTest elementTest x ∧
        CFGInterpret (M,Oi,Os)
          (CFG elementTest stateInterp context (x::ins) s
             outs)) ∨
     ∃ elementTest NS M Oi Os Out s context stateInterp x ins
       outs.
      (a0 = (M,Oi,Os)) ∧ (a1 = discard (inputList x)) ∧
      (a2 =
       CFG elementTest stateInterp context (x::ins) s outs) ∧
      (a3 =
        CFG elementTest stateInterp context ins
          (NS s (discard (inputList x)))
          (Out s (discard (inputList x))::outs)) ∧
      ¬authenticationTest elementTest x
```

[TR_discard_cmd_rule]

```
 ⊢ TR (M,Oi,Os) (discard (inputList x))
     (CFG elementTest stateInterp context (x::ins) s outs)
     (CFG elementTest stateInterp context ins
        (NS s (discard (inputList x)))
        (Out s (discard (inputList x))::outs)) ⟺
   ¬authenticationTest elementTest x
```

[TR_EQ_rules_thm]

```
 ⊢ (TR (M,Oi,Os) (exec (inputList x))
      (CFG elementTest stateInterp context (x::ins) s outs)
      (CFG elementTest stateInterp context ins
         (NS s (exec (inputList x)))
         (Out s (exec (inputList x))::outs)) ⟺
    authenticationTest elementTest x ∧
    CFGInterpret (M,Oi,Os)
      (CFG elementTest stateInterp context (x::ins) s outs)) ∧
   (TR (M,Oi,Os) (trap (inputList x))
      (CFG elementTest stateInterp context (x::ins) s outs)
      (CFG elementTest stateInterp context ins
         (NS s (trap (inputList x)))
         (Out s (trap (inputList x))::outs)) ⟺
    authenticationTest elementTest x ∧
    CFGInterpret (M,Oi,Os)
      (CFG elementTest stateInterp context (x::ins) s outs)) ∧
   (TR (M,Oi,Os) (discard (inputList x))
      (CFG elementTest stateInterp context (x::ins) s outs)
      (CFG elementTest stateInterp context ins
         (NS s (discard (inputList x)))
         (Out s (discard (inputList x))::outs)) ⟺
    ¬authenticationTest elementTest x)
```

[TR_exec_cmd_rule]

```
 ⊢ ∀elementTest context stateInterp x ins s outs.
     (∀M Oi Os.
        CFGInterpret (M,Oi,Os)
          (CFG elementTest stateInterp context (x::ins) s
             outs) ⇒
        (M,Oi,Os) satList propCommandList x) ⇒
     ∀NS Out M Oi Os.
       TR (M,Oi,Os) (exec (inputList x))
         (CFG elementTest stateInterp context (x::ins) s outs)
         (CFG elementTest stateInterp context ins
          (NS s (exec (inputList x)))
          (Out s (exec (inputList x))::outs)) ⇔
      authenticationTest elementTest x ∧
      CFGInterpret (M,Oi,Os)
        (CFG elementTest stateInterp context (x::ins) s outs) ∧
      (M,Oi,Os) satList propCommandList x
```

[TR_ind]

```
⊢ ∀ TR′.
    (∀ elementTest NS M Oi Os Out s context stateInterp x ins
       outs.
       authenticationTest elementTest x ∧
       CFGInterpret (M,Oi,Os)
         (CFG elementTest stateInterp context (x::ins) s
            outs) ⇒
       TR′ (M,Oi,Os) (exec (inputList x))
         (CFG elementTest stateInterp context (x::ins) s outs)
         (CFG elementTest stateInterp context ins
            (NS s (exec (inputList x)))
            (Out s (exec (inputList x))::outs))) ∧
    (∀ elementTest NS M Oi Os Out s context stateInterp x ins
       outs.
       authenticationTest elementTest x ∧
       CFGInterpret (M,Oi,Os)
         (CFG elementTest stateInterp context (x::ins) s
            outs) ⇒
       TR′ (M,Oi,Os) (trap (inputList x))
         (CFG elementTest stateInterp context (x::ins) s outs)
         (CFG elementTest stateInterp context ins
            (NS s (trap (inputList x)))
            (Out s (trap (inputList x))::outs))) ∧
    (∀ elementTest NS M Oi Os Out s context stateInterp x ins
       outs.
       ¬authenticationTest elementTest x ⇒
       TR′ (M,Oi,Os) (discard (inputList x))
         (CFG elementTest stateInterp context (x::ins) s outs)
         (CFG elementTest stateInterp context ins
            (NS s (discard (inputList x)))
            (Out s (discard (inputList x))::outs))) ⇒
    ∀ a0 a1 a2 a3. TR a0 a1 a2 a3 ⇒ TR′ a0 a1 a2 a3
```

[TR_rules]

```
⊢ (∀ elementTest NS M Oi Os Out s context stateInterp x ins
     outs.
     authenticationTest elementTest x ∧
     CFGInterpret (M,Oi,Os)
       (CFG elementTest stateInterp context (x::ins) s outs) ⇒
     TR (M,Oi,Os) (exec (inputList x))
       (CFG elementTest stateInterp context (x::ins) s outs)
```

```
        (CFG elementTest stateInterp context ins
           (NS s (exec (inputList x)))
           (Out s (exec (inputList x))::outs))) ∧
   (∀ elementTest NS M Oi Os Out s context stateInterp x ins
       outs.
      authenticationTest elementTest x ∧
      CFGInterpret (M,Oi,Os)
        (CFG elementTest stateInterp context (x::ins) s outs) ⇒
      TR (M,Oi,Os) (trap (inputList x))
        (CFG elementTest stateInterp context (x::ins) s outs)
        (CFG elementTest stateInterp context ins
           (NS s (trap (inputList x)))
           (Out s (trap (inputList x))::outs))) ∧
   ∀ elementTest NS M Oi Os Out s context stateInterp x ins outs.
     ¬authenticationTest elementTest x ⇒
     TR (M,Oi,Os) (discard (inputList x))
       (CFG elementTest stateInterp context (x::ins) s outs)
       (CFG elementTest stateInterp context ins
          (NS s (discard (inputList x)))
          (Out s (discard (inputList x))::outs))
```

[TR_strongind]

```
 ⊢ ∀ TR′.
     (∀ elementTest NS M Oi Os Out s context stateInterp x ins
         outs.
        authenticationTest elementTest x ∧
        CFGInterpret (M,Oi,Os)
          (CFG elementTest stateInterp context (x::ins) s
             outs) ⇒
        TR′ (M,Oi,Os) (exec (inputList x))
          (CFG elementTest stateInterp context (x::ins) s outs)
          (CFG elementTest stateInterp context ins
             (NS s (exec (inputList x)))
             (Out s (exec (inputList x))::outs))) ∧
     (∀ elementTest NS M Oi Os Out s context stateInterp x ins
         outs.
        authenticationTest elementTest x ∧
        CFGInterpret (M,Oi,Os)
          (CFG elementTest stateInterp context (x::ins) s
             outs) ⇒
        TR′ (M,Oi,Os) (trap (inputList x))
          (CFG elementTest stateInterp context (x::ins) s outs)
          (CFG elementTest stateInterp context ins
             (NS s (trap (inputList x)))
             (Out s (trap (inputList x))::outs))) ∧
     (∀ elementTest NS M Oi Os Out s context stateInterp x ins
         outs.
        ¬authenticationTest elementTest x ⇒
        TR′ (M,Oi,Os) (discard (inputList x))
```

```
          (CFG elementTest stateInterp context (x::ins) s outs)
          (CFG elementTest stateInterp context ins
             (NS s (discard (inputList x)))
             (Out s (discard (inputList x))::outs))) ⇒
     ∀ a0 a1 a2 a3. TR a0 a1 a2 a3 ⇒ TR' a0 a1 a2 a3
```

[TR_trap_cmd_rule]

```
 ⊢ ∀ elementTest context stateInterp x ins s outs.
     (∀ M Oi Os.
        CFGInterpret (M,Oi,Os)
          (CFG elementTest stateInterp context (x::ins) s
             outs) ⇒
        (M,Oi,Os) sat prop NONE) ⇒
     ∀ NS Out M Oi Os.
       TR (M,Oi,Os) (trap (inputList x))
         (CFG elementTest stateInterp context (x::ins) s outs)
         (CFG elementTest stateInterp context ins
            (NS s (trap (inputList x)))
            (Out s (trap (inputList x))::outs)) ⟺
       authenticationTest elementTest x ∧
       CFGInterpret (M,Oi,Os)
         (CFG elementTest stateInterp context (x::ins) s outs) ∧
       (M,Oi,Os) sat prop NONE
```

[TRrule0]

```
 ⊢ TR (M,Oi,Os) (exec (inputList x))
     (CFG elementTest stateInterp context (x::ins) s outs)
     (CFG elementTest stateInterp context ins
        (NS s (exec (inputList x)))
        (Out s (exec (inputList x))::outs)) ⟺
   authenticationTest elementTest x ∧
   CFGInterpret (M,Oi,Os)
     (CFG elementTest stateInterp context (x::ins) s outs)
```

[TRrule1]

```
 ⊢ TR (M,Oi,Os) (trap (inputList x))
     (CFG elementTest stateInterp context (x::ins) s outs)
     (CFG elementTest stateInterp context ins
        (NS s (trap (inputList x)))
        (Out s (trap (inputList x))::outs)) ⟺
   authenticationTest elementTest x ∧
   CFGInterpret (M,Oi,Os)
     (CFG elementTest stateInterp context (x::ins) s outs)
```

[trType_distinct_clauses]

```
 ⊢ (∀ a' a. discard a ≠ trap a') ∧ (∀ a' a. discard a ≠ exec a') ∧
   ∀ a' a. trap a ≠ exec a'
```

[trType_one_one]

$\vdash$ ($\forall a\ a'$. (discard a = discard a') $\iff$ (a = a')) $\wedge$
($\forall a\ a'$. (trap a = trap a') $\iff$ (a = a')) $\wedge$
$\forall a\ a'$. (exec a = exec a') $\iff$ (a = a')

4 satList Theory

Built: 10 June 2018
Parent Theories: aclDrules

4.1 Definitions

[satList_def]

$\vdash \forall M\ Oi\ Os\ formList$.
(M,Oi,Os) satList $formList$ $\iff$
FOLDR ($\lambda x\ y$. $x \wedge y$) T (MAP (λf. (M,Oi,Os) sat f) $formList$)

4.2 Theorems

[satList_conj]

$\vdash \forall l_1\ l_2\ M\ Oi\ Os$.
(M,Oi,Os) satList l_1 $\wedge$ (M,Oi,Os) satList l_2 $\iff$
(M,Oi,Os) satList (l_1 ++ l_2)

[satList_CONS]

$\vdash \forall h\ t\ M\ Oi\ Os$.
(M,Oi,Os) satList (h::t) $\iff$
(M,Oi,Os) sat h $\wedge$ (M,Oi,Os) satList t

[satList_nil]

$\vdash$ (M,Oi,Os) satList []

5 PBTypeIntegrated Theory

Built: 11 June 2018
Parent Theories: OMNIType

5.1 Datatypes

```
omniCommand = ssmPlanPBComplete | ssmMoveToORPComplete
            | ssmConductORPComplete | ssmMoveToPBComplete
            | ssmConductPBComplete | invalidOmniCommand

plCommand = crossLD | conductORP | moveToPB | conductPB
          | completePB | incomplete
```

```
slCommand =
    PL PBTypeIntegrated$plCommand
  | OMNI PBTypeIntegrated$omniCommand

slOutput = PlanPB | MoveToORP | ConductORP | MoveToPB
         | ConductPB | CompletePB | unAuthenticated
         | unAuthorized

slState = PLAN_PB | MOVE_TO_ORP | CONDUCT_ORP | MOVE_TO_PB
        | CONDUCT_PB | COMPLETE_PB

stateRole = PlatoonLeader | Omni
```

5.2 Theorems

[omniCommand_distinct_clauses]

```
⊢ ssmPlanPBComplete ≠ ssmMoveToORPComplete ∧
  ssmPlanPBComplete ≠ ssmConductORPComplete ∧
  ssmPlanPBComplete ≠ ssmMoveToPBComplete ∧
  ssmPlanPBComplete ≠ ssmConductPBComplete ∧
  ssmPlanPBComplete ≠ invalidOmniCommand ∧
  ssmMoveToORPComplete ≠ ssmConductORPComplete ∧
  ssmMoveToORPComplete ≠ ssmMoveToPBComplete ∧
  ssmMoveToORPComplete ≠ ssmConductPBComplete ∧
  ssmMoveToORPComplete ≠ invalidOmniCommand ∧
  ssmConductORPComplete ≠ ssmMoveToPBComplete ∧
  ssmConductORPComplete ≠ ssmConductPBComplete ∧
  ssmConductORPComplete ≠ invalidOmniCommand ∧
  ssmMoveToPBComplete ≠ ssmConductPBComplete ∧
  ssmMoveToPBComplete ≠ invalidOmniCommand ∧
  ssmConductPBComplete ≠ invalidOmniCommand
```

[plCommand_distinct_clauses]

```
⊢ crossLD ≠ conductORP ∧ crossLD ≠ moveToPB ∧
  crossLD ≠ conductPB ∧ crossLD ≠ completePB ∧
  crossLD ≠ incomplete ∧ conductORP ≠ moveToPB ∧
  conductORP ≠ conductPB ∧ conductORP ≠ completePB ∧
  conductORP ≠ incomplete ∧ moveToPB ≠ conductPB ∧
  moveToPB ≠ completePB ∧ moveToPB ≠ incomplete ∧
  conductPB ≠ completePB ∧ conductPB ≠ incomplete ∧
  completePB ≠ incomplete
```

[slCommand_distinct_clauses]

```
⊢ ∀a′ a. PL a ≠ OMNI a′
```

[slCommand_one_one]

```
⊢ (∀a a′. (PL a = PL a′) ⇔ (a = a′)) ∧
  ∀a a′. (OMNI a = OMNI a′) ⇔ (a = a′)
```

[slOutput_distinct_clauses]

```
⊢ PlanPB ≠ MoveToORP ∧ PlanPB ≠ ConductORP ∧
  PlanPB ≠ MoveToPB ∧ PlanPB ≠ ConductPB ∧
  PlanPB ≠ CompletePB ∧ PlanPB ≠ unAuthenticated ∧
  PlanPB ≠ unAuthorized ∧ MoveToORP ≠ ConductORP ∧
  MoveToORP ≠ MoveToPB ∧ MoveToORP ≠ ConductPB ∧
  MoveToORP ≠ CompletePB ∧ MoveToORP ≠ unAuthenticated ∧
  MoveToORP ≠ unAuthorized ∧ ConductORP ≠ MoveToPB ∧
  ConductORP ≠ ConductPB ∧ ConductORP ≠ CompletePB ∧
  ConductORP ≠ unAuthenticated ∧ ConductORP ≠ unAuthorized ∧
  MoveToPB ≠ ConductPB ∧ MoveToPB ≠ CompletePB ∧
  MoveToPB ≠ unAuthenticated ∧ MoveToPB ≠ unAuthorized ∧
  ConductPB ≠ CompletePB ∧ ConductPB ≠ unAuthenticated ∧
  ConductPB ≠ unAuthorized ∧ CompletePB ≠ unAuthenticated ∧
  CompletePB ≠ unAuthorized ∧ unAuthenticated ≠ unAuthorized
```

[slState_distinct_clauses]

```
⊢ PLAN_PB ≠ MOVE_TO_ORP ∧ PLAN_PB ≠ CONDUCT_ORP ∧
  PLAN_PB ≠ MOVE_TO_PB ∧ PLAN_PB ≠ CONDUCT_PB ∧
  PLAN_PB ≠ COMPLETE_PB ∧ MOVE_TO_ORP ≠ CONDUCT_ORP ∧
  MOVE_TO_ORP ≠ MOVE_TO_PB ∧ MOVE_TO_ORP ≠ CONDUCT_PB ∧
  MOVE_TO_ORP ≠ COMPLETE_PB ∧ CONDUCT_ORP ≠ MOVE_TO_PB ∧
  CONDUCT_ORP ≠ CONDUCT_PB ∧ CONDUCT_ORP ≠ COMPLETE_PB ∧
  MOVE_TO_PB ≠ CONDUCT_PB ∧ MOVE_TO_PB ≠ COMPLETE_PB ∧
  CONDUCT_PB ≠ COMPLETE_PB
```

[stateRole_distinct_clauses]

```
⊢ PlatoonLeader ≠ Omni
```

6 PBIntegratedDef Theory

Built: 11 June 2018

Parent Theories: PBTypeIntegrated, aclfoundation

6.1 Definitions

[secAuthorization_def]

```
⊢ ∀xs. secAuthorization xs = secHelper (getOmniCommand xs)
```

[secContext_def]

```
⊢ (∀xs.
     secContext PLAN_PB xs =
     if getOmniCommand xs = ssmPlanPBComplete then
       [prop (SOME (SLc (OMNI ssmPlanPBComplete))) impf
        Name PlatoonLeader controls
        prop (SOME (SLc (PL crossLD)))]
     else [prop NONE]) ∧
  (∀ xs.
     secContext MOVE_TO_ORP xs =
     if getOmniCommand xs = ssmMoveToORPComplete then
       [prop (SOME (SLc (OMNI ssmMoveToORPComplete))) impf
        Name PlatoonLeader controls
        prop (SOME (SLc (PL conductORP)))]
     else [prop NONE]) ∧
  (∀ xs.
     secContext CONDUCT_ORP xs =
     if getOmniCommand xs = ssmConductORPComplete then
       [prop (SOME (SLc (OMNI ssmConductORPComplete))) impf
        Name PlatoonLeader controls
        prop (SOME (SLc (PL moveToPB)))]
     else [prop NONE]) ∧
  (∀ xs.
     secContext MOVE_TO_PB xs =
     if getOmniCommand xs = ssmConductORPComplete then
       [prop (SOME (SLc (OMNI ssmMoveToPBComplete))) impf
        Name PlatoonLeader controls
        prop (SOME (SLc (PL conductPB)))]
     else [prop NONE]) ∧
  ∀ xs.
    secContext CONDUCT_PB xs =
    if getOmniCommand xs = ssmConductPBComplete then
      [prop (SOME (SLc (OMNI ssmConductPBComplete))) impf
       Name PlatoonLeader controls
       prop (SOME (SLc (PL completePB)))]
    else [prop NONE]
```

[secHelper_def]

```
⊢ ∀ cmd.
    secHelper cmd =
    [Name Omni controls prop (SOME (SLc (OMNI cmd)))]
```

6.2 Theorems

[getOmniCommand_def]

```
⊢ (getOmniCommand [] = invalidOmniCommand) ∧
  (∀ xs cmd.
     getOmniCommand
       (Name Omni says prop (SOME (SLc (OMNI cmd)))::xs) =
     cmd) ∧
  (∀ xs. getOmniCommand (TT::xs) = getOmniCommand xs) ∧
  (∀ xs. getOmniCommand (FF::xs) = getOmniCommand xs) ∧
  (∀ xs v2. getOmniCommand (prop v2::xs) = getOmniCommand xs) ∧
  (∀ xs v3. getOmniCommand (notf v3::xs) = getOmniCommand xs) ∧
  (∀ xs v5 v4.
   getOmniCommand ($v_4$ andf $v_5$::$xs$) = getOmniCommand $xs$) ∧
(∀ $xs$ $v_7$ $v_6$.
   getOmniCommand ($v_6$ orf $v_7$::$xs$) = getOmniCommand $xs$) ∧
(∀ $xs$ $v_9$ $v_8$.
   getOmniCommand ($v_8$ impf $v_9$::$xs$) = getOmniCommand $xs$) ∧
(∀ $xs$ $v_{11}$ $v_{10}$.
   getOmniCommand ($v_{10}$ eqf $v_{11}$::$xs$) = getOmniCommand $xs$) ∧
(∀ $xs$ $v_{12}$.
   getOmniCommand ($v_{12}$ says TT::$xs$) = getOmniCommand $xs$) ∧
(∀ $xs$ $v_{12}$.
   getOmniCommand ($v_{12}$ says FF::$xs$) = getOmniCommand $xs$) ∧
(∀ $xs$ $v134$.
   getOmniCommand (Name $v134$ says prop NONE::$xs$) =
   getOmniCommand $xs$) ∧
(∀ $xs$ $v144$.
   getOmniCommand
     (Name PlatoonLeader says prop (SOME $v144$)::$xs$) =
   getOmniCommand $xs$) ∧
(∀ $xs$ $v146$.
   getOmniCommand
     (Name Omni says prop (SOME (ESCc $v146$))::$xs$) =
   getOmniCommand $xs$) ∧
(∀ $xs$ $v150$.
   getOmniCommand
     (Name Omni says prop (SOME (SLc (PL $v150$)))::$xs$) =
   getOmniCommand $xs$) ∧
(∀ $xs$ $v_{68}$ $v136$ $v135$.
   getOmniCommand ($v135$ meet $v136$ says prop $v_{68}$::$xs$) =
   getOmniCommand $xs$) ∧
(∀ $xs$ $v_{68}$ $v138$ $v137$.
   getOmniCommand ($v137$ quoting $v138$ says prop $v_{68}$::$xs$) =
   getOmniCommand $xs$) ∧
(∀ $xs$ $v_{69}$ $v_{12}$.
   getOmniCommand ($v_{12}$ says notf $v_{69}$::$xs$) =
   getOmniCommand $xs$) ∧
(∀ $xs$ $v_{71}$ $v_{70}$ $v_{12}$.
   getOmniCommand ($v_{12}$ says ($v_{70}$ andf $v_{71}$)::$xs$) =
   getOmniCommand $xs$) ∧
(∀ $xs$ $v_{73}$ $v_{72}$ $v_{12}$.
   getOmniCommand ($v_{12}$ says ($v_{72}$ orf $v_{73}$)::$xs$) =
   getOmniCommand $xs$) ∧
(∀ $xs$ $v_{75}$ $v_{74}$ $v_{12}$.
   getOmniCommand ($v_{12}$ says ($v_{74}$ impf $v_{75}$)::$xs$) =
   getOmniCommand $xs$) ∧
(∀ $xs$ $v_{77}$ $v_{76}$ $v_{12}$.
   getOmniCommand ($v_{12}$ says ($v_{76}$ eqf $v_{77}$)::$xs$) =
   getOmniCommand $xs$) ∧
(∀ $xs$ $v_{79}$ $v_{78}$ $v_{12}$.
   getOmniCommand ($v_{12}$ says $v_{78}$ says $v_{79}$::$xs$) =
   getOmniCommand xs) ∧
(∀ xs v81 v80 v12.
   getOmniCommand (v12 says v80 speaks_for v81::xs) =
   getOmniCommand xs) ∧
(∀ xs v83 v82 v12.
   getOmniCommand (v12 says v82 controls v83::xs) =
   getOmniCommand xs) ∧
(∀ xs v86 v85 v84 v12.
   getOmniCommand (v12 says reps v84 v85 v86::xs) =
   getOmniCommand xs) ∧
(∀ xs v88 v87 v12.
   getOmniCommand (v12 says v87 domi v88::xs) =
   getOmniCommand xs) ∧
(∀ xs v90 v89 v12.
   getOmniCommand (v12 says v89 eqi v90::xs) =
   getOmniCommand xs) ∧
(∀ xs v92 v91 v12.
   getOmniCommand (v12 says v91 doms v92::xs) =
   getOmniCommand xs) ∧
(∀ xs v94 v93 v12.
   getOmniCommand (v12 says v93 eqs v94::xs) =
   getOmniCommand xs) ∧
(∀ xs v96 v95 v12.
   getOmniCommand (v12 says v95 eqn v96::xs) =
   getOmniCommand xs) ∧
(∀ xs v98 v97 v12.
   getOmniCommand (v12 says v97 lte v98::xs) =
   getOmniCommand xs) ∧
(∀ xs v99 v12 v100.
   getOmniCommand (v12 says v99 lt v100::xs) =
   getOmniCommand xs) ∧
(∀ xs v15 v14.
   getOmniCommand (v14 speaks_for v15::xs) =
   getOmniCommand xs) ∧
(∀ xs v17 v16.
   getOmniCommand (v16 controls v17::xs) =
   getOmniCommand xs) ∧
(∀ xs v20 v19 v18.
   getOmniCommand (reps v18 v19 v20::xs) =
   getOmniCommand xs) ∧
(∀ xs v22 v21.
   getOmniCommand (v21 domi v22::xs) = getOmniCommand xs) ∧
(∀ xs v24 v23.
   getOmniCommand (v23 eqi v24::xs) = getOmniCommand xs) ∧
(∀ xs v26 v25.
   getOmniCommand (v25 doms v26::xs) = getOmniCommand xs) ∧
(∀ xs v28 v27.
   getOmniCommand (v27 eqs v28::xs) = getOmniCommand xs) ∧
(∀ xs v30 v29.
      getOmniCommand ($v_{29}$ eqn $v_{30}$::$xs$) = getOmniCommand $xs$) ∧
   (∀ $xs$ $v_{32}$ $v_{31}$.
      getOmniCommand ($v_{31}$ lte $v_{32}$::$xs$) = getOmniCommand $xs$) ∧
   ∀ $xs$ $v_{34}$ $v_{33}$.
     getOmniCommand ($v_{33}$ lt $v_{34}$::$xs$) = getOmniCommand $xs$
```

[getOmniCommand_ind]

```
 ⊢ ∀ P.
     P [] ∧
     (∀ cmd xs.
        P (Name Omni says prop (SOME (SLc (OMNI cmd)))::xs)) ∧
     (∀ xs. P xs ⇒ P (TT::xs)) ∧ (∀ xs. P xs ⇒ P (FF::xs)) ∧
     (∀ $v_2$ xs. P xs ⇒ P (prop $v_2$::xs)) ∧
     (∀ $v_3$ xs. P xs ⇒ P (notf $v_3$::xs)) ∧
     (∀ $v_4$ $v_5$ xs. P xs ⇒ P ($v_4$ andf $v_5$::xs)) ∧
     (∀ $v_6$ $v_7$ xs. P xs ⇒ P ($v_6$ orf $v_7$::xs)) ∧
     (∀ $v_8$ $v_9$ xs. P xs ⇒ P ($v_8$ impf $v_9$::xs)) ∧
     (∀ $v_{10}$ $v_{11}$ xs. P xs ⇒ P ($v_{10}$ eqf $v_{11}$::xs)) ∧
     (∀ $v_{12}$ xs. P xs ⇒ P ($v_{12}$ says TT::xs)) ∧
     (∀ $v_{12}$ xs. P xs ⇒ P ($v_{12}$ says FF::xs)) ∧
     (∀ v134 xs. P xs ⇒ P (Name v134 says prop NONE::xs)) ∧
     (∀ v144 xs.
        P xs ⇒
        P (Name PlatoonLeader says prop (SOME v144)::xs)) ∧
     (∀ v146 xs.
        P xs ⇒ P (Name Omni says prop (SOME (ESCc v146))::xs)) ∧
     (∀ v150 xs.
        P xs ⇒
        P (Name Omni says prop (SOME (SLc (PL v150)))::xs)) ∧
     (∀ v135 v136 $v_{68}$ xs.
        P xs ⇒ P (v135 meet v136 says prop $v_{68}$::xs)) ∧
     (∀ v137 v138 $v_{68}$ xs.
        P xs ⇒ P (v137 quoting v138 says prop $v_{68}$::xs)) ∧
     (∀ $v_{12}$ $v_{69}$ xs. P xs ⇒ P ($v_{12}$ says notf $v_{69}$::xs)) ∧
     (∀ $v_{12}$ $v_{70}$ $v_{71}$ xs. P xs ⇒ P ($v_{12}$ says ($v_{70}$ andf $v_{71}$)::xs)) ∧
     (∀ $v_{12}$ $v_{72}$ $v_{73}$ xs. P xs ⇒ P ($v_{12}$ says ($v_{72}$ orf $v_{73}$)::xs)) ∧
     (∀ $v_{12}$ $v_{74}$ $v_{75}$ xs. P xs ⇒ P ($v_{12}$ says ($v_{74}$ impf $v_{75}$)::xs)) ∧
     (∀ $v_{12}$ $v_{76}$ $v_{77}$ xs. P xs ⇒ P ($v_{12}$ says ($v_{76}$ eqf $v_{77}$)::xs)) ∧
     (∀ $v_{12}$ $v_{78}$ $v_{79}$ xs. P xs ⇒ P ($v_{12}$ says $v_{78}$ says $v_{79}$::xs)) ∧
     (∀ $v_{12}$ $v_{80}$ $v_{81}$ xs.
        P xs ⇒ P ($v_{12}$ says $v_{80}$ speaks_for $v_{81}$::xs)) ∧
     (∀ $v_{12}$ $v_{82}$ $v_{83}$ xs.
        P xs ⇒ P ($v_{12}$ says $v_{82}$ controls $v_{83}$::xs)) ∧
     (∀ $v_{12}$ $v_{84}$ $v_{85}$ $v_{86}$ xs.
        P xs ⇒ P ($v_{12}$ says reps $v_{84}$ $v_{85}$ $v_{86}$::xs)) ∧
     (∀ $v_{12}$ $v_{87}$ $v_{88}$ xs. P xs ⇒ P ($v_{12}$ says $v_{87}$ domi $v_{88}$::xs)) ∧
     (∀ $v_{12}$ $v_{89}$ $v_{90}$ xs. P xs ⇒ P ($v_{12}$ says $v_{89}$ eqi $v_{90}$::xs)) ∧
     (∀ $v_{12}$ $v_{91}$ $v_{92}$ xs. P xs ⇒ P ($v_{12}$ says $v_{91}$ doms $v_{92}$::xs)) ∧
     (∀ $v_{12}$ $v_{93}$ $v_{94}$ xs. P xs ⇒ P ($v_{12}$ says $v_{93}$ eqs $v_{94}$::xs)) ∧
```

$$\begin{array}{l}
(\forall v_{12}\ v_{95}\ v_{96}\ xs.\ P\ xs \Rightarrow P\ (v_{12}\ \mathtt{says}\ v_{95}\ \mathtt{eqn}\ v_{96}\mathtt{::}xs))\ \land \\
(\forall v_{12}\ v_{97}\ v_{98}\ xs.\ P\ xs \Rightarrow P\ (v_{12}\ \mathtt{says}\ v_{97}\ \mathtt{lte}\ v_{98}\mathtt{::}xs))\ \land \\
(\forall v_{12}\ v_{99}\ v100\ xs.\ P\ xs \Rightarrow P\ (v_{12}\ \mathtt{says}\ v_{99}\ \mathtt{lt}\ v100\mathtt{::}xs))\ \land \\
(\forall v_{14}\ v_{15}\ xs.\ P\ xs \Rightarrow P\ (v_{14}\ \mathtt{speaks\_for}\ v_{15}\mathtt{::}xs))\ \land \\
(\forall v_{16}\ v_{17}\ xs.\ P\ xs \Rightarrow P\ (v_{16}\ \mathtt{controls}\ v_{17}\mathtt{::}xs))\ \land \\
(\forall v_{18}\ v_{19}\ v_{20}\ xs.\ P\ xs \Rightarrow P\ (\mathtt{reps}\ v_{18}\ v_{19}\ v_{20}\mathtt{::}xs))\ \land \\
(\forall v_{21}\ v_{22}\ xs.\ P\ xs \Rightarrow P\ (v_{21}\ \mathtt{domi}\ v_{22}\mathtt{::}xs))\ \land \\
(\forall v_{23}\ v_{24}\ xs.\ P\ xs \Rightarrow P\ (v_{23}\ \mathtt{eqi}\ v_{24}\mathtt{::}xs))\ \land \\
(\forall v_{25}\ v_{26}\ xs.\ P\ xs \Rightarrow P\ (v_{25}\ \mathtt{doms}\ v_{26}\mathtt{::}xs))\ \land \\
(\forall v_{27}\ v_{28}\ xs.\ P\ xs \Rightarrow P\ (v_{27}\ \mathtt{eqs}\ v_{28}\mathtt{::}xs))\ \land \\
(\forall v_{29}\ v_{30}\ xs.\ P\ xs \Rightarrow P\ (v_{29}\ \mathtt{eqn}\ v_{30}\mathtt{::}xs))\ \land \\
(\forall v_{31}\ v_{32}\ xs.\ P\ xs \Rightarrow P\ (v_{31}\ \mathtt{lte}\ v_{32}\mathtt{::}xs))\ \land \\
(\forall v_{33}\ v_{34}\ xs.\ P\ xs \Rightarrow P\ (v_{33}\ \mathtt{lt}\ v_{34}\mathtt{::}xs)) \Rightarrow \\
\forall v.\ P\ v
\end{array}$$

[getPlCom_def]

$$\begin{array}{l}
\vdash (\mathtt{getPlCom}\ \mathtt{[]}\ \mathtt{=}\ \mathtt{incomplete})\ \land \\
\quad (\forall xs\ cmd.\ \mathtt{getPlCom}\ (\mathtt{SOME}\ (\mathtt{SLc}\ (\mathtt{PL}\ cmd))\mathtt{::}xs)\ \mathtt{=}\ cmd)\ \land \\
\quad (\forall xs.\ \mathtt{getPlCom}\ (\mathtt{NONE}\mathtt{::}xs)\ \mathtt{=}\ \mathtt{getPlCom}\ xs)\ \land \\
\quad (\forall xs\ v_4.\ \mathtt{getPlCom}\ (\mathtt{SOME}\ (\mathtt{ESCc}\ v_4)\mathtt{::}xs)\ \mathtt{=}\ \mathtt{getPlCom}\ xs)\ \land \\
\quad \forall xs\ v_9.\ \mathtt{getPlCom}\ (\mathtt{SOME}\ (\mathtt{SLc}\ (\mathtt{OMNI}\ v_9))\mathtt{::}xs)\ \mathtt{=}\ \mathtt{getPlCom}\ xs
\end{array}$$

[getPlCom_ind]

$$\begin{array}{l}
\vdash \forall P. \\
\qquad P\ \mathtt{[]}\ \land\ (\forall cmd\ xs.\ P\ (\mathtt{SOME}\ (\mathtt{SLc}\ (\mathtt{PL}\ cmd))\mathtt{::}xs))\ \land \\
\qquad (\forall xs.\ P\ xs \Rightarrow P\ (\mathtt{NONE}\mathtt{::}xs))\ \land \\
\qquad (\forall v_4\ xs.\ P\ xs \Rightarrow P\ (\mathtt{SOME}\ (\mathtt{ESCc}\ v_4)\mathtt{::}xs))\ \land \\
\qquad (\forall v_9\ xs.\ P\ xs \Rightarrow P\ (\mathtt{SOME}\ (\mathtt{SLc}\ (\mathtt{OMNI}\ v_9))\mathtt{::}xs)) \Rightarrow \\
\qquad \forall v.\ P\ v
\end{array}$$

7 ssmPBIntegrated Theory

Built: 11 June 2018

Parent Theories: PBIntegratedDef, ssm

7.1 Theorems

[inputOK_cmd_reject_lemma]

$$\vdash \forall cmd.\ \neg\mathtt{inputOK}\ (\mathtt{prop}\ (\mathtt{SOME}\ cmd))$$

[inputOK_def]

$$\begin{array}{l}
\vdash (\mathtt{inputOK}\ (\mathtt{Name\ PlatoonLeader\ says\ prop}\ cmd) \iff \mathtt{T})\ \land \\
\quad (\mathtt{inputOK}\ (\mathtt{Name\ Omni\ says\ prop}\ cmd) \iff \mathtt{T})\ \land \\
\quad (\mathtt{inputOK}\ \mathtt{TT} \iff \mathtt{F})\ \land\ (\mathtt{inputOK}\ \mathtt{FF} \iff \mathtt{F})\ \land \\
\quad (\mathtt{inputOK}\ (\mathtt{prop}\ v) \iff \mathtt{F})\ \land\ (\mathtt{inputOK}\ (\mathtt{notf}\ v_1) \iff \mathtt{F})\ \land \\
\quad (\mathtt{inputOK}\ (v_2\ \mathtt{andf}\ v_3) \iff \mathtt{F})\ \land\ (\mathtt{inputOK}\ (v_4\ \mathtt{orf}\ v_5) \iff \mathtt{F})\ \land \\
\quad (\mathtt{inputOK}\ (v_6\ \mathtt{impf}\ v_7) \iff \mathtt{F})\ \land\ (\mathtt{inputOK}\ (v_8\ \mathtt{eqf}\ v_9) \iff \mathtt{F})\ \land
\end{array}$$

(inputOK (v_{10} says TT) $\iff$ F) $\wedge$ (inputOK (v_{10} says FF) $\iff$ F) $\wedge$
(inputOK ($v133$ meet $v134$ says prop v_{66}) $\iff$ F) $\wedge$
(inputOK ($v135$ quoting $v136$ says prop v_{66}) $\iff$ F) $\wedge$
(inputOK (v_{10} says notf v_{67}) $\iff$ F) $\wedge$
(inputOK (v_{10} says (v_{68} andf v_{69})) $\iff$ F) $\wedge$
(inputOK (v_{10} says (v_{70} orf v_{71})) $\iff$ F) $\wedge$
(inputOK (v_{10} says (v_{72} impf v_{73})) $\iff$ F) $\wedge$
(inputOK (v_{10} says (v_{74} eqf v_{75})) $\iff$ F) $\wedge$
(inputOK (v_{10} says v_{76} says v_{77}) $\iff$ F) $\wedge$
(inputOK (v_{10} says v_{78} speaks_for v_{79}) $\iff$ F) $\wedge$
(inputOK (v_{10} says v_{80} controls v_{81}) $\iff$ F) $\wedge$
(inputOK (v_{10} says reps v_{82} v_{83} v_{84}) $\iff$ F) $\wedge$
(inputOK (v_{10} says v_{85} domi v_{86}) $\iff$ F) $\wedge$
(inputOK (v_{10} says v_{87} eqi v_{88}) $\iff$ F) $\wedge$
(inputOK (v_{10} says v_{89} doms v_{90}) $\iff$ F) $\wedge$
(inputOK (v_{10} says v_{91} eqs v_{92}) $\iff$ F) $\wedge$
(inputOK (v_{10} says v_{93} eqn v_{94}) $\iff$ F) $\wedge$
(inputOK (v_{10} says v_{95} lte v_{96}) $\iff$ F) $\wedge$
(inputOK (v_{10} says v_{97} lt v_{98}) $\iff$ F) $\wedge$
(inputOK (v_{12} speaks_for v_{13}) $\iff$ F) $\wedge$
(inputOK (v_{14} controls v_{15}) $\iff$ F) $\wedge$
(inputOK (reps v_{16} v_{17} v_{18}) $\iff$ F) $\wedge$
(inputOK (v_{19} domi v_{20}) $\iff$ F) $\wedge$
(inputOK (v_{21} eqi v_{22}) $\iff$ F) $\wedge$
(inputOK (v_{23} doms v_{24}) $\iff$ F) $\wedge$
(inputOK (v_{25} eqs v_{26}) $\iff$ F) $\wedge$ (inputOK (v_{27} eqn v_{28}) $\iff$ F) $\wedge$
(inputOK (v_{29} lte v_{30}) $\iff$ F) $\wedge$ (inputOK (v_{31} lt v_{32}) $\iff$ F)

[inputOK_ind]

$\vdash$ $\forall P$.
($\forall$ cmd. P (Name PlatoonLeader says prop cmd)) $\wedge$
($\forall$ cmd. P (Name Omni says prop cmd)) $\wedge$ P TT $\wedge$ P FF $\wedge$
($\forall$ v. P (prop v)) $\wedge$ ($\forall$ v_1. P (notf v_1)) $\wedge$
($\forall$ v_2 v_3. P (v_2 andf v_3)) $\wedge$ ($\forall$ v_4 v_5. P (v_4 orf v_5)) $\wedge$
($\forall$ v_6 v_7. P (v_6 impf v_7)) $\wedge$ ($\forall$ v_8 v_9. P (v_8 eqf v_9)) $\wedge$
($\forall$ v_{10}. P (v_{10} says TT)) $\wedge$ ($\forall$ v_{10}. P (v_{10} says FF)) $\wedge$
($\forall$ $v133$ $v134$ v_{66}. P ($v133$ meet $v134$ says prop v_{66})) $\wedge$
($\forall$ $v135$ $v136$ v_{66}. P ($v135$ quoting $v136$ says prop v_{66})) $\wedge$
($\forall$ v_{10} v_{67}. P (v_{10} says notf v_{67})) $\wedge$
($\forall$ v_{10} v_{68} v_{69}. P (v_{10} says (v_{68} andf v_{69}))) $\wedge$
($\forall$ v_{10} v_{70} v_{71}. P (v_{10} says (v_{70} orf v_{71}))) $\wedge$
($\forall$ v_{10} v_{72} v_{73}. P (v_{10} says (v_{72} impf v_{73}))) $\wedge$
($\forall$ v_{10} v_{74} v_{75}. P (v_{10} says (v_{74} eqf v_{75}))) $\wedge$
($\forall$ v_{10} v_{76} v_{77}. P (v_{10} says v_{76} says v_{77})) $\wedge$
($\forall$ v_{10} v_{78} v_{79}. P (v_{10} says v_{78} speaks_for v_{79})) $\wedge$
($\forall$ v_{10} v_{80} v_{81}. P (v_{10} says v_{80} controls v_{81})) $\wedge$
($\forall$ v_{10} v_{82} v_{83} v_{84}. P (v_{10} says reps v_{82} v_{83} v_{84})) $\wedge$
($\forall$ v_{10} v_{85} v_{86}. P (v_{10} says v_{85} domi v_{86})) $\wedge$
($\forall$ v_{10} v_{87} v_{88}. P (v_{10} says v_{87} eqi v_{88})) $\wedge$

```
     (∀ $v_{10}$ $v_{89}$ $v_{90}$. $P$ ($v_{10}$ says $v_{89}$ doms $v_{90}$)) ∧
     (∀ $v_{10}$ $v_{91}$ $v_{92}$. $P$ ($v_{10}$ says $v_{91}$ eqs $v_{92}$)) ∧
     (∀ $v_{10}$ $v_{93}$ $v_{94}$. $P$ ($v_{10}$ says $v_{93}$ eqn $v_{94}$)) ∧
     (∀ $v_{10}$ $v_{95}$ $v_{96}$. $P$ ($v_{10}$ says $v_{95}$ lte $v_{96}$)) ∧
     (∀ $v_{10}$ $v_{97}$ $v_{98}$. $P$ ($v_{10}$ says $v_{97}$ lt $v_{98}$)) ∧
     (∀ $v_{12}$ $v_{13}$. $P$ ($v_{12}$ speaks_for $v_{13}$)) ∧
     (∀ $v_{14}$ $v_{15}$. $P$ ($v_{14}$ controls $v_{15}$)) ∧
     (∀ $v_{16}$ $v_{17}$ $v_{18}$. $P$ (reps $v_{16}$ $v_{17}$ $v_{18}$)) ∧
     (∀ $v_{19}$ $v_{20}$. $P$ ($v_{19}$ domi $v_{20}$)) ∧
     (∀ $v_{21}$ $v_{22}$. $P$ ($v_{21}$ eqi $v_{22}$)) ∧
     (∀ $v_{23}$ $v_{24}$. $P$ ($v_{23}$ doms $v_{24}$)) ∧
     (∀ $v_{25}$ $v_{26}$. $P$ ($v_{25}$ eqs $v_{26}$)) ∧ (∀ $v_{27}$ $v_{28}$. $P$ ($v_{27}$ eqn $v_{28}$)) ∧
     (∀ $v_{29}$ $v_{30}$. $P$ ($v_{29}$ lte $v_{30}$)) ∧ (∀ $v_{31}$ $v_{32}$. $P$ ($v_{31}$ lt $v_{32}$)) ⇒
     ∀ $v$. $P$ $v$
```

[PBNS_def]

```
 ⊢ (PBNS PLAN_PB (exec $x$) =
    if getPlCom $x$ = crossLD then MOVE_TO_ORP else PLAN_PB) ∧
   (PBNS MOVE_TO_ORP (exec $x$) =
    if getPlCom $x$ = conductORP then CONDUCT_ORP
    else MOVE_TO_ORP) ∧
   (PBNS CONDUCT_ORP (exec $x$) =
    if getPlCom $x$ = moveToPB then MOVE_TO_PB else CONDUCT_ORP) ∧
   (PBNS MOVE_TO_PB (exec $x$) =
    if getPlCom $x$ = conductPB then CONDUCT_PB else MOVE_TO_PB) ∧
   (PBNS CONDUCT_PB (exec $x$) =
    if getPlCom $x$ = completePB then COMPLETE_PB
    else CONDUCT_PB) ∧ (PBNS $s$ (trap $v_0$) = $s$) ∧
   (PBNS $s$ (discard $v_1$) = $s$)
```

[PBNS_ind]

```
 ⊢ ∀ $P$.
     (∀ $x$. $P$ PLAN_PB (exec $x$)) ∧ (∀ $x$. $P$ MOVE_TO_ORP (exec $x$)) ∧
     (∀ $x$. $P$ CONDUCT_ORP (exec $x$)) ∧
     (∀ $x$. $P$ MOVE_TO_PB (exec $x$)) ∧ (∀ $x$. $P$ CONDUCT_PB (exec $x$)) ∧
     (∀ $s$ $v_0$. $P$ $s$ (trap $v_0$)) ∧ (∀ $s$ $v_1$. $P$ $s$ (discard $v_1$)) ∧
     (∀ $v_6$. $P$ COMPLETE_PB (exec $v_6$)) ⇒
     ∀ $v$ $v_1$. $P$ $v$ $v_1$
```

[PBOut_def]

```
 ⊢ (PBOut PLAN_PB (exec $x$) =
    if getPlCom $x$ = crossLD then MoveToORP else PlanPB) ∧
   (PBOut MOVE_TO_ORP (exec $x$) =
    if getPlCom $x$ = conductORP then ConductORP else MoveToORP) ∧
   (PBOut CONDUCT_ORP (exec $x$) =
    if getPlCom $x$ = moveToPB then MoveToORP else ConductORP) ∧
   (PBOut MOVE_TO_PB (exec $x$) =
    if getPlCom $x$ = conductPB then ConductPB else MoveToPB) ∧
   (PBOut CONDUCT_PB (exec x) =
    if getPlCom x = completePB then CompletePB else ConductPB) ∧
   (PBOut s (trap v0) = unAuthorized) ∧
   (PBOut s (discard v1) = unAuthenticated)
```

[PBOut_ind]

```
 ⊢ ∀P.
     (∀x. P PLAN_PB (exec x)) ∧ (∀x. P MOVE_TO_ORP (exec x)) ∧
     (∀x. P CONDUCT_ORP (exec x)) ∧
     (∀x. P MOVE_TO_PB (exec x)) ∧ (∀x. P CONDUCT_PB (exec x)) ∧
     (∀s v0. P s (trap v0)) ∧ (∀s v1. P s (discard v1)) ∧
     (∀v6. P COMPLETE_PB (exec v6)) ⇒
     ∀v v1. P v v1
```

[PlatoonLeader_Omni_notDiscard_slCommand_thm]

```
 ⊢ ∀NS Out M Oi Os.
     ¬TR (M,Oi,Os)
        (discard
           [SOME (SLc (PL plCommand));
            SOME (SLc (OMNI omniCommand))])
        (CFG inputOK secContext secAuthorization
           ([Name Omni says prop (SOME (SLc (PL plCommand)));
             Name PlatoonLeader says
             prop (SOME (SLc (OMNI omniCommand)))]::ins) PLAN_PB
           outs)
        (CFG inputOK secContext secAuthorization ins
           (NS PLAN_PB
              (discard
                 [SOME (SLc (PL plCommand));
                  SOME (SLc (OMNI omniCommand))]))
           (Out PLAN_PB
              (discard
                 [SOME (SLc (PL plCommand));
                  SOME (SLc (OMNI omniCommand))])::outs))
```

[PlatoonLeader_PLAN_PB_exec_justified_lemma]

```
 ⊢ ∀NS Out M Oi Os.
     TR (M,Oi,Os)
       (exec
          (inputList
             [Name Omni says
              prop (SOME (SLc (OMNI ssmPlanPBComplete)));
              Name PlatoonLeader says
              prop (SOME (SLc (PL crossLD)))]))
       (CFG inputOK secContext secAuthorization
          ([Name Omni says
            prop (SOME (SLc (OMNI ssmPlanPBComplete)));
            Name PlatoonLeader says
          prop (SOME (SLc (PL crossLD)))]::ins) PLAN_PB outs)
      (CFG inputOK secContext secAuthorization ins
         (NS PLAN_PB
            (exec
               (inputList
                  [Name Omni says
                   prop (SOME (SLc (OMNI ssmPlanPBComplete)));
                   Name PlatoonLeader says
                   prop (SOME (SLc (PL crossLD)))])))
         (Out PLAN_PB
            (exec
               (inputList
                  [Name Omni says
                   prop (SOME (SLc (OMNI ssmPlanPBComplete)));
                   Name PlatoonLeader says
                   prop (SOME (SLc (PL crossLD)))]))::outs)) ⟺
    authenticationTest inputOK
      [Name Omni says
       prop (SOME (SLc (OMNI ssmPlanPBComplete)));
       Name PlatoonLeader says
       prop (SOME (SLc (PL crossLD)))] ∧
    CFGInterpret (M,Oi,Os)
      (CFG inputOK secContext secAuthorization
         ([Name Omni says
           prop (SOME (SLc (OMNI ssmPlanPBComplete)));
           Name PlatoonLeader says
           prop (SOME (SLc (PL crossLD)))]::ins) PLAN_PB
         outs) ∧
    (M,Oi,Os) satList
    propCommandList
      [Name Omni says
       prop (SOME (SLc (OMNI ssmPlanPBComplete)));
       Name PlatoonLeader says prop (SOME (SLc (PL crossLD)))]
```

[PlatoonLeader_PLAN_PB_exec_justified_thm]

```
 ⊢ ∀ NS Out M Oi Os.
    TR (M,Oi,Os)
      (exec
         [SOME (SLc (OMNI ssmPlanPBComplete));
          SOME (SLc (PL crossLD))])
      (CFG inputOK secContext secAuthorization
         ([Name Omni says
           prop (SOME (SLc (OMNI ssmPlanPBComplete)));
           Name PlatoonLeader says
           prop (SOME (SLc (PL crossLD)))]::ins) PLAN_PB outs)
      (CFG inputOK secContext secAuthorization ins
         (NS PLAN_PB
            (exec
               [SOME (SLc (OMNI ssmPlanPBComplete));
```

```
                SOME (SLc (PL crossLD))]))
         (Out PLAN_PB
            (exec
               [SOME (SLc (OMNI ssmPlanPBComplete));
                SOME (SLc (PL crossLD))])::outs)) ⟺
     authenticationTest inputOK
       [Name Omni says
        prop (SOME (SLc (OMNI ssmPlanPBComplete)));
        Name PlatoonLeader says
        prop (SOME (SLc (PL crossLD)))] ∧
     CFGInterpret (M,Oi,Os)
       (CFG inputOK secContext secAuthorization
          ([Name Omni says
            prop (SOME (SLc (OMNI ssmPlanPBComplete)));
            Name PlatoonLeader says
            prop (SOME (SLc (PL crossLD)))]::ins) PLAN_PB
          outs) ∧
     (M,Oi,Os) satList
     [prop (SOME (SLc (OMNI ssmPlanPBComplete)));
      prop (SOME (SLc (PL crossLD)))]
```

[PlatoonLeader_PLAN_PB_exec_lemma]

```
 ⊢ ∀ M  Oi  Os.
     CFGInterpret (M,Oi,Os)
       (CFG inputOK secContext secAuthorization
          ([Name Omni says
            prop (SOME (SLc (OMNI ssmPlanPBComplete)));
            Name PlatoonLeader says
            prop (SOME (SLc (PL crossLD)))]::ins) PLAN_PB
          outs) ⇒
     (M,Oi,Os) satList
     propCommandList
       [Name Omni says
        prop (SOME (SLc (OMNI ssmPlanPBComplete)));
        Name PlatoonLeader says prop (SOME (SLc (PL crossLD)))]
```

[PlatoonLeader_PLAN_PB_trap_justified_lemma]

```
 ⊢ omniCommand ≠ ssmPlanPBComplete ⇒
   (s = PLAN_PB) ⇒
   ∀ NS  Out  M  Oi  Os.
     TR (M,Oi,Os)
       (trap
          (inputList
             [Name Omni says
              prop (SOME (SLc (OMNI omniCommand)));
              Name PlatoonLeader says
              prop (SOME (SLc (PL crossLD)))]))
       (CFG inputOK secContext secAuthorization
          ([Name Omni says prop (SOME (SLc (OMNI omniCommand)));
          Name PlatoonLeader says
          prop (SOME (SLc (PL crossLD)))]::ins) PLAN_PB outs)
     (CFG inputOK secContext secAuthorization ins
        (NS PLAN_PB
           (trap
              (inputList
                 [Name Omni says
                  prop (SOME (SLc (OMNI omniCommand)));
                  Name PlatoonLeader says
                  prop (SOME (SLc (PL crossLD)))])))
        (Out PLAN_PB
           (trap
              (inputList
                 [Name Omni says
                  prop (SOME (SLc (OMNI omniCommand)));
                  Name PlatoonLeader says
                  prop (SOME (SLc (PL crossLD)))]))::outs)) ⇔
   authenticationTest inputOK
     [Name Omni says prop (SOME (SLc (OMNI omniCommand)));
      Name PlatoonLeader says
      prop (SOME (SLc (PL crossLD)))] ∧
   CFGInterpret (M,Oi,Os)
     (CFG inputOK secContext secAuthorization
        ([Name Omni says prop (SOME (SLc (OMNI omniCommand)));
          Name PlatoonLeader says
          prop (SOME (SLc (PL crossLD)))]::ins) PLAN_PB
        outs) ∧ (M,Oi,Os) sat prop NONE
```

[PlatoonLeader_PLAN_PB_trap_justified_thm]

```
 ⊢ omniCommand ≠ ssmPlanPBComplete ⇒
   (s = PLAN_PB) ⇒
   ∀ NS Out M Oi Os.
     TR (M,Oi,Os)
       (trap
          [SOME (SLc (OMNI omniCommand));
           SOME (SLc (PL crossLD))])
       (CFG inputOK secContext secAuthorization
          ([Name Omni says prop (SOME (SLc (OMNI omniCommand)));
            Name PlatoonLeader says
            prop (SOME (SLc (PL crossLD)))]::ins) PLAN_PB outs)
       (CFG inputOK secContext secAuthorization ins
          (NS PLAN_PB
             (trap
                [SOME (SLc (OMNI omniCommand));
                 SOME (SLc (PL crossLD))]))
          (Out PLAN_PB
             (trap
                [SOME (SLc (OMNI omniCommand));
                 SOME (SLc (PL crossLD))])::outs)) ⇔
     authenticationTest inputOK
       [Name Omni says prop (SOME (SLc (OMNI omniCommand)));
        Name PlatoonLeader says
        prop (SOME (SLc (PL crossLD)))] ∧
     CFGInterpret (M,Oi,Os)
       (CFG inputOK secContext secAuthorization
          ([Name Omni says prop (SOME (SLc (OMNI omniCommand)));
            Name PlatoonLeader says
            prop (SOME (SLc (PL crossLD)))]::ins) PLAN_PB
          outs) ∧ (M,Oi,Os) sat prop NONE
```

[PlatoonLeader_PLAN_PB_trap_lemma]

```
 ⊢ omniCommand ≠ ssmPlanPBComplete ⇒
   (s = PLAN_PB) ⇒
   ∀M Oi Os.
     CFGInterpret (M,Oi,Os)
       (CFG inputOK secContext secAuthorization
          ([Name Omni says prop (SOME (SLc (OMNI omniCommand)));
            Name PlatoonLeader says
            prop (SOME (SLc (PL crossLD)))]::ins) PLAN_PB
          outs) ⇒
     (M,Oi,Os) sat prop NONE
```

8 ssmConductORP Theory

Built: 11 June 2018

Parent Theories: ConductORPDef

8.1 Theorems

[conductORPNS_def]

```
 ⊢ (conductORPNS CONDUCT_ORP (exec x) =
    if getPlCom x = secure then SECURE else CONDUCT_ORP) ∧
   (conductORPNS SECURE (exec x) =
    if getPsgCom x = actionsIn then ACTIONS_IN else SECURE) ∧
   (conductORPNS ACTIONS_IN (exec x) =
    if getPlCom x = withdraw then WITHDRAW else ACTIONS_IN) ∧
   (conductORPNS WITHDRAW (exec x) =
    if getPlCom x = complete then COMPLETE else WITHDRAW) ∧
   (conductORPNS s (trap x) = s) ∧
   (conductORPNS s (discard x) = s)
```

[conductORPNS_ind]

```
 ⊢ ∀P.
     (∀x. P CONDUCT_ORP (exec x)) ∧ (∀x. P SECURE (exec x)) ∧
     (∀x. P ACTIONS_IN (exec x)) ∧ (∀x. P WITHDRAW (exec x)) ∧
     (∀s x. P s (trap x)) ∧ (∀s x. P s (discard x)) ∧
```

$(\forall v_5.\ P$ `COMPLETE (exec` $v_5)) \Rightarrow$
$\forall v\ v_1.\ P\ v\ v_1$

[conductORPOut_def]

$\vdash$ `(conductORPOut CONDUCT_ORP (exec` x`) =`
if `getPlCom` x `= secure` **then** `Secure` **else** `ConductORP)` $\wedge$
`(conductORPOut SECURE (exec` x`) =`
if `getPsgCom` x `= actionsIn` **then** `ActionsIn` **else** `Secure)` $\wedge$
`(conductORPOut ACTIONS_IN (exec` x`) =`
if `getPlCom` x `= withdraw` **then** `Withdraw` **else** `ActionsIn)` $\wedge$
`(conductORPOut WITHDRAW (exec` x`) =`
if `getPlCom` x `= complete` **then** `Complete` **else** `Withdraw)` $\wedge$
`(conductORPOut` s `(trap` x`) = unAuthorized)` $\wedge$
`(conductORPOut` s `(discard` x`) = unAuthenticated)`

[conductORPOut_ind]

$\vdash \forall P.$
$(\forall x.\ P$ `CONDUCT_ORP (exec` $x)) \wedge (\forall x.\ P$ `SECURE (exec` $x)) \wedge$
$(\forall x.\ P$ `ACTIONS_IN (exec` $x)) \wedge (\forall x.\ P$ `WITHDRAW (exec` $x)) \wedge$
$(\forall s\ x.\ P\ s$ `(trap` $x)) \wedge (\forall s\ x.\ P\ s$ `(discard` $x)) \wedge$
$(\forall v_5.\ P$ `COMPLETE (exec` $v_5)) \Rightarrow$
$\forall v\ v_1.\ P\ v\ v_1$

[inputOK_cmd_reject_lemma]

$\vdash \forall cmd.\ \neg$`inputOK (prop (SOME` cmd`))`

[inputOK_def]

$\vdash$ `(inputOK (Name PlatoonLeader says prop` cmd`)` $\iff$ `T)` $\wedge$
`(inputOK (Name PlatoonSergeant says prop` cmd`)` $\iff$ `T)` $\wedge$
`(inputOK (Name Omni says prop` cmd`)` $\iff$ `T)` $\wedge$
`(inputOK TT` $\iff$ `F)` $\wedge$ `(inputOK FF` $\iff$ `F)` $\wedge$
`(inputOK (prop` v`)` $\iff$ `F)` $\wedge$ `(inputOK (notf` v_1`)` $\iff$ `F)` $\wedge$
`(inputOK (`v_2 `andf` v_3`)` $\iff$ `F)` $\wedge$ `(inputOK (`v_4 `orf` v_5`)` $\iff$ `F)` $\wedge$
`(inputOK (`v_6 `impf` v_7`)` $\iff$ `F)` $\wedge$ `(inputOK (`v_8 `eqf` v_9`)` $\iff$ `F)` $\wedge$
`(inputOK (`v_{10} `says TT)` $\iff$ `F)` $\wedge$ `(inputOK (`v_{10} `says FF)` $\iff$ `F)` $\wedge$
`(inputOK (`$v133$ `meet` $v134$ `says prop` v_{66}`)` $\iff$ `F)` $\wedge$
`(inputOK (`$v135$ `quoting` $v136$ `says prop` v_{66}`)` $\iff$ `F)` $\wedge$
`(inputOK (`v_{10} `says notf` v_{67}`)` $\iff$ `F)` $\wedge$
`(inputOK (`v_{10} `says (`v_{68} `andf` v_{69}`))` $\iff$ `F)` $\wedge$
`(inputOK (`v_{10} `says (`v_{70} `orf` v_{71}`))` $\iff$ `F)` $\wedge$
`(inputOK (`v_{10} `says (`v_{72} `impf` v_{73}`))` $\iff$ `F)` $\wedge$
`(inputOK (`v_{10} `says (`v_{74} `eqf` v_{75}`))` $\iff$ `F)` $\wedge$
`(inputOK (`v_{10} `says` v_{76} `says` v_{77}`)` $\iff$ `F)` $\wedge$
`(inputOK (`v_{10} `says` v_{78} `speaks_for` v_{79}`)` $\iff$ `F)` $\wedge$
`(inputOK (`v_{10} `says` v_{80} `controls` v_{81}`)` $\iff$ `F)` $\wedge$
`(inputOK (`v_{10} `says reps` v_{82} v_{83} v_{84}`)` $\iff$ `F)` $\wedge$
`(inputOK (`v_{10} `says` v_{85} `domi` v_{86}`)` $\iff$ `F)` $\wedge$
`(inputOK (`v_{10} `says` v_{87} `eqi` v_{88}`)` $\iff$ `F)` $\wedge$

(inputOK (v_{10} says v_{89} doms v_{90}) $\iff$ F) $\land$
(inputOK (v_{10} says v_{91} eqs v_{92}) $\iff$ F) $\land$
(inputOK (v_{10} says v_{93} eqn v_{94}) $\iff$ F) $\land$
(inputOK (v_{10} says v_{95} lte v_{96}) $\iff$ F) $\land$
(inputOK (v_{10} says v_{97} lt v_{98}) $\iff$ F) $\land$
(inputOK (v_{12} speaks_for v_{13}) $\iff$ F) $\land$
(inputOK (v_{14} controls v_{15}) $\iff$ F) $\land$
(inputOK (reps v_{16} v_{17} v_{18}) $\iff$ F) $\land$
(inputOK (v_{19} domi v_{20}) $\iff$ F) $\land$
(inputOK (v_{21} eqi v_{22}) $\iff$ F) $\land$
(inputOK (v_{23} doms v_{24}) $\iff$ F) $\land$
(inputOK (v_{25} eqs v_{26}) $\iff$ F) $\land$ (inputOK (v_{27} eqn v_{28}) $\iff$ F) $\land$
(inputOK (v_{29} lte v_{30}) $\iff$ F) $\land$ (inputOK (v_{31} lt v_{32}) $\iff$ F)

[inputOK_ind]

$\vdash \forall P$.
($\forall$ *cmd*. P (Name PlatoonLeader says prop *cmd*)) $\land$
($\forall$ *cmd*. P (Name PlatoonSergeant says prop *cmd*)) $\land$
($\forall$ *cmd*. P (Name Omni says prop *cmd*)) $\land$ P TT $\land$ P FF $\land$
($\forall v$. P (prop v)) $\land$ ($\forall v_1$. P (notf v_1)) $\land$
($\forall v_2$ v_3. P (v_2 andf v_3)) $\land$ ($\forall v_4$ v_5. P (v_4 orf v_5)) $\land$
($\forall v_6$ v_7. P (v_6 impf v_7)) $\land$ ($\forall v_8$ v_9. P (v_8 eqf v_9)) $\land$
($\forall v_{10}$. P (v_{10} says TT)) $\land$ ($\forall v_{10}$. P (v_{10} says FF)) $\land$
($\forall$ *v133* *v134* v_{66}. P (*v133* meet *v134* says prop v_{66})) $\land$
($\forall$ *v135* *v136* v_{66}. P (*v135* quoting *v136* says prop v_{66})) $\land$
($\forall v_{10}$ v_{67}. P (v_{10} says notf v_{67})) $\land$
($\forall v_{10}$ v_{68} v_{69}. P (v_{10} says (v_{68} andf v_{69}))) $\land$
($\forall v_{10}$ v_{70} v_{71}. P (v_{10} says (v_{70} orf v_{71}))) $\land$
($\forall v_{10}$ v_{72} v_{73}. P (v_{10} says (v_{72} impf v_{73}))) $\land$
($\forall v_{10}$ v_{74} v_{75}. P (v_{10} says (v_{74} eqf v_{75}))) $\land$
($\forall v_{10}$ v_{76} v_{77}. P (v_{10} says v_{76} says v_{77})) $\land$
($\forall v_{10}$ v_{78} v_{79}. P (v_{10} says v_{78} speaks_for v_{79})) $\land$
($\forall v_{10}$ v_{80} v_{81}. P (v_{10} says v_{80} controls v_{81})) $\land$
($\forall v_{10}$ v_{82} v_{83} v_{84}. P (v_{10} says reps v_{82} v_{83} v_{84})) $\land$
($\forall v_{10}$ v_{85} v_{86}. P (v_{10} says v_{85} domi v_{86})) $\land$
($\forall v_{10}$ v_{87} v_{88}. P (v_{10} says v_{87} eqi v_{88})) $\land$
($\forall v_{10}$ v_{89} v_{90}. P (v_{10} says v_{89} doms v_{90})) $\land$
($\forall v_{10}$ v_{91} v_{92}. P (v_{10} says v_{91} eqs v_{92})) $\land$
($\forall v_{10}$ v_{93} v_{94}. P (v_{10} says v_{93} eqn v_{94})) $\land$
($\forall v_{10}$ v_{95} v_{96}. P (v_{10} says v_{95} lte v_{96})) $\land$
($\forall v_{10}$ v_{97} v_{98}. P (v_{10} says v_{97} lt v_{98})) $\land$
($\forall v_{12}$ v_{13}. P (v_{12} speaks_for v_{13})) $\land$
($\forall v_{14}$ v_{15}. P (v_{14} controls v_{15})) $\land$
($\forall v_{16}$ v_{17} v_{18}. P (reps v_{16} v_{17} v_{18})) $\land$
($\forall v_{19}$ v_{20}. P (v_{19} domi v_{20})) $\land$
($\forall v_{21}$ v_{22}. P (v_{21} eqi v_{22})) $\land$
($\forall v_{23}$ v_{24}. P (v_{23} doms v_{24})) $\land$
($\forall v_{25}$ v_{26}. P (v_{25} eqs v_{26})) $\land$ ($\forall v_{27}$ v_{28}. P (v_{27} eqn v_{28})) $\land$
($\forall v_{29}$ v_{30}. P (v_{29} lte v_{30})) $\land$ ($\forall v_{31}$ v_{32}. P (v_{31} lt v_{32})) $\Rightarrow$

```
     ∀ v. P v

[PlatoonLeader_ACTIONS_IN_exec_justified_lemma]
 ⊢ ∀ NS Out M Oi Os.
     TR (M,Oi,Os)
       (exec
          (inputList
             [Name Omni says
              prop (SOME (SLc (OMNI ssmActionsInComplete)));
              Name PlatoonLeader says
              prop (SOME (SLc (PL withdraw)))]))
       (CFG inputOK secContext secAuthorization
          ([Name Omni says
            prop (SOME (SLc (OMNI ssmActionsInComplete)));
            Name PlatoonLeader says
            prop (SOME (SLc (PL withdraw)))]::ins) ACTIONS_IN
          outs)
       (CFG inputOK secContext secAuthorization ins
          (NS ACTIONS_IN
             (exec
                (inputList
                   [Name Omni says
                    prop
                      (SOME (SLc (OMNI ssmActionsInComplete)));
                    Name PlatoonLeader says
                    prop (SOME (SLc (PL withdraw)))])))
          (Out ACTIONS_IN
             (exec
                (inputList
                   [Name Omni says
                    prop
                      (SOME (SLc (OMNI ssmActionsInComplete)));
                    Name PlatoonLeader says
                    prop (SOME (SLc (PL withdraw)))]))::
                outs)) ⟺
     authenticationTest inputOK
       [Name Omni says
        prop (SOME (SLc (OMNI ssmActionsInComplete)));
        Name PlatoonLeader says
        prop (SOME (SLc (PL withdraw)))] ∧
     CFGInterpret (M,Oi,Os)
       (CFG inputOK secContext secAuthorization
          ([Name Omni says
            prop (SOME (SLc (OMNI ssmActionsInComplete)));
            Name PlatoonLeader says
            prop (SOME (SLc (PL withdraw)))]::ins) ACTIONS_IN
          outs) ∧
     (M,Oi,Os) satList
     propCommandList
```

```
      [Name Omni says
       prop (SOME (SLc (OMNI ssmActionsInComplete)));
       Name PlatoonLeader says prop (SOME (SLc (PL withdraw)))]
```

[PlatoonLeader_ACTIONS_IN_exec_justified_thm]

```
 ⊢ ∀NS Out M Oi Os.
     TR (M,Oi,Os)
       (exec
          [SOME (SLc (OMNI ssmActionsInComplete));
           SOME (SLc (PL withdraw))])
       (CFG inputOK secContext secAuthorization
          ([Name Omni says
            prop (SOME (SLc (OMNI ssmActionsInComplete)));
            Name PlatoonLeader says
            prop (SOME (SLc (PL withdraw)))]::ins) ACTIONS_IN
          outs)
       (CFG inputOK secContext secAuthorization ins
          (NS ACTIONS_IN
             (exec
                [SOME (SLc (OMNI ssmActionsInComplete));
                 SOME (SLc (PL withdraw))]))
          (Out ACTIONS_IN
             (exec
                [SOME (SLc (OMNI ssmActionsInComplete));
                 SOME (SLc (PL withdraw))])::outs)) ⟺
     authenticationTest inputOK
       [Name Omni says
        prop (SOME (SLc (OMNI ssmActionsInComplete)));
        Name PlatoonLeader says
        prop (SOME (SLc (PL withdraw)))] ∧
     CFGInterpret (M,Oi,Os)
       (CFG inputOK secContext secAuthorization
          ([Name Omni says
            prop (SOME (SLc (OMNI ssmActionsInComplete)));
            Name PlatoonLeader says
            prop (SOME (SLc (PL withdraw)))]::ins) ACTIONS_IN
          outs) ∧
     (M,Oi,Os) satList
     [prop (SOME (SLc (OMNI ssmActionsInComplete)));
      prop (SOME (SLc (PL withdraw)))]
```

[PlatoonLeader_ACTIONS_IN_exec_lemma]

```
 ⊢ ∀M Oi Os.
     CFGInterpret (M,Oi,Os)
       (CFG inputOK secContext secAuthorization
          ([Name Omni says
            prop (SOME (SLc (OMNI ssmActionsInComplete)));
            Name PlatoonLeader says
            prop (SOME (SLc (PL withdraw)))]::ins) ACTIONS_IN
          outs) ⇒
     (M,Oi,Os) satList
     propCommandList
       [Name Omni says
        prop (SOME (SLc (OMNI ssmActionsInComplete)));
        Name PlatoonLeader says prop (SOME (SLc (PL withdraw)))]
```

[PlatoonLeader_ACTIONS_IN_trap_justified_lemma]

```
 ⊢ omniCommand ≠ ssmActionsInComplete ⇒
   (s = ACTIONS_IN) ⇒
   ∀ NS Out M Oi Os.
     TR (M,Oi,Os)
       (trap
          (inputList
             [Name Omni says
              prop (SOME (SLc (OMNI omniCommand)));
              Name PlatoonLeader says
              prop (SOME (SLc (PL withdraw)))]))
       (CFG inputOK secContext secAuthorization
          ([Name Omni says prop (SOME (SLc (OMNI omniCommand)));
            Name PlatoonLeader says
            prop (SOME (SLc (PL withdraw)))]::ins) ACTIONS_IN
          outs)
       (CFG inputOK secContext secAuthorization ins
          (NS ACTIONS_IN
             (trap
                (inputList
                   [Name Omni says
                    prop (SOME (SLc (OMNI omniCommand)));
                    Name PlatoonLeader says
                    prop (SOME (SLc (PL withdraw)))])))
          (Out ACTIONS_IN
             (trap
                (inputList
                   [Name Omni says
                    prop (SOME (SLc (OMNI omniCommand)));
                    Name PlatoonLeader says
                    prop (SOME (SLc (PL withdraw)))]))::
            outs)) ⟺
     authenticationTest inputOK
       [Name Omni says prop (SOME (SLc (OMNI omniCommand)));
        Name PlatoonLeader says
        prop (SOME (SLc (PL withdraw)))] ∧
     CFGInterpret (M,Oi,Os)
       (CFG inputOK secContext secAuthorization
          ([Name Omni says prop (SOME (SLc (OMNI omniCommand)));
            Name PlatoonLeader says
            prop (SOME (SLc (PL withdraw)))]::ins) ACTIONS_IN
          outs) ∧ (M,Oi,Os) sat prop NONE
```

[PlatoonLeader_ACTIONS_IN_trap_justified_thm]

```
⊢ omniCommand ≠ ssmActionsInComplete ⇒
  (s = ACTIONS_IN) ⇒
  ∀NS Out M Oi Os.
    TR (M,Oi,Os)
      (trap
         [SOME (SLc (OMNI omniCommand));
          SOME (SLc (PL withdraw))])
      (CFG inputOK secContext secAuthorization
         ([Name Omni says prop (SOME (SLc (OMNI omniCommand)));
           Name PlatoonLeader says
           prop (SOME (SLc (PL withdraw)))]::ins) ACTIONS_IN
         outs)
      (CFG inputOK secContext secAuthorization ins
         (NS ACTIONS_IN
            (trap
               [SOME (SLc (OMNI omniCommand));
                SOME (SLc (PL withdraw))]))
         (Out ACTIONS_IN
            (trap
               [SOME (SLc (OMNI omniCommand));
                SOME (SLc (PL withdraw))])::outs)) ⟺
    authenticationTest inputOK
      [Name Omni says prop (SOME (SLc (OMNI omniCommand)));
       Name PlatoonLeader says
       prop (SOME (SLc (PL withdraw)))] ∧
    CFGInterpret (M,Oi,Os)
      (CFG inputOK secContext secAuthorization
         ([Name Omni says prop (SOME (SLc (OMNI omniCommand)));
           Name PlatoonLeader says
           prop (SOME (SLc (PL withdraw)))]::ins) ACTIONS_IN
         outs) ∧ (M,Oi,Os) sat prop NONE
```

[PlatoonLeader_ACTIONS_IN_trap_lemma]

```
⊢ omniCommand ≠ ssmActionsInComplete ⇒
  (s = ACTIONS_IN) ⇒
  ∀M Oi Os.
    CFGInterpret (M,Oi,Os)
      (CFG inputOK secContext secAuthorization
         ([Name Omni says prop (SOME (SLc (OMNI omniCommand)));
           Name PlatoonLeader says
           prop (SOME (SLc (PL withdraw)))]::ins) ACTIONS_IN
         outs) ⇒
    (M,Oi,Os) sat prop NONE
```

[PlatoonLeader_CONDUCT_ORP_exec_secure_justified_thm]

```
⊢ ∀NS Out M Oi Os.
    TR (M,Oi,Os) (exec [SOME (SLc (PL secure))])
    (CFG inputOK secContext secAuthorization
       ([Name PlatoonLeader says
         prop (SOME (SLc (PL secure)))]::ins) CONDUCT_ORP
       outs)
    (CFG inputOK secContext secAuthorization ins
       (NS CONDUCT_ORP (exec [SOME (SLc (PL secure))]))
       (Out CONDUCT_ORP (exec [SOME (SLc (PL secure))])::
            outs)) ⟺
  authenticationTest inputOK
    [Name PlatoonLeader says prop (SOME (SLc (PL secure)))] ∧
  CFGInterpret (M,Oi,Os)
    (CFG inputOK secContext secAuthorization
       ([Name PlatoonLeader says
         prop (SOME (SLc (PL secure)))]::ins) CONDUCT_ORP
       outs) ∧
  (M,Oi,Os) satList [prop (SOME (SLc (PL secure)))]
```

[PlatoonLeader_CONDUCT_ORP_exec_secure_lemma]

```
⊢ ∀M Oi Os.
    CFGInterpret (M,Oi,Os)
      (CFG inputOK secContext secAuthorization
         ([Name PlatoonLeader says
           prop (SOME (SLc (PL secure)))]::ins) CONDUCT_ORP
         outs) ⇒
    (M,Oi,Os) satList
    propCommandList
      [Name PlatoonLeader says prop (SOME (SLc (PL secure)))]
```

[PlatoonSergeant_SECURE_exec_justified_lemma]

```
⊢ ∀NS Out M Oi Os.
    TR (M,Oi,Os)
      (exec
         (inputList
            [Name Omni says
             prop (SOME (SLc (OMNI ssmSecureComplete)));
             Name PlatoonSergeant says
             prop (SOME (SLc (PSG actionsIn)))]))
      (CFG inputOK secContext secAuthorization
         ([Name Omni says
           prop (SOME (SLc (OMNI ssmSecureComplete)));
           Name PlatoonSergeant says
           prop (SOME (SLc (PSG actionsIn)))]::ins) SECURE
         outs)
      (CFG inputOK secContext secAuthorization ins
         (NS SECURE
            (exec
               (inputList
                  [Name Omni says
                   prop (SOME (SLc (OMNI ssmSecureComplete)));
```

```
                    Name PlatoonSergeant says
                    prop (SOME (SLc (PSG actionsIn)))])))
          (Out SECURE
             (exec
                (inputList
                   [Name Omni says
                    prop (SOME (SLc (OMNI ssmSecureComplete)));
                    Name PlatoonSergeant says
                    prop (SOME (SLc (PSG actionsIn)))])))::
               outs)) ⟺
    authenticationTest inputOK
      [Name Omni says
       prop (SOME (SLc (OMNI ssmSecureComplete)));
       Name PlatoonSergeant says
       prop (SOME (SLc (PSG actionsIn)))] ∧
    CFGInterpret (M,Oi,Os)
      (CFG inputOK secContext secAuthorization
         ([Name Omni says
           prop (SOME (SLc (OMNI ssmSecureComplete)));
           Name PlatoonSergeant says
           prop (SOME (SLc (PSG actionsIn)))]::ins) SECURE
         outs) ∧
    (M,Oi,Os) satList
    propCommandList
      [Name Omni says
       prop (SOME (SLc (OMNI ssmSecureComplete)));
       Name PlatoonSergeant says
       prop (SOME (SLc (PSG actionsIn)))]
```

[PlatoonSergeant_SECURE_exec_justified_thm]

```
 ⊢ ∀ NS Out M Oi Os.
     TR (M,Oi,Os)
       (exec
          [SOME (SLc (OMNI ssmSecureComplete));
           SOME (SLc (PSG actionsIn))])
       (CFG inputOK secContext secAuthorization
          ([Name Omni says
            prop (SOME (SLc (OMNI ssmSecureComplete)));
            Name PlatoonSergeant says
            prop (SOME (SLc (PSG actionsIn)))]::ins) SECURE
          outs)
       (CFG inputOK secContext secAuthorization ins
          (NS SECURE
             (exec
                [SOME (SLc (OMNI ssmSecureComplete));
                 SOME (SLc (PSG actionsIn))]))
          (Out SECURE
             (exec
                [SOME (SLc (OMNI ssmSecureComplete));
```

```
                 SOME (SLc (PSG actionsIn))]):: outs)) ⟺
      authenticationTest inputOK
        [Name Omni says
         prop (SOME (SLc (OMNI ssmSecureComplete)));
         Name PlatoonSergeant says
         prop (SOME (SLc (PSG actionsIn)))] ∧
      CFGInterpret (M,Oi,Os)
        (CFG inputOK secContext secAuthorization
           ([Name Omni says
             prop (SOME (SLc (OMNI ssmSecureComplete)));
             Name PlatoonSergeant says
             prop (SOME (SLc (PSG actionsIn)))]::ins) SECURE
           outs) ∧
      (M,Oi,Os) satList
      [prop (SOME (SLc (OMNI ssmSecureComplete)));
       prop (SOME (SLc (PSG actionsIn)))]
```

[PlatoonSergeant_SECURE_exec_lemma]

```
 ⊢ ∀ M  Oi  Os.
      CFGInterpret (M,Oi,Os)
        (CFG inputOK secContext secAuthorization
           ([Name Omni says
             prop (SOME (SLc (OMNI ssmSecureComplete)));
             Name PlatoonSergeant says
             prop (SOME (SLc (PSG actionsIn)))]::ins) SECURE
           outs) ⇒
      (M,Oi,Os) satList
      propCommandList
        [Name Omni says
         prop (SOME (SLc (OMNI ssmSecureComplete)));
         Name PlatoonSergeant says
         prop (SOME (SLc (PSG actionsIn)))]
```

9 ConductORPType Theory

Built: 11 June 2018
Parent Theories: indexedLists, patternMatches

9.1 Datatypes

```
omniCommand = ssmSecureComplete | ssmActionsInComplete
            | ssmWithdrawComplete | invalidOmniCommand

plCommand = secure | withdraw | complete | plIncomplete

psgCommand = actionsIn | psgIncomplete
```

slCommand =
```
    PL ConductORPType$plCommand
  | PSG ConductORPType$psgCommand
  | OMNI omniCommand
```

slOutput = `ConductORP | Secure | ActionsIn | Withdraw | Complete | unAuthenticated | unAuthorized`

slState = `CONDUCT_ORP | SECURE | ACTIONS_IN | WITHDRAW | COMPLETE`

stateRole = `PlatoonLeader | PlatoonSergeant | Omni`

9.2 Theorems

[omniCommand_distinct_clauses]

$\vdash$ ssmSecureComplete $\neq$ ssmActionsInComplete $\wedge$
ssmSecureComplete $\neq$ ssmWithdrawComplete $\wedge$
ssmSecureComplete $\neq$ invalidOmniCommand $\wedge$
ssmActionsInComplete $\neq$ ssmWithdrawComplete $\wedge$
ssmActionsInComplete $\neq$ invalidOmniCommand $\wedge$
ssmWithdrawComplete $\neq$ invalidOmniCommand

[plCommand_distinct_clauses]

$\vdash$ secure $\neq$ withdraw $\wedge$ secure $\neq$ complete $\wedge$
secure $\neq$ plIncomplete $\wedge$ withdraw $\neq$ complete $\wedge$
withdraw $\neq$ plIncomplete $\wedge$ complete $\neq$ plIncomplete

[psgCommand_distinct_clauses]

$\vdash$ actionsIn $\neq$ psgIncomplete

[slCommand_distinct_clauses]

$\vdash$ ($\forall a'\ a$. PL a $\neq$ PSG a') $\wedge$ ($\forall a'\ a$. PL a $\neq$ OMNI a') $\wedge$
$\forall a'\ a$. PSG a $\neq$ OMNI a'

[slCommand_one_one]

$\vdash$ ($\forall a\ a'$. (PL a = PL a') $\iff$ (a = a')) $\wedge$
($\forall a\ a'$. (PSG a = PSG a') $\iff$ (a = a')) $\wedge$
$\forall a\ a'$. (OMNI a = OMNI a') $\iff$ (a = a')

[slOutput_distinct_clauses]

$\vdash$ ConductORP $\neq$ Secure $\wedge$ ConductORP $\neq$ ActionsIn $\wedge$
ConductORP $\neq$ Withdraw $\wedge$ ConductORP $\neq$ Complete $\wedge$
ConductORP $\neq$ unAuthenticated $\wedge$ ConductORP $\neq$ unAuthorized $\wedge$
Secure $\neq$ ActionsIn $\wedge$ Secure $\neq$ Withdraw $\wedge$ Secure $\neq$ Complete $\wedge$
Secure $\neq$ unAuthenticated $\wedge$ Secure $\neq$ unAuthorized $\wedge$
ActionsIn $\neq$ Withdraw $\wedge$ ActionsIn $\neq$ Complete $\wedge$
ActionsIn $\neq$ unAuthenticated $\wedge$ ActionsIn $\neq$ unAuthorized $\wedge$
Withdraw $\neq$ Complete $\wedge$ Withdraw $\neq$ unAuthenticated $\wedge$
Withdraw $\neq$ unAuthorized $\wedge$ Complete $\neq$ unAuthenticated $\wedge$
Complete $\neq$ unAuthorized $\wedge$ unAuthenticated $\neq$ unAuthorized

[slRole_distinct_clauses]

```
⊢ PlatoonLeader ≠ PlatoonSergeant ∧ PlatoonLeader ≠ Omni ∧
  PlatoonSergeant ≠ Omni
```

[slState_distinct_clauses]

```
⊢ CONDUCT_ORP ≠ SECURE ∧ CONDUCT_ORP ≠ ACTIONS_IN ∧
  CONDUCT_ORP ≠ WITHDRAW ∧ CONDUCT_ORP ≠ COMPLETE ∧
  SECURE ≠ ACTIONS_IN ∧ SECURE ≠ WITHDRAW ∧ SECURE ≠ COMPLETE ∧
  ACTIONS_IN ≠ WITHDRAW ∧ ACTIONS_IN ≠ COMPLETE ∧
  WITHDRAW ≠ COMPLETE
```

10 ConductORPDef Theory

Built: 11 June 2018

Parent Theories: ConductORPType, ssm, OMNIType

10.1 Definitions

[secAuthorization_def]

```
⊢ ∀xs. secAuthorization xs = secHelper (getOmniCommand xs)
```

[secContext_def]

```
⊢ (∀xs.
     secContext CONDUCT_ORP xs =
     [Name PlatoonLeader controls
      prop (SOME (SLc (PL secure)))]) ∧
  (∀xs.
     secContext SECURE xs =
     if getOmniCommand xs = ssmSecureComplete then
       [prop (SOME (SLc (OMNI ssmSecureComplete))) impf
        Name PlatoonSergeant controls
        prop (SOME (SLc (PSG actionsIn)))]
     else [prop NONE]) ∧
  (∀xs.
     secContext ACTIONS_IN xs =
     if getOmniCommand xs = ssmActionsInComplete then
       [prop (SOME (SLc (OMNI ssmActionsInComplete))) impf
        Name PlatoonLeader controls
        prop (SOME (SLc (PL withdraw)))]
     else [prop NONE]) ∧
  ∀xs.
    secContext WITHDRAW xs =
    if getOmniCommand xs = ssmWithdrawComplete then
      [prop (SOME (SLc (OMNI ssmWithdrawComplete))) impf
       Name PlatoonLeader controls
       prop (SOME (SLc (PL complete)))]
    else [prop NONE]
```

[secHelper_def]

```
⊢ ∀ cmd.
     secHelper cmd =
     [Name Omni controls prop (SOME (SLc (OMNI cmd)))]
```

10.2 Theorems

[getOmniCommand_def]

```
⊢ (getOmniCommand [] = invalidOmniCommand) ∧
  (∀ xs cmd.
     getOmniCommand
       (Name Omni says prop (SOME (SLc (OMNI cmd)))::xs) =
     cmd) ∧
  (∀ xs. getOmniCommand (TT::xs) = getOmniCommand xs) ∧
  (∀ xs. getOmniCommand (FF::xs) = getOmniCommand xs) ∧
  (∀ xs v2. getOmniCommand (prop v2::xs) = getOmniCommand xs) ∧
  (∀ xs v3. getOmniCommand (notf v3::xs) = getOmniCommand xs) ∧
  (∀ xs v5 v4.
     getOmniCommand (v4 andf v5::xs) = getOmniCommand xs) ∧
  (∀ xs v7 v6.
     getOmniCommand (v6 orf v7::xs) = getOmniCommand xs) ∧
  (∀ xs v9 v8.
     getOmniCommand (v8 impf v9::xs) = getOmniCommand xs) ∧
  (∀ xs v11 v10.
     getOmniCommand (v10 eqf v11::xs) = getOmniCommand xs) ∧
  (∀ xs v12.
     getOmniCommand (v12 says TT::xs) = getOmniCommand xs) ∧
  (∀ xs v12.
     getOmniCommand (v12 says FF::xs) = getOmniCommand xs) ∧
  (∀ xs v134.
     getOmniCommand (Name v134 says prop NONE::xs) =
     getOmniCommand xs) ∧
  (∀ xs v144.
     getOmniCommand
       (Name PlatoonLeader says prop (SOME v144)::xs) =
     getOmniCommand xs) ∧
  (∀ xs v144.
     getOmniCommand
       (Name PlatoonSergeant says prop (SOME v144)::xs) =
     getOmniCommand xs) ∧
  (∀ xs v146.
     getOmniCommand
       (Name Omni says prop (SOME (ESCc v146))::xs) =
     getOmniCommand xs) ∧
  (∀ xs v150.
     getOmniCommand
       (Name Omni says prop (SOME (SLc (PL v150)))::xs) =
     getOmniCommand xs) ∧
```

$(\forall xs\ \ v151\,.$
  getOmniCommand
    (Name Omni says prop (SOME (SLc (PSG $v151$)))::xs) =
  getOmniCommand xs) $\wedge$
$(\forall xs\ \ v_{68}\ \ v136\ \ v135\,.$
  getOmniCommand ($v135$ meet $v136$ says prop v_{68}::xs) =
  getOmniCommand xs) $\wedge$
$(\forall xs\ \ v_{68}\ \ v138\ \ v137\,.$
  getOmniCommand ($v137$ quoting $v138$ says prop v_{68}::xs) =
  getOmniCommand xs) $\wedge$
$(\forall xs\ \ v_{69}\ \ v_{12}\,.$
  getOmniCommand (v_{12} says notf v_{69}::xs) =
  getOmniCommand xs) $\wedge$
$(\forall xs\ \ v_{71}\ \ v_{70}\ \ v_{12}\,.$
  getOmniCommand (v_{12} says (v_{70} andf v_{71})::xs) =
  getOmniCommand xs) $\wedge$
$(\forall xs\ \ v_{73}\ \ v_{72}\ \ v_{12}\,.$
  getOmniCommand (v_{12} says (v_{72} orf v_{73})::xs) =
  getOmniCommand xs) $\wedge$
$(\forall xs\ \ v_{75}\ \ v_{74}\ \ v_{12}\,.$
  getOmniCommand (v_{12} says (v_{74} impf v_{75})::xs) =
  getOmniCommand xs) $\wedge$
$(\forall xs\ \ v_{77}\ \ v_{76}\ \ v_{12}\,.$
  getOmniCommand (v_{12} says (v_{76} eqf v_{77})::xs) =
  getOmniCommand xs) $\wedge$
$(\forall xs\ \ v_{79}\ \ v_{78}\ \ v_{12}\,.$
  getOmniCommand (v_{12} says v_{78} says v_{79}::xs) =
  getOmniCommand xs) $\wedge$
$(\forall xs\ \ v_{81}\ \ v_{80}\ \ v_{12}\,.$
  getOmniCommand (v_{12} says v_{80} speaks_for v_{81}::xs) =
  getOmniCommand xs) $\wedge$
$(\forall xs\ \ v_{83}\ \ v_{82}\ \ v_{12}\,.$
  getOmniCommand (v_{12} says v_{82} controls v_{83}::xs) =
  getOmniCommand xs) $\wedge$
$(\forall xs\ \ v_{86}\ \ v_{85}\ \ v_{84}\ \ v_{12}\,.$
  getOmniCommand (v_{12} says reps v_{84} v_{85} v_{86}::xs) =
  getOmniCommand xs) $\wedge$
$(\forall xs\ \ v_{88}\ \ v_{87}\ \ v_{12}\,.$
  getOmniCommand (v_{12} says v_{87} domi v_{88}::xs) =
  getOmniCommand xs) $\wedge$
$(\forall xs\ \ v_{90}\ \ v_{89}\ \ v_{12}\,.$
  getOmniCommand (v_{12} says v_{89} eqi v_{90}::xs) =
  getOmniCommand xs) $\wedge$
$(\forall xs\ \ v_{92}\ \ v_{91}\ \ v_{12}\,.$
  getOmniCommand (v_{12} says v_{91} doms v_{92}::xs) =
  getOmniCommand xs) $\wedge$
$(\forall xs\ \ v_{94}\ \ v_{93}\ \ v_{12}\,.$
  getOmniCommand (v_{12} says v_{93} eqs v_{94}::xs) =
  getOmniCommand xs) $\wedge$

```
  (∀ xs v_96 v_95 v_12.
     getOmniCommand (v_12 says v_95 eqn v_96::xs) =
     getOmniCommand xs) ∧
  (∀ xs v_98 v_97 v_12.
     getOmniCommand (v_12 says v_97 lte v_98::xs) =
     getOmniCommand xs) ∧
  (∀ xs v_99 v_12 v100.
     getOmniCommand (v_12 says v_99 lt v100::xs) =
     getOmniCommand xs) ∧
  (∀ xs v_15 v_14.
     getOmniCommand (v_14 speaks_for v_15::xs) =
     getOmniCommand xs) ∧
  (∀ xs v_17 v_16.
     getOmniCommand (v_16 controls v_17::xs) =
     getOmniCommand xs) ∧
  (∀ xs v_20 v_19 v_18.
     getOmniCommand (reps v_18 v_19 v_20::xs) =
     getOmniCommand xs) ∧
  (∀ xs v_22 v_21.
     getOmniCommand (v_21 domi v_22::xs) = getOmniCommand xs) ∧
  (∀ xs v_24 v_23.
     getOmniCommand (v_23 eqi v_24::xs) = getOmniCommand xs) ∧
  (∀ xs v_26 v_25.
     getOmniCommand (v_25 doms v_26::xs) = getOmniCommand xs) ∧
  (∀ xs v_28 v_27.
     getOmniCommand (v_27 eqs v_28::xs) = getOmniCommand xs) ∧
  (∀ xs v_30 v_29.
     getOmniCommand (v_29 eqn v_30::xs) = getOmniCommand xs) ∧
  (∀ xs v_32 v_31.
     getOmniCommand (v_31 lte v_32::xs) = getOmniCommand xs) ∧
  ∀ xs v_34 v_33.
    getOmniCommand (v_33 lt v_34::xs) = getOmniCommand xs
```

[getOmniCommand_ind]

```
⊢ ∀ P.
    P [] ∧
    (∀ cmd xs.
       P (Name Omni says prop (SOME (SLc (OMNI cmd)))::xs)) ∧
    (∀ xs. P xs ⇒ P (TT::xs)) ∧ (∀ xs. P xs ⇒ P (FF::xs)) ∧
    (∀ v_2 xs. P xs ⇒ P (prop v_2::xs)) ∧
    (∀ v_3 xs. P xs ⇒ P (notf v_3::xs)) ∧
    (∀ v_4 v_5 xs. P xs ⇒ P (v_4 andf v_5::xs)) ∧
    (∀ v_6 v_7 xs. P xs ⇒ P (v_6 orf v_7::xs)) ∧
    (∀ v_8 v_9 xs. P xs ⇒ P (v_8 impf v_9::xs)) ∧
    (∀ v_10 v_11 xs. P xs ⇒ P (v_10 eqf v_11::xs)) ∧
    (∀ v_12 xs. P xs ⇒ P (v_12 says TT::xs)) ∧
    (∀ v_12 xs. P xs ⇒ P (v_12 says FF::xs)) ∧
    (∀ v134 xs. P xs ⇒ P (Name v134 says prop NONE::xs)) ∧
    (∀ v144 xs.
     P xs ⇒
     P (Name PlatoonLeader says prop (SOME v144)::xs)) ∧
  (∀ v144 xs.
     P xs ⇒
     P (Name PlatoonSergeant says prop (SOME v144)::xs)) ∧
  (∀ v146 xs.
     P xs ⇒ P (Name Omni says prop (SOME (ESCc v146))::xs)) ∧
  (∀ v150 xs.
     P xs ⇒
     P (Name Omni says prop (SOME (SLc (PL v150)))::xs)) ∧
  (∀ v151 xs.
     P xs ⇒
     P (Name Omni says prop (SOME (SLc (PSG v151)))::xs)) ∧
  (∀ v135 v136 v68 xs.
     P xs ⇒ P (v135 meet v136 says prop v68::xs)) ∧
  (∀ v137 v138 v68 xs.
     P xs ⇒ P (v137 quoting v138 says prop v68::xs)) ∧
  (∀ v12 v69 xs. P xs ⇒ P (v12 says notf v69::xs)) ∧
  (∀ v12 v70 v71 xs. P xs ⇒ P (v12 says (v70 andf v71)::xs)) ∧
  (∀ v12 v72 v73 xs. P xs ⇒ P (v12 says (v72 orf v73)::xs)) ∧
  (∀ v12 v74 v75 xs. P xs ⇒ P (v12 says (v74 impf v75)::xs)) ∧
  (∀ v12 v76 v77 xs. P xs ⇒ P (v12 says (v76 eqf v77)::xs)) ∧
  (∀ v12 v78 v79 xs. P xs ⇒ P (v12 says v78 says v79::xs)) ∧
  (∀ v12 v80 v81 xs.
     P xs ⇒ P (v12 says v80 speaks_for v81::xs)) ∧
  (∀ v12 v82 v83 xs.
     P xs ⇒ P (v12 says v82 controls v83::xs)) ∧
  (∀ v12 v84 v85 v86 xs.
     P xs ⇒ P (v12 says reps v84 v85 v86::xs)) ∧
  (∀ v12 v87 v88 xs. P xs ⇒ P (v12 says v87 domi v88::xs)) ∧
  (∀ v12 v89 v90 xs. P xs ⇒ P (v12 says v89 eqi v90::xs)) ∧
  (∀ v12 v91 v92 xs. P xs ⇒ P (v12 says v91 doms v92::xs)) ∧
  (∀ v12 v93 v94 xs. P xs ⇒ P (v12 says v93 eqs v94::xs)) ∧
  (∀ v12 v95 v96 xs. P xs ⇒ P (v12 says v95 eqn v96::xs)) ∧
  (∀ v12 v97 v98 xs. P xs ⇒ P (v12 says v97 lte v98::xs)) ∧
  (∀ v12 v99 v100 xs. P xs ⇒ P (v12 says v99 lt v100::xs)) ∧
  (∀ v14 v15 xs. P xs ⇒ P (v14 speaks_for v15::xs)) ∧
  (∀ v16 v17 xs. P xs ⇒ P (v16 controls v17::xs)) ∧
  (∀ v18 v19 v20 xs. P xs ⇒ P (reps v18 v19 v20::xs)) ∧
  (∀ v21 v22 xs. P xs ⇒ P (v21 domi v22::xs)) ∧
  (∀ v23 v24 xs. P xs ⇒ P (v23 eqi v24::xs)) ∧
  (∀ v25 v26 xs. P xs ⇒ P (v25 doms v26::xs)) ∧
  (∀ v27 v28 xs. P xs ⇒ P (v27 eqs v28::xs)) ∧
  (∀ v29 v30 xs. P xs ⇒ P (v29 eqn v30::xs)) ∧
  (∀ v31 v32 xs. P xs ⇒ P (v31 lte v32::xs)) ∧
  (∀ v33 v34 xs. P xs ⇒ P (v33 lt v34::xs)) ⇒
  ∀ v. P v
```

[getPlCom_def]

⊢ (getPlCom [] = plIncomplete) ∧
  (∀ *xs* *cmd*. getPlCom (SOME (SLc (PL *cmd*))::*xs*) = *cmd*) ∧
  (∀ *xs*. getPlCom (NONE::*xs*) = getPlCom *xs*) ∧
  (∀ *xs* v_4. getPlCom (SOME (ESCc v_4)::*xs*) = getPlCom *xs*) ∧
  (∀ *xs* v_9. getPlCom (SOME (SLc (PSG v_9))::*xs*) = getPlCom *xs*) ∧
  ∀ *xs* v_{10}. getPlCom (SOME (SLc (OMNI v_{10}))::*xs*) = getPlCom *xs*

[getPlCom_ind]

⊢ ∀ *P*.
    P [] ∧ (∀ *cmd* *xs*. *P* (SOME (SLc (PL *cmd*))::*xs*)) ∧
    (∀ *xs*. *P* *xs* ⇒ *P* (NONE::*xs*)) ∧
    (∀ v_4 *xs*. *P* *xs* ⇒ *P* (SOME (ESCc v_4)::*xs*)) ∧
    (∀ v_9 *xs*. *P* *xs* ⇒ *P* (SOME (SLc (PSG v_9))::*xs*)) ∧
    (∀ v_{10} *xs*. *P* *xs* ⇒ *P* (SOME (SLc (OMNI v_{10}))::*xs*)) ⇒
    ∀ *v*. *P* *v*

[getPsgCom_def]

⊢ (getPsgCom [] = psgIncomplete) ∧
  (∀ *xs* *cmd*. getPsgCom (SOME (SLc (PSG *cmd*))::*xs*) = *cmd*) ∧
  (∀ *xs*. getPsgCom (NONE::*xs*) = getPsgCom *xs*) ∧
  (∀ *xs* v_4. getPsgCom (SOME (ESCc v_4)::*xs*) = getPsgCom *xs*) ∧
  (∀ *xs* v_8. getPsgCom (SOME (SLc (PL v_8))::*xs*) = getPsgCom *xs*) ∧
  ∀ *xs* v_{10}. getPsgCom (SOME (SLc (OMNI v_{10}))::*xs*) = getPsgCom *xs*

[getPsgCom_ind]

⊢ ∀ *P*.
    P [] ∧ (∀ *cmd* *xs*. *P* (SOME (SLc (PSG *cmd*))::*xs*)) ∧
    (∀ *xs*. *P* *xs* ⇒ *P* (NONE::*xs*)) ∧
    (∀ v_4 *xs*. *P* *xs* ⇒ *P* (SOME (ESCc v_4)::*xs*)) ∧
    (∀ v_8 *xs*. *P* *xs* ⇒ *P* (SOME (SLc (PL v_8))::*xs*)) ∧
    (∀ v_{10} *xs*. *P* *xs* ⇒ *P* (SOME (SLc (OMNI v_{10}))::*xs*)) ⇒
    ∀ *v*. *P* *v*

11 ssmConductPB Theory

Built: 10 June 2018
Parent Theories: ConductPBType, ssm11, OMNIType

11.1 Definitions

[secContextConductPB_def]

⊢ ∀ *plcmd* *psgcmd* *incomplete*.
    secContextConductPB *plcmd* *psgcmd* *incomplete* =
    [Name PlatoonLeader controls prop (SOME (SLc (PL *plcmd*)));
     Name PlatoonSergeant controls
     prop (SOME (SLc (PSG *psgcmd*)));
     Name PlatoonLeader says

```
prop (SOME (SLc (PSG psgcmd))) impf prop NONE;
Name PlatoonSergeant says
prop (SOME (SLc (PL plcmd))) impf prop NONE]
```

[ssmConductPBStateInterp_def]

$\vdash$ ∀ *slState*. `ssmConductPBStateInterp` *slState* `= TT`

11.2 Theorems

[authTestConductPB_cmd_reject_lemma]

$\vdash$ ∀ *cmd*. ¬`authTestConductPB (prop (SOME` *cmd*`))`

[authTestConductPB_def]

```
⊢ (authTestConductPB (Name PlatoonLeader says prop cmd) ⇔ T) ∧
  (authTestConductPB (Name PlatoonSergeant says prop cmd) ⇔
   T) ∧ (authTestConductPB TT ⇔ F) ∧
  (authTestConductPB FF ⇔ F) ∧
  (authTestConductPB (prop v) ⇔ F) ∧
  (authTestConductPB (notf v1) ⇔ F) ∧
  (authTestConductPB (v2 andf v3) ⇔ F) ∧
  (authTestConductPB (v4 orf v5) ⇔ F) ∧
  (authTestConductPB (v6 impf v7) ⇔ F) ∧
  (authTestConductPB (v8 eqf v9) ⇔ F) ∧
  (authTestConductPB (v10 says TT) ⇔ F) ∧
  (authTestConductPB (v10 says FF) ⇔ F) ∧
  (authTestConductPB (v133 meet v134 says prop v66) ⇔ F) ∧
  (authTestConductPB (v135 quoting v136 says prop v66) ⇔ F) ∧
  (authTestConductPB (v10 says notf v67) ⇔ F) ∧
  (authTestConductPB (v10 says (v68 andf v69)) ⇔ F) ∧
  (authTestConductPB (v10 says (v70 orf v71)) ⇔ F) ∧
  (authTestConductPB (v10 says (v72 impf v73)) ⇔ F) ∧
  (authTestConductPB (v10 says (v74 eqf v75)) ⇔ F) ∧
  (authTestConductPB (v10 says v76 says v77) ⇔ F) ∧
  (authTestConductPB (v10 says v78 speaks_for v79) ⇔ F) ∧
  (authTestConductPB (v10 says v80 controls v81) ⇔ F) ∧
  (authTestConductPB (v10 says reps v82 v83 v84) ⇔ F) ∧
  (authTestConductPB (v10 says v85 domi v86) ⇔ F) ∧
  (authTestConductPB (v10 says v87 eqi v88) ⇔ F) ∧
  (authTestConductPB (v10 says v89 doms v90) ⇔ F) ∧
  (authTestConductPB (v10 says v91 eqs v92) ⇔ F) ∧
  (authTestConductPB (v10 says v93 eqn v94) ⇔ F) ∧
  (authTestConductPB (v10 says v95 lte v96) ⇔ F) ∧
  (authTestConductPB (v10 says v97 lt v98) ⇔ F) ∧
  (authTestConductPB (v12 speaks_for v13) ⇔ F) ∧
  (authTestConductPB (v14 controls v15) ⇔ F) ∧
  (authTestConductPB (reps v16 v17 v18) ⇔ F) ∧
  (authTestConductPB (v19 domi v20) ⇔ F) ∧
  (authTestConductPB (v21 eqi v22) ⇔ F) ∧
```

(authTestConductPB (v_{23} doms v_{24}) $\iff$ F) $\land$
(authTestConductPB (v_{25} eqs v_{26}) $\iff$ F) $\land$
(authTestConductPB (v_{27} eqn v_{28}) $\iff$ F) $\land$
(authTestConductPB (v_{29} lte v_{30}) $\iff$ F) $\land$
(authTestConductPB (v_{31} lt v_{32}) $\iff$ F)

[authTestConductPB_ind]

$\vdash$ $\forall P$.
(∀ *cmd*. P (Name PlatoonLeader says prop *cmd*)) $\land$
(∀ *cmd*. P (Name PlatoonSergeant says prop *cmd*)) $\land$ P TT $\land$
P FF $\land$ ($\forall v$. P (prop v)) $\land$ ($\forall v_1$. P (notf v_1)) $\land$
($\forall v_2\ v_3$. P (v_2 andf v_3)) $\land$ ($\forall v_4\ v_5$. P (v_4 orf v_5)) $\land$
($\forall v_6\ v_7$. P (v_6 impf v_7)) $\land$ ($\forall v_8\ v_9$. P (v_8 eqf v_9)) $\land$
($\forall v_{10}$. P (v_{10} says TT)) $\land$ ($\forall v_{10}$. P (v_{10} says FF)) $\land$
($\forall$ *v133* *v134* v_{66}. P (*v133* meet *v134* says prop v_{66})) $\land$
($\forall$ *v135* *v136* v_{66}. P (*v135* quoting *v136* says prop v_{66})) $\land$
($\forall v_{10}\ v_{67}$. P (v_{10} says notf v_{67})) $\land$
($\forall v_{10}\ v_{68}\ v_{69}$. P (v_{10} says (v_{68} andf v_{69}))) $\land$
($\forall v_{10}\ v_{70}\ v_{71}$. P (v_{10} says (v_{70} orf v_{71}))) $\land$
($\forall v_{10}\ v_{72}\ v_{73}$. P (v_{10} says (v_{72} impf v_{73}))) $\land$
($\forall v_{10}\ v_{74}\ v_{75}$. P (v_{10} says (v_{74} eqf v_{75}))) $\land$
($\forall v_{10}\ v_{76}\ v_{77}$. P (v_{10} says v_{76} says v_{77})) $\land$
($\forall v_{10}\ v_{78}\ v_{79}$. P (v_{10} says v_{78} speaks_for v_{79})) $\land$
($\forall v_{10}\ v_{80}\ v_{81}$. P (v_{10} says v_{80} controls v_{81})) $\land$
($\forall v_{10}\ v_{82}\ v_{83}\ v_{84}$. P (v_{10} says reps v_{82} v_{83} v_{84})) $\land$
($\forall v_{10}\ v_{85}\ v_{86}$. P (v_{10} says v_{85} domi v_{86})) $\land$
($\forall v_{10}\ v_{87}\ v_{88}$. P (v_{10} says v_{87} eqi v_{88})) $\land$
($\forall v_{10}\ v_{89}\ v_{90}$. P (v_{10} says v_{89} doms v_{90})) $\land$
($\forall v_{10}\ v_{91}\ v_{92}$. P (v_{10} says v_{91} eqs v_{92})) $\land$
($\forall v_{10}\ v_{93}\ v_{94}$. P (v_{10} says v_{93} eqn v_{94})) $\land$
($\forall v_{10}\ v_{95}\ v_{96}$. P (v_{10} says v_{95} lte v_{96})) $\land$
($\forall v_{10}\ v_{97}\ v_{98}$. P (v_{10} says v_{97} lt v_{98})) $\land$
($\forall v_{12}\ v_{13}$. P (v_{12} speaks_for v_{13})) $\land$
($\forall v_{14}\ v_{15}$. P (v_{14} controls v_{15})) $\land$
($\forall v_{16}\ v_{17}\ v_{18}$. P (reps v_{16} v_{17} v_{18})) $\land$
($\forall v_{19}\ v_{20}$. P (v_{19} domi v_{20})) $\land$
($\forall v_{21}\ v_{22}$. P (v_{21} eqi v_{22})) $\land$
($\forall v_{23}\ v_{24}$. P (v_{23} doms v_{24})) $\land$
($\forall v_{25}\ v_{26}$. P (v_{25} eqs v_{26})) $\land$ ($\forall v_{27}\ v_{28}$. P (v_{27} eqn v_{28})) $\land$
($\forall v_{29}\ v_{30}$. P (v_{29} lte v_{30})) $\land$ ($\forall v_{31}\ v_{32}$. P (v_{31} lt v_{32})) $\Rightarrow$
$\forall v$. P v

[conductPBNS_def]

```
⊢ (conductPBNS CONDUCT_PB (exec (PL securePB)) = SECURE_PB) ∧
  (conductPBNS CONDUCT_PB (exec (PL plIncompletePB)) =
   CONDUCT_PB) ∧
  (conductPBNS SECURE_PB (exec (PSG actionsInPB)) =
   ACTIONS_IN_PB) ∧
  (conductPBNS SECURE_PB (exec (PSG psgIncompletePB)) =
     SECURE_PB) ∧
    (conductPBNS ACTIONS_IN_PB (exec (PL withdrawPB)) =
     WITHDRAW_PB) ∧
    (conductPBNS ACTIONS_IN_PB (exec (PL plIncompletePB)) =
     ACTIONS_IN_PB) ∧
    (conductPBNS WITHDRAW_PB (exec (PL completePB)) =
     COMPLETE_PB) ∧
    (conductPBNS WITHDRAW_PB (exec (PL plIncompletePB)) =
     WITHDRAW_PB) ∧ (conductPBNS s (trap (PL cmd′)) = s) ∧
    (conductPBNS s (trap (PSG cmd)) = s) ∧
    (conductPBNS s (discard (PL cmd′)) = s) ∧
    (conductPBNS s (discard (PSG cmd)) = s)
```

[conductPBNS_ind]

```
  ⊢ ∀P.
       P CONDUCT_PB (exec (PL securePB)) ∧
       P CONDUCT_PB (exec (PL plIncompletePB)) ∧
       P SECURE_PB (exec (PSG actionsInPB)) ∧
       P SECURE_PB (exec (PSG psgIncompletePB)) ∧
       P ACTIONS_IN_PB (exec (PL withdrawPB)) ∧
       P ACTIONS_IN_PB (exec (PL plIncompletePB)) ∧
       P WITHDRAW_PB (exec (PL completePB)) ∧
       P WITHDRAW_PB (exec (PL plIncompletePB)) ∧
       (∀s cmd. P s (trap (PL cmd))) ∧
       (∀s cmd. P s (trap (PSG cmd))) ∧
       (∀s cmd. P s (discard (PL cmd))) ∧
       (∀s cmd. P s (discard (PSG cmd))) ∧
       P CONDUCT_PB (exec (PL withdrawPB)) ∧
       P CONDUCT_PB (exec (PL completePB)) ∧
       (∀v11. P CONDUCT_PB (exec (PSG v11))) ∧
       (∀v13. P SECURE_PB (exec (PL v13))) ∧
       P ACTIONS_IN_PB (exec (PL securePB)) ∧
       P ACTIONS_IN_PB (exec (PL completePB)) ∧
       (∀v17. P ACTIONS_IN_PB (exec (PSG v17))) ∧
       P WITHDRAW_PB (exec (PL securePB)) ∧
       P WITHDRAW_PB (exec (PL withdrawPB)) ∧
       (∀v20. P WITHDRAW_PB (exec (PSG v20))) ∧
       (∀v21. P COMPLETE_PB (exec v21)) ⇒
       ∀v v1. P v v1
```

[conductPBOut_def]

```
  ⊢ (conductPBOut CONDUCT_PB (exec (PL securePB)) = ConductPB) ∧
    (conductPBOut CONDUCT_PB (exec (PL plIncompletePB)) =
     ConductPB) ∧
    (conductPBOut SECURE_PB (exec (PSG actionsInPB)) =
     SecurePB) ∧
    (conductPBOut SECURE_PB (exec (PSG psgIncompletePB)) =
     SecurePB) ∧
    (conductPBOut ACTIONS_IN_PB (exec (PL withdrawPB)) =
    ActionsInPB) ∧
   (conductPBOut ACTIONS_IN_PB (exec (PL plIncompletePB)) =
    ActionsInPB) ∧
   (conductPBOut WITHDRAW_PB (exec (PL completePB)) =
    WithdrawPB) ∧
   (conductPBOut WITHDRAW_PB (exec (PL plIncompletePB)) =
    WithdrawPB) ∧
   (conductPBOut s (trap (PL cmd′)) = unAuthorized) ∧
   (conductPBOut s (trap (PSG cmd)) = unAuthorized) ∧
   (conductPBOut s (discard (PL cmd′)) = unAuthenticated) ∧
   (conductPBOut s (discard (PSG cmd)) = unAuthenticated)
```

[conductPBOut_ind]

```
 ⊢ ∀P.
     P CONDUCT_PB (exec (PL securePB)) ∧
     P CONDUCT_PB (exec (PL plIncompletePB)) ∧
     P SECURE_PB (exec (PSG actionsInPB)) ∧
     P SECURE_PB (exec (PSG psgIncompletePB)) ∧
     P ACTIONS_IN_PB (exec (PL withdrawPB)) ∧
     P ACTIONS_IN_PB (exec (PL plIncompletePB)) ∧
     P WITHDRAW_PB (exec (PL completePB)) ∧
     P WITHDRAW_PB (exec (PL plIncompletePB)) ∧
     (∀s cmd. P s (trap (PL cmd))) ∧
     (∀s cmd. P s (trap (PSG cmd))) ∧
     (∀s cmd. P s (discard (PL cmd))) ∧
     (∀s cmd. P s (discard (PSG cmd))) ∧
     P CONDUCT_PB (exec (PL withdrawPB)) ∧
     P CONDUCT_PB (exec (PL completePB)) ∧
     (∀v11. P CONDUCT_PB (exec (PSG v11))) ∧
     (∀v13. P SECURE_PB (exec (PL v13))) ∧
     P ACTIONS_IN_PB (exec (PL securePB)) ∧
     P ACTIONS_IN_PB (exec (PL completePB)) ∧
     (∀v17. P ACTIONS_IN_PB (exec (PSG v17))) ∧
     P WITHDRAW_PB (exec (PL securePB)) ∧
     P WITHDRAW_PB (exec (PL withdrawPB)) ∧
     (∀v20. P WITHDRAW_PB (exec (PSG v20))) ∧
     (∀v21. P COMPLETE_PB (exec v21)) ⇒
     ∀v v1. P v v1
```

[PlatoonLeader_exec_plCommandPB_justified_thm]

```
 ⊢ ∀NS Out M Oi Os.
     TR (M,Oi,Os) (exec (SLc (PL plCommand)))
       (CFG authTestConductPB ssmConductPBStateInterp
          (secContextConductPB plCommand psgCommand incomplete)
          (Name PlatoonLeader says
           prop (SOME (SLc (PL plCommand)))::ins) s outs)
       (CFG authTestConductPB ssmConductPBStateInterp
          (secContextConductPB plCommand psgCommand incomplete)
          ins (NS s (exec (SLc (PL plCommand))))
```

```
        (Out s (exec (SLc (PL plCommand)))::outs)) ⟺
     authTestConductPB
       (Name PlatoonLeader says
        prop (SOME (SLc (PL plCommand)))) ∧
     CFGInterpret (M,Oi,Os)
       (CFG authTestConductPB ssmConductPBStateInterp
          (secContextConductPB plCommand psgCommand incomplete)
          (Name PlatoonLeader says
           prop (SOME (SLc (PL plCommand)))::ins) s outs) ∧
     (M,Oi,Os) sat prop (SOME (SLc (PL plCommand)))
```

[PlatoonLeader_plCommandPB_lemma]

```
 ⊢ CFGInterpret (M,Oi,Os)
     (CFG authTestConductPB ssmConductPBStateInterp
        (secContextConductPB plCommand psgCommand incomplete)
        (Name PlatoonLeader says
         prop (SOME (SLc (PL plCommand)))::ins) s outs) ⇒
   (M,Oi,Os) sat prop (SOME (SLc (PL plCommand)))
```

[PlatoonSergeant_exec_psgCommandPB_justified_thm]

```
 ⊢ ∀NS Out M Oi Os.
     TR (M,Oi,Os) (exec (SLc (PSG psgCommand)))
       (CFG authTestConductPB ssmConductPBStateInterp
          (secContextConductPB plCommand psgCommand incomplete)
          (Name PlatoonSergeant says
           prop (SOME (SLc (PSG psgCommand)))::ins) s outs)
       (CFG authTestConductPB ssmConductPBStateInterp
          (secContextConductPB plCommand psgCommand incomplete)
          ins (NS s (exec (SLc (PSG psgCommand))))
          (Out s (exec (SLc (PSG psgCommand)))::outs)) ⟺
     authTestConductPB
       (Name PlatoonSergeant says
        prop (SOME (SLc (PSG psgCommand)))) ∧
     CFGInterpret (M,Oi,Os)
       (CFG authTestConductPB ssmConductPBStateInterp
          (secContextConductPB plCommand psgCommand incomplete)
          (Name PlatoonSergeant says
           prop (SOME (SLc (PSG psgCommand)))::ins) s outs) ∧
     (M,Oi,Os) sat prop (SOME (SLc (PSG psgCommand)))
```

[PlatoonSergeant_psgCommandPB_lemma]

```
 ⊢ CFGInterpret (M,Oi,Os)
     (CFG authTestConductPB ssmConductPBStateInterp
        (secContextConductPB plCommand psgCommand incomplete)
        (Name PlatoonSergeant says
         prop (SOME (SLc (PSG psgCommand)))::ins) s outs) ⇒
   (M,Oi,Os) sat prop (SOME (SLc (PSG psgCommand)))
```

12 ConductPBType Theory

Built: 10 June 2018

Parent Theories: indexedLists, patternMatches

12.1 Datatypes

```
plCommandPB = securePB | withdrawPB | completePB
            | plIncompletePB

psgCommandPB = actionsInPB | psgIncompletePB

slCommand = PL plCommandPB | PSG psgCommandPB

slOutput = ConductPB | SecurePB | ActionsInPB | WithdrawPB
         | CompletePB | unAuthenticated | unAuthorized

slState = CONDUCT_PB | SECURE_PB | ACTIONS_IN_PB | WITHDRAW_PB
        | COMPLETE_PB

stateRole = PlatoonLeader | PlatoonSergeant
```

12.2 Theorems

[plCommandPB_distinct_clauses]

```
⊢ securePB ≠ withdrawPB ∧ securePB ≠ completePB ∧
  securePB ≠ plIncompletePB ∧ withdrawPB ≠ completePB ∧
  withdrawPB ≠ plIncompletePB ∧ completePB ≠ plIncompletePB
```

[psgCommandPB_distinct_clauses]

```
⊢ actionsInPB ≠ psgIncompletePB
```

[slCommand_distinct_clauses]

```
⊢ ∀ a' a. PL a ≠ PSG a'
```

[slCommand_one_one]

```
⊢ (∀ a a'. (PL a = PL a') ⇔ (a = a')) ∧
  ∀ a a'. (PSG a = PSG a') ⇔ (a = a')
```

[slOutput_distinct_clauses]

```
⊢ ConductPB ≠ SecurePB ∧ ConductPB ≠ ActionsInPB ∧
  ConductPB ≠ WithdrawPB ∧ ConductPB ≠ CompletePB ∧
  ConductPB ≠ unAuthenticated ∧ ConductPB ≠ unAuthorized ∧
  SecurePB ≠ ActionsInPB ∧ SecurePB ≠ WithdrawPB ∧
  SecurePB ≠ CompletePB ∧ SecurePB ≠ unAuthenticated ∧
  SecurePB ≠ unAuthorized ∧ ActionsInPB ≠ WithdrawPB ∧
  ActionsInPB ≠ CompletePB ∧ ActionsInPB ≠ unAuthenticated ∧
  ActionsInPB ≠ unAuthorized ∧ WithdrawPB ≠ CompletePB ∧
  WithdrawPB ≠ unAuthenticated ∧ WithdrawPB ≠ unAuthorized ∧
  CompletePB ≠ unAuthenticated ∧ CompletePB ≠ unAuthorized ∧
  unAuthenticated ≠ unAuthorized
```

[slRole_distinct_clauses]

```
⊢ PlatoonLeader ≠ PlatoonSergeant
```

[slState_distinct_clauses]

```
⊢ CONDUCT_PB ≠ SECURE_PB ∧ CONDUCT_PB ≠ ACTIONS_IN_PB ∧
  CONDUCT_PB ≠ WITHDRAW_PB ∧ CONDUCT_PB ≠ COMPLETE_PB ∧
  SECURE_PB ≠ ACTIONS_IN_PB ∧ SECURE_PB ≠ WITHDRAW_PB ∧
  SECURE_PB ≠ COMPLETE_PB ∧ ACTIONS_IN_PB ≠ WITHDRAW_PB ∧
  ACTIONS_IN_PB ≠ COMPLETE_PB ∧ WITHDRAW_PB ≠ COMPLETE_PB
```

13 ssmMoveToORP Theory

Built: 10 June 2018

Parent Theories: MoveToORPType, ssm11, OMNIType

13.1 Definitions

[secContextMoveToORP_def]

```
⊢ ∀cmd.
    secContextMoveToORP cmd =
    [Name PlatoonLeader controls prop (SOME (SLc cmd))]
```

[ssmMoveToORPStateInterp_def]

```
⊢ ∀state. ssmMoveToORPStateInterp state = TT
```

13.2 Theorems

[authTestMoveToORP_cmd_reject_lemma]

```
⊢ ∀cmd. ¬authTestMoveToORP (prop (SOME cmd))
```

[authTestMoveToORP_def]

```
⊢ (authTestMoveToORP (Name PlatoonLeader says prop cmd) ⇔ T) ∧
  (authTestMoveToORP TT ⇔ F) ∧ (authTestMoveToORP FF ⇔ F) ∧
  (authTestMoveToORP (prop v) ⇔ F) ∧
  (authTestMoveToORP (notf v1) ⇔ F) ∧
  (authTestMoveToORP (v2 andf v3) ⇔ F) ∧
  (authTestMoveToORP (v4 orf v5) ⇔ F) ∧
  (authTestMoveToORP (v6 impf v7) ⇔ F) ∧
  (authTestMoveToORP (v8 eqf v9) ⇔ F) ∧
  (authTestMoveToORP (v10 says TT) ⇔ F) ∧
  (authTestMoveToORP (v10 says FF) ⇔ F) ∧
  (authTestMoveToORP (v133 meet v134 says prop v66) ⇔ F) ∧
  (authTestMoveToORP (v135 quoting v136 says prop v66) ⇔ F) ∧
  (authTestMoveToORP (v10 says notf v67) ⇔ F) ∧
  (authTestMoveToORP (v10 says (v68 andf v69)) ⇔ F) ∧
  (authTestMoveToORP (v10 says (v70 orf v71)) ⇔ F) ∧
```

(authTestMoveToORP (v_{10} says (v_{72} impf v_{73})) $\iff$ F) $\wedge$
(authTestMoveToORP (v_{10} says (v_{74} eqf v_{75})) $\iff$ F) $\wedge$
(authTestMoveToORP (v_{10} says v_{76} says v_{77}) $\iff$ F) $\wedge$
(authTestMoveToORP (v_{10} says v_{78} speaks_for v_{79}) $\iff$ F) $\wedge$
(authTestMoveToORP (v_{10} says v_{80} controls v_{81}) $\iff$ F) $\wedge$
(authTestMoveToORP (v_{10} says reps v_{82} v_{83} v_{84}) $\iff$ F) $\wedge$
(authTestMoveToORP (v_{10} says v_{85} domi v_{86}) $\iff$ F) $\wedge$
(authTestMoveToORP (v_{10} says v_{87} eqi v_{88}) $\iff$ F) $\wedge$
(authTestMoveToORP (v_{10} says v_{89} doms v_{90}) $\iff$ F) $\wedge$
(authTestMoveToORP (v_{10} says v_{91} eqs v_{92}) $\iff$ F) $\wedge$
(authTestMoveToORP (v_{10} says v_{93} eqn v_{94}) $\iff$ F) $\wedge$
(authTestMoveToORP (v_{10} says v_{95} lte v_{96}) $\iff$ F) $\wedge$
(authTestMoveToORP (v_{10} says v_{97} lt v_{98}) $\iff$ F) $\wedge$
(authTestMoveToORP (v_{12} speaks_for v_{13}) $\iff$ F) $\wedge$
(authTestMoveToORP (v_{14} controls v_{15}) $\iff$ F) $\wedge$
(authTestMoveToORP (reps v_{16} v_{17} v_{18}) $\iff$ F) $\wedge$
(authTestMoveToORP (v_{19} domi v_{20}) $\iff$ F) $\wedge$
(authTestMoveToORP (v_{21} eqi v_{22}) $\iff$ F) $\wedge$
(authTestMoveToORP (v_{23} doms v_{24}) $\iff$ F) $\wedge$
(authTestMoveToORP (v_{25} eqs v_{26}) $\iff$ F) $\wedge$
(authTestMoveToORP (v_{27} eqn v_{28}) $\iff$ F) $\wedge$
(authTestMoveToORP (v_{29} lte v_{30}) $\iff$ F) $\wedge$
(authTestMoveToORP (v_{31} lt v_{32}) $\iff$ F)

[authTestMoveToORP_ind]

$\vdash \forall P.$
($\forall$ *cmd*. P (Name PlatoonLeader says prop *cmd*)) $\wedge$ P TT $\wedge$
P FF $\wedge$ ($\forall v$. P (prop v)) $\wedge$ ($\forall v_1$. P (notf v_1)) $\wedge$
($\forall v_2$ v_3. P (v_2 andf v_3)) $\wedge$ ($\forall v_4$ v_5. P (v_4 orf v_5)) $\wedge$
($\forall v_6$ v_7. P (v_6 impf v_7)) $\wedge$ ($\forall v_8$ v_9. P (v_8 eqf v_9)) $\wedge$
($\forall v_{10}$. P (v_{10} says TT)) $\wedge$ ($\forall v_{10}$. P (v_{10} says FF)) $\wedge$
($\forall$ *v133* *v134* v_{66}. P (*v133* meet *v134* says prop v_{66})) $\wedge$
($\forall$ *v135* *v136* v_{66}. P (*v135* quoting *v136* says prop v_{66})) $\wedge$
($\forall v_{10}$ v_{67}. P (v_{10} says notf v_{67})) $\wedge$
($\forall v_{10}$ v_{68} v_{69}. P (v_{10} says (v_{68} andf v_{69}))) $\wedge$
($\forall v_{10}$ v_{70} v_{71}. P (v_{10} says (v_{70} orf v_{71}))) $\wedge$
($\forall v_{10}$ v_{72} v_{73}. P (v_{10} says (v_{72} impf v_{73}))) $\wedge$
($\forall v_{10}$ v_{74} v_{75}. P (v_{10} says (v_{74} eqf v_{75}))) $\wedge$
($\forall v_{10}$ v_{76} v_{77}. P (v_{10} says v_{76} says v_{77})) $\wedge$
($\forall v_{10}$ v_{78} v_{79}. P (v_{10} says v_{78} speaks_for v_{79})) $\wedge$
($\forall v_{10}$ v_{80} v_{81}. P (v_{10} says v_{80} controls v_{81})) $\wedge$
($\forall v_{10}$ v_{82} v_{83} v_{84}. P (v_{10} says reps v_{82} v_{83} v_{84})) $\wedge$
($\forall v_{10}$ v_{85} v_{86}. P (v_{10} says v_{85} domi v_{86})) $\wedge$
($\forall v_{10}$ v_{87} v_{88}. P (v_{10} says v_{87} eqi v_{88})) $\wedge$
($\forall v_{10}$ v_{89} v_{90}. P (v_{10} says v_{89} doms v_{90})) $\wedge$
($\forall v_{10}$ v_{91} v_{92}. P (v_{10} says v_{91} eqs v_{92})) $\wedge$
($\forall v_{10}$ v_{93} v_{94}. P (v_{10} says v_{93} eqn v_{94})) $\wedge$
($\forall v_{10}$ v_{95} v_{96}. P (v_{10} says v_{95} lte v_{96})) $\wedge$
($\forall v_{10}$ v_{97} v_{98}. P (v_{10} says v_{97} lt v_{98})) $\wedge$

```
     (∀ $v_{12}$ $v_{13}$. $P$ ($v_{12}$ speaks_for $v_{13}$)) ∧
     (∀ $v_{14}$ $v_{15}$. $P$ ($v_{14}$ controls $v_{15}$)) ∧
     (∀ $v_{16}$ $v_{17}$ $v_{18}$. $P$ (reps $v_{16}$ $v_{17}$ $v_{18}$)) ∧
     (∀ $v_{19}$ $v_{20}$. $P$ ($v_{19}$ domi $v_{20}$)) ∧
     (∀ $v_{21}$ $v_{22}$. $P$ ($v_{21}$ eqi $v_{22}$)) ∧
     (∀ $v_{23}$ $v_{24}$. $P$ ($v_{23}$ doms $v_{24}$)) ∧
     (∀ $v_{25}$ $v_{26}$. $P$ ($v_{25}$ eqs $v_{26}$)) ∧ (∀ $v_{27}$ $v_{28}$. $P$ ($v_{27}$ eqn $v_{28}$)) ∧
     (∀ $v_{29}$ $v_{30}$. $P$ ($v_{29}$ lte $v_{30}$)) ∧ (∀ $v_{31}$ $v_{32}$. $P$ ($v_{31}$ lt $v_{32}$)) ⇒
     ∀ $v$. $P$ $v$
```

[moveToORPNS_def]

```
 ⊢ (moveToORPNS MOVE_TO_ORP (exec (SLc pltForm)) = PLT_FORM) ∧
   (moveToORPNS MOVE_TO_ORP (exec (SLc incomplete)) =
    MOVE_TO_ORP) ∧
   (moveToORPNS PLT_FORM (exec (SLc pltMove)) = PLT_MOVE) ∧
   (moveToORPNS PLT_FORM (exec (SLc incomplete)) = PLT_FORM) ∧
   (moveToORPNS PLT_MOVE (exec (SLc pltSecureHalt)) =
    PLT_SECURE_HALT) ∧
   (moveToORPNS PLT_MOVE (exec (SLc incomplete)) = PLT_MOVE) ∧
   (moveToORPNS PLT_SECURE_HALT (exec (SLc complete)) =
    COMPLETE) ∧
   (moveToORPNS PLT_SECURE_HALT (exec (SLc incomplete)) =
    PLT_SECURE_HALT) ∧ (moveToORPNS $s$ (trap (SLc $cmd$)) = $s$) ∧
   (moveToORPNS $s$ (discard (SLc $cmd$)) = $s$)
```

[moveToORPNS_ind]

```
 ⊢ ∀ $P$.
     $P$ MOVE_TO_ORP (exec (SLc pltForm)) ∧
     $P$ MOVE_TO_ORP (exec (SLc incomplete)) ∧
     $P$ PLT_FORM (exec (SLc pltMove)) ∧
     $P$ PLT_FORM (exec (SLc incomplete)) ∧
     $P$ PLT_MOVE (exec (SLc pltSecureHalt)) ∧
     $P$ PLT_MOVE (exec (SLc incomplete)) ∧
     $P$ PLT_SECURE_HALT (exec (SLc complete)) ∧
     $P$ PLT_SECURE_HALT (exec (SLc incomplete)) ∧
     (∀ $s$ $cmd$. $P$ $s$ (trap (SLc $cmd$))) ∧
     (∀ $s$ $cmd$. $P$ $s$ (discard (SLc $cmd$))) ∧
     (∀ $s$ $v_6$. $P$ $s$ (discard (ESCc $v_6$))) ∧
     (∀ $s$ $v_9$. $P$ $s$ (trap (ESCc $v_9$))) ∧
     (∀ $v_{12}$. $P$ MOVE_TO_ORP (exec (ESCc $v_{12}$))) ∧
     $P$ MOVE_TO_ORP (exec (SLc pltMove)) ∧
     $P$ MOVE_TO_ORP (exec (SLc pltSecureHalt)) ∧
     $P$ MOVE_TO_ORP (exec (SLc complete)) ∧
     (∀ $v_{15}$. $P$ PLT_FORM (exec (ESCc $v_{15}$))) ∧
     $P$ PLT_FORM (exec (SLc pltForm)) ∧
     $P$ PLT_FORM (exec (SLc pltSecureHalt)) ∧
     $P$ PLT_FORM (exec (SLc complete)) ∧
     (∀ $v_{18}$. $P$ PLT_MOVE (exec (ESCc $v_{18}$))) ∧
     $P$ PLT_MOVE (exec (SLc pltForm)) ∧
```

```
    P PLT_MOVE (exec (SLc pltMove)) ∧
    P PLT_MOVE (exec (SLc complete)) ∧
    (∀ v21. P PLT_SECURE_HALT (exec (ESCc v21))) ∧
    P PLT_SECURE_HALT (exec (SLc pltForm)) ∧
    P PLT_SECURE_HALT (exec (SLc pltMove)) ∧
    P PLT_SECURE_HALT (exec (SLc pltSecureHalt)) ∧
    (∀ v23. P COMPLETE (exec v23)) ⇒
    ∀ v v1. P v v1
```

[moveToORPOut_def]

```
 ⊢ (moveToORPOut MOVE_TO_ORP (exec (SLc pltForm)) = PLTForm) ∧
   (moveToORPOut MOVE_TO_ORP (exec (SLc incomplete)) =
    MoveToORP) ∧
   (moveToORPOut PLT_FORM (exec (SLc pltMove)) = PLTMove) ∧
   (moveToORPOut PLT_FORM (exec (SLc incomplete)) = PLTForm) ∧
   (moveToORPOut PLT_MOVE (exec (SLc pltSecureHalt)) =
    PLTSecureHalt) ∧
   (moveToORPOut PLT_MOVE (exec (SLc incomplete)) = PLTMove) ∧
   (moveToORPOut PLT_SECURE_HALT (exec (SLc complete)) =
    Complete) ∧
   (moveToORPOut PLT_SECURE_HALT (exec (SLc incomplete)) =
    PLTSecureHalt) ∧
   (moveToORPOut s (trap (SLc cmd)) = unAuthorized) ∧
   (moveToORPOut s (discard (SLc cmd)) = unAuthenticated)
```

[moveToORPOut_ind]

```
 ⊢ ∀ P.
    P MOVE_TO_ORP (exec (SLc pltForm)) ∧
    P MOVE_TO_ORP (exec (SLc incomplete)) ∧
    P PLT_FORM (exec (SLc pltMove)) ∧
    P PLT_FORM (exec (SLc incomplete)) ∧
    P PLT_MOVE (exec (SLc pltSecureHalt)) ∧
    P PLT_MOVE (exec (SLc incomplete)) ∧
    P PLT_SECURE_HALT (exec (SLc complete)) ∧
    P PLT_SECURE_HALT (exec (SLc incomplete)) ∧
    (∀ s cmd. P s (trap (SLc cmd))) ∧
    (∀ s cmd. P s (discard (SLc cmd))) ∧
    (∀ s v6. P s (discard (ESCc v6))) ∧
    (∀ s v9. P s (trap (ESCc v9))) ∧
    (∀ v12. P MOVE_TO_ORP (exec (ESCc v12))) ∧
    P MOVE_TO_ORP (exec (SLc pltMove)) ∧
    P MOVE_TO_ORP (exec (SLc pltSecureHalt)) ∧
    P MOVE_TO_ORP (exec (SLc complete)) ∧
    (∀ v15. P PLT_FORM (exec (ESCc v15))) ∧
    P PLT_FORM (exec (SLc pltForm)) ∧
    P PLT_FORM (exec (SLc pltSecureHalt)) ∧
    P PLT_FORM (exec (SLc complete)) ∧
    (∀ v18. P PLT_MOVE (exec (ESCc v18))) ∧
    P PLT_MOVE (exec (SLc pltForm)) ∧
    P PLT_MOVE (exec (SLc pltMove)) ∧
    P PLT_MOVE (exec (SLc complete)) ∧
    (∀v21. P PLT_SECURE_HALT (exec (ESCc v21))) ∧
    P PLT_SECURE_HALT (exec (SLc pltForm)) ∧
    P PLT_SECURE_HALT (exec (SLc pltMove)) ∧
    P PLT_SECURE_HALT (exec (SLc pltSecureHalt)) ∧
    (∀v23. P COMPLETE (exec v23)) ⇒
    ∀v v1. P v v1
```

[PlatoonLeader_exec_slCommand_justified_thm]

```
 ⊢ ∀NS Out M Oi Os.
     TR (M,Oi,Os) (exec (SLc slCommand))
       (CFG authTestMoveToORP ssmMoveToORPStateInterp
          (secContextMoveToORP slCommand)
          (Name PlatoonLeader says prop (SOME (SLc slCommand))::
               ins) s outs)
       (CFG authTestMoveToORP ssmMoveToORPStateInterp
          (secContextMoveToORP slCommand) ins
          (NS s (exec (SLc slCommand)))
          (Out s (exec (SLc slCommand))::outs)) ⇔
     authTestMoveToORP
       (Name PlatoonLeader says prop (SOME (SLc slCommand))) ∧
     CFGInterpret (M,Oi,Os)
       (CFG authTestMoveToORP ssmMoveToORPStateInterp
          (secContextMoveToORP slCommand)
          (Name PlatoonLeader says prop (SOME (SLc slCommand))::
               ins) s outs) ∧
     (M,Oi,Os) sat prop (SOME (SLc slCommand))
```

[PlatoonLeader_slCommand_lemma]

```
 ⊢ CFGInterpret (M,Oi,Os)
     (CFG authTestMoveToORP ssmMoveToORPStateInterp
        (secContextMoveToORP slCommand)
        (Name PlatoonLeader says prop (SOME (SLc slCommand))::
             ins) s outs) ⇒
   (M,Oi,Os) sat prop (SOME (SLc slCommand))
```

14 MoveToORPType Theory

Built: 10 June 2018

Parent Theories: indexedLists, patternMatches

14.1 Datatypes

```
slCommand = pltForm | pltMove | pltSecureHalt | complete
          | incomplete
```

```
slOutput = MoveToORP | PLTForm | PLTMove | PLTSecureHalt
         | Complete | unAuthorized | unAuthenticated

slState = MOVE_TO_ORP | PLT_FORM | PLT_MOVE | PLT_SECURE_HALT
        | COMPLETE

stateRole = PlatoonLeader
```

14.2 Theorems

[slCommand_distinct_clauses]

```
⊢ pltForm ≠ pltMove ∧ pltForm ≠ pltSecureHalt ∧
  pltForm ≠ complete ∧ pltForm ≠ incomplete ∧
  pltMove ≠ pltSecureHalt ∧ pltMove ≠ complete ∧
  pltMove ≠ incomplete ∧ pltSecureHalt ≠ complete ∧
  pltSecureHalt ≠ incomplete ∧ complete ≠ incomplete
```

[slOutput_distinct_clauses]

```
⊢ MoveToORP ≠ PLTForm ∧ MoveToORP ≠ PLTMove ∧
  MoveToORP ≠ PLTSecureHalt ∧ MoveToORP ≠ Complete ∧
  MoveToORP ≠ unAuthorized ∧ MoveToORP ≠ unAuthenticated ∧
  PLTForm ≠ PLTMove ∧ PLTForm ≠ PLTSecureHalt ∧
  PLTForm ≠ Complete ∧ PLTForm ≠ unAuthorized ∧
  PLTForm ≠ unAuthenticated ∧ PLTMove ≠ PLTSecureHalt ∧
  PLTMove ≠ Complete ∧ PLTMove ≠ unAuthorized ∧
  PLTMove ≠ unAuthenticated ∧ PLTSecureHalt ≠ Complete ∧
  PLTSecureHalt ≠ unAuthorized ∧
  PLTSecureHalt ≠ unAuthenticated ∧ Complete ≠ unAuthorized ∧
  Complete ≠ unAuthenticated ∧ unAuthorized ≠ unAuthenticated
```

[slState_distinct_clauses]

```
⊢ MOVE_TO_ORP ≠ PLT_FORM ∧ MOVE_TO_ORP ≠ PLT_MOVE ∧
  MOVE_TO_ORP ≠ PLT_SECURE_HALT ∧ MOVE_TO_ORP ≠ COMPLETE ∧
  PLT_FORM ≠ PLT_MOVE ∧ PLT_FORM ≠ PLT_SECURE_HALT ∧
  PLT_FORM ≠ COMPLETE ∧ PLT_MOVE ≠ PLT_SECURE_HALT ∧
  PLT_MOVE ≠ COMPLETE ∧ PLT_SECURE_HALT ≠ COMPLETE
```

15 ssmMoveToPB Theory

Built: 10 June 2018

Parent Theories: MoveToPBType, ssm11, OMNIType

15.1 Definitions

[secContextMoveToPB_def]

```
⊢ ∀ cmd.
    secContextMoveToPB cmd =
    [Name PlatoonLeader controls prop (SOME (SLc cmd))]
```

[ssmMoveToPBStateInterp_def]

⊢ ∀ *state*. ssmMoveToPBStateInterp *state* = TT

15.2 Theorems

[authTestMoveToPB_cmd_reject_lemma]

⊢ ∀ *cmd*. ¬authTestMoveToPB (prop (SOME *cmd*))

[authTestMoveToPB_def]

⊢ (authTestMoveToPB (Name PlatoonLeader says prop *cmd*) ⟺ T) ∧
(authTestMoveToPB TT ⟺ F) ∧ (authTestMoveToPB FF ⟺ F) ∧
(authTestMoveToPB (prop v) ⟺ F) ∧
(authTestMoveToPB (notf v_1) ⟺ F) ∧
(authTestMoveToPB (v_2 andf v_3) ⟺ F) ∧
(authTestMoveToPB (v_4 orf v_5) ⟺ F) ∧
(authTestMoveToPB (v_6 impf v_7) ⟺ F) ∧
(authTestMoveToPB (v_8 eqf v_9) ⟺ F) ∧
(authTestMoveToPB (v_{10} says TT) ⟺ F) ∧
(authTestMoveToPB (v_{10} says FF) ⟺ F) ∧
(authTestMoveToPB (*v133* meet *v134* says prop v_{66}) ⟺ F) ∧
(authTestMoveToPB (*v135* quoting *v136* says prop v_{66}) ⟺ F) ∧
(authTestMoveToPB (v_{10} says notf v_{67}) ⟺ F) ∧
(authTestMoveToPB (v_{10} says (v_{68} andf v_{69})) ⟺ F) ∧
(authTestMoveToPB (v_{10} says (v_{70} orf v_{71})) ⟺ F) ∧
(authTestMoveToPB (v_{10} says (v_{72} impf v_{73})) ⟺ F) ∧
(authTestMoveToPB (v_{10} says (v_{74} eqf v_{75})) ⟺ F) ∧
(authTestMoveToPB (v_{10} says v_{76} says v_{77}) ⟺ F) ∧
(authTestMoveToPB (v_{10} says v_{78} speaks_for v_{79}) ⟺ F) ∧
(authTestMoveToPB (v_{10} says v_{80} controls v_{81}) ⟺ F) ∧
(authTestMoveToPB (v_{10} says reps v_{82} v_{83} v_{84}) ⟺ F) ∧
(authTestMoveToPB (v_{10} says v_{85} domi v_{86}) ⟺ F) ∧
(authTestMoveToPB (v_{10} says v_{87} eqi v_{88}) ⟺ F) ∧
(authTestMoveToPB (v_{10} says v_{89} doms v_{90}) ⟺ F) ∧
(authTestMoveToPB (v_{10} says v_{91} eqs v_{92}) ⟺ F) ∧
(authTestMoveToPB (v_{10} says v_{93} eqn v_{94}) ⟺ F) ∧
(authTestMoveToPB (v_{10} says v_{95} lte v_{96}) ⟺ F) ∧
(authTestMoveToPB (v_{10} says v_{97} lt v_{98}) ⟺ F) ∧
(authTestMoveToPB (v_{12} speaks_for v_{13}) ⟺ F) ∧
(authTestMoveToPB (v_{14} controls v_{15}) ⟺ F) ∧
(authTestMoveToPB (reps v_{16} v_{17} v_{18}) ⟺ F) ∧
(authTestMoveToPB (v_{19} domi v_{20}) ⟺ F) ∧
(authTestMoveToPB (v_{21} eqi v_{22}) ⟺ F) ∧
(authTestMoveToPB (v_{23} doms v_{24}) ⟺ F) ∧
(authTestMoveToPB (v_{25} eqs v_{26}) ⟺ F) ∧
(authTestMoveToPB (v_{27} eqn v_{28}) ⟺ F) ∧
(authTestMoveToPB (v_{29} lte v_{30}) ⟺ F) ∧
(authTestMoveToPB (v_{31} lt v_{32}) ⟺ F)

[authTestMoveToPB_ind]

```
⊢ ∀P.
     (∀cmd. P (Name PlatoonLeader says prop cmd)) ∧ P TT ∧
     P FF ∧ (∀v. P (prop v)) ∧ (∀v_1. P (notf v_1)) ∧
     (∀v_2 v_3. P (v_2 andf v_3)) ∧ (∀v_4 v_5. P (v_4 orf v_5)) ∧
     (∀v_6 v_7. P (v_6 impf v_7)) ∧ (∀v_8 v_9. P (v_8 eqf v_9)) ∧
     (∀v_10. P (v_10 says TT)) ∧ (∀v_10. P (v_10 says FF)) ∧
     (∀v133 v134 v_66. P (v133 meet v134 says prop v_66)) ∧
     (∀v135 v136 v_66. P (v135 quoting v136 says prop v_66)) ∧
     (∀v_10 v_67. P (v_10 says notf v_67)) ∧
     (∀v_10 v_68 v_69. P (v_10 says (v_68 andf v_69))) ∧
     (∀v_10 v_70 v_71. P (v_10 says (v_70 orf v_71))) ∧
     (∀v_10 v_72 v_73. P (v_10 says (v_72 impf v_73))) ∧
     (∀v_10 v_74 v_75. P (v_10 says (v_74 eqf v_75))) ∧
     (∀v_10 v_76 v_77. P (v_10 says v_76 says v_77)) ∧
     (∀v_10 v_78 v_79. P (v_10 says v_78 speaks_for v_79)) ∧
     (∀v_10 v_80 v_81. P (v_10 says v_80 controls v_81)) ∧
     (∀v_10 v_82 v_83 v_84. P (v_10 says reps v_82 v_83 v_84)) ∧
     (∀v_10 v_85 v_86. P (v_10 says v_85 domi v_86)) ∧
     (∀v_10 v_87 v_88. P (v_10 says v_87 eqi v_88)) ∧
     (∀v_10 v_89 v_90. P (v_10 says v_89 doms v_90)) ∧
     (∀v_10 v_91 v_92. P (v_10 says v_91 eqs v_92)) ∧
     (∀v_10 v_93 v_94. P (v_10 says v_93 eqn v_94)) ∧
     (∀v_10 v_95 v_96. P (v_10 says v_95 lte v_96)) ∧
     (∀v_10 v_97 v_98. P (v_10 says v_97 lt v_98)) ∧
     (∀v_12 v_13. P (v_12 speaks_for v_13)) ∧
     (∀v_14 v_15. P (v_14 controls v_15)) ∧
     (∀v_16 v_17 v_18. P (reps v_16 v_17 v_18)) ∧
     (∀v_19 v_20. P (v_19 domi v_20)) ∧
     (∀v_21 v_22. P (v_21 eqi v_22)) ∧
     (∀v_23 v_24. P (v_23 doms v_24)) ∧
     (∀v_25 v_26. P (v_25 eqs v_26)) ∧ (∀v_27 v_28. P (v_27 eqn v_28)) ∧
     (∀v_29 v_30. P (v_29 lte v_30)) ∧ (∀v_31 v_32. P (v_31 lt v_32)) ⇒
     ∀v. P v
```

[moveToPBNS_def]

```
⊢ (moveToPBNS MOVE_TO_PB (exec (SLc pltForm)) = PLT_FORM) ∧
  (moveToPBNS MOVE_TO_PB (exec (SLc incomplete)) =
   MOVE_TO_PB) ∧
  (moveToPBNS PLT_FORM (exec (SLc pltMove)) = PLT_MOVE) ∧
  (moveToPBNS PLT_FORM (exec (SLc incomplete)) = PLT_FORM) ∧
  (moveToPBNS PLT_MOVE (exec (SLc pltHalt)) = PLT_HALT) ∧
  (moveToPBNS PLT_MOVE (exec (SLc incomplete)) = PLT_MOVE) ∧
  (moveToPBNS PLT_HALT (exec (SLc complete)) = COMPLETE) ∧
  (moveToPBNS PLT_HALT (exec (SLc incomplete)) = PLT_HALT) ∧
  (moveToPBNS s (trap (SLc cmd)) = s) ∧
  (moveToPBNS s (discard (SLc cmd)) = s)
```

[moveToPBNS_ind]

$\vdash \forall P.$
$\quad P$ `MOVE_TO_PB (exec (SLc pltForm))` $\land$
$\quad P$ `MOVE_TO_PB (exec (SLc incomplete))` $\land$
$\quad P$ `PLT_FORM (exec (SLc pltMove))` $\land$
$\quad P$ `PLT_FORM (exec (SLc incomplete))` $\land$
$\quad P$ `PLT_MOVE (exec (SLc pltHalt))` $\land$
$\quad P$ `PLT_MOVE (exec (SLc incomplete))` $\land$
$\quad P$ `PLT_HALT (exec (SLc complete))` $\land$
$\quad P$ `PLT_HALT (exec (SLc incomplete))` $\land$
$\quad (\forall s\ cmd.\ P\ s$ `(trap (SLc` cmd`)))` $\land$
$\quad (\forall s\ cmd.\ P\ s$ `(discard (SLc` cmd`)))` $\land$
$\quad (\forall s\ v_6.\ P\ s$ `(discard (ESCc` v_6`)))` $\land$
$\quad (\forall s\ v_9.\ P\ s$ `(trap (ESCc` v_9`)))` $\land$
$\quad (\forall v_{12}.\ P$ `MOVE_TO_PB (exec (ESCc` v_{12}`)))` $\land$
$\quad P$ `MOVE_TO_PB (exec (SLc pltMove))` $\land$
$\quad P$ `MOVE_TO_PB (exec (SLc pltHalt))` $\land$
$\quad P$ `MOVE_TO_PB (exec (SLc complete))` $\land$
$\quad (\forall v_{15}.\ P$ `PLT_FORM (exec (ESCc` v_{15}`)))` $\land$
$\quad P$ `PLT_FORM (exec (SLc pltForm))` $\land$
$\quad P$ `PLT_FORM (exec (SLc pltHalt))` $\land$
$\quad P$ `PLT_FORM (exec (SLc complete))` $\land$
$\quad (\forall v_{18}.\ P$ `PLT_MOVE (exec (ESCc` v_{18}`)))` $\land$
$\quad P$ `PLT_MOVE (exec (SLc pltForm))` $\land$
$\quad P$ `PLT_MOVE (exec (SLc pltMove))` $\land$
$\quad P$ `PLT_MOVE (exec (SLc complete))` $\land$
$\quad (\forall v_{21}.\ P$ `PLT_HALT (exec (ESCc` v_{21}`)))` $\land$
$\quad P$ `PLT_HALT (exec (SLc pltForm))` $\land$
$\quad P$ `PLT_HALT (exec (SLc pltMove))` $\land$
$\quad P$ `PLT_HALT (exec (SLc pltHalt))` $\land$
$\quad (\forall v_{23}.\ P$ `COMPLETE (exec` v_{23}`))` $\Rightarrow$
$\quad \forall v\ v_1.\ P\ v\ v_1$

[moveToPBOut_def]

$\vdash$ `(moveToPBOut MOVE_TO_PB (exec (SLc pltForm)) = PLTForm)` $\land$
`(moveToPBOut MOVE_TO_PB (exec (SLc incomplete)) = MoveToPB)` $\land$
`(moveToPBOut PLT_FORM (exec (SLc pltMove)) = PLTMove)` $\land$
`(moveToPBOut PLT_FORM (exec (SLc incomplete)) = PLTForm)` $\land$
`(moveToPBOut PLT_MOVE (exec (SLc pltHalt)) = PLTHalt)` $\land$
`(moveToPBOut PLT_MOVE (exec (SLc incomplete)) = PLTMove)` $\land$
`(moveToPBOut PLT_HALT (exec (SLc complete)) = Complete)` $\land$
`(moveToPBOut PLT_HALT (exec (SLc incomplete)) = PLTHalt)` $\land$
`(moveToPBOut` s `(trap (SLc` cmd`)) = unAuthorized)` $\land$
`(moveToPBOut` s `(discard (SLc` cmd`)) = unAuthenticated)`

[moveToPBOut_ind]

$\vdash \forall P.$
$\quad P$ `MOVE_TO_PB (exec (SLc pltForm))` $\land$

```
    P MOVE_TO_PB (exec (SLc incomplete)) ∧
    P PLT_FORM (exec (SLc pltMove)) ∧
    P PLT_FORM (exec (SLc incomplete)) ∧
    P PLT_MOVE (exec (SLc pltHalt)) ∧
    P PLT_MOVE (exec (SLc incomplete)) ∧
    P PLT_HALT (exec (SLc complete)) ∧
    P PLT_HALT (exec (SLc incomplete)) ∧
    (∀s cmd. P s (trap (SLc cmd))) ∧
    (∀s cmd. P s (discard (SLc cmd))) ∧
    (∀s v6. P s (discard (ESCc v6))) ∧
    (∀s v9. P s (trap (ESCc v9))) ∧
    (∀v12. P MOVE_TO_PB (exec (ESCc v12))) ∧
    P MOVE_TO_PB (exec (SLc pltMove)) ∧
    P MOVE_TO_PB (exec (SLc pltHalt)) ∧
    P MOVE_TO_PB (exec (SLc complete)) ∧
    (∀v15. P PLT_FORM (exec (ESCc v15))) ∧
    P PLT_FORM (exec (SLc pltForm)) ∧
    P PLT_FORM (exec (SLc pltHalt)) ∧
    P PLT_FORM (exec (SLc complete)) ∧
    (∀v18. P PLT_MOVE (exec (ESCc v18))) ∧
    P PLT_MOVE (exec (SLc pltForm)) ∧
    P PLT_MOVE (exec (SLc pltMove)) ∧
    P PLT_MOVE (exec (SLc complete)) ∧
    (∀v21. P PLT_HALT (exec (ESCc v21))) ∧
    P PLT_HALT (exec (SLc pltForm)) ∧
    P PLT_HALT (exec (SLc pltMove)) ∧
    P PLT_HALT (exec (SLc pltHalt)) ∧
    (∀v23. P COMPLETE (exec v23)) ⇒
    ∀v v1. P v v1
```

[PlatoonLeader_exec_slCommand_justified_thm]

```
 ⊢ ∀NS Out M Oi Os.
     TR (M,Oi,Os) (exec (SLc slCommand))
       (CFG authTestMoveToPB ssmMoveToPBStateInterp
          (secContextMoveToPB slCommand)
          (Name PlatoonLeader says prop (SOME (SLc slCommand))::
               ins) s outs)
       (CFG authTestMoveToPB ssmMoveToPBStateInterp
          (secContextMoveToPB slCommand) ins
          (NS s (exec (SLc slCommand)))
          (Out s (exec (SLc slCommand))::outs)) ⟺
     authTestMoveToPB
       (Name PlatoonLeader says prop (SOME (SLc slCommand))) ∧
     CFGInterpret (M,Oi,Os)
       (CFG authTestMoveToPB ssmMoveToPBStateInterp
          (secContextMoveToPB slCommand)
          (Name PlatoonLeader says prop (SOME (SLc slCommand))::
               ins) s outs) ∧
     (M,Oi,Os) sat prop (SOME (SLc slCommand))
```

[PlatoonLeader_slCommand_lemma]

```
⊢ CFGInterpret (M,Oi,Os)
    (CFG authTestMoveToPB ssmMoveToPBStateInterp
       (secContextMoveToPB slCommand)
       (Name PlatoonLeader says prop (SOME (SLc slCommand))::
            ins) s outs) ⇒
  (M,Oi,Os) sat prop (SOME (SLc slCommand))
```

16 MoveToPBType Theory

Built: 10 June 2018
Parent Theories: indexedLists, patternMatches

16.1 Datatypes

```
slCommand = pltForm | pltMove | pltHalt | complete | incomplete

slOutput = MoveToPB | PLTForm | PLTMove | PLTHalt | Complete
         | unAuthorized | unAuthenticated

slState = MOVE_TO_PB | PLT_FORM | PLT_MOVE | PLT_HALT | COMPLETE

stateRole = PlatoonLeader
```

16.2 Theorems

[slCommand_distinct_clauses]

```
⊢ pltForm ≠ pltMove ∧ pltForm ≠ pltHalt ∧ pltForm ≠ complete ∧
  pltForm ≠ incomplete ∧ pltMove ≠ pltHalt ∧
  pltMove ≠ complete ∧ pltMove ≠ incomplete ∧
  pltHalt ≠ complete ∧ pltHalt ≠ incomplete ∧
  complete ≠ incomplete
```

[slOutput_distinct_clauses]

```
⊢ MoveToPB ≠ PLTForm ∧ MoveToPB ≠ PLTMove ∧
  MoveToPB ≠ PLTHalt ∧ MoveToPB ≠ Complete ∧
  MoveToPB ≠ unAuthorized ∧ MoveToPB ≠ unAuthenticated ∧
  PLTForm ≠ PLTMove ∧ PLTForm ≠ PLTHalt ∧ PLTForm ≠ Complete ∧
  PLTForm ≠ unAuthorized ∧ PLTForm ≠ unAuthenticated ∧
  PLTMove ≠ PLTHalt ∧ PLTMove ≠ Complete ∧
  PLTMove ≠ unAuthorized ∧ PLTMove ≠ unAuthenticated ∧
  PLTHalt ≠ Complete ∧ PLTHalt ≠ unAuthorized ∧
  PLTHalt ≠ unAuthenticated ∧ Complete ≠ unAuthorized ∧
  Complete ≠ unAuthenticated ∧ unAuthorized ≠ unAuthenticated
```

[slState_distinct_clauses]

```
⊢ MOVE_TO_PB ≠ PLT_FORM ∧ MOVE_TO_PB ≠ PLT_MOVE ∧
  MOVE_TO_PB ≠ PLT_HALT ∧ MOVE_TO_PB ≠ COMPLETE ∧
  PLT_FORM ≠ PLT_MOVE ∧ PLT_FORM ≠ PLT_HALT ∧
  PLT_FORM ≠ COMPLETE ∧ PLT_MOVE ≠ PLT_HALT ∧
  PLT_MOVE ≠ COMPLETE ∧ PLT_HALT ≠ COMPLETE
```

17 ssmPlanPB Theory

Built: 10 June 2018
Parent Theories: PlanPBDef, ssm

17.1 Theorems

[inputOK_def]

```
⊢ (inputOK (Name PlatoonLeader says prop cmd) ⇔ T) ∧
  (inputOK (Name PlatoonSergeant says prop 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 (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)
```

[inputOK_ind]

```
 ⊢ ∀P.
     (∀cmd. P (Name PlatoonLeader says prop cmd)) ∧
     (∀cmd. P (Name PlatoonSergeant says prop cmd)) ∧ P TT ∧
     P FF ∧ (∀v. P (prop v)) ∧ (∀v1. P (notf v1)) ∧
     (∀v2 v3. P (v2 andf v3)) ∧ (∀v4 v5. P (v4 orf v5)) ∧
     (∀v6 v7. P (v6 impf v7)) ∧ (∀v8 v9. P (v8 eqf v9)) ∧
     (∀v10. P (v10 says TT)) ∧ (∀v10. P (v10 says FF)) ∧
     (∀v133 v134 v66. P (v133 meet v134 says prop v66)) ∧
     (∀v135 v136 v66. P (v135 quoting v136 says prop v66)) ∧
     (∀v10 v67. P (v10 says notf v67)) ∧
     (∀v10 v68 v69. P (v10 says (v68 andf v69))) ∧
     (∀v10 v70 v71. P (v10 says (v70 orf v71))) ∧
     (∀v10 v72 v73. P (v10 says (v72 impf v73))) ∧
     (∀v10 v74 v75. P (v10 says (v74 eqf v75))) ∧
     (∀v10 v76 v77. P (v10 says v76 says v77)) ∧
     (∀v10 v78 v79. P (v10 says v78 speaks_for v79)) ∧
     (∀v10 v80 v81. P (v10 says v80 controls v81)) ∧
     (∀v10 v82 v83 v84. P (v10 says reps v82 v83 v84)) ∧
     (∀v10 v85 v86. P (v10 says v85 domi v86)) ∧
     (∀v10 v87 v88. P (v10 says v87 eqi v88)) ∧
     (∀v10 v89 v90. P (v10 says v89 doms v90)) ∧
     (∀v10 v91 v92. P (v10 says v91 eqs v92)) ∧
     (∀v10 v93 v94. P (v10 says v93 eqn v94)) ∧
     (∀v10 v95 v96. P (v10 says v95 lte v96)) ∧
     (∀v10 v97 v98. P (v10 says v97 lt v98)) ∧
     (∀v12 v13. P (v12 speaks_for v13)) ∧
     (∀v14 v15. P (v14 controls v15)) ∧
     (∀v16 v17 v18. P (reps v16 v17 v18)) ∧
     (∀v19 v20. P (v19 domi v20)) ∧
     (∀v21 v22. P (v21 eqi v22)) ∧
     (∀v23 v24. P (v23 doms v24)) ∧
     (∀v25 v26. P (v25 eqs v26)) ∧ (∀v27 v28. P (v27 eqn v28)) ∧
     (∀v29 v30. P (v29 lte v30)) ∧ (∀v31 v32. P (v31 lt v32)) ⇒
     ∀v. P v
```

[planPBNS_def]

```
 ⊢ (planPBNS WARNO (exec x) =
    if
      (getRecon x = [SOME (SLc (PL recon))]) ∧
      (getTenativePlan x = [SOME (SLc (PL tentativePlan))]) ∧
      (getReport x = [SOME (SLc (PL report1))]) ∧
      (getInitMove x = [SOME (SLc (PSG initiateMovement))])
    then
      REPORT1
    else WARNO) ∧
```

```
(planPBNS PLAN_PB (exec x) =
 if getPlCom x = receiveMission then RECEIVE_MISSION
 else PLAN_PB) ∧
(planPBNS RECEIVE_MISSION (exec x) =
 if getPlCom x = warno then WARNO else RECEIVE_MISSION) ∧
(planPBNS REPORT1 (exec x) =
 if getPlCom x = completePlan then COMPLETE_PLAN
 else REPORT1) ∧
(planPBNS COMPLETE_PLAN (exec x) =
 if getPlCom x = opoid then OPOID else COMPLETE_PLAN) ∧
(planPBNS OPOID (exec x) =
 if getPlCom x = supervise then SUPERVISE else OPOID) ∧
(planPBNS SUPERVISE (exec x) =
 if getPlCom x = report2 then REPORT2 else SUPERVISE) ∧
(planPBNS REPORT2 (exec x) =
 if getPlCom x = complete then COMPLETE else REPORT2) ∧
(planPBNS s (trap v0) = s) ∧ (planPBNS s (discard v1) = s)
```

[planPBNS_ind]

```
⊢ ∀P.
    (∀x. P WARNO (exec x)) ∧ (∀x. P PLAN_PB (exec x)) ∧
    (∀x. P RECEIVE_MISSION (exec x)) ∧
    (∀x. P REPORT1 (exec x)) ∧ (∀x. P COMPLETE_PLAN (exec x)) ∧
    (∀x. P OPOID (exec x)) ∧ (∀x. P SUPERVISE (exec x)) ∧
    (∀x. P REPORT2 (exec x)) ∧ (∀s v0. P s (trap v0)) ∧
    (∀s v1. P s (discard v1)) ∧
    (∀v6. P TENTATIVE_PLAN (exec v6)) ∧
    (∀v7. P INITIATE_MOVEMENT (exec v7)) ∧
    (∀v8. P RECON (exec v8)) ∧ (∀v9. P COMPLETE (exec v9)) ⇒
    ∀v v1. P v v1
```

[planPBOut_def]

```
⊢ (planPBOut WARNO (exec x) =
   if
     (getRecon x = [SOME (SLc (PL recon))]) ∧
     (getTenativePlan x = [SOME (SLc (PL tentativePlan))]) ∧
     (getReport x = [SOME (SLc (PL report1))]) ∧
     (getInitMove x = [SOME (SLc (PSG initiateMovement))])
   then
     Report1
   else unAuthorized) ∧
  (planPBOut PLAN_PB (exec x) =
   if getPlCom x = receiveMission then ReceiveMission
   else unAuthorized) ∧
  (planPBOut RECEIVE_MISSION (exec x) =
   if getPlCom x = warno then Warno else unAuthorized) ∧
  (planPBOut REPORT1 (exec x) =
   if getPlCom x = completePlan then CompletePlan
   else unAuthorized) ∧
```

```
(planPBOut COMPLETE_PLAN (exec x) =
 if getPlCom x = opoid then Opoid else unAuthorized) ∧
(planPBOut OPOID (exec x) =
 if getPlCom x = supervise then Supervise
 else unAuthorized) ∧
(planPBOut SUPERVISE (exec x) =
 if getPlCom x = report2 then Report2 else unAuthorized) ∧
(planPBOut REPORT2 (exec x) =
 if getPlCom x = complete then Complete else unAuthorized) ∧
(planPBOut s (trap v0) = unAuthorized) ∧
(planPBOut s (discard v1) = unAuthenticated)
```

[planPBOut_ind]

```
⊢ ∀P.
    (∀x. P WARNO (exec x)) ∧ (∀x. P PLAN_PB (exec x)) ∧
    (∀x. P RECEIVE_MISSION (exec x)) ∧
    (∀x. P REPORT1 (exec x)) ∧ (∀x. P COMPLETE_PLAN (exec x)) ∧
    (∀x. P OPOID (exec x)) ∧ (∀x. P SUPERVISE (exec x)) ∧
    (∀x. P REPORT2 (exec x)) ∧ (∀s v0. P s (trap v0)) ∧
    (∀s v1. P s (discard v1)) ∧
    (∀v6. P TENTATIVE_PLAN (exec v6)) ∧
    (∀v7. P INITIATE_MOVEMENT (exec v7)) ∧
    (∀v8. P RECON (exec v8)) ∧ (∀v9. P COMPLETE (exec v9)) ⇒
    ∀v v1. P v v1
```

[PlatoonLeader_notWARNO_notreport1_exec_plCommand_justified_lemma]

```
⊢ s ≠ WARNO ⇒
  plCommand ≠ invalidPlCommand ⇒
  plCommand ≠ report1 ⇒
  ∀NS Out M Oi Os.
    TR (M,Oi,Os)
      (exec
         (inputList
            [Name PlatoonLeader says
             prop (SOME (SLc (PL plCommand)))]))
      (CFG inputOK secContext secContextNull
         ([Name PlatoonLeader says
           prop (SOME (SLc (PL plCommand)))]::ins) s outs)
      (CFG inputOK secContext secContextNull ins
         (NS s
            (exec
               (inputList
                  [Name PlatoonLeader says
                   prop (SOME (SLc (PL plCommand)))])))
         (Out s
            (exec
               (inputList
                  [Name PlatoonLeader says
                   prop (SOME (SLc (PL plCommand)))]))::
```

```
              outs)) ⟺
    authenticationTest inputOK
      [Name PlatoonLeader says
       prop (SOME (SLc (PL plCommand)))] ∧
    CFGInterpret (M,Oi,Os)
      (CFG inputOK secContext secContextNull
         ([Name PlatoonLeader says
           prop (SOME (SLc (PL plCommand)))]::ins) s outs) ∧
    (M,Oi,Os) satList
    propCommandList
      [Name PlatoonLeader says
       prop (SOME (SLc (PL plCommand)))]
```

[PlatoonLeader_notWARNO_notreport1_exec_plCommand_justified_thm]

```
 ⊢ s ≠ WARNO ⇒
   plCommand ≠ invalidPlCommand ⇒
   plCommand ≠ report1 ⇒
   ∀NS Out M Oi Os.
     TR (M,Oi,Os) (exec [SOME (SLc (PL plCommand))])
       (CFG inputOK secContext secContextNull
          ([Name PlatoonLeader says
            prop (SOME (SLc (PL plCommand)))]::ins) s outs)
       (CFG inputOK secContext secContextNull ins
          (NS s (exec [SOME (SLc (PL plCommand))]))
          (Out s (exec [SOME (SLc (PL plCommand))])::outs)) ⟺
     authenticationTest inputOK
       [Name PlatoonLeader says
        prop (SOME (SLc (PL plCommand)))] ∧
     CFGInterpret (M,Oi,Os)
       (CFG inputOK secContext secContextNull
          ([Name PlatoonLeader says
            prop (SOME (SLc (PL plCommand)))]::ins) s outs) ∧
     (M,Oi,Os) satList [prop (SOME (SLc (PL plCommand)))]
```

[PlatoonLeader_notWARNO_notreport1_exec_plCommand_lemma]

```
 ⊢ s ≠ WARNO ⇒
   plCommand ≠ invalidPlCommand ⇒
   plCommand ≠ report1 ⇒
   ∀M Oi Os.
     CFGInterpret (M,Oi,Os)
       (CFG inputOK secContext secContextNull
          ([Name PlatoonLeader says
            prop (SOME (SLc (PL plCommand)))]::ins) s outs) ⇒
     (M,Oi,Os) satList
     propCommandList
       [Name PlatoonLeader says
        prop (SOME (SLc (PL plCommand)))]
```

[PlatoonLeader_psgCommand_notDiscard_thm]
```
 ⊢ ∀NS Out M Oi Os.
     ¬TR (M,Oi,Os) (discard [SOME (SLc (PSG psgCommand))])
        (CFG inputOK secContext secContextNull
           ([Name PlatoonLeader says
             prop (SOME (SLc (PSG psgCommand)))]::ins) s outs)
        (CFG inputOK secContext secContextNull ins
           (NS s (discard [SOME (SLc (PSG psgCommand))]))
           (Out s (discard [SOME (SLc (PSG psgCommand))])::
                outs))
```

[PlatoonLeader_trap_psgCommand_justified_lemma]
```
 ⊢ ∀NS Out M Oi Os.
     TR (M,Oi,Os)
       (trap
          (inputList
             [Name PlatoonLeader says
              prop (SOME (SLc (PSG psgCommand)))]))
       (CFG inputOK secContext secContextNull
          ([Name PlatoonLeader says
            prop (SOME (SLc (PSG psgCommand)))]::ins) s outs)
       (CFG inputOK secContext secContextNull ins
          (NS s
             (trap
                (inputList
                   [Name PlatoonLeader says
                    prop (SOME (SLc (PSG psgCommand)))])))
          (Out s
             (trap
                (inputList
                   [Name PlatoonLeader says
                    prop (SOME (SLc (PSG psgCommand)))]))::
               outs)) ⟺
     authenticationTest inputOK
       [Name PlatoonLeader says
        prop (SOME (SLc (PSG psgCommand)))] ∧
     CFGInterpret (M,Oi,Os)
       (CFG inputOK secContext secContextNull
          ([Name PlatoonLeader says
            prop (SOME (SLc (PSG psgCommand)))]::ins) s outs) ∧
     (M,Oi,Os) sat prop NONE
```

[PlatoonLeader_trap_psgCommand_lemma]
```
 ⊢ ∀M Oi Os.
     CFGInterpret (M,Oi,Os)
       (CFG inputOK secContext secContextNull
          ([Name PlatoonLeader says
            prop (SOME (SLc (PSG psgCommand)))]::ins) s outs) ⇒
     (M,Oi,Os) sat prop NONE
```

[PlatoonLeader_WARNO_exec_report1_justified_lemma]

```
 ⊢ ∀NS Out M Oi Os.
     TR (M,Oi,Os)
       (exec
          (inputList
             [Name PlatoonLeader says
              prop (SOME (SLc (PL recon)));
              Name PlatoonLeader says
              prop (SOME (SLc (PL tentativePlan)));
              Name PlatoonSergeant says
              prop (SOME (SLc (PSG initiateMovement)));
              Name PlatoonLeader says
              prop (SOME (SLc (PL report1)))]))
       (CFG inputOK secContext secContextNull
          ([Name PlatoonLeader says
            prop (SOME (SLc (PL recon)));
            Name PlatoonLeader says
            prop (SOME (SLc (PL tentativePlan)));
            Name PlatoonSergeant says
            prop (SOME (SLc (PSG initiateMovement)));
            Name PlatoonLeader says
            prop (SOME (SLc (PL report1)))]::ins) WARNO outs)
       (CFG inputOK secContext secContextNull ins
          (NS WARNO
             (exec
                (inputList
                   [Name PlatoonLeader says
                    prop (SOME (SLc (PL recon)));
                    Name PlatoonLeader says
                    prop (SOME (SLc (PL tentativePlan)));
                    Name PlatoonSergeant says
                    prop (SOME (SLc (PSG initiateMovement)));
                    Name PlatoonLeader says
                    prop (SOME (SLc (PL report1)))])))
          (Out WARNO
             (exec
                (inputList
                   [Name PlatoonLeader says
                    prop (SOME (SLc (PL recon)));
                    Name PlatoonLeader says
                    prop (SOME (SLc (PL tentativePlan)));
                    Name PlatoonSergeant says
                    prop (SOME (SLc (PSG initiateMovement)));
                    Name PlatoonLeader says
                    prop (SOME (SLc (PL report1)))]))::outs)) ⇔
     authenticationTest inputOK
       [Name PlatoonLeader says prop (SOME (SLc (PL recon)));
        Name PlatoonLeader says
        prop (SOME (SLc (PL tentativePlan)));
```

```
      Name PlatoonSergeant says
      prop (SOME (SLc (PSG initiateMovement)));
      Name PlatoonLeader says
      prop (SOME (SLc (PL report1)))] ∧
   CFGInterpret (M,Oi,Os)
     (CFG inputOK secContext secContextNull
        ([Name PlatoonLeader says
          prop (SOME (SLc (PL recon)));
          Name PlatoonLeader says
          prop (SOME (SLc (PL tentativePlan)));
          Name PlatoonSergeant says
          prop (SOME (SLc (PSG initiateMovement)));
          Name PlatoonLeader says
          prop (SOME (SLc (PL report1)))]::ins) WARNO outs) ∧
   (M,Oi,Os) satList
   propCommandList
     [Name PlatoonLeader says prop (SOME (SLc (PL recon)));
      Name PlatoonLeader says
      prop (SOME (SLc (PL tentativePlan)));
      Name PlatoonSergeant says
      prop (SOME (SLc (PSG initiateMovement)));
      Name PlatoonLeader says prop (SOME (SLc (PL report1)))]
```

[PlatoonLeader_WARNO_exec_report1_justified_thm]

```
 ⊢ ∀NS Out M Oi Os.
     TR (M,Oi,Os)
       (exec
          [SOME (SLc (PL recon)); SOME (SLc (PL tentativePlan));
           SOME (SLc (PSG initiateMovement));
           SOME (SLc (PL report1))])
       (CFG inputOK secContext secContextNull
          ([Name PlatoonLeader says
            prop (SOME (SLc (PL recon)));
            Name PlatoonLeader says
            prop (SOME (SLc (PL tentativePlan)));
            Name PlatoonSergeant says
            prop (SOME (SLc (PSG initiateMovement)));
            Name PlatoonLeader says
            prop (SOME (SLc (PL report1)))]::ins) WARNO outs)
       (CFG inputOK secContext secContextNull ins
          (NS WARNO
             (exec
                [SOME (SLc (PL recon));
                 SOME (SLc (PL tentativePlan));
                 SOME (SLc (PSG initiateMovement));
                 SOME (SLc (PL report1))]))
          (Out WARNO
             (exec
                [SOME (SLc (PL recon));
                SOME (SLc (PL tentativePlan));
                SOME (SLc (PSG initiateMovement));
                SOME (SLc (PL report1))]):: outs)) ⟺
    authenticationTest inputOK
      [Name PlatoonLeader says prop (SOME (SLc (PL recon)));
       Name PlatoonLeader says
       prop (SOME (SLc (PL tentativePlan)));
       Name PlatoonSergeant says
       prop (SOME (SLc (PSG initiateMovement)));
       Name PlatoonLeader says
       prop (SOME (SLc (PL report1)))] ∧
    CFGInterpret (M,Oi,Os)
      (CFG inputOK secContext secContextNull
         ([Name PlatoonLeader says
           prop (SOME (SLc (PL recon)));
           Name PlatoonLeader says
           prop (SOME (SLc (PL tentativePlan)));
           Name PlatoonSergeant says
           prop (SOME (SLc (PSG initiateMovement)));
           Name PlatoonLeader says
           prop (SOME (SLc (PL report1)))]:: ins) WARNO outs) ∧
    (M,Oi,Os) satList
    [prop (SOME (SLc (PL recon)));
     prop (SOME (SLc (PL tentativePlan)));
     prop (SOME (SLc (PSG initiateMovement)));
     prop (SOME (SLc (PL report1)))]
```

[PlatoonLeader_WARNO_exec_report1_lemma]

```
 ⊢ ∀ M  Oi  Os.
    CFGInterpret (M,Oi,Os)
      (CFG inputOK secContext secContextNull
         ([Name PlatoonLeader says
           prop (SOME (SLc (PL recon)));
           Name PlatoonLeader says
           prop (SOME (SLc (PL tentativePlan)));
           Name PlatoonSergeant says
           prop (SOME (SLc (PSG initiateMovement)));
           Name PlatoonLeader says
           prop (SOME (SLc (PL report1)))]:: ins) WARNO outs) ⇒
    (M,Oi,Os) satList
    propCommandList
      [Name PlatoonLeader says prop (SOME (SLc (PL recon)));
       Name PlatoonLeader says
       prop (SOME (SLc (PL tentativePlan)));
       Name PlatoonSergeant says
       prop (SOME (SLc (PSG initiateMovement)));
       Name PlatoonLeader says prop (SOME (SLc (PL report1)))]
```

[PlatoonSergeant_trap_plCommand_justified_lemma]

```
⊢ ∀NS Out M Oi Os.
    TR (M,Oi,Os)
      (trap
         (inputList
            [Name PlatoonSergeant says
             prop (SOME (SLc (PL plCommand)))]))
      (CFG inputOK secContext secContextNull
         ([Name PlatoonSergeant says
           prop (SOME (SLc (PL plCommand)))]::ins) s outs)
      (CFG inputOK secContext secContextNull ins
         (NS s
            (trap
               (inputList
                  [Name PlatoonSergeant says
                   prop (SOME (SLc (PL plCommand)))])))
         (Out s
            (trap
               (inputList
                  [Name PlatoonSergeant says
                   prop (SOME (SLc (PL plCommand)))]))::
             outs)) ⟺
    authenticationTest inputOK
      [Name PlatoonSergeant says
       prop (SOME (SLc (PL plCommand)))] ∧
    CFGInterpret (M,Oi,Os)
      (CFG inputOK secContext secContextNull
         ([Name PlatoonSergeant says
           prop (SOME (SLc (PL plCommand)))]::ins) s outs) ∧
    (M,Oi,Os) sat prop NONE
```

[PlatoonSergeant_trap_plCommand_justified_thm]

```
⊢ ∀NS Out M Oi Os.
    TR (M,Oi,Os) (trap [SOME (SLc (PL plCommand))])
      (CFG inputOK secContext secContextNull
         ([Name PlatoonSergeant says
           prop (SOME (SLc (PL plCommand)))]::ins) s outs)
      (CFG inputOK secContext secContextNull ins
         (NS s (trap [SOME (SLc (PL plCommand))]))
         (Out s (trap [SOME (SLc (PL plCommand))])::outs)) ⟺
    authenticationTest inputOK
      [Name PlatoonSergeant says
       prop (SOME (SLc (PL plCommand)))] ∧
    CFGInterpret (M,Oi,Os)
      (CFG inputOK secContext secContextNull
         ([Name PlatoonSergeant says
           prop (SOME (SLc (PL plCommand)))]::ins) s outs) ∧
    (M,Oi,Os) sat prop NONE
```

[PlatoonSergeant_trap_plCommand_lemma]

```
⊢ ∀M Oi Os.
    CFGInterpret (M,Oi,Os)
      (CFG inputOK secContext secContextNull
         ([Name PlatoonSergeant says
           prop (SOME (SLc (PL plCommand)))]::ins) s outs) ⇒
    (M,Oi,Os) sat prop NONE
```

18 PlanPBType Theory

Built: 10 June 2018

Parent Theories: indexedLists, patternMatches

18.1 Datatypes

```
plCommand = receiveMission | warno | tentativePlan | recon
          | report1 | completePlan | opoid | supervise | report2
          | complete | plIncomplete | invalidPlCommand

psgCommand = initiateMovement | psgIncomplete
           | invalidPsgCommand

slCommand = PL plCommand | PSG psgCommand

slOutput = PlanPB | ReceiveMission | Warno | TentativePlan
         | InitiateMovement | Recon | Report1 | CompletePlan
         | Opoid | Supervise | Report2 | Complete
         | unAuthenticated | unAuthorized

slState = PLAN_PB | RECEIVE_MISSION | WARNO | TENTATIVE_PLAN
        | INITIATE_MOVEMENT | RECON | REPORT1 | COMPLETE_PLAN
        | OPOID | SUPERVISE | REPORT2 | COMPLETE

stateRole = PlatoonLeader | PlatoonSergeant
```

18.2 Theorems

[plCommand_distinct_clauses]

```
⊢ receiveMission ≠ warno ∧ receiveMission ≠ tentativePlan ∧
  receiveMission ≠ recon ∧ receiveMission ≠ report1 ∧
  receiveMission ≠ completePlan ∧ receiveMission ≠ opoid ∧
  receiveMission ≠ supervise ∧ receiveMission ≠ report2 ∧
  receiveMission ≠ complete ∧ receiveMission ≠ plIncomplete ∧
  receiveMission ≠ invalidPlCommand ∧ warno ≠ tentativePlan ∧
  warno ≠ recon ∧ warno ≠ report1 ∧ warno ≠ completePlan ∧
  warno ≠ opoid ∧ warno ≠ supervise ∧ warno ≠ report2 ∧
  warno ≠ complete ∧ warno ≠ plIncomplete ∧
  warno ≠ invalidPlCommand ∧ tentativePlan ≠ recon ∧
  tentativePlan ≠ report1 ∧ tentativePlan ≠ completePlan ∧
tentativePlan ≠ opoid ∧ tentativePlan ≠ supervise ∧
tentativePlan ≠ report2 ∧ tentativePlan ≠ complete ∧
tentativePlan ≠ plIncomplete ∧
tentativePlan ≠ invalidPlCommand ∧ recon ≠ report1 ∧
recon ≠ completePlan ∧ recon ≠ opoid ∧ recon ≠ supervise ∧
recon ≠ report2 ∧ recon ≠ complete ∧ recon ≠ plIncomplete ∧
recon ≠ invalidPlCommand ∧ report1 ≠ completePlan ∧
report1 ≠ opoid ∧ report1 ≠ supervise ∧ report1 ≠ report2 ∧
report1 ≠ complete ∧ report1 ≠ plIncomplete ∧
report1 ≠ invalidPlCommand ∧ completePlan ≠ opoid ∧
completePlan ≠ supervise ∧ completePlan ≠ report2 ∧
completePlan ≠ complete ∧ completePlan ≠ plIncomplete ∧
completePlan ≠ invalidPlCommand ∧ opoid ≠ supervise ∧
opoid ≠ report2 ∧ opoid ≠ complete ∧ opoid ≠ plIncomplete ∧
opoid ≠ invalidPlCommand ∧ supervise ≠ report2 ∧
supervise ≠ complete ∧ supervise ≠ plIncomplete ∧
supervise ≠ invalidPlCommand ∧ report2 ≠ complete ∧
report2 ≠ plIncomplete ∧ report2 ≠ invalidPlCommand ∧
complete ≠ plIncomplete ∧ complete ≠ invalidPlCommand ∧
plIncomplete ≠ invalidPlCommand
```

[psgCommand_distinct_clauses]

```
⊢ initiateMovement ≠ psgIncomplete ∧
  initiateMovement ≠ invalidPsgCommand ∧
  psgIncomplete ≠ invalidPsgCommand
```

[slCommand_distinct_clauses]

```
⊢ ∀ a′ a. PL a ≠ PSG a′
```

[slCommand_one_one]

```
⊢ (∀ a a′. (PL a = PL a′) ⇔ (a = a′)) ∧
  ∀ a a′. (PSG a = PSG a′) ⇔ (a = a′)
```

[slOutput_distinct_clauses]

```
⊢ PlanPB ≠ ReceiveMission ∧ PlanPB ≠ Warno ∧
  PlanPB ≠ TentativePlan ∧ PlanPB ≠ InitiateMovement ∧
  PlanPB ≠ Recon ∧ PlanPB ≠ Report1 ∧ PlanPB ≠ CompletePlan ∧
  PlanPB ≠ Opoid ∧ PlanPB ≠ Supervise ∧ PlanPB ≠ Report2 ∧
  PlanPB ≠ Complete ∧ PlanPB ≠ unAuthenticated ∧
  PlanPB ≠ unAuthorized ∧ ReceiveMission ≠ Warno ∧
  ReceiveMission ≠ TentativePlan ∧
  ReceiveMission ≠ InitiateMovement ∧ ReceiveMission ≠ Recon ∧
  ReceiveMission ≠ Report1 ∧ ReceiveMission ≠ CompletePlan ∧
  ReceiveMission ≠ Opoid ∧ ReceiveMission ≠ Supervise ∧
  ReceiveMission ≠ Report2 ∧ ReceiveMission ≠ Complete ∧
  ReceiveMission ≠ unAuthenticated ∧
  ReceiveMission ≠ unAuthorized ∧ Warno ≠ TentativePlan ∧
  Warno ≠ InitiateMovement ∧ Warno ≠ Recon ∧ Warno ≠ Report1 ∧
```

```
   Warno ≠ CompletePlan ∧ Warno ≠ Opoid ∧ Warno ≠ Supervise ∧
   Warno ≠ Report2 ∧ Warno ≠ Complete ∧
   Warno ≠ unAuthenticated ∧ Warno ≠ unAuthorized ∧
   TentativePlan ≠ InitiateMovement ∧ TentativePlan ≠ Recon ∧
   TentativePlan ≠ Report1 ∧ TentativePlan ≠ CompletePlan ∧
   TentativePlan ≠ Opoid ∧ TentativePlan ≠ Supervise ∧
   TentativePlan ≠ Report2 ∧ TentativePlan ≠ Complete ∧
   TentativePlan ≠ unAuthenticated ∧
   TentativePlan ≠ unAuthorized ∧ InitiateMovement ≠ Recon ∧
   InitiateMovement ≠ Report1 ∧
   InitiateMovement ≠ CompletePlan ∧ InitiateMovement ≠ Opoid ∧
   InitiateMovement ≠ Supervise ∧ InitiateMovement ≠ Report2 ∧
   InitiateMovement ≠ Complete ∧
   InitiateMovement ≠ unAuthenticated ∧
   InitiateMovement ≠ unAuthorized ∧ Recon ≠ Report1 ∧
   Recon ≠ CompletePlan ∧ Recon ≠ Opoid ∧ Recon ≠ Supervise ∧
   Recon ≠ Report2 ∧ Recon ≠ Complete ∧
   Recon ≠ unAuthenticated ∧ Recon ≠ unAuthorized ∧
   Report1 ≠ CompletePlan ∧ Report1 ≠ Opoid ∧
   Report1 ≠ Supervise ∧ Report1 ≠ Report2 ∧
   Report1 ≠ Complete ∧ Report1 ≠ unAuthenticated ∧
   Report1 ≠ unAuthorized ∧ CompletePlan ≠ Opoid ∧
   CompletePlan ≠ Supervise ∧ CompletePlan ≠ Report2 ∧
   CompletePlan ≠ Complete ∧ CompletePlan ≠ unAuthenticated ∧
   CompletePlan ≠ unAuthorized ∧ Opoid ≠ Supervise ∧
   Opoid ≠ Report2 ∧ Opoid ≠ Complete ∧
   Opoid ≠ unAuthenticated ∧ Opoid ≠ unAuthorized ∧
   Supervise ≠ Report2 ∧ Supervise ≠ Complete ∧
   Supervise ≠ unAuthenticated ∧ Supervise ≠ unAuthorized ∧
   Report2 ≠ Complete ∧ Report2 ≠ unAuthenticated ∧
   Report2 ≠ unAuthorized ∧ Complete ≠ unAuthenticated ∧
   Complete ≠ unAuthorized ∧ unAuthenticated ≠ unAuthorized
```

[slRole_distinct_clauses]

```
 ⊢ PlatoonLeader ≠ PlatoonSergeant
```

[slState_distinct_clauses]

```
 ⊢ PLAN_PB ≠ RECEIVE_MISSION ∧ PLAN_PB ≠ WARNO ∧
   PLAN_PB ≠ TENTATIVE_PLAN ∧ PLAN_PB ≠ INITIATE_MOVEMENT ∧
   PLAN_PB ≠ RECON ∧ PLAN_PB ≠ REPORT1 ∧
   PLAN_PB ≠ COMPLETE_PLAN ∧ PLAN_PB ≠ OPOID ∧
   PLAN_PB ≠ SUPERVISE ∧ PLAN_PB ≠ REPORT2 ∧
   PLAN_PB ≠ COMPLETE ∧ RECEIVE_MISSION ≠ WARNO ∧
   RECEIVE_MISSION ≠ TENTATIVE_PLAN ∧
   RECEIVE_MISSION ≠ INITIATE_MOVEMENT ∧
   RECEIVE_MISSION ≠ RECON ∧ RECEIVE_MISSION ≠ REPORT1 ∧
   RECEIVE_MISSION ≠ COMPLETE_PLAN ∧ RECEIVE_MISSION ≠ OPOID ∧
   RECEIVE_MISSION ≠ SUPERVISE ∧ RECEIVE_MISSION ≠ REPORT2 ∧
   RECEIVE_MISSION ≠ COMPLETE ∧ WARNO ≠ TENTATIVE_PLAN ∧
```

```
WARNO ≠ INITIATE_MOVEMENT ∧ WARNO ≠ RECON ∧ WARNO ≠ REPORT1 ∧
WARNO ≠ COMPLETE_PLAN ∧ WARNO ≠ OPOID ∧ WARNO ≠ SUPERVISE ∧
WARNO ≠ REPORT2 ∧ WARNO ≠ COMPLETE ∧
TENTATIVE_PLAN ≠ INITIATE_MOVEMENT ∧ TENTATIVE_PLAN ≠ RECON ∧
TENTATIVE_PLAN ≠ REPORT1 ∧ TENTATIVE_PLAN ≠ COMPLETE_PLAN ∧
TENTATIVE_PLAN ≠ OPOID ∧ TENTATIVE_PLAN ≠ SUPERVISE ∧
TENTATIVE_PLAN ≠ REPORT2 ∧ TENTATIVE_PLAN ≠ COMPLETE ∧
INITIATE_MOVEMENT ≠ RECON ∧ INITIATE_MOVEMENT ≠ REPORT1 ∧
INITIATE_MOVEMENT ≠ COMPLETE_PLAN ∧
INITIATE_MOVEMENT ≠ OPOID ∧ INITIATE_MOVEMENT ≠ SUPERVISE ∧
INITIATE_MOVEMENT ≠ REPORT2 ∧ INITIATE_MOVEMENT ≠ COMPLETE ∧
RECON ≠ REPORT1 ∧ RECON ≠ COMPLETE_PLAN ∧ RECON ≠ OPOID ∧
RECON ≠ SUPERVISE ∧ RECON ≠ REPORT2 ∧ RECON ≠ COMPLETE ∧
REPORT1 ≠ COMPLETE_PLAN ∧ REPORT1 ≠ OPOID ∧
REPORT1 ≠ SUPERVISE ∧ REPORT1 ≠ REPORT2 ∧
REPORT1 ≠ COMPLETE ∧ COMPLETE_PLAN ≠ OPOID ∧
COMPLETE_PLAN ≠ SUPERVISE ∧ COMPLETE_PLAN ≠ REPORT2 ∧
COMPLETE_PLAN ≠ COMPLETE ∧ OPOID ≠ SUPERVISE ∧
OPOID ≠ REPORT2 ∧ OPOID ≠ COMPLETE ∧ SUPERVISE ≠ REPORT2 ∧
SUPERVISE ≠ COMPLETE ∧ REPORT2 ≠ COMPLETE
```

19 PlanPBDef Theory

Built: 10 June 2018

Parent Theories: PlanPBType, aclfoundation, OMNIType

19.1 Definitions

[PL_notWARNO_Auth_def]

```
⊢ ∀ cmd.
    PL_notWARNO_Auth cmd =
    if cmd = report1 then prop NONE
    else
      Name PlatoonLeader says prop (SOME (SLc (PL cmd))) impf
      Name PlatoonLeader controls prop (SOME (SLc (PL cmd)))
```

[PL_WARNO_Auth_def]

```
⊢ PL_WARNO_Auth =
  prop (SOME (SLc (PL recon))) impf
  prop (SOME (SLc (PL tentativePlan))) impf
  prop (SOME (SLc (PSG initiateMovement))) impf
  Name PlatoonLeader controls prop (SOME (SLc (PL report1)))
```

[secContext_def]

```
⊢ ∀ s x.
    secContext s x =
    if s = WARNO then
```

```
      if
        (getRecon x = [SOME (SLc (PL recon))]) ∧
        (getTenativePlan x = [SOME (SLc (PL tentativePlan))]) ∧
        (getReport x = [SOME (SLc (PL report1))]) ∧
        (getInitMove x = [SOME (SLc (PSG initiateMovement))])
      then
        [PL_WARNO_Auth;
         Name PlatoonLeader controls
         prop (SOME (SLc (PL recon)));
         Name PlatoonLeader controls
         prop (SOME (SLc (PL tentativePlan)));
         Name PlatoonSergeant controls
         prop (SOME (SLc (PSG initiateMovement)))]
      else [prop NONE]
    else if getPlCom x = invalidPlCommand then [prop NONE]
    else [PL_notWARNO_Auth (getPlCom x)]
```

[secContextNull_def]

```
 ⊢ ∀x. secContextNull x = [TT]
```

19.2 Theorems

[getInitMove_def]

```
 ⊢ (getInitMove [] = [NONE]) ∧
   (∀xs.
      getInitMove
        (Name PlatoonSergeant says
         prop (SOME (SLc (PSG initiateMovement)))::xs) =
      [SOME (SLc (PSG initiateMovement))]) ∧
   (∀xs. getInitMove (TT::xs) = getInitMove xs) ∧
   (∀xs. getInitMove (FF::xs) = getInitMove xs) ∧
   (∀xs v2. getInitMove (prop v2::xs) = getInitMove xs) ∧
   (∀xs v3. getInitMove (notf v3::xs) = getInitMove xs) ∧
   (∀xs v5 v4. getInitMove (v4 andf v5::xs) = getInitMove xs) ∧
   (∀xs v7 v6. getInitMove (v6 orf v7::xs) = getInitMove xs) ∧
   (∀xs v9 v8. getInitMove (v8 impf v9::xs) = getInitMove xs) ∧
   (∀xs v11 v10.
      getInitMove (v10 eqf v11::xs) = getInitMove xs) ∧
   (∀xs v12. getInitMove (v12 says TT::xs) = getInitMove xs) ∧
   (∀xs v12. getInitMove (v12 says FF::xs) = getInitMove xs) ∧
   (∀xs v134.
      getInitMove (Name v134 says prop NONE::xs) =
      getInitMove xs) ∧
   (∀xs v144.
      getInitMove
        (Name PlatoonLeader says prop (SOME v144)::xs) =
      getInitMove xs) ∧
   (∀xs v146.
    getInitMove
      (Name PlatoonSergeant says prop (SOME (ESCc v146))::
           xs) =
    getInitMove xs) ∧
 (∀ xs v150.
    getInitMove
      (Name PlatoonSergeant says prop (SOME (SLc (PL v150)))::
           xs) =
    getInitMove xs) ∧
 (∀ xs.
    getInitMove
      (Name PlatoonSergeant says
       prop (SOME (SLc (PSG psgIncomplete)))::xs) =
    getInitMove xs) ∧
 (∀ xs.
    getInitMove
      (Name PlatoonSergeant says
       prop (SOME (SLc (PSG invalidPsgCommand)))::xs) =
    getInitMove xs) ∧
 (∀ xs v68 v136 v135.
    getInitMove (v135 meet v136 says prop v68::xs) =
    getInitMove xs) ∧
 (∀ xs v68 v138 v137.
    getInitMove (v137 quoting v138 says prop v68::xs) =
    getInitMove xs) ∧
 (∀ xs v69 v12.
    getInitMove (v12 says notf v69::xs) = getInitMove xs) ∧
 (∀ xs v71 v70 v12.
    getInitMove (v12 says (v70 andf v71)::xs) =
    getInitMove xs) ∧
 (∀ xs v73 v72 v12.
    getInitMove (v12 says (v72 orf v73)::xs) =
    getInitMove xs) ∧
 (∀ xs v75 v74 v12.
    getInitMove (v12 says (v74 impf v75)::xs) =
    getInitMove xs) ∧
 (∀ xs v77 v76 v12.
    getInitMove (v12 says (v76 eqf v77)::xs) =
    getInitMove xs) ∧
 (∀ xs v79 v78 v12.
    getInitMove (v12 says v78 says v79::xs) =
    getInitMove xs) ∧
 (∀ xs v81 v80 v12.
    getInitMove (v12 says v80 speaks_for v81::xs) =
    getInitMove xs) ∧
 (∀ xs v83 v82 v12.
    getInitMove (v12 says v82 controls v83::xs) =
    getInitMove xs) ∧
 (∀ xs v86 v85 v84 v12.
      getInitMove (v12 says reps v84 v85 v86::xs) =
      getInitMove xs) ∧
   (∀xs v88 v87 v12.
      getInitMove (v12 says v87 domi v88::xs) =
      getInitMove xs) ∧
   (∀xs v90 v89 v12.
      getInitMove (v12 says v89 eqi v90::xs) = getInitMove xs) ∧
   (∀xs v92 v91 v12.
      getInitMove (v12 says v91 doms v92::xs) =
      getInitMove xs) ∧
   (∀xs v94 v93 v12.
      getInitMove (v12 says v93 eqs v94::xs) = getInitMove xs) ∧
   (∀xs v96 v95 v12.
      getInitMove (v12 says v95 eqn v96::xs) = getInitMove xs) ∧
   (∀xs v98 v97 v12.
      getInitMove (v12 says v97 lte v98::xs) = getInitMove xs) ∧
   (∀xs v99 v12 v100.
      getInitMove (v12 says v99 lt v100::xs) = getInitMove xs) ∧
   (∀xs v15 v14.
      getInitMove (v14 speaks_for v15::xs) = getInitMove xs) ∧
   (∀xs v17 v16.
      getInitMove (v16 controls v17::xs) = getInitMove xs) ∧
   (∀xs v20 v19 v18.
      getInitMove (reps v18 v19 v20::xs) = getInitMove xs) ∧
   (∀xs v22 v21.
      getInitMove (v21 domi v22::xs) = getInitMove xs) ∧
   (∀xs v24 v23.
      getInitMove (v23 eqi v24::xs) = getInitMove xs) ∧
   (∀xs v26 v25.
      getInitMove (v25 doms v26::xs) = getInitMove xs) ∧
   (∀xs v28 v27.
      getInitMove (v27 eqs v28::xs) = getInitMove xs) ∧
   (∀xs v30 v29.
      getInitMove (v29 eqn v30::xs) = getInitMove xs) ∧
   (∀xs v32 v31.
      getInitMove (v31 lte v32::xs) = getInitMove xs) ∧
   ∀xs v34 v33. getInitMove (v33 lt v34::xs) = getInitMove xs
```

[getInitMove_ind]

```
 ⊢ ∀P.
     P [] ∧
     (∀xs.
        P
          (Name PlatoonSergeant says
           prop (SOME (SLc (PSG initiateMovement)))::xs)) ∧
     (∀xs. P xs ⇒ P (TT::xs)) ∧ (∀xs. P xs ⇒ P (FF::xs)) ∧
     (∀v2 xs. P xs ⇒ P (prop v2::xs)) ∧
     (∀v3 xs. P xs ⇒ P (notf v3::xs)) ∧
     (∀v4 v5 xs. P xs ⇒ P (v4 andf v5::xs)) ∧
```

```
(∀v6 v7 xs. P xs ⇒ P (v6 orf v7::xs)) ∧
(∀v8 v9 xs. P xs ⇒ P (v8 impf v9::xs)) ∧
(∀v10 v11 xs. P xs ⇒ P (v10 eqf v11::xs)) ∧
(∀v12 xs. P xs ⇒ P (v12 says TT::xs)) ∧
(∀v12 xs. P xs ⇒ P (v12 says FF::xs)) ∧
(∀v134 xs. P xs ⇒ P (Name v134 says prop NONE::xs)) ∧
(∀v144 xs.
   P xs ⇒
   P (Name PlatoonLeader says prop (SOME v144)::xs)) ∧
(∀v146 xs.
   P xs ⇒
   P
     (Name PlatoonSergeant says prop (SOME (ESCc v146))::
          xs)) ∧
(∀v150 xs.
   P xs ⇒
   P
     (Name PlatoonSergeant says
      prop (SOME (SLc (PL v150)))::xs)) ∧
(∀xs.
   P xs ⇒
   P
     (Name PlatoonSergeant says
      prop (SOME (SLc (PSG psgIncomplete)))::xs)) ∧
(∀xs.
   P xs ⇒
   P
     (Name PlatoonSergeant says
      prop (SOME (SLc (PSG invalidPsgCommand)))::xs)) ∧
(∀v135 v136 v68 xs.
   P xs ⇒ P (v135 meet v136 says prop v68::xs)) ∧
(∀v137 v138 v68 xs.
   P xs ⇒ P (v137 quoting v138 says prop v68::xs)) ∧
(∀v12 v69 xs. P xs ⇒ P (v12 says notf v69::xs)) ∧
(∀v12 v70 v71 xs. P xs ⇒ P (v12 says (v70 andf v71)::xs)) ∧
(∀v12 v72 v73 xs. P xs ⇒ P (v12 says (v72 orf v73)::xs)) ∧
(∀v12 v74 v75 xs. P xs ⇒ P (v12 says (v74 impf v75)::xs)) ∧
(∀v12 v76 v77 xs. P xs ⇒ P (v12 says (v76 eqf v77)::xs)) ∧
(∀v12 v78 v79 xs. P xs ⇒ P (v12 says v78 says v79::xs)) ∧
(∀v12 v80 v81 xs.
   P xs ⇒ P (v12 says v80 speaks_for v81::xs)) ∧
(∀v12 v82 v83 xs.
   P xs ⇒ P (v12 says v82 controls v83::xs)) ∧
(∀v12 v84 v85 v86 xs.
   P xs ⇒ P (v12 says reps v84 v85 v86::xs)) ∧
(∀v12 v87 v88 xs. P xs ⇒ P (v12 says v87 domi v88::xs)) ∧
(∀v12 v89 v90 xs. P xs ⇒ P (v12 says v89 eqi v90::xs)) ∧
(∀v12 v91 v92 xs. P xs ⇒ P (v12 says v91 doms v92::xs)) ∧
(∀v12 v93 v94 xs. P xs ⇒ P (v12 says v93 eqs v94::xs)) ∧
```

```
  (∀v12 v95 v96 xs. P xs ⇒ P (v12 says v95 eqn v96::xs)) ∧
  (∀v12 v97 v98 xs. P xs ⇒ P (v12 says v97 lte v98::xs)) ∧
  (∀v12 v99 v100 xs. P xs ⇒ P (v12 says v99 lt v100::xs)) ∧
  (∀v14 v15 xs. P xs ⇒ P (v14 speaks_for v15::xs)) ∧
  (∀v16 v17 xs. P xs ⇒ P (v16 controls v17::xs)) ∧
  (∀v18 v19 v20 xs. P xs ⇒ P (reps v18 v19 v20::xs)) ∧
  (∀v21 v22 xs. P xs ⇒ P (v21 domi v22::xs)) ∧
  (∀v23 v24 xs. P xs ⇒ P (v23 eqi v24::xs)) ∧
  (∀v25 v26 xs. P xs ⇒ P (v25 doms v26::xs)) ∧
  (∀v27 v28 xs. P xs ⇒ P (v27 eqs v28::xs)) ∧
  (∀v29 v30 xs. P xs ⇒ P (v29 eqn v30::xs)) ∧
  (∀v31 v32 xs. P xs ⇒ P (v31 lte v32::xs)) ∧
  (∀v33 v34 xs. P xs ⇒ P (v33 lt v34::xs)) ⇒
  ∀v. P v
```

[getPlCom_def]

```
⊢ (getPlCom [] = invalidPlCommand) ∧
  (∀xs cmd.
     getPlCom
       (Name PlatoonLeader says prop (SOME (SLc (PL cmd)))::
            xs) =
     cmd) ∧ (∀xs. getPlCom (TT::xs) = getPlCom xs) ∧
  (∀xs. getPlCom (FF::xs) = getPlCom xs) ∧
  (∀xs v2. getPlCom (prop v2::xs) = getPlCom xs) ∧
  (∀xs v3. getPlCom (notf v3::xs) = getPlCom xs) ∧
  (∀xs v5 v4. getPlCom (v4 andf v5::xs) = getPlCom xs) ∧
  (∀xs v7 v6. getPlCom (v6 orf v7::xs) = getPlCom xs) ∧
  (∀xs v9 v8. getPlCom (v8 impf v9::xs) = getPlCom xs) ∧
  (∀xs v11 v10. getPlCom (v10 eqf v11::xs) = getPlCom xs) ∧
  (∀xs v12. getPlCom (v12 says TT::xs) = getPlCom xs) ∧
  (∀xs v12. getPlCom (v12 says FF::xs) = getPlCom xs) ∧
  (∀xs v134.
     getPlCom (Name v134 says prop NONE::xs) = getPlCom xs) ∧
  (∀xs v146.
     getPlCom
       (Name PlatoonLeader says prop (SOME (ESCc v146))::xs) =
     getPlCom xs) ∧
  (∀xs v151.
     getPlCom
       (Name PlatoonLeader says prop (SOME (SLc (PSG v151)))::
            xs) =
     getPlCom xs) ∧
  (∀xs v144.
     getPlCom
       (Name PlatoonSergeant says prop (SOME v144)::xs) =
     getPlCom xs) ∧
  (∀xs v68 v136 v135.
     getPlCom (v135 meet v136 says prop v68::xs) =
     getPlCom xs) ∧
```

$(\forall xs\ v_{68}\ v138\ v137.$
  getPlCom ($v137$ quoting $v138$ says prop v_{68}::xs) =
  getPlCom xs) $\land$
$(\forall xs\ v_{69}\ v_{12}.$
  getPlCom (v_{12} says notf v_{69}::xs) = getPlCom xs) $\land$
$(\forall xs\ v_{71}\ v_{70}\ v_{12}.$
  getPlCom (v_{12} says (v_{70} andf v_{71})::xs) = getPlCom xs) $\land$
$(\forall xs\ v_{73}\ v_{72}\ v_{12}.$
  getPlCom (v_{12} says (v_{72} orf v_{73})::xs) = getPlCom xs) $\land$
$(\forall xs\ v_{75}\ v_{74}\ v_{12}.$
  getPlCom (v_{12} says (v_{74} impf v_{75})::xs) = getPlCom xs) $\land$
$(\forall xs\ v_{77}\ v_{76}\ v_{12}.$
  getPlCom (v_{12} says (v_{76} eqf v_{77})::xs) = getPlCom xs) $\land$
$(\forall xs\ v_{79}\ v_{78}\ v_{12}.$
  getPlCom (v_{12} says v_{78} says v_{79}::xs) = getPlCom xs) $\land$
$(\forall xs\ v_{81}\ v_{80}\ v_{12}.$
  getPlCom (v_{12} says v_{80} speaks_for v_{81}::xs) =
  getPlCom xs) $\land$
$(\forall xs\ v_{83}\ v_{82}\ v_{12}.$
  getPlCom (v_{12} says v_{82} controls v_{83}::xs) = getPlCom xs) $\land$
$(\forall xs\ v_{86}\ v_{85}\ v_{84}\ v_{12}.$
  getPlCom (v_{12} says reps v_{84} v_{85} v_{86}::xs) = getPlCom xs) $\land$
$(\forall xs\ v_{88}\ v_{87}\ v_{12}.$
  getPlCom (v_{12} says v_{87} domi v_{88}::xs) = getPlCom xs) $\land$
$(\forall xs\ v_{90}\ v_{89}\ v_{12}.$
  getPlCom (v_{12} says v_{89} eqi v_{90}::xs) = getPlCom xs) $\land$
$(\forall xs\ v_{92}\ v_{91}\ v_{12}.$
  getPlCom (v_{12} says v_{91} doms v_{92}::xs) = getPlCom xs) $\land$
$(\forall xs\ v_{94}\ v_{93}\ v_{12}.$
  getPlCom (v_{12} says v_{93} eqs v_{94}::xs) = getPlCom xs) $\land$
$(\forall xs\ v_{96}\ v_{95}\ v_{12}.$
  getPlCom (v_{12} says v_{95} eqn v_{96}::xs) = getPlCom xs) $\land$
$(\forall xs\ v_{98}\ v_{97}\ v_{12}.$
  getPlCom (v_{12} says v_{97} lte v_{98}::xs) = getPlCom xs) $\land$
$(\forall xs\ v_{99}\ v_{12}\ v100.$
  getPlCom (v_{12} says v_{99} lt $v100$::xs) = getPlCom xs) $\land$
$(\forall xs\ v_{15}\ v_{14}.$
  getPlCom (v_{14} speaks_for v_{15}::xs) = getPlCom xs) $\land$
$(\forall xs\ v_{17}\ v_{16}.$
  getPlCom (v_{16} controls v_{17}::xs) = getPlCom xs) $\land$
$(\forall xs\ v_{20}\ v_{19}\ v_{18}.$
  getPlCom (reps v_{18} v_{19} v_{20}::xs) = getPlCom xs) $\land$
$(\forall xs\ v_{22}\ v_{21}.$ getPlCom (v_{21} domi v_{22}::xs) = getPlCom xs) $\land$
$(\forall xs\ v_{24}\ v_{23}.$ getPlCom (v_{23} eqi v_{24}::xs) = getPlCom xs) $\land$
$(\forall xs\ v_{26}\ v_{25}.$ getPlCom (v_{25} doms v_{26}::xs) = getPlCom xs) $\land$
$(\forall xs\ v_{28}\ v_{27}.$ getPlCom (v_{27} eqs v_{28}::xs) = getPlCom xs) $\land$
$(\forall xs\ v_{30}\ v_{29}.$ getPlCom (v_{29} eqn v_{30}::xs) = getPlCom xs) $\land$
$(\forall xs\ v_{32}\ v_{31}.$ getPlCom (v_{31} lte v_{32}::xs) = getPlCom xs) $\land$
$\forall xs\ v_{34}\ v_{33}.$ getPlCom (v_{33} lt v_{34}::xs) = getPlCom xs

[getPlCom_ind]

```
⊢ ∀P.
      P [] ∧
      (∀cmd xs.
         P
           (Name PlatoonLeader says prop (SOME (SLc (PL cmd)))::
                xs)) ∧ (∀xs. P xs ⇒ P (TT::xs)) ∧
      (∀xs. P xs ⇒ P (FF::xs)) ∧
      (∀v2 xs. P xs ⇒ P (prop v2::xs)) ∧
      (∀v3 xs. P xs ⇒ P (notf v3::xs)) ∧
      (∀v4 v5 xs. P xs ⇒ P (v4 andf v5::xs)) ∧
      (∀v6 v7 xs. P xs ⇒ P (v6 orf v7::xs)) ∧
      (∀v8 v9 xs. P xs ⇒ P (v8 impf v9::xs)) ∧
      (∀v10 v11 xs. P xs ⇒ P (v10 eqf v11::xs)) ∧
      (∀v12 xs. P xs ⇒ P (v12 says TT::xs)) ∧
      (∀v12 xs. P xs ⇒ P (v12 says FF::xs)) ∧
      (∀v134 xs. P xs ⇒ P (Name v134 says prop NONE::xs)) ∧
      (∀v146 xs.
         P xs ⇒
         P
           (Name PlatoonLeader says prop (SOME (ESCc v146))::
                xs)) ∧
      (∀v151 xs.
         P xs ⇒
         P
           (Name PlatoonLeader says
             prop (SOME (SLc (PSG v151)))::xs)) ∧
      (∀v144 xs.
         P xs ⇒
         P (Name PlatoonSergeant says prop (SOME v144)::xs)) ∧
      (∀v135 v136 v68 xs.
         P xs ⇒ P (v135 meet v136 says prop v68::xs)) ∧
      (∀v137 v138 v68 xs.
         P xs ⇒ P (v137 quoting v138 says prop v68::xs)) ∧
      (∀v12 v69 xs. P xs ⇒ P (v12 says notf v69::xs)) ∧
      (∀v12 v70 v71 xs. P xs ⇒ P (v12 says (v70 andf v71)::xs)) ∧
      (∀v12 v72 v73 xs. P xs ⇒ P (v12 says (v72 orf v73)::xs)) ∧
      (∀v12 v74 v75 xs. P xs ⇒ P (v12 says (v74 impf v75)::xs)) ∧
      (∀v12 v76 v77 xs. P xs ⇒ P (v12 says (v76 eqf v77)::xs)) ∧
      (∀v12 v78 v79 xs. P xs ⇒ P (v12 says v78 says v79::xs)) ∧
      (∀v12 v80 v81 xs.
         P xs ⇒ P (v12 says v80 speaks_for v81::xs)) ∧
      (∀v12 v82 v83 xs.
         P xs ⇒ P (v12 says v82 controls v83::xs)) ∧
      (∀v12 v84 v85 v86 xs.
         P xs ⇒ P (v12 says reps v84 v85 v86::xs)) ∧
      (∀v12 v87 v88 xs. P xs ⇒ P (v12 says v87 domi v88::xs)) ∧
      (∀v12 v89 v90 xs. P xs ⇒ P (v12 says v89 eqi v90::xs)) ∧
      (∀v12 v91 v92 xs. P xs ⇒ P (v12 says v91 doms v92::xs)) ∧
```

```
(∀ v12 v93 v94 xs. P xs ⇒ P (v12 says v93 eqs v94::xs)) ∧
(∀ v12 v95 v96 xs. P xs ⇒ P (v12 says v95 eqn v96::xs)) ∧
(∀ v12 v97 v98 xs. P xs ⇒ P (v12 says v97 lte v98::xs)) ∧
(∀ v12 v99 v100 xs. P xs ⇒ P (v12 says v99 lt v100::xs)) ∧
(∀ v14 v15 xs. P xs ⇒ P (v14 speaks_for v15::xs)) ∧
(∀ v16 v17 xs. P xs ⇒ P (v16 controls v17::xs)) ∧
(∀ v18 v19 v20 xs. P xs ⇒ P (reps v18 v19 v20::xs)) ∧
(∀ v21 v22 xs. P xs ⇒ P (v21 domi v22::xs)) ∧
(∀ v23 v24 xs. P xs ⇒ P (v23 eqi v24::xs)) ∧
(∀ v25 v26 xs. P xs ⇒ P (v25 doms v26::xs)) ∧
(∀ v27 v28 xs. P xs ⇒ P (v27 eqs v28::xs)) ∧
(∀ v29 v30 xs. P xs ⇒ P (v29 eqn v30::xs)) ∧
(∀ v31 v32 xs. P xs ⇒ P (v31 lte v32::xs)) ∧
(∀ v33 v34 xs. P xs ⇒ P (v33 lt v34::xs)) ⇒
∀ v. P v
```

[getPsgCom_def]

```
⊢ (getPsgCom [] = invalidPsgCommand) ∧
  (∀ xs cmd.
     getPsgCom
       (Name PlatoonSergeant says prop (SOME (SLc (PSG cmd)))::
            xs) =
     cmd) ∧ (∀ xs. getPsgCom (TT::xs) = getPsgCom xs) ∧
  (∀ xs. getPsgCom (FF::xs) = getPsgCom xs) ∧
  (∀ xs v2. getPsgCom (prop v2::xs) = getPsgCom xs) ∧
  (∀ xs v3. getPsgCom (notf v3::xs) = getPsgCom xs) ∧
  (∀ xs v5 v4. getPsgCom (v4 andf v5::xs) = getPsgCom xs) ∧
  (∀ xs v7 v6. getPsgCom (v6 orf v7::xs) = getPsgCom xs) ∧
  (∀ xs v9 v8. getPsgCom (v8 impf v9::xs) = getPsgCom xs) ∧
  (∀ xs v11 v10. getPsgCom (v10 eqf v11::xs) = getPsgCom xs) ∧
  (∀ xs v12. getPsgCom (v12 says TT::xs) = getPsgCom xs) ∧
  (∀ xs v12. getPsgCom (v12 says FF::xs) = getPsgCom xs) ∧
  (∀ xs v134.
     getPsgCom (Name v134 says prop NONE::xs) = getPsgCom xs) ∧
  (∀ xs v144.
     getPsgCom (Name PlatoonLeader says prop (SOME v144)::xs) =
     getPsgCom xs) ∧
  (∀ xs v146.
     getPsgCom
       (Name PlatoonSergeant says prop (SOME (ESCc v146))::
            xs) =
     getPsgCom xs) ∧
  (∀ xs v150.
     getPsgCom
       (Name PlatoonSergeant says prop (SOME (SLc (PL v150)))::
            xs) =
     getPsgCom xs) ∧
  (∀ xs v68 v136 v135.
     getPsgCom (v135 meet v136 says prop v68::xs) =
```

getPsgCom xs) $\wedge$
($\forall xs$ v_{68} $v138$ $v137$.
getPsgCom ($v137$ quoting $v138$ says prop v_{68}::xs) =
getPsgCom xs) $\wedge$
($\forall xs$ v_{69} v_{12}.
getPsgCom (v_{12} says notf v_{69}::xs) = getPsgCom xs) $\wedge$
($\forall xs$ v_{71} v_{70} v_{12}.
getPsgCom (v_{12} says (v_{70} andf v_{71})::xs) = getPsgCom xs) $\wedge$
($\forall xs$ v_{73} v_{72} v_{12}.
getPsgCom (v_{12} says (v_{72} orf v_{73})::xs) = getPsgCom xs) $\wedge$
($\forall xs$ v_{75} v_{74} v_{12}.
getPsgCom (v_{12} says (v_{74} impf v_{75})::xs) = getPsgCom xs) $\wedge$
($\forall xs$ v_{77} v_{76} v_{12}.
getPsgCom (v_{12} says (v_{76} eqf v_{77})::xs) = getPsgCom xs) $\wedge$
($\forall xs$ v_{79} v_{78} v_{12}.
getPsgCom (v_{12} says v_{78} says v_{79}::xs) = getPsgCom xs) $\wedge$
($\forall xs$ v_{81} v_{80} v_{12}.
getPsgCom (v_{12} says v_{80} speaks_for v_{81}::xs) =
getPsgCom xs) $\wedge$
($\forall xs$ v_{83} v_{82} v_{12}.
getPsgCom (v_{12} says v_{82} controls v_{83}::xs) =
getPsgCom xs) $\wedge$
($\forall xs$ v_{86} v_{85} v_{84} v_{12}.
getPsgCom (v_{12} says reps v_{84} v_{85} v_{86}::xs) =
getPsgCom xs) $\wedge$
($\forall xs$ v_{88} v_{87} v_{12}.
getPsgCom (v_{12} says v_{87} domi v_{88}::xs) = getPsgCom xs) $\wedge$
($\forall xs$ v_{90} v_{89} v_{12}.
getPsgCom (v_{12} says v_{89} eqi v_{90}::xs) = getPsgCom xs) $\wedge$
($\forall xs$ v_{92} v_{91} v_{12}.
getPsgCom (v_{12} says v_{91} doms v_{92}::xs) = getPsgCom xs) $\wedge$
($\forall xs$ v_{94} v_{93} v_{12}.
getPsgCom (v_{12} says v_{93} eqs v_{94}::xs) = getPsgCom xs) $\wedge$
($\forall xs$ v_{96} v_{95} v_{12}.
getPsgCom (v_{12} says v_{95} eqn v_{96}::xs) = getPsgCom xs) $\wedge$
($\forall xs$ v_{98} v_{97} v_{12}.
getPsgCom (v_{12} says v_{97} lte v_{98}::xs) = getPsgCom xs) $\wedge$
($\forall xs$ v_{99} v_{12} $v100$.
getPsgCom (v_{12} says v_{99} lt $v100$::xs) = getPsgCom xs) $\wedge$
($\forall xs$ v_{15} v_{14}.
getPsgCom (v_{14} speaks_for v_{15}::xs) = getPsgCom xs) $\wedge$
($\forall xs$ v_{17} v_{16}.
getPsgCom (v_{16} controls v_{17}::xs) = getPsgCom xs) $\wedge$
($\forall xs$ v_{20} v_{19} v_{18}.
getPsgCom (reps v_{18} v_{19} v_{20}::xs) = getPsgCom xs) $\wedge$
($\forall xs$ v_{22} v_{21}. getPsgCom (v_{21} domi v_{22}::xs) = getPsgCom xs) $\wedge$
($\forall xs$ v_{24} v_{23}. getPsgCom (v_{23} eqi v_{24}::xs) = getPsgCom xs) $\wedge$
($\forall xs$ v_{26} v_{25}. getPsgCom (v_{25} doms v_{26}::xs) = getPsgCom xs) $\wedge$
($\forall xs$ v_{28} v_{27}. getPsgCom (v_{27} eqs v_{28}::xs) = getPsgCom xs) $\wedge$

```
   (∀ xs v30 v29. getPsgCom (v29 eqn v30::xs) = getPsgCom xs) ∧
   (∀ xs v32 v31. getPsgCom (v31 lte v32::xs) = getPsgCom xs) ∧
   ∀ xs v34 v33. getPsgCom (v33 lt v34::xs) = getPsgCom xs
```

[getPsgCom_ind]

```
 ⊢ ∀ P.
     P [] ∧
     (∀ cmd xs.
        P
          (Name PlatoonSergeant says
           prop (SOME (SLc (PSG cmd)))::xs)) ∧
     (∀ xs. P xs ⇒ P (TT::xs)) ∧ (∀ xs. P xs ⇒ P (FF::xs)) ∧
     (∀ v2 xs. P xs ⇒ P (prop v2::xs)) ∧
     (∀ v3 xs. P xs ⇒ P (notf v3::xs)) ∧
     (∀ v4 v5 xs. P xs ⇒ P (v4 andf v5::xs)) ∧
     (∀ v6 v7 xs. P xs ⇒ P (v6 orf v7::xs)) ∧
     (∀ v8 v9 xs. P xs ⇒ P (v8 impf v9::xs)) ∧
     (∀ v10 v11 xs. P xs ⇒ P (v10 eqf v11::xs)) ∧
     (∀ v12 xs. P xs ⇒ P (v12 says TT::xs)) ∧
     (∀ v12 xs. P xs ⇒ P (v12 says FF::xs)) ∧
     (∀ v134 xs. P xs ⇒ P (Name v134 says prop NONE::xs)) ∧
     (∀ v144 xs.
        P xs ⇒
        P (Name PlatoonLeader says prop (SOME v144)::xs)) ∧
     (∀ v146 xs.
        P xs ⇒
        P
          (Name PlatoonSergeant says prop (SOME (ESCc v146))::
               xs)) ∧
     (∀ v150 xs.
        P xs ⇒
        P
          (Name PlatoonSergeant says
           prop (SOME (SLc (PL v150)))::xs)) ∧
     (∀ v135 v136 v68 xs.
        P xs ⇒ P (v135 meet v136 says prop v68::xs)) ∧
     (∀ v137 v138 v68 xs.
        P xs ⇒ P (v137 quoting v138 says prop v68::xs)) ∧
     (∀ v12 v69 xs. P xs ⇒ P (v12 says notf v69::xs)) ∧
     (∀ v12 v70 v71 xs. P xs ⇒ P (v12 says (v70 andf v71)::xs)) ∧
     (∀ v12 v72 v73 xs. P xs ⇒ P (v12 says (v72 orf v73)::xs)) ∧
     (∀ v12 v74 v75 xs. P xs ⇒ P (v12 says (v74 impf v75)::xs)) ∧
     (∀ v12 v76 v77 xs. P xs ⇒ P (v12 says (v76 eqf v77)::xs)) ∧
     (∀ v12 v78 v79 xs. P xs ⇒ P (v12 says v78 says v79::xs)) ∧
     (∀ v12 v80 v81 xs.
        P xs ⇒ P (v12 says v80 speaks_for v81::xs)) ∧
     (∀ v12 v82 v83 xs.
        P xs ⇒ P (v12 says v82 controls v83::xs)) ∧
     (∀ v12 v84 v85 v86 xs.
      P xs ⇒ P (v12 says reps v84 v85 v86::xs)) ∧
   (∀ v12 v87 v88 xs. P xs ⇒ P (v12 says v87 domi v88::xs)) ∧
   (∀ v12 v89 v90 xs. P xs ⇒ P (v12 says v89 eqi v90::xs)) ∧
   (∀ v12 v91 v92 xs. P xs ⇒ P (v12 says v91 doms v92::xs)) ∧
   (∀ v12 v93 v94 xs. P xs ⇒ P (v12 says v93 eqs v94::xs)) ∧
   (∀ v12 v95 v96 xs. P xs ⇒ P (v12 says v95 eqn v96::xs)) ∧
   (∀ v12 v97 v98 xs. P xs ⇒ P (v12 says v97 lte v98::xs)) ∧
   (∀ v12 v99 v100 xs. P xs ⇒ P (v12 says v99 lt v100::xs)) ∧
   (∀ v14 v15 xs. P xs ⇒ P (v14 speaks_for v15::xs)) ∧
   (∀ v16 v17 xs. P xs ⇒ P (v16 controls v17::xs)) ∧
   (∀ v18 v19 v20 xs. P xs ⇒ P (reps v18 v19 v20::xs)) ∧
   (∀ v21 v22 xs. P xs ⇒ P (v21 domi v22::xs)) ∧
   (∀ v23 v24 xs. P xs ⇒ P (v23 eqi v24::xs)) ∧
   (∀ v25 v26 xs. P xs ⇒ P (v25 doms v26::xs)) ∧
   (∀ v27 v28 xs. P xs ⇒ P (v27 eqs v28::xs)) ∧
   (∀ v29 v30 xs. P xs ⇒ P (v29 eqn v30::xs)) ∧
   (∀ v31 v32 xs. P xs ⇒ P (v31 lte v32::xs)) ∧
   (∀ v33 v34 xs. P xs ⇒ P (v33 lt v34::xs)) ⇒
   ∀ v. P v
```

[getRecon_def]

```
 ⊢ (getRecon [] = [NONE]) ∧
   (∀ xs.
      getRecon
        (Name PlatoonLeader says prop (SOME (SLc (PL recon)))::
             xs) =
      [SOME (SLc (PL recon))]) ∧
   (∀ xs. getRecon (TT::xs) = getRecon xs) ∧
   (∀ xs. getRecon (FF::xs) = getRecon xs) ∧
   (∀ xs v2. getRecon (prop v2::xs) = getRecon xs) ∧
   (∀ xs v3. getRecon (notf v3::xs) = getRecon xs) ∧
   (∀ xs v5 v4. getRecon (v4 andf v5::xs) = getRecon xs) ∧
   (∀ xs v7 v6. getRecon (v6 orf v7::xs) = getRecon xs) ∧
   (∀ xs v9 v8. getRecon (v8 impf v9::xs) = getRecon xs) ∧
   (∀ xs v11 v10. getRecon (v10 eqf v11::xs) = getRecon xs) ∧
   (∀ xs v12. getRecon (v12 says TT::xs) = getRecon xs) ∧
   (∀ xs v12. getRecon (v12 says FF::xs) = getRecon xs) ∧
   (∀ xs v134.
      getRecon (Name v134 says prop NONE::xs) = getRecon xs) ∧
   (∀ xs v146.
      getRecon
        (Name PlatoonLeader says prop (SOME (ESCc v146))::xs) =
      getRecon xs) ∧
   (∀ xs.
      getRecon
        (Name PlatoonLeader says
         prop (SOME (SLc (PL receiveMission)))::xs) =
      getRecon xs) ∧
   (∀ xs.
    getRecon
      (Name PlatoonLeader says prop (SOME (SLc (PL warno)))::
           xs) =
    getRecon xs) ∧
 (∀ xs.
    getRecon
      (Name PlatoonLeader says
       prop (SOME (SLc (PL tentativePlan)))::xs) =
    getRecon xs) ∧
 (∀ xs.
    getRecon
      (Name PlatoonLeader says
       prop (SOME (SLc (PL report1)))::xs) =
    getRecon xs) ∧
 (∀ xs.
    getRecon
      (Name PlatoonLeader says
       prop (SOME (SLc (PL completePlan)))::xs) =
    getRecon xs) ∧
 (∀ xs.
    getRecon
      (Name PlatoonLeader says prop (SOME (SLc (PL opoid)))::
           xs) =
    getRecon xs) ∧
 (∀ xs.
    getRecon
      (Name PlatoonLeader says
       prop (SOME (SLc (PL supervise)))::xs) =
    getRecon xs) ∧
 (∀ xs.
    getRecon
      (Name PlatoonLeader says
       prop (SOME (SLc (PL report2)))::xs) =
    getRecon xs) ∧
 (∀ xs.
    getRecon
      (Name PlatoonLeader says
       prop (SOME (SLc (PL complete)))::xs) =
    getRecon xs) ∧
 (∀ xs.
    getRecon
      (Name PlatoonLeader says
       prop (SOME (SLc (PL plIncomplete)))::xs) =
    getRecon xs) ∧
 (∀ xs.
    getRecon
      (Name PlatoonLeader says
       prop (SOME (SLc (PL invalidPlCommand)))::xs) =
    getRecon xs) ∧
```

```
(∀ xs v151.
   getRecon
     (Name PlatoonLeader says prop (SOME (SLc (PSG v151)))::
          xs) =
   getRecon xs) ∧
(∀ xs v144.
   getRecon
     (Name PlatoonSergeant says prop (SOME v144)::xs) =
   getRecon xs) ∧
(∀ xs v68 v136 v135.
   getRecon (v135 meet v136 says prop v68::xs) =
   getRecon xs) ∧
(∀ xs v68 v138 v137.
   getRecon (v137 quoting v138 says prop v68::xs) =
   getRecon xs) ∧
(∀ xs v69 v12.
   getRecon (v12 says notf v69::xs) = getRecon xs) ∧
(∀ xs v71 v70 v12.
   getRecon (v12 says (v70 andf v71)::xs) = getRecon xs) ∧
(∀ xs v73 v72 v12.
   getRecon (v12 says (v72 orf v73)::xs) = getRecon xs) ∧
(∀ xs v75 v74 v12.
   getRecon (v12 says (v74 impf v75)::xs) = getRecon xs) ∧
(∀ xs v77 v76 v12.
   getRecon (v12 says (v76 eqf v77)::xs) = getRecon xs) ∧
(∀ xs v79 v78 v12.
   getRecon (v12 says v78 says v79::xs) = getRecon xs) ∧
(∀ xs v81 v80 v12.
   getRecon (v12 says v80 speaks_for v81::xs) =
   getRecon xs) ∧
(∀ xs v83 v82 v12.
   getRecon (v12 says v82 controls v83::xs) = getRecon xs) ∧
(∀ xs v86 v85 v84 v12.
   getRecon (v12 says reps v84 v85 v86::xs) = getRecon xs) ∧
(∀ xs v88 v87 v12.
   getRecon (v12 says v87 domi v88::xs) = getRecon xs) ∧
(∀ xs v90 v89 v12.
   getRecon (v12 says v89 eqi v90::xs) = getRecon xs) ∧
(∀ xs v92 v91 v12.
   getRecon (v12 says v91 doms v92::xs) = getRecon xs) ∧
(∀ xs v94 v93 v12.
   getRecon (v12 says v93 eqs v94::xs) = getRecon xs) ∧
(∀ xs v96 v95 v12.
   getRecon (v12 says v95 eqn v96::xs) = getRecon xs) ∧
(∀ xs v98 v97 v12.
   getRecon (v12 says v97 lte v98::xs) = getRecon xs) ∧
(∀ xs v99 v12 v100.
   getRecon (v12 says v99 lt v100::xs) = getRecon xs) ∧
(∀ xs v15 v14.
      getRecon (v_14 speaks_for v_15::xs) = getRecon xs) ∧
   (∀xs v_17 v_16.
      getRecon (v_16 controls v_17::xs) = getRecon xs) ∧
   (∀xs v_20 v_19 v_18.
      getRecon (reps v_18 v_19 v_20::xs) = getRecon xs) ∧
   (∀xs v_22 v_21. getRecon (v_21 domi v_22::xs) = getRecon xs) ∧
   (∀xs v_24 v_23. getRecon (v_23 eqi v_24::xs) = getRecon xs) ∧
   (∀xs v_26 v_25. getRecon (v_25 doms v_26::xs) = getRecon xs) ∧
   (∀xs v_28 v_27. getRecon (v_27 eqs v_28::xs) = getRecon xs) ∧
   (∀xs v_30 v_29. getRecon (v_29 eqn v_30::xs) = getRecon xs) ∧
   (∀xs v_32 v_31. getRecon (v_31 lte v_32::xs) = getRecon xs) ∧
   ∀xs v_34 v_33. getRecon (v_33 lt v_34::xs) = getRecon xs
```

[getRecon_ind]

```
 ⊢ ∀P.
     P [] ∧
     (∀xs.
        P
          (Name PlatoonLeader says
           prop (SOME (SLc (PL recon)))::xs)) ∧
     (∀xs. P xs ⇒ P (TT::xs)) ∧ (∀xs. P xs ⇒ P (FF::xs)) ∧
     (∀v_2 xs. P xs ⇒ P (prop v_2::xs)) ∧
     (∀v_3 xs. P xs ⇒ P (notf v_3::xs)) ∧
     (∀v_4 v_5 xs. P xs ⇒ P (v_4 andf v_5::xs)) ∧
     (∀v_6 v_7 xs. P xs ⇒ P (v_6 orf v_7::xs)) ∧
     (∀v_8 v_9 xs. P xs ⇒ P (v_8 impf v_9::xs)) ∧
     (∀v_10 v_11 xs. P xs ⇒ P (v_10 eqf v_11::xs)) ∧
     (∀v_12 xs. P xs ⇒ P (v_12 says TT::xs)) ∧
     (∀v_12 xs. P xs ⇒ P (v_12 says FF::xs)) ∧
     (∀v134 xs. P xs ⇒ P (Name v134 says prop NONE::xs)) ∧
     (∀v146 xs.
        P xs ⇒
        P
          (Name PlatoonLeader says prop (SOME (ESCc v146))::
               xs)) ∧
     (∀xs.
        P xs ⇒
        P
          (Name PlatoonLeader says
           prop (SOME (SLc (PL receiveMission)))::xs)) ∧
     (∀xs.
        P xs ⇒
        P
          (Name PlatoonLeader says
           prop (SOME (SLc (PL warno)))::xs)) ∧
     (∀xs.
        P xs ⇒
        P
          (Name PlatoonLeader says
       prop (SOME (SLc (PL tentativePlan)))::xs)) ∧
(∀ xs.
   P xs ⇒
   P
     (Name PlatoonLeader says
      prop (SOME (SLc (PL report1)))::xs)) ∧
(∀ xs.
   P xs ⇒
   P
     (Name PlatoonLeader says
      prop (SOME (SLc (PL completePlan)))::xs)) ∧
(∀ xs.
   P xs ⇒
   P
     (Name PlatoonLeader says
      prop (SOME (SLc (PL opoid)))::xs)) ∧
(∀ xs.
   P xs ⇒
   P
     (Name PlatoonLeader says
      prop (SOME (SLc (PL supervise)))::xs)) ∧
(∀ xs.
   P xs ⇒
   P
     (Name PlatoonLeader says
      prop (SOME (SLc (PL report2)))::xs)) ∧
(∀ xs.
   P xs ⇒
   P
     (Name PlatoonLeader says
      prop (SOME (SLc (PL complete)))::xs)) ∧
(∀ xs.
   P xs ⇒
   P
     (Name PlatoonLeader says
      prop (SOME (SLc (PL plIncomplete)))::xs)) ∧
(∀ xs.
   P xs ⇒
   P
     (Name PlatoonLeader says
      prop (SOME (SLc (PL invalidPlCommand)))::xs)) ∧
(∀ v151 xs.
   P xs ⇒
   P
     (Name PlatoonLeader says
      prop (SOME (SLc (PSG v151)))::xs)) ∧
(∀ v144 xs.
   P xs ⇒
   P (Name PlatoonSergeant says prop (SOME v144)::xs)) ∧
```

```
      (∀ v135 v136 v68 xs.
         P xs ⇒ P (v135 meet v136 says prop v68::xs)) ∧
      (∀ v137 v138 v68 xs.
         P xs ⇒ P (v137 quoting v138 says prop v68::xs)) ∧
      (∀ v12 v69 xs. P xs ⇒ P (v12 says notf v69::xs)) ∧
      (∀ v12 v70 v71 xs. P xs ⇒ P (v12 says (v70 andf v71)::xs)) ∧
      (∀ v12 v72 v73 xs. P xs ⇒ P (v12 says (v72 orf v73)::xs)) ∧
      (∀ v12 v74 v75 xs. P xs ⇒ P (v12 says (v74 impf v75)::xs)) ∧
      (∀ v12 v76 v77 xs. P xs ⇒ P (v12 says (v76 eqf v77)::xs)) ∧
      (∀ v12 v78 v79 xs. P xs ⇒ P (v12 says v78 says v79::xs)) ∧
      (∀ v12 v80 v81 xs.
         P xs ⇒ P (v12 says v80 speaks_for v81::xs)) ∧
      (∀ v12 v82 v83 xs.
         P xs ⇒ P (v12 says v82 controls v83::xs)) ∧
      (∀ v12 v84 v85 v86 xs.
         P xs ⇒ P (v12 says reps v84 v85 v86::xs)) ∧
      (∀ v12 v87 v88 xs. P xs ⇒ P (v12 says v87 domi v88::xs)) ∧
      (∀ v12 v89 v90 xs. P xs ⇒ P (v12 says v89 eqi v90::xs)) ∧
      (∀ v12 v91 v92 xs. P xs ⇒ P (v12 says v91 doms v92::xs)) ∧
      (∀ v12 v93 v94 xs. P xs ⇒ P (v12 says v93 eqs v94::xs)) ∧
      (∀ v12 v95 v96 xs. P xs ⇒ P (v12 says v95 eqn v96::xs)) ∧
      (∀ v12 v97 v98 xs. P xs ⇒ P (v12 says v97 lte v98::xs)) ∧
      (∀ v12 v99 v100 xs. P xs ⇒ P (v12 says v99 lt v100::xs)) ∧
      (∀ v14 v15 xs. P xs ⇒ P (v14 speaks_for v15::xs)) ∧
      (∀ v16 v17 xs. P xs ⇒ P (v16 controls v17::xs)) ∧
      (∀ v18 v19 v20 xs. P xs ⇒ P (reps v18 v19 v20::xs)) ∧
      (∀ v21 v22 xs. P xs ⇒ P (v21 domi v22::xs)) ∧
      (∀ v23 v24 xs. P xs ⇒ P (v23 eqi v24::xs)) ∧
      (∀ v25 v26 xs. P xs ⇒ P (v25 doms v26::xs)) ∧
      (∀ v27 v28 xs. P xs ⇒ P (v27 eqs v28::xs)) ∧
      (∀ v29 v30 xs. P xs ⇒ P (v29 eqn v30::xs)) ∧
      (∀ v31 v32 xs. P xs ⇒ P (v31 lte v32::xs)) ∧
      (∀ v33 v34 xs. P xs ⇒ P (v33 lt v34::xs)) ⇒
      ∀ v. P v
```

[getReport_def]

```
 ⊢ (getReport [] = [NONE]) ∧
   (∀ xs.
      getReport
        (Name PlatoonLeader says
         prop (SOME (SLc (PL report1)))::xs) =
      [SOME (SLc (PL report1))]) ∧
   (∀ xs. getReport (TT::xs) = getReport xs) ∧
   (∀ xs. getReport (FF::xs) = getReport xs) ∧
   (∀ xs v2. getReport (prop v2::xs) = getReport xs) ∧
   (∀ xs v3. getReport (notf v3::xs) = getReport xs) ∧
   (∀ xs v5 v4. getReport (v4 andf v5::xs) = getReport xs) ∧
   (∀ xs v7 v6. getReport (v6 orf v7::xs) = getReport xs) ∧
   (∀ xs v9 v8. getReport (v8 impf v9::xs) = getReport xs) ∧
```

```
(∀xs v11 v10. getReport (v10 eqf v11::xs) = getReport xs) ∧
(∀xs v12. getReport (v12 says TT::xs) = getReport xs) ∧
(∀xs v12. getReport (v12 says FF::xs) = getReport xs) ∧
(∀xs v134.
   getReport (Name v134 says prop NONE::xs) = getReport xs) ∧
(∀xs v146.
   getReport
     (Name PlatoonLeader says prop (SOME (ESCc v146))::xs) =
   getReport xs) ∧
(∀xs.
   getReport
     (Name PlatoonLeader says
      prop (SOME (SLc (PL receiveMission)))::xs) =
   getReport xs) ∧
(∀xs.
   getReport
     (Name PlatoonLeader says prop (SOME (SLc (PL warno)))::
          xs) =
   getReport xs) ∧
(∀xs.
   getReport
     (Name PlatoonLeader says
      prop (SOME (SLc (PL tentativePlan)))::xs) =
   getReport xs) ∧
(∀xs.
   getReport
     (Name PlatoonLeader says prop (SOME (SLc (PL recon)))::
          xs) =
   getReport xs) ∧
(∀xs.
   getReport
     (Name PlatoonLeader says
      prop (SOME (SLc (PL completePlan)))::xs) =
   getReport xs) ∧
(∀xs.
   getReport
     (Name PlatoonLeader says prop (SOME (SLc (PL opoid)))::
          xs) =
   getReport xs) ∧
(∀xs.
   getReport
     (Name PlatoonLeader says
      prop (SOME (SLc (PL supervise)))::xs) =
   getReport xs) ∧
(∀xs.
   getReport
     (Name PlatoonLeader says
      prop (SOME (SLc (PL report2)))::xs) =
   getReport xs) ∧
```

```
(∀ xs.
   getReport
     (Name PlatoonLeader says
      prop (SOME (SLc (PL complete)))::xs) =
   getReport xs) ∧
(∀ xs.
   getReport
     (Name PlatoonLeader says
      prop (SOME (SLc (PL plIncomplete)))::xs) =
   getReport xs) ∧
(∀ xs.
   getReport
     (Name PlatoonLeader says
      prop (SOME (SLc (PL invalidPlCommand)))::xs) =
   getReport xs) ∧
(∀ xs v151.
   getReport
     (Name PlatoonLeader says prop (SOME (SLc (PSG v151)))::
          xs) =
   getReport xs) ∧
(∀ xs v144.
   getReport
     (Name PlatoonSergeant says prop (SOME v144)::xs) =
   getReport xs) ∧
(∀ xs $v_{68}$ v136 v135.
   getReport (v135 meet v136 says prop $v_{68}$::xs) =
   getReport xs) ∧
(∀ xs $v_{68}$ v138 v137.
   getReport (v137 quoting v138 says prop $v_{68}$::xs) =
   getReport xs) ∧
(∀ xs $v_{69}$ $v_{12}$.
   getReport ($v_{12}$ says notf $v_{69}$::xs) = getReport xs) ∧
(∀ xs $v_{71}$ $v_{70}$ $v_{12}$.
   getReport ($v_{12}$ says ($v_{70}$ andf $v_{71}$)::xs) = getReport xs) ∧
(∀ xs $v_{73}$ $v_{72}$ $v_{12}$.
   getReport ($v_{12}$ says ($v_{72}$ orf $v_{73}$)::xs) = getReport xs) ∧
(∀ xs $v_{75}$ $v_{74}$ $v_{12}$.
   getReport ($v_{12}$ says ($v_{74}$ impf $v_{75}$)::xs) = getReport xs) ∧
(∀ xs $v_{77}$ $v_{76}$ $v_{12}$.
   getReport ($v_{12}$ says ($v_{76}$ eqf $v_{77}$)::xs) = getReport xs) ∧
(∀ xs $v_{79}$ $v_{78}$ $v_{12}$.
   getReport ($v_{12}$ says $v_{78}$ says $v_{79}$::xs) = getReport xs) ∧
(∀ xs $v_{81}$ $v_{80}$ $v_{12}$.
   getReport ($v_{12}$ says $v_{80}$ speaks_for $v_{81}$::xs) =
   getReport xs) ∧
(∀ xs $v_{83}$ $v_{82}$ $v_{12}$.
   getReport ($v_{12}$ says $v_{82}$ controls $v_{83}$::xs) =
   getReport xs) ∧
(∀ xs $v_{86}$ $v_{85}$ $v_{84}$ $v_{12}$.
     getReport (v12 says reps v84 v85 v86::xs) =
     getReport xs) ∧
  (∀xs v88 v87 v12.
     getReport (v12 says v87 domi v88::xs) = getReport xs) ∧
  (∀xs v90 v89 v12.
     getReport (v12 says v89 eqi v90::xs) = getReport xs) ∧
  (∀xs v92 v91 v12.
     getReport (v12 says v91 doms v92::xs) = getReport xs) ∧
  (∀xs v94 v93 v12.
     getReport (v12 says v93 eqs v94::xs) = getReport xs) ∧
  (∀xs v96 v95 v12.
     getReport (v12 says v95 eqn v96::xs) = getReport xs) ∧
  (∀xs v98 v97 v12.
     getReport (v12 says v97 lte v98::xs) = getReport xs) ∧
  (∀xs v99 v12 v100.
     getReport (v12 says v99 lt v100::xs) = getReport xs) ∧
  (∀xs v15 v14.
     getReport (v14 speaks_for v15::xs) = getReport xs) ∧
  (∀xs v17 v16.
     getReport (v16 controls v17::xs) = getReport xs) ∧
  (∀xs v20 v19 v18.
     getReport (reps v18 v19 v20::xs) = getReport xs) ∧
  (∀xs v22 v21. getReport (v21 domi v22::xs) = getReport xs) ∧
  (∀xs v24 v23. getReport (v23 eqi v24::xs) = getReport xs) ∧
  (∀xs v26 v25. getReport (v25 doms v26::xs) = getReport xs) ∧
  (∀xs v28 v27. getReport (v27 eqs v28::xs) = getReport xs) ∧
  (∀xs v30 v29. getReport (v29 eqn v30::xs) = getReport xs) ∧
  (∀xs v32 v31. getReport (v31 lte v32::xs) = getReport xs) ∧
  ∀xs v34 v33. getReport (v33 lt v34::xs) = getReport xs
```

[getReport_ind]

```
⊢ ∀P.
     P [] ∧
     (∀xs.
        P
          (Name PlatoonLeader says
           prop (SOME (SLc (PL report1)))::xs)) ∧
     (∀xs. P xs ⇒ P (TT::xs)) ∧ (∀xs. P xs ⇒ P (FF::xs)) ∧
     (∀v2 xs. P xs ⇒ P (prop v2::xs)) ∧
     (∀v3 xs. P xs ⇒ P (notf v3::xs)) ∧
     (∀v4 v5 xs. P xs ⇒ P (v4 andf v5::xs)) ∧
     (∀v6 v7 xs. P xs ⇒ P (v6 orf v7::xs)) ∧
     (∀v8 v9 xs. P xs ⇒ P (v8 impf v9::xs)) ∧
     (∀v10 v11 xs. P xs ⇒ P (v10 eqf v11::xs)) ∧
     (∀v12 xs. P xs ⇒ P (v12 says TT::xs)) ∧
     (∀v12 xs. P xs ⇒ P (v12 says FF::xs)) ∧
     (∀v134 xs. P xs ⇒ P (Name v134 says prop NONE::xs)) ∧
     (∀v146 xs.
        P xs ⇒
```

```
    P
      (Name PlatoonLeader says prop (SOME (ESCc v146))::
           xs)) ∧
 (∀ xs.
    P xs ⇒
    P
      (Name PlatoonLeader says
       prop (SOME (SLc (PL receiveMission)))::xs)) ∧
 (∀ xs.
    P xs ⇒
    P
      (Name PlatoonLeader says
       prop (SOME (SLc (PL warno)))::xs)) ∧
 (∀ xs.
    P xs ⇒
    P
      (Name PlatoonLeader says
       prop (SOME (SLc (PL tentativePlan)))::xs)) ∧
 (∀ xs.
    P xs ⇒
    P
      (Name PlatoonLeader says
       prop (SOME (SLc (PL recon)))::xs)) ∧
 (∀ xs.
    P xs ⇒
    P
      (Name PlatoonLeader says
       prop (SOME (SLc (PL completePlan)))::xs)) ∧
 (∀ xs.
    P xs ⇒
    P
      (Name PlatoonLeader says
       prop (SOME (SLc (PL opoid)))::xs)) ∧
 (∀ xs.
    P xs ⇒
    P
      (Name PlatoonLeader says
       prop (SOME (SLc (PL supervise)))::xs)) ∧
 (∀ xs.
    P xs ⇒
    P
      (Name PlatoonLeader says
       prop (SOME (SLc (PL report2)))::xs)) ∧
 (∀ xs.
    P xs ⇒
    P
      (Name PlatoonLeader says
       prop (SOME (SLc (PL complete)))::xs)) ∧
 (∀ xs.
      P xs ⇒
      P
        (Name PlatoonLeader says
         prop (SOME (SLc (PL plIncomplete)))::xs)) ∧
   (∀ xs.
      P xs ⇒
      P
        (Name PlatoonLeader says
         prop (SOME (SLc (PL invalidPlCommand)))::xs)) ∧
   (∀ v151 xs.
      P xs ⇒
      P
        (Name PlatoonLeader says
         prop (SOME (SLc (PSG v151)))::xs)) ∧
   (∀ v144 xs.
      P xs ⇒
      P (Name PlatoonSergeant says prop (SOME v144)::xs)) ∧
   (∀ v135 v136 v68 xs.
      P xs ⇒ P (v135 meet v136 says prop v68::xs)) ∧
   (∀ v137 v138 v68 xs.
      P xs ⇒ P (v137 quoting v138 says prop v68::xs)) ∧
   (∀ v12 v69 xs. P xs ⇒ P (v12 says notf v69::xs)) ∧
   (∀ v12 v70 v71 xs. P xs ⇒ P (v12 says (v70 andf v71)::xs)) ∧
   (∀ v12 v72 v73 xs. P xs ⇒ P (v12 says (v72 orf v73)::xs)) ∧
   (∀ v12 v74 v75 xs. P xs ⇒ P (v12 says (v74 impf v75)::xs)) ∧
   (∀ v12 v76 v77 xs. P xs ⇒ P (v12 says (v76 eqf v77)::xs)) ∧
   (∀ v12 v78 v79 xs. P xs ⇒ P (v12 says v78 says v79::xs)) ∧
   (∀ v12 v80 v81 xs.
      P xs ⇒ P (v12 says v80 speaks_for v81::xs)) ∧
   (∀ v12 v82 v83 xs.
      P xs ⇒ P (v12 says v82 controls v83::xs)) ∧
   (∀ v12 v84 v85 v86 xs.
      P xs ⇒ P (v12 says reps v84 v85 v86::xs)) ∧
   (∀ v12 v87 v88 xs. P xs ⇒ P (v12 says v87 domi v88::xs)) ∧
   (∀ v12 v89 v90 xs. P xs ⇒ P (v12 says v89 eqi v90::xs)) ∧
   (∀ v12 v91 v92 xs. P xs ⇒ P (v12 says v91 doms v92::xs)) ∧
   (∀ v12 v93 v94 xs. P xs ⇒ P (v12 says v93 eqs v94::xs)) ∧
   (∀ v12 v95 v96 xs. P xs ⇒ P (v12 says v95 eqn v96::xs)) ∧
   (∀ v12 v97 v98 xs. P xs ⇒ P (v12 says v97 lte v98::xs)) ∧
   (∀ v12 v99 v100 xs. P xs ⇒ P (v12 says v99 lt v100::xs)) ∧
   (∀ v14 v15 xs. P xs ⇒ P (v14 speaks_for v15::xs)) ∧
   (∀ v16 v17 xs. P xs ⇒ P (v16 controls v17::xs)) ∧
   (∀ v18 v19 v20 xs. P xs ⇒ P (reps v18 v19 v20::xs)) ∧
   (∀ v21 v22 xs. P xs ⇒ P (v21 domi v22::xs)) ∧
   (∀ v23 v24 xs. P xs ⇒ P (v23 eqi v24::xs)) ∧
   (∀ v25 v26 xs. P xs ⇒ P (v25 doms v26::xs)) ∧
   (∀ v27 v28 xs. P xs ⇒ P (v27 eqs v28::xs)) ∧
   (∀ v29 v30 xs. P xs ⇒ P (v29 eqn v30::xs)) ∧
   (∀ v31 v32 xs. P xs ⇒ P (v31 lte v32::xs)) ∧
```

```
    (∀ v33 v34 xs. P xs ⇒ P (v33 lt v34::xs)) ⇒
    ∀ v. P v
```

[getTenativePlan_def]

```
 ⊢ (getTenativePlan [] = [NONE]) ∧
   (∀ xs.
      getTenativePlan
        (Name PlatoonLeader says
         prop (SOME (SLc (PL tentativePlan)))::xs) =
      [SOME (SLc (PL tentativePlan))]) ∧
   (∀ xs. getTenativePlan (TT::xs) = getTenativePlan xs) ∧
   (∀ xs. getTenativePlan (FF::xs) = getTenativePlan xs) ∧
   (∀ xs v2.
      getTenativePlan (prop v2::xs) = getTenativePlan xs) ∧
   (∀ xs v3.
      getTenativePlan (notf v3::xs) = getTenativePlan xs) ∧
   (∀ xs v5 v4.
      getTenativePlan (v4 andf v5::xs) = getTenativePlan xs) ∧
   (∀ xs v7 v6.
      getTenativePlan (v6 orf v7::xs) = getTenativePlan xs) ∧
   (∀ xs v9 v8.
      getTenativePlan (v8 impf v9::xs) = getTenativePlan xs) ∧
   (∀ xs v11 v10.
      getTenativePlan (v10 eqf v11::xs) = getTenativePlan xs) ∧
   (∀ xs v12.
      getTenativePlan (v12 says TT::xs) = getTenativePlan xs) ∧
   (∀ xs v12.
      getTenativePlan (v12 says FF::xs) = getTenativePlan xs) ∧
   (∀ xs v134.
      getTenativePlan (Name v134 says prop NONE::xs) =
      getTenativePlan xs) ∧
   (∀ xs v146.
      getTenativePlan
        (Name PlatoonLeader says prop (SOME (ESCc v146))::xs) =
      getTenativePlan xs) ∧
   (∀ xs.
      getTenativePlan
        (Name PlatoonLeader says
         prop (SOME (SLc (PL receiveMission)))::xs) =
      getTenativePlan xs) ∧
   (∀ xs.
      getTenativePlan
        (Name PlatoonLeader says prop (SOME (SLc (PL warno)))::
           xs) =
      getTenativePlan xs) ∧
   (∀ xs.
      getTenativePlan
        (Name PlatoonLeader says prop (SOME (SLc (PL recon)))::
           xs) =
    getTenativePlan xs) ∧
 (∀ xs.
    getTenativePlan
      (Name PlatoonLeader says
       prop (SOME (SLc (PL report1)))::xs) =
    getTenativePlan xs) ∧
 (∀ xs.
    getTenativePlan
      (Name PlatoonLeader says
       prop (SOME (SLc (PL completePlan)))::xs) =
    getTenativePlan xs) ∧
 (∀ xs.
    getTenativePlan
      (Name PlatoonLeader says prop (SOME (SLc (PL opoid)))::
           xs) =
    getTenativePlan xs) ∧
 (∀ xs.
    getTenativePlan
      (Name PlatoonLeader says
       prop (SOME (SLc (PL supervise)))::xs) =
    getTenativePlan xs) ∧
 (∀ xs.
    getTenativePlan
      (Name PlatoonLeader says
       prop (SOME (SLc (PL report2)))::xs) =
    getTenativePlan xs) ∧
 (∀ xs.
    getTenativePlan
      (Name PlatoonLeader says
       prop (SOME (SLc (PL complete)))::xs) =
    getTenativePlan xs) ∧
 (∀ xs.
    getTenativePlan
      (Name PlatoonLeader says
       prop (SOME (SLc (PL plIncomplete)))::xs) =
    getTenativePlan xs) ∧
 (∀ xs.
    getTenativePlan
      (Name PlatoonLeader says
       prop (SOME (SLc (PL invalidPlCommand)))::xs) =
    getTenativePlan xs) ∧
 (∀ xs v151.
    getTenativePlan
      (Name PlatoonLeader says prop (SOME (SLc (PSG v151)))::
           xs) =
    getTenativePlan xs) ∧
 (∀ xs v144.
    getTenativePlan
      (Name PlatoonSergeant says prop (SOME v144)::xs) =
     getTenativePlan xs) ∧
  (∀ xs v68 v136 v135.
     getTenativePlan (v135 meet v136 says prop v68::xs) =
     getTenativePlan xs) ∧
  (∀ xs v68 v138 v137.
     getTenativePlan (v137 quoting v138 says prop v68::xs) =
     getTenativePlan xs) ∧
  (∀ xs v69 v12.
     getTenativePlan (v12 says notf v69::xs) =
     getTenativePlan xs) ∧
  (∀ xs v71 v70 v12.
     getTenativePlan (v12 says (v70 andf v71)::xs) =
     getTenativePlan xs) ∧
  (∀ xs v73 v72 v12.
     getTenativePlan (v12 says (v72 orf v73)::xs) =
     getTenativePlan xs) ∧
  (∀ xs v75 v74 v12.
     getTenativePlan (v12 says (v74 impf v75)::xs) =
     getTenativePlan xs) ∧
  (∀ xs v77 v76 v12.
     getTenativePlan (v12 says (v76 eqf v77)::xs) =
     getTenativePlan xs) ∧
  (∀ xs v79 v78 v12.
     getTenativePlan (v12 says v78 says v79::xs) =
     getTenativePlan xs) ∧
  (∀ xs v81 v80 v12.
     getTenativePlan (v12 says v80 speaks_for v81::xs) =
     getTenativePlan xs) ∧
  (∀ xs v83 v82 v12.
     getTenativePlan (v12 says v82 controls v83::xs) =
     getTenativePlan xs) ∧
  (∀ xs v86 v85 v84 v12.
     getTenativePlan (v12 says reps v84 v85 v86::xs) =
     getTenativePlan xs) ∧
  (∀ xs v88 v87 v12.
     getTenativePlan (v12 says v87 domi v88::xs) =
     getTenativePlan xs) ∧
  (∀ xs v90 v89 v12.
     getTenativePlan (v12 says v89 eqi v90::xs) =
     getTenativePlan xs) ∧
  (∀ xs v92 v91 v12.
     getTenativePlan (v12 says v91 doms v92::xs) =
     getTenativePlan xs) ∧
  (∀ xs v94 v93 v12.
     getTenativePlan (v12 says v93 eqs v94::xs) =
     getTenativePlan xs) ∧
  (∀ xs v96 v95 v12.
     getTenativePlan (v12 says v95 eqn v96::xs) =
     getTenativePlan xs) ∧
```

```
(∀ xs v98 v97 v12.
   getTenativePlan (v12 says v97 lte v98::xs) =
   getTenativePlan xs) ∧
(∀ xs v99 v12 v100.
   getTenativePlan (v12 says v99 lt v100::xs) =
   getTenativePlan xs) ∧
(∀ xs v15 v14.
   getTenativePlan (v14 speaks_for v15::xs) =
   getTenativePlan xs) ∧
(∀ xs v17 v16.
   getTenativePlan (v16 controls v17::xs) =
   getTenativePlan xs) ∧
(∀ xs v20 v19 v18.
   getTenativePlan (reps v18 v19 v20::xs) =
   getTenativePlan xs) ∧
(∀ xs v22 v21.
   getTenativePlan (v21 domi v22::xs) = getTenativePlan xs) ∧
(∀ xs v24 v23.
   getTenativePlan (v23 eqi v24::xs) = getTenativePlan xs) ∧
(∀ xs v26 v25.
   getTenativePlan (v25 doms v26::xs) = getTenativePlan xs) ∧
(∀ xs v28 v27.
   getTenativePlan (v27 eqs v28::xs) = getTenativePlan xs) ∧
(∀ xs v30 v29.
   getTenativePlan (v29 eqn v30::xs) = getTenativePlan xs) ∧
(∀ xs v32 v31.
   getTenativePlan (v31 lte v32::xs) = getTenativePlan xs) ∧
∀ xs v34 v33.
  getTenativePlan (v33 lt v34::xs) = getTenativePlan xs
```

[getTenativePlan_ind]

```
⊢ ∀ P.
    P [] ∧
    (∀ xs.
       P
         (Name PlatoonLeader says
          prop (SOME (SLc (PL tentativePlan)))::xs)) ∧
    (∀ xs. P xs ⇒ P (TT::xs)) ∧ (∀ xs. P xs ⇒ P (FF::xs)) ∧
    (∀ v2 xs. P xs ⇒ P (prop v2::xs)) ∧
    (∀ v3 xs. P xs ⇒ P (notf v3::xs)) ∧
    (∀ v4 v5 xs. P xs ⇒ P (v4 andf v5::xs)) ∧
    (∀ v6 v7 xs. P xs ⇒ P (v6 orf v7::xs)) ∧
    (∀ v8 v9 xs. P xs ⇒ P (v8 impf v9::xs)) ∧
    (∀ v10 v11 xs. P xs ⇒ P (v10 eqf v11::xs)) ∧
    (∀ v12 xs. P xs ⇒ P (v12 says TT::xs)) ∧
    (∀ v12 xs. P xs ⇒ P (v12 says FF::xs)) ∧
    (∀ v134 xs. P xs ⇒ P (Name v134 says prop NONE::xs)) ∧
    (∀ v146 xs.
       P xs ⇒
    P
      (Name PlatoonLeader says prop (SOME (ESCc v146))::
           xs)) ∧
(∀ xs.
    P xs ⇒
    P
      (Name PlatoonLeader says
       prop (SOME (SLc (PL receiveMission)))::xs)) ∧
(∀ xs.
    P xs ⇒
    P
      (Name PlatoonLeader says
       prop (SOME (SLc (PL warno)))::xs)) ∧
(∀ xs.
    P xs ⇒
    P
      (Name PlatoonLeader says
       prop (SOME (SLc (PL recon)))::xs)) ∧
(∀ xs.
    P xs ⇒
    P
      (Name PlatoonLeader says
       prop (SOME (SLc (PL report1)))::xs)) ∧
(∀ xs.
    P xs ⇒
    P
      (Name PlatoonLeader says
       prop (SOME (SLc (PL completePlan)))::xs)) ∧
(∀ xs.
    P xs ⇒
    P
      (Name PlatoonLeader says
       prop (SOME (SLc (PL opoid)))::xs)) ∧
(∀ xs.
    P xs ⇒
    P
      (Name PlatoonLeader says
       prop (SOME (SLc (PL supervise)))::xs)) ∧
(∀ xs.
    P xs ⇒
    P
      (Name PlatoonLeader says
       prop (SOME (SLc (PL report2)))::xs)) ∧
(∀ xs.
    P xs ⇒
    P
      (Name PlatoonLeader says
       prop (SOME (SLc (PL complete)))::xs)) ∧
(∀ xs.
   P xs ⇒
   P
     (Name PlatoonLeader says
      prop (SOME (SLc (PL plIncomplete)))::xs)) ∧
(∀ xs.
   P xs ⇒
   P
     (Name PlatoonLeader says
      prop (SOME (SLc (PL invalidPlCommand)))::xs)) ∧
(∀ v151 xs.
   P xs ⇒
   P
     (Name PlatoonLeader says
      prop (SOME (SLc (PSG v151)))::xs)) ∧
(∀ v144 xs.
   P xs ⇒
   P (Name PlatoonSergeant says prop (SOME v144)::xs)) ∧
(∀ v135 v136 v68 xs.
   P xs ⇒ P (v135 meet v136 says prop v68::xs)) ∧
(∀ v137 v138 v68 xs.
   P xs ⇒ P (v137 quoting v138 says prop v68::xs)) ∧
(∀ v12 v69 xs. P xs ⇒ P (v12 says notf v69::xs)) ∧
(∀ v12 v70 v71 xs. P xs ⇒ P (v12 says (v70 andf v71)::xs)) ∧
(∀ v12 v72 v73 xs. P xs ⇒ P (v12 says (v72 orf v73)::xs)) ∧
(∀ v12 v74 v75 xs. P xs ⇒ P (v12 says (v74 impf v75)::xs)) ∧
(∀ v12 v76 v77 xs. P xs ⇒ P (v12 says (v76 eqf v77)::xs)) ∧
(∀ v12 v78 v79 xs. P xs ⇒ P (v12 says v78 says v79::xs)) ∧
(∀ v12 v80 v81 xs.
   P xs ⇒ P (v12 says v80 speaks_for v81::xs)) ∧
(∀ v12 v82 v83 xs.
   P xs ⇒ P (v12 says v82 controls v83::xs)) ∧
(∀ v12 v84 v85 v86 xs.
   P xs ⇒ P (v12 says reps v84 v85 v86::xs)) ∧
(∀ v12 v87 v88 xs. P xs ⇒ P (v12 says v87 domi v88::xs)) ∧
(∀ v12 v89 v90 xs. P xs ⇒ P (v12 says v89 eqi v90::xs)) ∧
(∀ v12 v91 v92 xs. P xs ⇒ P (v12 says v91 doms v92::xs)) ∧
(∀ v12 v93 v94 xs. P xs ⇒ P (v12 says v93 eqs v94::xs)) ∧
(∀ v12 v95 v96 xs. P xs ⇒ P (v12 says v95 eqn v96::xs)) ∧
(∀ v12 v97 v98 xs. P xs ⇒ P (v12 says v97 lte v98::xs)) ∧
(∀ v12 v99 v100 xs. P xs ⇒ P (v12 says v99 lt v100::xs)) ∧
(∀ v14 v15 xs. P xs ⇒ P (v14 speaks_for v15::xs)) ∧
(∀ v16 v17 xs. P xs ⇒ P (v16 controls v17::xs)) ∧
(∀ v18 v19 v20 xs. P xs ⇒ P (reps v18 v19 v20::xs)) ∧
(∀ v21 v22 xs. P xs ⇒ P (v21 domi v22::xs)) ∧
(∀ v23 v24 xs. P xs ⇒ P (v23 eqi v24::xs)) ∧
(∀ v25 v26 xs. P xs ⇒ P (v25 doms v26::xs)) ∧
(∀ v27 v28 xs. P xs ⇒ P (v27 eqs v28::xs)) ∧
(∀ v29 v30 xs. P xs ⇒ P (v29 eqn v30::xs)) ∧
(∀ v31 v32 xs. P xs ⇒ P (v31 lte v32::xs)) ∧
```

$(\forall v_{33}\ v_{34}\ xs.\ P\ xs \Rightarrow P\ (v_{33}\ \texttt{lt}\ v_{34}\texttt{::}xs)) \Rightarrow$
$\forall v.\ P\ v$

Index