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]

$\vdash \forall a'\ a.$ ESCc $a \neq$ SLc a'

[command_one_one]

$\vdash$ $(\forall a\ a'.$ (ESCc a = ESCc $a') \iff (a = a')) \wedge$
$\forall a\ a'.$ (SLc a = SLc $a') \iff (a = a')$

[escCommand_distinct_clauses]

$\vdash$ returnToBase $\neq$ changeMission $\wedge$ returnToBase $\neq$ resupply $\wedge$
returnToBase $\neq$ reactToContact $\wedge$ changeMission $\neq$ resupply $\wedge$
changeMission $\neq$ reactToContact $\wedge$ resupply $\neq$ reactToContact

[escOutput_distinct_clauses]

$\vdash$ ReturnToBase $\neq$ ChangeMission $\wedge$ ReturnToBase $\neq$ Resupply $\wedge$
ReturnToBase $\neq$ ReactToContact $\wedge$ ChangeMission $\neq$ Resupply $\wedge$
ChangeMission $\neq$ ReactToContact $\wedge$ Resupply $\neq$ ReactToContact

[escState_distinct_clauses]

$\vdash$ RTB $\neq$ CM $\wedge$ RTB $\neq$ RESUPPLY $\wedge$ RTB $\neq$ RTC $\wedge$ CM $\neq$ RESUPPLY $\wedge$
CM $\neq$ RTC $\wedge$ RESUPPLY $\neq$ RTC

[output_distinct_clauses]

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

[output_one_one]

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

[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]

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

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]

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

```
        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* $\wedge$ (M, Oi, Os) `sat` *input* $\wedge$
(M, Oi, Os) `sat` *stateInterp state*

[CFGInterpret_ind]

$\vdash \forall P.$
$(\forall M\ Oi\ Os\ authenticationTest\ stateInterp\ securityContext$
$input\ ins\ state\ outputStream.$
$P\ (M, Oi, Os)$
$(\texttt{CFG}\ authenticationTest\ stateInterp\ securityContext$
$(input :: ins)\ state\ outputStream)) \wedge$
$(\forall v_{15}\ v_{10}\ v_{11}\ v_{12}\ v_{13}\ v_{14}.$
$P\ v_{15}\ (\texttt{CFG}\ v_{10}\ v_{11}\ v_{12}\ []\ v_{13}\ v_{14})) \Rightarrow$
$\forall v\ v_1\ v_2\ v_3.\ P\ (v, v_1, v_2)\ v_3$

[configuration_one_one]

$\vdash \forall a_0\ a_1\ a_2\ a_3\ a_4\ a_5\ a_0'\ a_1'\ a_2'\ a_3'\ a_4'\ a_5'.$
$(\texttt{CFG}\ a_0\ a_1\ a_2\ a_3\ a_4\ a_5 = \texttt{CFG}\ a_0'\ a_1'\ a_2'\ a_3'\ a_4'\ a_5') \iff$
$(a_0 = a_0') \wedge (a_1 = a_1') \wedge (a_2 = a_2') \wedge (a_3 = a_3') \wedge$
$(a_4 = a_4') \wedge (a_5 = a_5')$

[order_distinct_clauses]

$\vdash \forall a.\ \texttt{SOME}\ a \neq \texttt{NONE}$

[order_one_one]

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

[TR_cases]

$\vdash \forall a_0\ a_1\ a_2\ a_3.$
$\texttt{TR}\ a_0\ a_1\ a_2\ a_3 \iff$
$(\exists authenticationTest\ P\ NS\ M\ Oi\ Os\ Out\ s\ securityContext$
$stateInterp\ cmd\ ins\ outs.$
$(a_0 = (M, Oi, Os)) \wedge (a_1 = \texttt{exec}\ cmd) \wedge$
$(a_2 =$
$\texttt{CFG}\ authenticationTest\ stateInterp\ securityContext$
$(P\ \texttt{says prop (SOME}\ cmd) :: ins)\ s\ outs) \wedge$
$(a_3 =$
$\texttt{CFG}\ authenticationTest\ stateInterp\ securityContext\ ins$
$(NS\ s\ (\texttt{exec}\ cmd))\ (Out\ s\ (\texttt{exec}\ cmd) :: outs)) \wedge$
$authenticationTest\ (P\ \texttt{says prop (SOME}\ cmd))\ \wedge$
$\texttt{CFGInterpret}\ (M, Oi, Os)$
$(\texttt{CFG}\ authenticationTest\ stateInterp\ securityContext$
$(P\ \texttt{says prop (SOME}\ cmd) :: ins)\ s\ outs)) \vee$
$(\exists authenticationTest\ P\ NS\ M\ Oi\ Os\ Out\ s\ securityContext$
$stateInterp\ cmd\ ins\ outs.$
$(a_0 = (M, Oi, Os)) \wedge (a_1 = \texttt{trap}\ cmd) \wedge$
$(a_2 =$
$\texttt{CFG}\ authenticationTest\ stateInterp\ securityContext$
$(P\ \texttt{says prop (SOME}\ cmd) :: ins)\ s\ outs) \wedge$

$(a_3 =$
$\texttt{CFG}\ \mathit{authenticationTest}\ \mathit{stateInterp}\ \mathit{securityContext}\ \mathit{ins}$
$(\mathit{NS}\ s\ (\texttt{trap}\ \mathit{cmd}))\ (\mathit{Out}\ s\ (\texttt{trap}\ \mathit{cmd})::\mathit{outs})) \wedge$
$\mathit{authenticationTest}\ (P\ \texttt{says prop (SOME}\ \mathit{cmd})) \wedge$
$\texttt{CFGInterpret}\ (M, Oi, Os)$
$(\texttt{CFG}\ \mathit{authenticationTest}\ \mathit{stateInterp}\ \mathit{securityContext}$
$(P\ \texttt{says prop (SOME}\ \mathit{cmd})::\mathit{ins})\ s\ \mathit{outs})) \vee$
$\exists\, \mathit{authenticationTest}\ \mathit{NS}\ M\ \mathit{Oi}\ \mathit{Os}\ \mathit{Out}\ s\ \mathit{securityContext}$
$\mathit{stateInterp}\ \mathit{cmd}\ x\ \mathit{ins}\ \mathit{outs}.$
$(a_0 = (M, Oi, Os)) \wedge (a_1 = \texttt{discard}\ \mathit{cmd}) \wedge$
$(a_2 =$
$\texttt{CFG}\ \mathit{authenticationTest}\ \mathit{stateInterp}\ \mathit{securityContext}$
$(x::\mathit{ins})\ s\ \mathit{outs}) \wedge$
$(a_3 =$
$\texttt{CFG}\ \mathit{authenticationTest}\ \mathit{stateInterp}\ \mathit{securityContext}\ \mathit{ins}$
$(\mathit{NS}\ s\ (\texttt{discard}\ \mathit{cmd}))\ (\mathit{Out}\ s\ (\texttt{discard}\ \mathit{cmd})::\mathit{outs})) \wedge$
$\neg \mathit{authenticationTest}\ x$

[TR_discard_cmd_rule]

$\vdash \texttt{TR}\ (M, Oi, Os)\ (\texttt{discard}\ \mathit{cmd})$
$(\texttt{CFG}\ \mathit{authenticationTest}\ \mathit{stateInterp}\ \mathit{securityContext}$
$(x::\mathit{ins})\ s\ \mathit{outs})$
$(\texttt{CFG}\ \mathit{authenticationTest}\ \mathit{stateInterp}\ \mathit{securityContext}\ \mathit{ins}$
$(\mathit{NS}\ s\ (\texttt{discard}\ \mathit{cmd}))\ (\mathit{Out}\ s\ (\texttt{discard}\ \mathit{cmd})::\mathit{outs})) \iff$
$\neg \mathit{authenticationTest}\ x$

[TR_EQ_rules_thm]

$\vdash (\texttt{TR}\ (M, Oi, Os)\ (\texttt{exec}\ \mathit{cmd})$
$(\texttt{CFG}\ \mathit{authenticationTest}\ \mathit{stateInterp}\ \mathit{securityContext}$
$(P\ \texttt{says prop (SOME}\ \mathit{cmd})::\mathit{ins})\ s\ \mathit{outs})$
$(\texttt{CFG}\ \mathit{authenticationTest}\ \mathit{stateInterp}\ \mathit{securityContext}\ \mathit{ins}$
$(\mathit{NS}\ s\ (\texttt{exec}\ \mathit{cmd}))\ (\mathit{Out}\ s\ (\texttt{exec}\ \mathit{cmd})::\mathit{outs})) \iff$
$\mathit{authenticationTest}\ (P\ \texttt{says prop (SOME}\ \mathit{cmd})) \wedge$
$\texttt{CFGInterpret}\ (M, Oi, Os)$
$(\texttt{CFG}\ \mathit{authenticationTest}\ \mathit{stateInterp}\ \mathit{securityContext}$
$(P\ \texttt{says prop (SOME}\ \mathit{cmd})::\mathit{ins})\ s\ \mathit{outs})) \wedge$
$(\texttt{TR}\ (M, Oi, Os)\ (\texttt{trap}\ \mathit{cmd})$
$(\texttt{CFG}\ \mathit{authenticationTest}\ \mathit{stateInterp}\ \mathit{securityContext}$
$(P\ \texttt{says prop (SOME}\ \mathit{cmd})::\mathit{ins})\ s\ \mathit{outs})$
$(\texttt{CFG}\ \mathit{authenticationTest}\ \mathit{stateInterp}\ \mathit{securityContext}\ \mathit{ins}$
$(\mathit{NS}\ s\ (\texttt{trap}\ \mathit{cmd}))\ (\mathit{Out}\ s\ (\texttt{trap}\ \mathit{cmd})::\mathit{outs})) \iff$
$\mathit{authenticationTest}\ (P\ \texttt{says prop (SOME}\ \mathit{cmd})) \wedge$
$\texttt{CFGInterpret}\ (M, Oi, Os)$
$(\texttt{CFG}\ \mathit{authenticationTest}\ \mathit{stateInterp}\ \mathit{securityContext}$
$(P\ \texttt{says prop (SOME}\ \mathit{cmd})::\mathit{ins})\ s\ \mathit{outs})) \wedge$
$(\texttt{TR}\ (M, Oi, Os)\ (\texttt{discard}\ \mathit{cmd})$
$(\texttt{CFG}\ \mathit{authenticationTest}\ \mathit{stateInterp}\ \mathit{securityContext}$
$(x::\mathit{ins})\ s\ \mathit{outs})$
$(\texttt{CFG}\ \mathit{authenticationTest}\ \mathit{stateInterp}\ \mathit{securityContext}\ \mathit{ins}$

$$(NS\ s\ (\texttt{discard}\ cmd))\ (Out\ s\ (\texttt{discard}\ cmd)::outs)) \iff \neg authenticationTest\ x)$$

[TR_exec_cmd_rule]

$$\begin{array}{l}\vdash \forall\, authenticationTest\ securityContext\ stateInterp\ P\ cmd\ ins\ s\ outs.\\ (\forall\, M\ Oi\ Os.\\ \quad \texttt{CFGInterpret}\ (M,Oi,Os)\\ \quad\quad (\texttt{CFG}\ authenticationTest\ stateInterp\ securityContext\\ \quad\quad\quad (P\ \texttt{says prop (SOME}\ cmd)::ins)\ s\ outs) \Rightarrow\\ \quad (M,Oi,Os)\ \texttt{sat prop (SOME}\ cmd)) \Rightarrow\\ \forall\, NS\ Out\ M\ Oi\ Os.\\ \quad \texttt{TR}\ (M,Oi,Os)\ (\texttt{exec}\ cmd)\\ \quad\quad (\texttt{CFG}\ authenticationTest\ stateInterp\ securityContext\\ \quad\quad\quad (P\ \texttt{says prop (SOME}\ cmd)::ins)\ s\ outs)\\ \quad\quad (\texttt{CFG}\ authenticationTest\ stateInterp\ securityContext\ ins\\ \quad\quad\quad (NS\ s\ (\texttt{exec}\ cmd))\ (Out\ s\ (\texttt{exec}\ cmd)::outs)) \iff\\ \quad authenticationTest\ (P\ \texttt{says prop (SOME}\ cmd)) \wedge\\ \quad \texttt{CFGInterpret}\ (M,Oi,Os)\\ \quad\quad (\texttt{CFG}\ authenticationTest\ stateInterp\ securityContext\\ \quad\quad\quad (P\ \texttt{says prop (SOME}\ cmd)::ins)\ s\ outs) \wedge\\ \quad (M,Oi,Os)\ \texttt{sat prop (SOME}\ cmd)\end{array}$$

[TR_ind]

$$\begin{array}{l}\vdash \forall\, TR'.\\ (\forall\, authenticationTest\ P\ NS\ M\ Oi\ Os\ Out\ s\ securityContext\\ \quad stateInterp\ cmd\ ins\ outs.\\ \quad authenticationTest\ (P\ \texttt{says prop (SOME}\ cmd)) \wedge\\ \quad \texttt{CFGInterpret}\ (M,Oi,Os)\\ \quad\quad (\texttt{CFG}\ authenticationTest\ stateInterp\ securityContext\\ \quad\quad\quad (P\ \texttt{says prop (SOME}\ cmd)::ins)\ s\ outs) \Rightarrow\\ \quad TR'\ (M,Oi,Os)\ (\texttt{exec}\ cmd)\\ \quad\quad (\texttt{CFG}\ authenticationTest\ stateInterp\ securityContext\\ \quad\quad\quad (P\ \texttt{says prop (SOME}\ cmd)::ins)\ s\ outs)\\ \quad\quad (\texttt{CFG}\ authenticationTest\ stateInterp\ securityContext\\ \quad\quad\quad ins\ (NS\ s\ (\texttt{exec}\ cmd))\ (Out\ s\ (\texttt{exec}\ cmd)::outs))) \wedge\\ (\forall\, authenticationTest\ P\ NS\ M\ Oi\ Os\ Out\ s\ securityContext\\ \quad stateInterp\ cmd\ ins\ outs.\\ \quad authenticationTest\ (P\ \texttt{says prop (SOME}\ cmd)) \wedge\\ \quad \texttt{CFGInterpret}\ (M,Oi,Os)\\ \quad\quad (\texttt{CFG}\ authenticationTest\ stateInterp\ securityContext\\ \quad\quad\quad (P\ \texttt{says prop (SOME}\ cmd)::ins)\ s\ outs) \Rightarrow\\ \quad TR'\ (M,Oi,Os)\ (\texttt{trap}\ cmd)\\ \quad\quad (\texttt{CFG}\ authenticationTest\ stateInterp\ securityContext\\ \quad\quad\quad (P\ \texttt{says prop (SOME}\ cmd)::ins)\ s\ outs)\\ \quad\quad (\texttt{CFG}\ authenticationTest\ stateInterp\ securityContext\\ \quad\quad\quad ins\ (NS\ s\ (\texttt{trap}\ cmd))\ (Out\ s\ (\texttt{trap}\ cmd)::outs))) \wedge\\ (\forall\, authenticationTest\ NS\ M\ Oi\ Os\ Out\ s\ securityContext\\ \quad stateInterp\ cmd\ x\ ins\ outs.\end{array}$$

$\neg authenticationTest\ x \Rightarrow$
$TR'\ (M, Oi, Os)$ (discard cmd)
(CFG $authenticationTest\ stateInterp\ securityContext$
(x::ins) $s\ outs$)
(CFG $authenticationTest\ stateInterp\ securityContext$
ins ($NS\ s$ (discard cmd))
($Out\ s$ (discard cmd)::$outs$))) $\Rightarrow$
$\forall a_0\ a_1\ a_2\ a_3.$ TR $a_0\ a_1\ a_2\ a_3 \Rightarrow TR'\ a_0\ a_1\ a_2\ a_3$

[TR_rules]

$\vdash$ ($\forall$ $authenticationTest\ P\ NS\ M\ Oi\ Os\ Out\ s\ securityContext$
$stateInterp\ cmd\ ins\ outs.$
$authenticationTest$ (P says prop (SOME cmd)) $\wedge$
CFGInterpret (M, Oi, Os)
(CFG $authenticationTest\ stateInterp\ securityContext$
(P says prop (SOME cmd)::ins) $s\ outs$) $\Rightarrow$
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)::$outs$))) $\wedge$
($\forall$ $authenticationTest\ P\ NS\ M\ Oi\ Os\ Out\ s\ securityContext$
$stateInterp\ cmd\ ins\ outs.$
$authenticationTest$ (P says prop (SOME cmd)) $\wedge$
CFGInterpret (M, Oi, Os)
(CFG $authenticationTest\ stateInterp\ securityContext$
(P says prop (SOME cmd)::ins) $s\ outs$) $\Rightarrow$
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)::$outs$))) $\wedge$
$\forall$ $authenticationTest\ NS\ M\ Oi\ Os\ Out\ s\ securityContext$
$stateInterp\ cmd\ x\ ins\ outs.$
$\neg authenticationTest\ x \Rightarrow$
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]

$\vdash \forall TR'.$
($\forall$ $authenticationTest\ P\ NS\ M\ Oi\ Os\ Out\ s\ securityContext$
$stateInterp\ cmd\ ins\ outs.$
$authenticationTest$ (P says prop (SOME cmd)) $\wedge$
CFGInterpret (M, Oi, Os)
(CFG $authenticationTest\ stateInterp\ securityContext$
(P says prop (SOME cmd)::ins) $s\ outs$) $\Rightarrow$

$$
\begin{array}{l}
\quad TR'\ (M, Oi, Os)\ (\texttt{exec}\ cmd) \\
\quad\quad (\texttt{CFG}\ authenticationTest\ stateInterp\ securityContext \\
\quad\quad\quad (P\ \texttt{says prop (SOME}\ cmd\texttt{)::}ins)\ s\ outs) \\
\quad\quad (\texttt{CFG}\ authenticationTest\ stateInterp\ securityContext \\
\quad\quad\quad ins\ (NS\ s\ (\texttt{exec}\ cmd))\ (Out\ s\ (\texttt{exec}\ cmd)\texttt{::}outs))) \wedge \\
(\forall\, authenticationTest\ P\ NS\ M\ Oi\ Os\ Out\ s\ securityContext \\
\quad stateInterp\ cmd\ ins\ outs. \\
\quad authenticationTest\ (P\ \texttt{says prop (SOME}\ cmd\texttt{)}) \wedge \\
\quad \texttt{CFGInterpret}\ (M, Oi, Os) \\
\quad\quad (\texttt{CFG}\ authenticationTest\ stateInterp\ securityContext \\
\quad\quad\quad (P\ \texttt{says prop (SOME}\ cmd\texttt{)::}ins)\ s\ outs) \Rightarrow \\
\quad TR'\ (M, Oi, Os)\ (\texttt{trap}\ cmd) \\
\quad\quad (\texttt{CFG}\ authenticationTest\ stateInterp\ securityContext \\
\quad\quad\quad (P\ \texttt{says prop (SOME}\ cmd\texttt{)::}ins)\ s\ outs) \\
\quad\quad (\texttt{CFG}\ authenticationTest\ stateInterp\ securityContext \\
\quad\quad\quad ins\ (NS\ s\ (\texttt{trap}\ cmd))\ (Out\ s\ (\texttt{trap}\ cmd)\texttt{::}outs))) \wedge \\
(\forall\, authenticationTest\ NS\ M\ Oi\ Os\ Out\ s\ securityContext \\
\quad stateInterp\ cmd\ x\ ins\ outs. \\
\quad \neg authenticationTest\ x \Rightarrow \\
\quad TR'\ (M, Oi, Os)\ (\texttt{discard}\ cmd) \\
\quad\quad (\texttt{CFG}\ authenticationTest\ stateInterp\ securityContext \\
\quad\quad\quad (x\texttt{::}ins)\ s\ outs) \\
\quad\quad (\texttt{CFG}\ authenticationTest\ stateInterp\ securityContext \\
\quad\quad\quad ins\ (NS\ s\ (\texttt{discard}\ cmd)) \\
\quad\quad\quad (Out\ s\ (\texttt{discard}\ cmd)\texttt{::}outs))) \Rightarrow \\
\forall\, a_0\ a_1\ a_2\ a_3.\ \texttt{TR}\ a_0\ a_1\ a_2\ a_3 \Rightarrow TR'\ a_0\ a_1\ a_2\ a_3
\end{array}
$$

[TR_trap_cmd_rule]

$$
\begin{array}{l}
\vdash \forall\, authenticationTest\ stateInterp\ securityContext\ P\ cmd\ ins\ s \\
\quad outs. \\
\quad (\forall\, M\ Oi\ Os. \\
\quad\quad \texttt{CFGInterpret}\ (M, Oi, Os) \\
\quad\quad\quad (\texttt{CFG}\ authenticationTest\ stateInterp\ securityContext \\
\quad\quad\quad\quad (P\ \texttt{says prop (SOME}\ cmd\texttt{)::}ins)\ s\ outs) \Rightarrow \\
\quad\quad (M, Oi, Os)\ \texttt{sat prop NONE}) \Rightarrow \\
\quad \forall\, NS\ Out\ M\ Oi\ Os. \\
\quad\quad \texttt{TR}\ (M, Oi, Os)\ (\texttt{trap}\ cmd) \\
\quad\quad\quad (\texttt{CFG}\ authenticationTest\ stateInterp\ securityContext \\
\quad\quad\quad\quad (P\ \texttt{says prop (SOME}\ cmd\texttt{)::}ins)\ s\ outs) \\
\quad\quad\quad (\texttt{CFG}\ authenticationTest\ stateInterp\ securityContext\ ins \\
\quad\quad\quad\quad (NS\ s\ (\texttt{trap}\ cmd))\ (Out\ s\ (\texttt{trap}\ cmd)\texttt{::}outs)) \iff \\
\quad\quad authenticationTest\ (P\ \texttt{says prop (SOME}\ cmd\texttt{)}) \wedge \\
\quad\quad \texttt{CFGInterpret}\ (M, Oi, Os) \\
\quad\quad\quad (\texttt{CFG}\ authenticationTest\ stateInterp\ securityContext \\
\quad\quad\quad\quad (P\ \texttt{says prop (SOME}\ cmd\texttt{)::}ins)\ s\ outs) \wedge \\
\quad\quad (M, Oi, Os)\ \texttt{sat prop NONE}
\end{array}
$$

[TRrule0]

$$
\begin{array}{l}
\vdash \texttt{TR}\ (M, Oi, Os)\ (\texttt{exec}\ cmd) \\
\quad (\texttt{CFG}\ authenticationTest\ stateInterp\ securityContext
\end{array}
$$

```
      (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]

$$\vdash \forall\, elementTest\ x.\ \texttt{authenticationTest}\ elementTest\ x \iff \texttt{FOLDR}\ (\lambda p\ q.\ p \wedge q)\ \texttt{T}\ (\texttt{MAP}\ elementTest\ x)$$

[commandList_def]

$$\vdash \forall x.\ \texttt{commandList}\ x = \texttt{MAP extractCommand}\ x$$

[inputList_def]

$$\vdash \forall xs.\ \texttt{inputList}\ xs = \texttt{MAP extractInput}\ xs$$

[propCommandList_def]

$$\vdash \forall x.\ \texttt{propCommandList}\ x = \texttt{MAP extractPropCommand}\ x$$

[TR_def]

$$\begin{array}{l}
\vdash \texttt{TR} = \\
\quad (\lambda\, a_0\ a_1\ a_2\ a_3.\ \\
\qquad \forall\, TR'.\ \\
\qquad\quad (\forall\, a_0\ a_1\ a_2\ a_3.\ \\
\qquad\qquad (\exists\, elementTest\ NS\ M\ Oi\ Os\ Out\ s\ context\ stateInterp\ x\ ins\ outs.\ \\
\qquad\qquad\quad (a_0 = (M, Oi, Os)) \wedge (a_1 = \texttt{exec (inputList}\ x\texttt{)}) \wedge \\
\qquad\qquad\quad (a_2 = \\
\qquad\qquad\quad\ \texttt{CFG}\ elementTest\ stateInterp\ context\ (x :: ins)\ s\ outs) \wedge \\
\qquad\qquad\quad (a_3 = \\
\qquad\qquad\quad\ \texttt{CFG}\ elementTest\ stateInterp\ context\ ins\ (NS\ s\ \texttt{(exec (inputList}\ x\texttt{))})\ (Out\ s\ \texttt{(exec (inputList}\ x\texttt{))} :: outs)) \wedge \\
\qquad\qquad\quad \texttt{authenticationTest}\ elementTest\ x \wedge \\
\qquad\qquad\quad \texttt{CFGInterpret}\ (M, Oi, Os)\ (\texttt{CFG}\ elementTest\ stateInterp\ context\ (x :: ins)\ s\ outs)) \vee \\
\qquad\qquad (\exists\, elementTest\ NS\ M\ Oi\ Os\ Out\ s\ context\ stateInterp\ x\ ins\ outs.\ \\
\qquad\qquad\quad (a_0 = (M, Oi, Os)) \wedge (a_1 = \texttt{trap (inputList}\ x\texttt{)}) \wedge \\
\qquad\qquad\quad (a_2 = \\
\qquad\qquad\quad\ \texttt{CFG}\ elementTest\ stateInterp\ context\ (x :: ins)\ s\ outs) \wedge \\
\qquad\qquad\quad (a_3 = \\
\qquad\qquad\quad\ \texttt{CFG}\ elementTest\ stateInterp\ context\ ins\ (NS\ s\ \texttt{(trap (inputList}\ x\texttt{))})\ (Out\ s\ \texttt{(trap (inputList}\ x\texttt{))} :: outs)) \wedge \\
\qquad\qquad\quad \texttt{authenticationTest}\ elementTest\ x \wedge \\
\qquad\qquad\quad \texttt{CFGInterpret}\ (M, Oi, Os)\ (\texttt{CFG}\ elementTest\ stateInterp\ context\ (x :: ins)\ s
\end{array}$$

```
            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′ a0 a1 a2 a3) ⇒
     TR′ a0 a1 a2 a3)
```

3.3 Theorems

[CFGInterpret_def]

```
⊢ CFGInterpret (M,Oi,Os)
    (CFG elementTest stateInterp context (x::ins) state
       outStream) ⇐⇒
  (M,Oi,Os) satList context x ∧ (M,Oi,Os) satList x ∧
  (M,Oi,Os) satList stateInterp state x
```

[CFGInterpret_ind]

```
⊢ ∀ P.
    (∀ M Oi Os elementTest stateInterp context x ins state
        outStream.
       P (M,Oi,Os)
         (CFG elementTest stateInterp context (x::ins) state
            outStream)) ∧
    (∀ 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′)
```

[extractCommand_def]

```
⊢ extractCommand (P says prop (SOME cmd)) = cmd
```

[extractCommand_ind]

```
⊢ ∀ P′.
    (∀ P cmd. P′ (P says prop (SOME cmd))) ∧ P′ TT ∧ P′ FF ∧
    (∀ v1. P′ (prop v1)) ∧ (∀ v3. P′ (notf v3)) ∧
```

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

[extractInput_def]

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

[extractInput_ind]

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

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

[extractPropCommand_def]

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

[extractPropCommand_ind]

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

$(\forall\, v147\ v123\ v124.\ P'\ (v147\ \texttt{says}\ v123\ \texttt{lte}\ v124)) \wedge$
$(\forall\, v148\ v127\ v128.\ P'\ (v148\ \texttt{says}\ v127\ \texttt{lt}\ v128)) \wedge$
$(\forall\, v_{24}\ v_{25}.\ P'\ (v_{24}\ \texttt{speaks\_for}\ v_{25})) \wedge$
$(\forall\, v_{28}\ v_{29}.\ P'\ (v_{28}\ \texttt{controls}\ v_{29})) \wedge$
$(\forall\, v_{33}\ v_{34}\ v_{35}.\ P'\ (\texttt{reps}\ v_{33}\ v_{34}\ v_{35})) \wedge$
$(\forall\, v_{38}\ v_{39}.\ P'\ (v_{38}\ \texttt{domi}\ v_{39})) \wedge$
$(\forall\, v_{42}\ v_{43}.\ P'\ (v_{42}\ \texttt{eqi}\ v_{43})) \wedge$
$(\forall\, v_{46}\ v_{47}.\ P'\ (v_{46}\ \texttt{doms}\ v_{47})) \wedge$
$(\forall\, v_{50}\ v_{51}.\ P'\ (v_{50}\ \texttt{eqs}\ v_{51})) \wedge$
$(\forall\, v_{54}\ v_{55}.\ P'\ (v_{54}\ \texttt{eqn}\ v_{55})) \wedge$
$(\forall\, v_{58}\ v_{59}.\ P'\ (v_{58}\ \texttt{lte}\ v_{59})) \wedge$
$(\forall\, v_{62}\ v_{63}.\ P'\ (v_{62}\ \texttt{lt}\ v_{63})) \Rightarrow$
$\forall\, v.\ P'\ v$

[TR_cases]

$\vdash \forall\, a_0\ a_1\ a_2\ a_3.$
$\texttt{TR}\ a_0\ a_1\ a_2\ a_3 \iff$
$(\exists\, elementTest\ NS\ M\ Oi\ Os\ Out\ s\ context\ stateInterp\ x\ ins\ outs.$
$(a_0 = (M, Oi, Os)) \wedge (a_1 = \texttt{exec (inputList}\ x)) \wedge$
$(a_2 =$
$\texttt{CFG}\ elementTest\ stateInterp\ context\ (x::ins)\ s\ outs) \wedge$
$(a_3 =$
$\texttt{CFG}\ elementTest\ stateInterp\ context\ ins$
$(NS\ s\ \texttt{(exec (inputList}\ x))))$
$(Out\ s\ \texttt{(exec (inputList}\ x))::outs)) \wedge$
$\texttt{authenticationTest}\ elementTest\ x \wedge$
$\texttt{CFGInterpret}\ (M, Oi, Os)$
$(\texttt{CFG}\ elementTest\ stateInterp\ context\ (x::ins)\ s$
$outs)) \vee$
$(\exists\, elementTest\ NS\ M\ Oi\ Os\ Out\ s\ context\ stateInterp\ x\ ins\ outs.$
$(a_0 = (M, Oi, Os)) \wedge (a_1 = \texttt{trap (inputList}\ x)) \wedge$
$(a_2 =$
$\texttt{CFG}\ elementTest\ stateInterp\ context\ (x::ins)\ s\ outs) \wedge$
$(a_3 =$
$\texttt{CFG}\ elementTest\ stateInterp\ context\ ins$
$(NS\ s\ \texttt{(trap (inputList}\ x))))$
$(Out\ s\ \texttt{(trap (inputList}\ x))::outs)) \wedge$
$\texttt{authenticationTest}\ elementTest\ x \wedge$
$\texttt{CFGInterpret}\ (M, Oi, Os)$
$(\texttt{CFG}\ elementTest\ stateInterp\ context\ (x::ins)\ s$
$outs)) \vee$
$\exists\, elementTest\ NS\ M\ Oi\ Os\ Out\ s\ context\ stateInterp\ x\ ins$
$outs.$
$(a_0 = (M, Oi, Os)) \wedge (a_1 = \texttt{discard (inputList}\ x)) \wedge$
$(a_2 =$
$\texttt{CFG}\ elementTest\ stateInterp\ context\ (x::ins)\ s\ outs) \wedge$
$(a_3 =$

```
        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 $(\lambda f.\ (M, Oi, Os)$ 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 (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 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 v68 v136 v135.
   getOmniCommand (v135 meet v136 says prop v68::xs) =
   getOmniCommand xs) ∧
(∀ xs v68 v138 v137.
   getOmniCommand (v137 quoting v138 says prop v68::xs) =
   getOmniCommand xs) ∧
(∀ xs v69 v12.
   getOmniCommand (v12 says notf v69::xs) =
   getOmniCommand xs) ∧
(∀ xs v71 v70 v12.
   getOmniCommand (v12 says (v70 andf v71)::xs) =
   getOmniCommand xs) ∧
(∀ xs v73 v72 v12.
   getOmniCommand (v12 says (v72 orf v73)::xs) =
   getOmniCommand xs) ∧
(∀ xs v75 v74 v12.
   getOmniCommand (v12 says (v74 impf v75)::xs) =
   getOmniCommand xs) ∧
(∀ xs v77 v76 v12.
   getOmniCommand (v12 says (v76 eqf v77)::xs) =
   getOmniCommand xs) ∧
(∀ xs v79 v78 v12.
   getOmniCommand (v12 says v78 says v79::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 (v29 eqn v30::xs) = getOmniCommand xs) ∧
  (∀ xs v32 v31.
    getOmniCommand (v31 lte v32::xs) = getOmniCommand xs) ∧
  ∀ xs v34 v33.
    getOmniCommand (v33 lt v34::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)) ∧
    (∀ 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 Omni says prop (SOME (ESCc v146))::xs)) ∧
    (∀ v150 xs.
       P xs ⇒
       P (Name Omni 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)) ∧
```

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

[getPlCom_def]

$\vdash$ (getPlCom [] = incomplete) $\wedge$
$(\forall xs\ cmd.$ getPlCom (SOME (SLc (PL cmd))::xs) = cmd) $\wedge$
$(\forall xs.$ getPlCom (NONE::xs) = getPlCom xs) $\wedge$
$(\forall xs\ v_4.$ getPlCom (SOME (ESCc v_4)::xs) = getPlCom xs) $\wedge$
$\forall xs\ v_9.$ getPlCom (SOME (SLc (OMNI v_9))::xs) = getPlCom xs

[getPlCom_ind]

$\vdash \forall P.$
P [] $\wedge$ $(\forall cmd\ xs.\ P$ (SOME (SLc (PL cmd))::xs)) $\wedge$
$(\forall xs.\ P\ xs \Rightarrow P$ (NONE::xs)) $\wedge$
$(\forall v_4\ xs.\ P\ xs \Rightarrow P$ (SOME (ESCc v_4)::xs)) $\wedge$
$(\forall v_9\ xs.\ P\ xs \Rightarrow P$ (SOME (SLc (OMNI v_9))::xs)) $\Rightarrow$
$\forall v.\ P\ v$

7 ssmPBIntegrated Theory

Built: 11 June 2018
Parent Theories: PBIntegratedDef, ssm

7.1 Theorems

[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 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) $\wedge$
(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]

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

$(\forall\, v_{10}\ v_{89}\ v_{90}.\ P\ (v_{10}\ \texttt{says}\ v_{89}\ \texttt{doms}\ v_{90})) \wedge$
$(\forall\, v_{10}\ v_{91}\ v_{92}.\ P\ (v_{10}\ \texttt{says}\ v_{91}\ \texttt{eqs}\ v_{92})) \wedge$
$(\forall\, v_{10}\ v_{93}\ v_{94}.\ P\ (v_{10}\ \texttt{says}\ v_{93}\ \texttt{eqn}\ v_{94})) \wedge$
$(\forall\, v_{10}\ v_{95}\ v_{96}.\ P\ (v_{10}\ \texttt{says}\ v_{95}\ \texttt{lte}\ v_{96})) \wedge$
$(\forall\, v_{10}\ v_{97}\ v_{98}.\ P\ (v_{10}\ \texttt{says}\ v_{97}\ \texttt{lt}\ v_{98})) \wedge$
$(\forall\, v_{12}\ v_{13}.\ P\ (v_{12}\ \texttt{speaks\_for}\ v_{13})) \wedge$
$(\forall\, v_{14}\ v_{15}.\ P\ (v_{14}\ \texttt{controls}\ v_{15})) \wedge$
$(\forall\, v_{16}\ v_{17}\ v_{18}.\ P\ (\texttt{reps}\ v_{16}\ v_{17}\ v_{18})) \wedge$
$(\forall\, v_{19}\ v_{20}.\ P\ (v_{19}\ \texttt{domi}\ v_{20})) \wedge$
$(\forall\, v_{21}\ v_{22}.\ P\ (v_{21}\ \texttt{eqi}\ v_{22})) \wedge$
$(\forall\, v_{23}\ v_{24}.\ P\ (v_{23}\ \texttt{doms}\ v_{24})) \wedge$
$(\forall\, v_{25}\ v_{26}.\ P\ (v_{25}\ \texttt{eqs}\ v_{26})) \wedge (\forall\, v_{27}\ v_{28}.\ P\ (v_{27}\ \texttt{eqn}\ v_{28})) \wedge$
$(\forall\, v_{29}\ v_{30}.\ P\ (v_{29}\ \texttt{lte}\ v_{30})) \wedge (\forall\, v_{31}\ v_{32}.\ P\ (v_{31}\ \texttt{lt}\ v_{32})) \Rightarrow$
$\forall\, v.\ P\ v$

[PBNS_def]

$\vdash$ (PBNS PLAN_PB (exec x) =
if getPlCom x = crossLD **then** MOVE_TO_ORP **else** PLAN_PB) $\wedge$
(PBNS MOVE_TO_ORP (exec x) =
if getPlCom x = conductORP **then** CONDUCT_ORP
else MOVE_TO_ORP) $\wedge$
(PBNS CONDUCT_ORP (exec x) =
if getPlCom x = moveToPB **then** MOVE_TO_PB **else** CONDUCT_ORP) $\wedge$
(PBNS MOVE_TO_PB (exec x) =
if getPlCom x = conductPB **then** CONDUCT_PB **else** MOVE_TO_PB) $\wedge$
(PBNS CONDUCT_PB (exec x) =
if getPlCom x = completePB **then** COMPLETE_PB
else CONDUCT_PB) $\wedge$ (PBNS s (trap v_0) = s) $\wedge$
(PBNS s (discard v_1) = s)

[PBNS_ind]

$\vdash \forall P.$
$(\forall x.\ P$ PLAN_PB (exec x)) $\wedge$ $(\forall x.\ P$ MOVE_TO_ORP (exec x)) $\wedge$
$(\forall x.\ P$ CONDUCT_ORP (exec x)) $\wedge$
$(\forall x.\ P$ MOVE_TO_PB (exec x)) $\wedge$ $(\forall x.\ P$ CONDUCT_PB (exec x)) $\wedge$
$(\forall s\ v_0.\ P\ s$ (trap v_0)) $\wedge$ $(\forall s\ v_1.\ P\ s$ (discard v_1)) $\wedge$
$(\forall v_6.\ P$ COMPLETE_PB (exec v_6)) $\Rightarrow$
$\forall v\ v_1.\ P\ v\ v_1$

[PBOut_def]

$\vdash$ (PBOut PLAN_PB (exec x) =
if getPlCom x = crossLD **then** MoveToORP **else** PlanPB) $\wedge$
(PBOut MOVE_TO_ORP (exec x) =
if getPlCom x = conductORP **then** ConductORP **else** MoveToORP) $\wedge$
(PBOut CONDUCT_ORP (exec x) =
if getPlCom x = moveToPB **then** MoveToORP **else** ConductORP) $\wedge$
(PBOut MOVE_TO_PB (exec x) =
if getPlCom x = conductPB **then** ConductPB **else** MoveToPB) $\wedge$

```
(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)) ∧
```

```
    (∀ v5. P COMPLETE (exec v5)) ⇒
    ∀ v v1. P v v1
```

[conductORPOut_def]

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

[conductORPOut_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)) ∧
    (∀ v5. P COMPLETE (exec v5)) ⇒
    ∀ v v1. P v v1
```

[inputOK_cmd_reject_lemma]

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

[inputOK_def]

```
⊢ (inputOK (Name PlatoonLeader says prop cmd) ⟺ T) ∧
  (inputOK (Name PlatoonSergeant says prop cmd) ⟺ T) ∧
  (inputOK (Name Omni 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 (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]

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

$\forall\, v\,.\;\; P\;\; v$

[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]
⊢ ssmSecureComplete ≠ ssmActionsInComplete ∧
  ssmSecureComplete ≠ ssmWithdrawComplete ∧
  ssmSecureComplete ≠ invalidOmniCommand ∧
  ssmActionsInComplete ≠ ssmWithdrawComplete ∧
  ssmActionsInComplete ≠ invalidOmniCommand ∧
  ssmWithdrawComplete ≠ invalidOmniCommand

[plCommand_distinct_clauses]
⊢ secure ≠ withdraw ∧ secure ≠ complete ∧
  secure ≠ plIncomplete ∧ withdraw ≠ complete ∧
  withdraw ≠ plIncomplete ∧ complete ≠ plIncomplete

[psgCommand_distinct_clauses]
⊢ actionsIn ≠ psgIncomplete

[slCommand_distinct_clauses]
⊢ (∀ a′ a. PL a ≠ PSG a′) ∧ (∀ a′ a. PL a ≠ OMNI a′) ∧
  ∀ a′ a. PSG a ≠ OMNI a′

[slCommand_one_one]
⊢ (∀ a a′. (PL a = PL a′) ⟺ (a = a′)) ∧
  (∀ a a′. (PSG a = PSG a′) ⟺ (a = a′)) ∧
  ∀ a a′. (OMNI a = OMNI a′) ⟺ (a = a′)

[slOutput_distinct_clauses]
⊢ ConductORP ≠ Secure ∧ ConductORP ≠ ActionsIn ∧
  ConductORP ≠ Withdraw ∧ ConductORP ≠ Complete ∧
  ConductORP ≠ unAuthenticated ∧ ConductORP ≠ unAuthorized ∧
  Secure ≠ ActionsIn ∧ Secure ≠ Withdraw ∧ Secure ≠ Complete ∧
  Secure ≠ unAuthenticated ∧ Secure ≠ unAuthorized ∧
  ActionsIn ≠ Withdraw ∧ ActionsIn ≠ Complete ∧
  ActionsIn ≠ unAuthenticated ∧ ActionsIn ≠ unAuthorized ∧
  Withdraw ≠ Complete ∧ Withdraw ≠ unAuthenticated ∧
  Withdraw ≠ unAuthorized ∧ Complete ≠ unAuthenticated ∧
  Complete ≠ unAuthorized ∧ unAuthenticated ≠ 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 ssmConductPB Theory

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

10.1 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]

⊢ ∀ *slState*. ssmConductPBStateInterp *slState* = TT

10.2 Theorems

[authTestConductPB_cmd_reject_lemma]

⊢ ∀ *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 v_1) ⟺ F) ∧
(authTestConductPB (v_2 andf v_3) ⟺ F) ∧
(authTestConductPB (v_4 orf v_5) ⟺ F) ∧

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

[authTestConductPB_ind]

⊢ ∀ P.
(∀ *cmd*. P (Name PlatoonLeader says prop *cmd*)) ∧
(∀ *cmd*. P (Name PlatoonSergeant 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})) ∧

$(\forall\, v_{10}\ v_{78}\ v_{79}.\ P\ (v_{10}\ \texttt{says}\ v_{78}\ \texttt{speaks\_for}\ v_{79}))\ \wedge$
$(\forall\, v_{10}\ v_{80}\ v_{81}.\ P\ (v_{10}\ \texttt{says}\ v_{80}\ \texttt{controls}\ v_{81}))\ \wedge$
$(\forall\, v_{10}\ v_{82}\ v_{83}\ v_{84}.\ P\ (v_{10}\ \texttt{says reps}\ v_{82}\ v_{83}\ v_{84}))\ \wedge$
$(\forall\, v_{10}\ v_{85}\ v_{86}.\ P\ (v_{10}\ \texttt{says}\ v_{85}\ \texttt{domi}\ v_{86}))\ \wedge$
$(\forall\, v_{10}\ v_{87}\ v_{88}.\ P\ (v_{10}\ \texttt{says}\ v_{87}\ \texttt{eqi}\ v_{88}))\ \wedge$
$(\forall\, v_{10}\ v_{89}\ v_{90}.\ P\ (v_{10}\ \texttt{says}\ v_{89}\ \texttt{doms}\ v_{90}))\ \wedge$
$(\forall\, v_{10}\ v_{91}\ v_{92}.\ P\ (v_{10}\ \texttt{says}\ v_{91}\ \texttt{eqs}\ v_{92}))\ \wedge$
$(\forall\, v_{10}\ v_{93}\ v_{94}.\ P\ (v_{10}\ \texttt{says}\ v_{93}\ \texttt{eqn}\ v_{94}))\ \wedge$
$(\forall\, v_{10}\ v_{95}\ v_{96}.\ P\ (v_{10}\ \texttt{says}\ v_{95}\ \texttt{lte}\ v_{96}))\ \wedge$
$(\forall\, v_{10}\ v_{97}\ v_{98}.\ P\ (v_{10}\ \texttt{says}\ v_{97}\ \texttt{lt}\ v_{98}))\ \wedge$
$(\forall\, v_{12}\ v_{13}.\ P\ (v_{12}\ \texttt{speaks\_for}\ v_{13}))\ \wedge$
$(\forall\, v_{14}\ v_{15}.\ P\ (v_{14}\ \texttt{controls}\ v_{15}))\ \wedge$
$(\forall\, v_{16}\ v_{17}\ v_{18}.\ P\ (\texttt{reps}\ v_{16}\ v_{17}\ v_{18}))\ \wedge$
$(\forall\, v_{19}\ v_{20}.\ P\ (v_{19}\ \texttt{domi}\ v_{20}))\ \wedge$
$(\forall\, v_{21}\ v_{22}.\ P\ (v_{21}\ \texttt{eqi}\ v_{22}))\ \wedge$
$(\forall\, v_{23}\ v_{24}.\ P\ (v_{23}\ \texttt{doms}\ v_{24}))\ \wedge$
$(\forall\, v_{25}\ v_{26}.\ P\ (v_{25}\ \texttt{eqs}\ v_{26}))\ \wedge\ (\forall\, v_{27}\ v_{28}.\ P\ (v_{27}\ \texttt{eqn}\ v_{28}))\ \wedge$
$(\forall\, v_{29}\ v_{30}.\ P\ (v_{29}\ \texttt{lte}\ v_{30}))\ \wedge\ (\forall\, v_{31}\ v_{32}.\ P\ (v_{31}\ \texttt{lt}\ v_{32}))\ \Rightarrow$
$\forall\, v.\ P\ v$

[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)) ∧
($\forall s\ cmd.\ P\ s$ (trap (PL cmd))) ∧
($\forall s\ cmd.\ P\ s$ (trap (PSG cmd))) ∧
($\forall s\ cmd.\ P\ s$ (discard (PL cmd))) ∧
($\forall s\ cmd.\ P\ s$ (discard (PSG cmd))) ∧
P CONDUCT_PB (exec (PL withdrawPB)) ∧
P CONDUCT_PB (exec (PL completePB)) ∧
($\forall v_{11}.\ P$ CONDUCT_PB (exec (PSG v_{11}))) ∧
($\forall v_{13}.\ P$ SECURE_PB (exec (PL v_{13}))) ∧
P ACTIONS_IN_PB (exec (PL securePB)) ∧
P ACTIONS_IN_PB (exec (PL completePB)) ∧
($\forall v_{17}.\ P$ ACTIONS_IN_PB (exec (PSG v_{17}))) ∧
P WITHDRAW_PB (exec (PL securePB)) ∧
P WITHDRAW_PB (exec (PL withdrawPB)) ∧
($\forall v_{20}.\ P$ WITHDRAW_PB (exec (PSG v_{20}))) ∧
($\forall v_{21}.\ P$ COMPLETE_PB (exec v_{21})) ⇒
$\forall v\ v_1.\ P\ v\ v_1$

[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]

⊢ $\forall P.$
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)))
```

11 ConductPBType Theory

Built: 10 June 2018

Parent Theories: indexedLists, patternMatches

11.1 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
```

11.2 Theorems

[plCommandPB_distinct_clauses]

$\vdash$ securePB $\neq$ withdrawPB $\wedge$ securePB $\neq$ completePB $\wedge$
securePB $\neq$ plIncompletePB $\wedge$ withdrawPB $\neq$ completePB $\wedge$
withdrawPB $\neq$ plIncompletePB $\wedge$ completePB $\neq$ plIncompletePB

[psgCommandPB_distinct_clauses]

$\vdash$ actionsInPB $\neq$ psgIncompletePB

[slCommand_distinct_clauses]

$\vdash \forall a'\ a.$ PL $a \neq$ PSG a'

[slCommand_one_one]

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

[slOutput_distinct_clauses]

$\vdash$ ConductPB $\neq$ SecurePB $\wedge$ ConductPB $\neq$ ActionsInPB $\wedge$
ConductPB $\neq$ WithdrawPB $\wedge$ ConductPB $\neq$ CompletePB $\wedge$
ConductPB $\neq$ unAuthenticated $\wedge$ ConductPB $\neq$ unAuthorized $\wedge$
SecurePB $\neq$ ActionsInPB $\wedge$ SecurePB $\neq$ WithdrawPB $\wedge$
SecurePB $\neq$ CompletePB $\wedge$ SecurePB $\neq$ unAuthenticated $\wedge$
SecurePB $\neq$ unAuthorized $\wedge$ ActionsInPB $\neq$ WithdrawPB $\wedge$
ActionsInPB $\neq$ CompletePB $\wedge$ ActionsInPB $\neq$ unAuthenticated $\wedge$
ActionsInPB $\neq$ unAuthorized $\wedge$ WithdrawPB $\neq$ CompletePB $\wedge$
WithdrawPB $\neq$ unAuthenticated $\wedge$ WithdrawPB $\neq$ unAuthorized $\wedge$
CompletePB $\neq$ unAuthenticated $\wedge$ CompletePB $\neq$ unAuthorized $\wedge$
unAuthenticated $\neq$ unAuthorized

[slRole_distinct_clauses]

$\vdash$ PlatoonLeader $\neq$ PlatoonSergeant

[slState_distinct_clauses]

$\vdash$ CONDUCT_PB $\neq$ SECURE_PB $\wedge$ CONDUCT_PB $\neq$ ACTIONS_IN_PB $\wedge$
CONDUCT_PB $\neq$ WITHDRAW_PB $\wedge$ CONDUCT_PB $\neq$ COMPLETE_PB $\wedge$
SECURE_PB $\neq$ ACTIONS_IN_PB $\wedge$ SECURE_PB $\neq$ WITHDRAW_PB $\wedge$
SECURE_PB $\neq$ COMPLETE_PB $\wedge$ ACTIONS_IN_PB $\neq$ WITHDRAW_PB $\wedge$
ACTIONS_IN_PB $\neq$ COMPLETE_PB $\wedge$ WITHDRAW_PB $\neq$ COMPLETE_PB

12 ssmMoveToORP Theory

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

12.1 Definitions

[secContextMoveToORP_def]

$\vdash \forall cmd.$
secContextMoveToORP cmd =
[Name PlatoonLeader controls prop (SOME (SLc cmd))]

[ssmMoveToORPStateInterp_def]

$\vdash \forall state.$ ssmMoveToORPStateInterp $state$ = TT

12.2 Theorems

[authTestMoveToORP_cmd_reject_lemma]

$\vdash \forall cmd.$ ¬authTestMoveToORP (prop (SOME cmd))

[authTestMoveToORP_def]

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

(authTestMoveToORP (v_{21} eqi v_{22}) ⟺ F) ∧
(authTestMoveToORP (v_{23} doms v_{24}) ⟺ F) ∧
(authTestMoveToORP (v_{25} eqs v_{26}) ⟺ F) ∧
(authTestMoveToORP (v_{27} eqn v_{28}) ⟺ F) ∧
(authTestMoveToORP (v_{29} lte v_{30}) ⟺ F) ∧
(authTestMoveToORP (v_{31} lt v_{32}) ⟺ F)

[authTestMoveToORP_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

[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 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
```

[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))
```

13 MoveToORPType Theory

Built: 10 June 2018
Parent Theories: indexedLists, patternMatches

13.1 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
```

13.2 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

14 ssmMoveToPB Theory

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

14.1 Definitions

[secContextMoveToPB_def]

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

[ssmMoveToPBStateInterp_def]

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

14.2 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})) ∧

$(\forall v_{10}\ v_{78}\ v_{79}.\ P\ (v_{10}\ \texttt{says}\ v_{78}\ \texttt{speaks\_for}\ v_{79}))\ \wedge$
$(\forall v_{10}\ v_{80}\ v_{81}.\ P\ (v_{10}\ \texttt{says}\ v_{80}\ \texttt{controls}\ v_{81}))\ \wedge$
$(\forall v_{10}\ v_{82}\ v_{83}\ v_{84}.\ P\ (v_{10}\ \texttt{says reps}\ v_{82}\ v_{83}\ v_{84}))\ \wedge$
$(\forall v_{10}\ v_{85}\ v_{86}.\ P\ (v_{10}\ \texttt{says}\ v_{85}\ \texttt{domi}\ v_{86}))\ \wedge$
$(\forall v_{10}\ v_{87}\ v_{88}.\ P\ (v_{10}\ \texttt{says}\ v_{87}\ \texttt{eqi}\ v_{88}))\ \wedge$
$(\forall v_{10}\ v_{89}\ v_{90}.\ P\ (v_{10}\ \texttt{says}\ v_{89}\ \texttt{doms}\ v_{90}))\ \wedge$
$(\forall v_{10}\ v_{91}\ v_{92}.\ P\ (v_{10}\ \texttt{says}\ v_{91}\ \texttt{eqs}\ v_{92}))\ \wedge$
$(\forall v_{10}\ v_{93}\ v_{94}.\ P\ (v_{10}\ \texttt{says}\ v_{93}\ \texttt{eqn}\ v_{94}))\ \wedge$
$(\forall v_{10}\ v_{95}\ v_{96}.\ P\ (v_{10}\ \texttt{says}\ v_{95}\ \texttt{lte}\ v_{96}))\ \wedge$
$(\forall v_{10}\ v_{97}\ v_{98}.\ P\ (v_{10}\ \texttt{says}\ v_{97}\ \texttt{lt}\ v_{98}))\ \wedge$
$(\forall v_{12}\ v_{13}.\ P\ (v_{12}\ \texttt{speaks\_for}\ v_{13}))\ \wedge$
$(\forall v_{14}\ v_{15}.\ P\ (v_{14}\ \texttt{controls}\ v_{15}))\ \wedge$
$(\forall v_{16}\ v_{17}\ v_{18}.\ P\ (\texttt{reps}\ v_{16}\ v_{17}\ v_{18}))\ \wedge$
$(\forall v_{19}\ v_{20}.\ P\ (v_{19}\ \texttt{domi}\ v_{20}))\ \wedge$
$(\forall v_{21}\ v_{22}.\ P\ (v_{21}\ \texttt{eqi}\ v_{22}))\ \wedge$
$(\forall v_{23}\ v_{24}.\ P\ (v_{23}\ \texttt{doms}\ v_{24}))\ \wedge$
$(\forall v_{25}\ v_{26}.\ P\ (v_{25}\ \texttt{eqs}\ v_{26}))\ \wedge\ (\forall v_{27}\ v_{28}.\ P\ (v_{27}\ \texttt{eqn}\ v_{28}))\ \wedge$
$(\forall v_{29}\ v_{30}.\ P\ (v_{29}\ \texttt{lte}\ v_{30}))\ \wedge\ (\forall v_{31}\ v_{32}.\ P\ (v_{31}\ \texttt{lt}\ v_{32}))\ \Rightarrow$
$\forall v.\ P\ v$

[moveToPBNS_def]

$\vdash$ (moveToPBNS MOVE_TO_PB (exec (SLc pltForm)) = PLT_FORM) $\wedge$
(moveToPBNS MOVE_TO_PB (exec (SLc incomplete)) =
MOVE_TO_PB) $\wedge$
(moveToPBNS PLT_FORM (exec (SLc pltMove)) = PLT_MOVE) $\wedge$
(moveToPBNS PLT_FORM (exec (SLc incomplete)) = PLT_FORM) $\wedge$
(moveToPBNS PLT_MOVE (exec (SLc pltHalt)) = PLT_HALT) $\wedge$
(moveToPBNS PLT_MOVE (exec (SLc incomplete)) = PLT_MOVE) $\wedge$
(moveToPBNS PLT_HALT (exec (SLc complete)) = COMPLETE) $\wedge$
(moveToPBNS PLT_HALT (exec (SLc incomplete)) = PLT_HALT) $\wedge$
(moveToPBNS s (trap (SLc cmd)) = s) $\wedge$
(moveToPBNS s (discard (SLc cmd)) = s)

[moveToPBNS_ind]

$\vdash \forall P.$
P MOVE_TO_PB (exec (SLc pltForm)) $\wedge$
P MOVE_TO_PB (exec (SLc incomplete)) $\wedge$
P PLT_FORM (exec (SLc pltMove)) $\wedge$
P PLT_FORM (exec (SLc incomplete)) $\wedge$
P PLT_MOVE (exec (SLc pltHalt)) $\wedge$
P PLT_MOVE (exec (SLc incomplete)) $\wedge$
P PLT_HALT (exec (SLc complete)) $\wedge$
P PLT_HALT (exec (SLc incomplete)) $\wedge$
$(\forall s\ cmd.\ P\ s\ (\texttt{trap}\ (\texttt{SLc}\ cmd)))\ \wedge$
$(\forall s\ cmd.\ P\ s\ (\texttt{discard}\ (\texttt{SLc}\ cmd)))\ \wedge$
$(\forall s\ v_6.\ P\ s\ (\texttt{discard}\ (\texttt{ESCc}\ v_6)))\ \wedge$
$(\forall s\ v_9.\ P\ s\ (\texttt{trap}\ (\texttt{ESCc}\ v_9)))\ \wedge$
$(\forall v_{12}.\ P$ MOVE_TO_PB (exec (ESCc v_{12}))) $\wedge$
P MOVE_TO_PB (exec (SLc pltMove)) $\wedge$

P MOVE_TO_PB (exec (SLc pltHalt)) ∧
P MOVE_TO_PB (exec (SLc complete)) ∧
(∀ v_{15}. P PLT_FORM (exec (ESCc v_{15}))) ∧
P PLT_FORM (exec (SLc pltForm)) ∧
P PLT_FORM (exec (SLc pltHalt)) ∧
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)) ∧
(∀ v_{21}. P PLT_HALT (exec (ESCc v_{21}))) ∧
P PLT_HALT (exec (SLc pltForm)) ∧
P PLT_HALT (exec (SLc pltMove)) ∧
P PLT_HALT (exec (SLc pltHalt)) ∧
(∀ v_{23}. P COMPLETE (exec v_{23})) ⇒
∀ v v_1. P v v_1

[moveToPBOut_def]

⊢ (moveToPBOut MOVE_TO_PB (exec (SLc pltForm)) = PLTForm) ∧
(moveToPBOut MOVE_TO_PB (exec (SLc incomplete)) = MoveToPB) ∧
(moveToPBOut PLT_FORM (exec (SLc pltMove)) = PLTMove) ∧
(moveToPBOut PLT_FORM (exec (SLc incomplete)) = PLTForm) ∧
(moveToPBOut PLT_MOVE (exec (SLc pltHalt)) = PLTHalt) ∧
(moveToPBOut PLT_MOVE (exec (SLc incomplete)) = PLTMove) ∧
(moveToPBOut PLT_HALT (exec (SLc complete)) = Complete) ∧
(moveToPBOut PLT_HALT (exec (SLc incomplete)) = PLTHalt) ∧
(moveToPBOut s (trap (SLc cmd)) = unAuthorized) ∧
(moveToPBOut s (discard (SLc cmd)) = unAuthenticated)

[moveToPBOut_ind]

⊢ ∀ P.
P MOVE_TO_PB (exec (SLc pltForm)) ∧
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 v_6. P s (discard (ESCc v_6))) ∧
(∀ s v_9. P s (trap (ESCc v_9))) ∧
(∀ v_{12}. P MOVE_TO_PB (exec (ESCc v_{12}))) ∧
P MOVE_TO_PB (exec (SLc pltMove)) ∧
P MOVE_TO_PB (exec (SLc pltHalt)) ∧
P MOVE_TO_PB (exec (SLc complete)) ∧
(∀ v_{15}. P PLT_FORM (exec (ESCc v_{15}))) ∧
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))
```

15 MoveToPBType Theory

Built: 10 June 2018

Parent Theories: indexedLists, patternMatches

15.1 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

15.2 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

16 ssmPlanPB Theory

Built: 10 June 2018
Parent Theories: PlanPBDef, ssm

16.1 Theorems

[inputOK_def]

$\vdash$ (inputOK (Name PlatoonLeader says prop cmd) $\iff$ T) $\wedge$
(inputOK (Name PlatoonSergeant 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) $\wedge$
(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 PlatoonSergeant 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}))) $\wedge$

$(\forall\, v_{10}\ v_{70}\ v_{71}.\ P\ (v_{10}\ \texttt{says}\ (v_{70}\ \texttt{orf}\ v_{71}))) \wedge$
$(\forall\, v_{10}\ v_{72}\ v_{73}.\ P\ (v_{10}\ \texttt{says}\ (v_{72}\ \texttt{impf}\ v_{73}))) \wedge$
$(\forall\, v_{10}\ v_{74}\ v_{75}.\ P\ (v_{10}\ \texttt{says}\ (v_{74}\ \texttt{eqf}\ v_{75}))) \wedge$
$(\forall\, v_{10}\ v_{76}\ v_{77}.\ P\ (v_{10}\ \texttt{says}\ v_{76}\ \texttt{says}\ v_{77})) \wedge$
$(\forall\, v_{10}\ v_{78}\ v_{79}.\ P\ (v_{10}\ \texttt{says}\ v_{78}\ \texttt{speaks\_for}\ v_{79})) \wedge$
$(\forall\, v_{10}\ v_{80}\ v_{81}.\ P\ (v_{10}\ \texttt{says}\ v_{80}\ \texttt{controls}\ v_{81})) \wedge$
$(\forall\, v_{10}\ v_{82}\ v_{83}\ v_{84}.\ P\ (v_{10}\ \texttt{says}\ \texttt{reps}\ v_{82}\ v_{83}\ v_{84})) \wedge$
$(\forall\, v_{10}\ v_{85}\ v_{86}.\ P\ (v_{10}\ \texttt{says}\ v_{85}\ \texttt{domi}\ v_{86})) \wedge$
$(\forall\, v_{10}\ v_{87}\ v_{88}.\ P\ (v_{10}\ \texttt{says}\ v_{87}\ \texttt{eqi}\ v_{88})) \wedge$
$(\forall\, v_{10}\ v_{89}\ v_{90}.\ P\ (v_{10}\ \texttt{says}\ v_{89}\ \texttt{doms}\ v_{90})) \wedge$
$(\forall\, v_{10}\ v_{91}\ v_{92}.\ P\ (v_{10}\ \texttt{says}\ v_{91}\ \texttt{eqs}\ v_{92})) \wedge$
$(\forall\, v_{10}\ v_{93}\ v_{94}.\ P\ (v_{10}\ \texttt{says}\ v_{93}\ \texttt{eqn}\ v_{94})) \wedge$
$(\forall\, v_{10}\ v_{95}\ v_{96}.\ P\ (v_{10}\ \texttt{says}\ v_{95}\ \texttt{lte}\ v_{96})) \wedge$
$(\forall\, v_{10}\ v_{97}\ v_{98}.\ P\ (v_{10}\ \texttt{says}\ v_{97}\ \texttt{lt}\ v_{98})) \wedge$
$(\forall\, v_{12}\ v_{13}.\ P\ (v_{12}\ \texttt{speaks\_for}\ v_{13})) \wedge$
$(\forall\, v_{14}\ v_{15}.\ P\ (v_{14}\ \texttt{controls}\ v_{15})) \wedge$
$(\forall\, v_{16}\ v_{17}\ v_{18}.\ P\ (\texttt{reps}\ v_{16}\ v_{17}\ v_{18})) \wedge$
$(\forall\, v_{19}\ v_{20}.\ P\ (v_{19}\ \texttt{domi}\ v_{20})) \wedge$
$(\forall\, v_{21}\ v_{22}.\ P\ (v_{21}\ \texttt{eqi}\ v_{22})) \wedge$
$(\forall\, v_{23}\ v_{24}.\ P\ (v_{23}\ \texttt{doms}\ v_{24})) \wedge$
$(\forall\, v_{25}\ v_{26}.\ P\ (v_{25}\ \texttt{eqs}\ v_{26})) \wedge (\forall\, v_{27}\ v_{28}.\ P\ (v_{27}\ \texttt{eqn}\ v_{28})) \wedge$
$(\forall\, v_{29}\ v_{30}.\ P\ (v_{29}\ \texttt{lte}\ v_{30})) \wedge (\forall\, v_{31}\ v_{32}.\ P\ (v_{31}\ \texttt{lt}\ v_{32})) \Rightarrow$
$\forall\, v.\ P\ v$

[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
```

17 PlanPBType Theory

Built: 10 June 2018

Parent Theories: indexedLists, patternMatches

17.1 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

17.2 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
```

18 PlanPBDef Theory

Built: 10 June 2018

Parent Theories: PlanPBType, aclfoundation, OMNIType

18.1 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]
```

18.2 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))::
```

$$
\begin{array}{l}
\quad\quad xs)) \wedge \\
(\forall\, v150\ xs. \\
\quad P\ xs \Rightarrow \\
\quad P \\
\quad\quad (\texttt{Name PlatoonSergeant says} \\
\quad\quad\ \texttt{prop (SOME (SLc (PL } v150\texttt{)))::}xs)) \wedge \\
(\forall\, xs. \\
\quad P\ xs \Rightarrow \\
\quad P \\
\quad\quad (\texttt{Name PlatoonSergeant says} \\
\quad\quad\ \texttt{prop (SOME (SLc (PSG psgIncomplete)))::}xs)) \wedge \\
(\forall\, xs. \\
\quad P\ xs \Rightarrow \\
\quad P \\
\quad\quad (\texttt{Name PlatoonSergeant says} \\
\quad\quad\ \texttt{prop (SOME (SLc (PSG invalidPsgCommand)))::}xs)) \wedge \\
(\forall\, v135\ v136\ v_{68}\ xs. \\
\quad P\ xs \Rightarrow P\ (v135\ \texttt{meet}\ v136\ \texttt{says prop}\ v_{68}\texttt{::}xs)) \wedge \\
(\forall\, v137\ v138\ v_{68}\ xs. \\
\quad P\ xs \Rightarrow P\ (v137\ \texttt{quoting}\ v138\ \texttt{says prop}\ v_{68}\texttt{::}xs)) \wedge \\
(\forall\, v_{12}\ v_{69}\ xs.\ P\ xs \Rightarrow P\ (v_{12}\ \texttt{says notf}\ v_{69}\texttt{::}xs)) \wedge \\
(\forall\, v_{12}\ v_{70}\ v_{71}\ xs.\ P\ xs \Rightarrow P\ (v_{12}\ \texttt{says (}v_{70}\ \texttt{andf}\ v_{71}\texttt{)::}xs)) \wedge \\
(\forall\, v_{12}\ v_{72}\ v_{73}\ xs.\ P\ xs \Rightarrow P\ (v_{12}\ \texttt{says (}v_{72}\ \texttt{orf}\ v_{73}\texttt{)::}xs)) \wedge \\
(\forall\, v_{12}\ v_{74}\ v_{75}\ xs.\ P\ xs \Rightarrow P\ (v_{12}\ \texttt{says (}v_{74}\ \texttt{impf}\ v_{75}\texttt{)::}xs)) \wedge \\
(\forall\, v_{12}\ v_{76}\ v_{77}\ xs.\ P\ xs \Rightarrow P\ (v_{12}\ \texttt{says (}v_{76}\ \texttt{eqf}\ v_{77}\texttt{)::}xs)) \wedge \\
(\forall\, v_{12}\ v_{78}\ v_{79}\ xs.\ P\ xs \Rightarrow P\ (v_{12}\ \texttt{says}\ v_{78}\ \texttt{says}\ v_{79}\texttt{::}xs)) \wedge \\
(\forall\, v_{12}\ v_{80}\ v_{81}\ xs. \\
\quad P\ xs \Rightarrow P\ (v_{12}\ \texttt{says}\ v_{80}\ \texttt{speaks\_for}\ v_{81}\texttt{::}xs)) \wedge \\
(\forall\, v_{12}\ v_{82}\ v_{83}\ xs. \\
\quad P\ xs \Rightarrow P\ (v_{12}\ \texttt{says}\ v_{82}\ \texttt{controls}\ v_{83}\texttt{::}xs)) \wedge \\
(\forall\, v_{12}\ v_{84}\ v_{85}\ v_{86}\ xs. \\
\quad P\ xs \Rightarrow P\ (v_{12}\ \texttt{says reps}\ v_{84}\ v_{85}\ v_{86}\texttt{::}xs)) \wedge \\
(\forall\, v_{12}\ v_{87}\ v_{88}\ xs.\ P\ xs \Rightarrow P\ (v_{12}\ \texttt{says}\ v_{87}\ \texttt{domi}\ v_{88}\texttt{::}xs)) \wedge \\
(\forall\, v_{12}\ v_{89}\ v_{90}\ xs.\ P\ xs \Rightarrow P\ (v_{12}\ \texttt{says}\ v_{89}\ \texttt{eqi}\ v_{90}\texttt{::}xs)) \wedge \\
(\forall\, v_{12}\ v_{91}\ v_{92}\ xs.\ P\ xs \Rightarrow P\ (v_{12}\ \texttt{says}\ v_{91}\ \texttt{doms}\ v_{92}\texttt{::}xs)) \wedge \\
(\forall\, v_{12}\ v_{93}\ v_{94}\ xs.\ P\ xs \Rightarrow P\ (v_{12}\ \texttt{says}\ v_{93}\ \texttt{eqs}\ v_{94}\texttt{::}xs)) \wedge \\
(\forall\, v_{12}\ v_{95}\ v_{96}\ xs.\ P\ xs \Rightarrow P\ (v_{12}\ \texttt{says}\ v_{95}\ \texttt{eqn}\ v_{96}\texttt{::}xs)) \wedge \\
(\forall\, v_{12}\ v_{97}\ v_{98}\ xs.\ P\ xs \Rightarrow P\ (v_{12}\ \texttt{says}\ v_{97}\ \texttt{lte}\ v_{98}\texttt{::}xs)) \wedge \\
(\forall\, v_{12}\ v_{99}\ v100\ xs.\ P\ xs \Rightarrow P\ (v_{12}\ \texttt{says}\ v_{99}\ \texttt{lt}\ v100\texttt{::}xs)) \wedge \\
(\forall\, v_{14}\ v_{15}\ xs.\ P\ xs \Rightarrow P\ (v_{14}\ \texttt{speaks\_for}\ v_{15}\texttt{::}xs)) \wedge \\
(\forall\, v_{16}\ v_{17}\ xs.\ P\ xs \Rightarrow P\ (v_{16}\ \texttt{controls}\ v_{17}\texttt{::}xs)) \wedge \\
(\forall\, v_{18}\ v_{19}\ v_{20}\ xs.\ P\ xs \Rightarrow P\ (\texttt{reps}\ v_{18}\ v_{19}\ v_{20}\texttt{::}xs)) \wedge \\
(\forall\, v_{21}\ v_{22}\ xs.\ P\ xs \Rightarrow P\ (v_{21}\ \texttt{domi}\ v_{22}\texttt{::}xs)) \wedge \\
(\forall\, v_{23}\ v_{24}\ xs.\ P\ xs \Rightarrow P\ (v_{23}\ \texttt{eqi}\ v_{24}\texttt{::}xs)) \wedge \\
(\forall\, v_{25}\ v_{26}\ xs.\ P\ xs \Rightarrow P\ (v_{25}\ \texttt{doms}\ v_{26}\texttt{::}xs)) \wedge \\
(\forall\, v_{27}\ v_{28}\ xs.\ P\ xs \Rightarrow P\ (v_{27}\ \texttt{eqs}\ v_{28}\texttt{::}xs)) \wedge \\
(\forall\, v_{29}\ v_{30}\ xs.\ P\ xs \Rightarrow P\ (v_{29}\ \texttt{eqn}\ v_{30}\texttt{::}xs)) \wedge \\
(\forall\, v_{31}\ v_{32}\ xs.\ P\ xs \Rightarrow P\ (v_{31}\ \texttt{lte}\ v_{32}\texttt{::}xs)) \wedge \\
(\forall\, v_{33}\ v_{34}\ xs.\ P\ xs \Rightarrow P\ (v_{33}\ \texttt{lt}\ v_{34}\texttt{::}xs)) \Rightarrow
\end{array}
$$

$\forall v.\ P\ v$

[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) ∧
  (∀xs v68 v138 v137.
     getPlCom (v137 quoting v138 says prop v68::xs) =
     getPlCom xs) ∧
  (∀xs v69 v12.
     getPlCom (v12 says notf v69::xs) = getPlCom xs) ∧
  (∀xs v71 v70 v12.
     getPlCom (v12 says (v70 andf v71)::xs) = getPlCom xs) ∧
  (∀xs v73 v72 v12.
     getPlCom (v12 says (v72 orf v73)::xs) = getPlCom xs) ∧
  (∀xs v75 v74 v12.
     getPlCom (v12 says (v74 impf v75)::xs) = getPlCom xs) ∧
  (∀xs v77 v76 v12.
     getPlCom (v12 says (v76 eqf v77)::xs) = getPlCom xs) ∧
```

$(\forall\, xs\ v_{79}\ v_{78}\ v_{12}.$
$\quad$ getPlCom $(v_{12}$ says v_{78} says v_{79}::$xs)$ = getPlCom $xs)\ \wedge$
$(\forall\, xs\ v_{81}\ v_{80}\ v_{12}.$
$\quad$ getPlCom $(v_{12}$ says v_{80} speaks_for v_{81}::$xs)$ =
$\quad$ getPlCom $xs)\ \wedge$
$(\forall\, xs\ v_{83}\ v_{82}\ v_{12}.$
$\quad$ getPlCom $(v_{12}$ says v_{82} controls v_{83}::$xs)$ = getPlCom $xs)\ \wedge$
$(\forall\, xs\ v_{86}\ v_{85}\ v_{84}\ v_{12}.$
$\quad$ getPlCom $(v_{12}$ says reps $v_{84}\ v_{85}\ v_{86}$::$xs)$ = getPlCom $xs)\ \wedge$
$(\forall\, xs\ v_{88}\ v_{87}\ v_{12}.$
$\quad$ getPlCom $(v_{12}$ says v_{87} domi v_{88}::$xs)$ = getPlCom $xs)\ \wedge$
$(\forall\, xs\ v_{90}\ v_{89}\ v_{12}.$
$\quad$ getPlCom $(v_{12}$ says v_{89} eqi v_{90}::$xs)$ = getPlCom $xs)\ \wedge$
$(\forall\, xs\ v_{92}\ v_{91}\ v_{12}.$
$\quad$ getPlCom $(v_{12}$ says v_{91} doms v_{92}::$xs)$ = getPlCom $xs)\ \wedge$
$(\forall\, xs\ v_{94}\ v_{93}\ v_{12}.$
$\quad$ getPlCom $(v_{12}$ says v_{93} eqs v_{94}::$xs)$ = getPlCom $xs)\ \wedge$
$(\forall\, xs\ v_{96}\ v_{95}\ v_{12}.$
$\quad$ getPlCom $(v_{12}$ says v_{95} eqn v_{96}::$xs)$ = getPlCom $xs)\ \wedge$
$(\forall\, xs\ v_{98}\ v_{97}\ v_{12}.$
$\quad$ getPlCom $(v_{12}$ says v_{97} lte v_{98}::$xs)$ = getPlCom $xs)\ \wedge$
$(\forall\, xs\ v_{99}\ v_{12}\ v100.$
$\quad$ getPlCom $(v_{12}$ says v_{99} lt $v100$::$xs)$ = getPlCom $xs)\ \wedge$
$(\forall\, xs\ v_{15}\ v_{14}.$
$\quad$ getPlCom $(v_{14}$ speaks_for v_{15}::$xs)$ = getPlCom $xs)\ \wedge$
$(\forall\, xs\ v_{17}\ v_{16}.$
$\quad$ getPlCom $(v_{16}$ controls v_{17}::$xs)$ = getPlCom $xs)\ \wedge$
$(\forall\, xs\ v_{20}\ v_{19}\ v_{18}.$
$\quad$ getPlCom (reps $v_{18}\ v_{19}\ v_{20}$::$xs)$ = getPlCom $xs)\ \wedge$
$(\forall\, xs\ v_{22}\ v_{21}.$ getPlCom $(v_{21}$ domi v_{22}::$xs)$ = getPlCom $xs)\ \wedge$
$(\forall\, xs\ v_{24}\ v_{23}.$ getPlCom $(v_{23}$ eqi v_{24}::$xs)$ = getPlCom $xs)\ \wedge$
$(\forall\, xs\ v_{26}\ v_{25}.$ getPlCom $(v_{25}$ doms v_{26}::$xs)$ = getPlCom $xs)\ \wedge$
$(\forall\, xs\ v_{28}\ v_{27}.$ getPlCom $(v_{27}$ eqs v_{28}::$xs)$ = getPlCom $xs)\ \wedge$
$(\forall\, xs\ v_{30}\ v_{29}.$ getPlCom $(v_{29}$ eqn v_{30}::$xs)$ = getPlCom $xs)\ \wedge$
$(\forall\, xs\ v_{32}\ v_{31}.$ getPlCom $(v_{31}$ lte v_{32}::$xs)$ = getPlCom $xs)\ \wedge$
$\forall\, xs\ v_{34}\ v_{33}.$ getPlCom $(v_{33}$ lt v_{34}::$xs)$ = getPlCom xs

[getPlCom_ind]

$\vdash\ \forall P.$
$\quad P$ [] $\wedge$
$\quad(\forall\, cmd\ xs.$
$\quad\quad P$
$\quad\quad$ (Name PlatoonLeader says prop (SOME (SLc (PL cmd)))::
$\quad\quad\quad xs)) \wedge (\forall xs.\ P\ xs \Rightarrow P$ (TT::xs)) $\wedge$
$\quad(\forall xs.\ P\ xs \Rightarrow P$ (FF::xs)) $\wedge$
$\quad(\forall v_2\ xs.\ P\ xs \Rightarrow P$ (prop v_2::xs)) $\wedge$
$\quad(\forall v_3\ xs.\ P\ xs \Rightarrow P$ (notf v_3::xs)) $\wedge$
$\quad(\forall v_4\ v_5\ xs.\ P\ xs \Rightarrow P\ (v_4$ andf v_5::xs)) $\wedge$
$\quad(\forall v_6\ v_7\ xs.\ P\ xs \Rightarrow P\ (v_6$ orf v_7::xs)) $\wedge$

$(\forall\, v_8\ v_9\ xs.\ P\ xs \Rightarrow P\ (v_8\ \texttt{impf}\ v_9\texttt{::}xs)) \wedge$
$(\forall\, v_{10}\ v_{11}\ xs.\ P\ xs \Rightarrow P\ (v_{10}\ \texttt{eqf}\ v_{11}\texttt{::}xs)) \wedge$
$(\forall\, v_{12}\ xs.\ P\ xs \Rightarrow P\ (v_{12}\ \texttt{says TT::}xs)) \wedge$
$(\forall\, v_{12}\ xs.\ P\ xs \Rightarrow P\ (v_{12}\ \texttt{says FF::}xs)) \wedge$
$(\forall\, v134\ xs.\ P\ xs \Rightarrow P\ (\texttt{Name}\ v134\ \texttt{says prop NONE::}xs)) \wedge$
$(\forall\, v146\ xs.$
$\quad P\ xs \Rightarrow$
$\quad P$
$\quad\quad (\texttt{Name PlatoonLeader says prop (SOME (ESCc}\ v146\texttt{))::}$
$\quad\quad\quad xs)) \wedge$
$(\forall\, v151\ xs.$
$\quad P\ xs \Rightarrow$
$\quad P$
$\quad\quad (\texttt{Name PlatoonLeader says}$
$\quad\quad\ \texttt{prop (SOME (SLc (PSG}\ v151\texttt{)))::}xs)) \wedge$
$(\forall\, v144\ xs.$
$\quad P\ xs \Rightarrow$
$\quad P\ (\texttt{Name PlatoonSergeant says prop (SOME}\ v144\texttt{)::}xs)) \wedge$
$(\forall\, v135\ v136\ v_{68}\ xs.$
$\quad P\ xs \Rightarrow P\ (v135\ \texttt{meet}\ v136\ \texttt{says prop}\ v_{68}\texttt{::}xs)) \wedge$
$(\forall\, v137\ v138\ v_{68}\ xs.$
$\quad P\ xs \Rightarrow P\ (v137\ \texttt{quoting}\ v138\ \texttt{says prop}\ v_{68}\texttt{::}xs)) \wedge$
$(\forall\, v_{12}\ v_{69}\ xs.\ P\ xs \Rightarrow P\ (v_{12}\ \texttt{says notf}\ v_{69}\texttt{::}xs)) \wedge$
$(\forall\, v_{12}\ v_{70}\ v_{71}\ xs.\ P\ xs \Rightarrow P\ (v_{12}\ \texttt{says (}v_{70}\ \texttt{andf}\ v_{71}\texttt{)::}xs)) \wedge$
$(\forall\, v_{12}\ v_{72}\ v_{73}\ xs.\ P\ xs \Rightarrow P\ (v_{12}\ \texttt{says (}v_{72}\ \texttt{orf}\ v_{73}\texttt{)::}xs)) \wedge$
$(\forall\, v_{12}\ v_{74}\ v_{75}\ xs.\ P\ xs \Rightarrow P\ (v_{12}\ \texttt{says (}v_{74}\ \texttt{impf}\ v_{75}\texttt{)::}xs)) \wedge$
$(\forall\, v_{12}\ v_{76}\ v_{77}\ xs.\ P\ xs \Rightarrow P\ (v_{12}\ \texttt{says (}v_{76}\ \texttt{eqf}\ v_{77}\texttt{)::}xs)) \wedge$
$(\forall\, v_{12}\ v_{78}\ v_{79}\ xs.\ P\ xs \Rightarrow P\ (v_{12}\ \texttt{says}\ v_{78}\ \texttt{says}\ v_{79}\texttt{::}xs)) \wedge$
$(\forall\, v_{12}\ v_{80}\ v_{81}\ xs.$
$\quad P\ xs \Rightarrow P\ (v_{12}\ \texttt{says}\ v_{80}\ \texttt{speaks\_for}\ v_{81}\texttt{::}xs)) \wedge$
$(\forall\, v_{12}\ v_{82}\ v_{83}\ xs.$
$\quad P\ xs \Rightarrow P\ (v_{12}\ \texttt{says}\ v_{82}\ \texttt{controls}\ v_{83}\texttt{::}xs)) \wedge$
$(\forall\, v_{12}\ v_{84}\ v_{85}\ v_{86}\ xs.$
$\quad P\ xs \Rightarrow P\ (v_{12}\ \texttt{says reps}\ v_{84}\ v_{85}\ v_{86}\texttt{::}xs)) \wedge$
$(\forall\, v_{12}\ v_{87}\ v_{88}\ xs.\ P\ xs \Rightarrow P\ (v_{12}\ \texttt{says}\ v_{87}\ \texttt{domi}\ v_{88}\texttt{::}xs)) \wedge$
$(\forall\, v_{12}\ v_{89}\ v_{90}\ xs.\ P\ xs \Rightarrow P\ (v_{12}\ \texttt{says}\ v_{89}\ \texttt{eqi}\ v_{90}\texttt{::}xs)) \wedge$
$(\forall\, v_{12}\ v_{91}\ v_{92}\ xs.\ P\ xs \Rightarrow P\ (v_{12}\ \texttt{says}\ v_{91}\ \texttt{doms}\ v_{92}\texttt{::}xs)) \wedge$
$(\forall\, v_{12}\ v_{93}\ v_{94}\ xs.\ P\ xs \Rightarrow P\ (v_{12}\ \texttt{says}\ v_{93}\ \texttt{eqs}\ v_{94}\texttt{::}xs)) \wedge$
$(\forall\, v_{12}\ v_{95}\ v_{96}\ xs.\ P\ xs \Rightarrow P\ (v_{12}\ \texttt{says}\ v_{95}\ \texttt{eqn}\ v_{96}\texttt{::}xs)) \wedge$
$(\forall\, v_{12}\ v_{97}\ v_{98}\ xs.\ P\ xs \Rightarrow P\ (v_{12}\ \texttt{says}\ v_{97}\ \texttt{lte}\ v_{98}\texttt{::}xs)) \wedge$
$(\forall\, v_{12}\ v_{99}\ v100\ xs.\ P\ xs \Rightarrow P\ (v_{12}\ \texttt{says}\ v_{99}\ \texttt{lt}\ v100\texttt{::}xs)) \wedge$
$(\forall\, v_{14}\ v_{15}\ xs.\ P\ xs \Rightarrow P\ (v_{14}\ \texttt{speaks\_for}\ v_{15}\texttt{::}xs)) \wedge$
$(\forall\, v_{16}\ v_{17}\ xs.\ P\ xs \Rightarrow P\ (v_{16}\ \texttt{controls}\ v_{17}\texttt{::}xs)) \wedge$
$(\forall\, v_{18}\ v_{19}\ v_{20}\ xs.\ P\ xs \Rightarrow P\ (\texttt{reps}\ v_{18}\ v_{19}\ v_{20}\texttt{::}xs)) \wedge$
$(\forall\, v_{21}\ v_{22}\ xs.\ P\ xs \Rightarrow P\ (v_{21}\ \texttt{domi}\ v_{22}\texttt{::}xs)) \wedge$
$(\forall\, v_{23}\ v_{24}\ xs.\ P\ xs \Rightarrow P\ (v_{23}\ \texttt{eqi}\ v_{24}\texttt{::}xs)) \wedge$
$(\forall\, v_{25}\ v_{26}\ xs.\ P\ xs \Rightarrow P\ (v_{25}\ \texttt{doms}\ v_{26}\texttt{::}xs)) \wedge$
$(\forall\, v_{27}\ v_{28}\ xs.\ P\ xs \Rightarrow P\ (v_{27}\ \texttt{eqs}\ v_{28}\texttt{::}xs)) \wedge$
$(\forall\, v_{29}\ v_{30}\ xs.\ P\ xs \Rightarrow P\ (v_{29}\ \texttt{eqn}\ v_{30}\texttt{::}xs)) \wedge$

```
   (∀ 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) ∧
  (∀ xs v68 v138 v137.
     getPsgCom (v137 quoting v138 says prop v68::xs) =
     getPsgCom xs) ∧
  (∀ xs v69 v12.
     getPsgCom (v12 says notf v69::xs) = getPsgCom xs) ∧
  (∀ xs v71 v70 v12.
     getPsgCom (v12 says (v70 andf v71)::xs) = getPsgCom xs) ∧
  (∀ xs v73 v72 v12.
     getPsgCom (v12 says (v72 orf v73)::xs) = getPsgCom xs) ∧
  (∀ xs v75 v74 v12.
     getPsgCom (v12 says (v74 impf v75)::xs) = getPsgCom xs) ∧
```

```
(∀ xs v77 v76 v12.
   getPsgCom (v12 says (v76 eqf v77)::xs) = getPsgCom xs) ∧
(∀ xs v79 v78 v12.
   getPsgCom (v12 says v78 says v79::xs) = getPsgCom xs) ∧
(∀ xs v81 v80 v12.
   getPsgCom (v12 says v80 speaks_for v81::xs) =
   getPsgCom xs) ∧
(∀ xs v83 v82 v12.
   getPsgCom (v12 says v82 controls v83::xs) =
   getPsgCom xs) ∧
(∀ xs v86 v85 v84 v12.
   getPsgCom (v12 says reps v84 v85 v86::xs) =
   getPsgCom xs) ∧
(∀ xs v88 v87 v12.
   getPsgCom (v12 says v87 domi v88::xs) = getPsgCom xs) ∧
(∀ xs v90 v89 v12.
   getPsgCom (v12 says v89 eqi v90::xs) = getPsgCom xs) ∧
(∀ xs v92 v91 v12.
   getPsgCom (v12 says v91 doms v92::xs) = getPsgCom xs) ∧
(∀ xs v94 v93 v12.
   getPsgCom (v12 says v93 eqs v94::xs) = getPsgCom xs) ∧
(∀ xs v96 v95 v12.
   getPsgCom (v12 says v95 eqn v96::xs) = getPsgCom xs) ∧
(∀ xs v98 v97 v12.
   getPsgCom (v12 says v97 lte v98::xs) = getPsgCom xs) ∧
(∀ xs v99 v12 v100.
   getPsgCom (v12 says v99 lt v100::xs) = getPsgCom xs) ∧
(∀ xs v15 v14.
   getPsgCom (v14 speaks_for v15::xs) = getPsgCom xs) ∧
(∀ xs v17 v16.
   getPsgCom (v16 controls v17::xs) = getPsgCom xs) ∧
(∀ xs v20 v19 v18.
   getPsgCom (reps v18 v19 v20::xs) = getPsgCom xs) ∧
(∀ xs v22 v21. getPsgCom (v21 domi v22::xs) = getPsgCom xs) ∧
(∀ xs v24 v23. getPsgCom (v23 eqi v24::xs) = getPsgCom xs) ∧
(∀ xs v26 v25. getPsgCom (v25 doms v26::xs) = getPsgCom xs) ∧
(∀ xs v28 v27. getPsgCom (v27 eqs v28::xs) = getPsgCom xs) ∧
(∀ 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)) ∧
```

$(\forall\, v_2\ xs.\ P\ xs \Rightarrow P\ (\texttt{prop}\ v_2\texttt{::}xs)) \wedge$
$(\forall\, v_3\ xs.\ P\ xs \Rightarrow P\ (\texttt{notf}\ v_3\texttt{::}xs)) \wedge$
$(\forall\, v_4\ v_5\ xs.\ P\ xs \Rightarrow P\ (v_4\ \texttt{andf}\ v_5\texttt{::}xs)) \wedge$
$(\forall\, v_6\ v_7\ xs.\ P\ xs \Rightarrow P\ (v_6\ \texttt{orf}\ v_7\texttt{::}xs)) \wedge$
$(\forall\, v_8\ v_9\ xs.\ P\ xs \Rightarrow P\ (v_8\ \texttt{impf}\ v_9\texttt{::}xs)) \wedge$
$(\forall\, v_{10}\ v_{11}\ xs.\ P\ xs \Rightarrow P\ (v_{10}\ \texttt{eqf}\ v_{11}\texttt{::}xs)) \wedge$
$(\forall\, v_{12}\ xs.\ P\ xs \Rightarrow P\ (v_{12}\ \texttt{says TT::}xs)) \wedge$
$(\forall\, v_{12}\ xs.\ P\ xs \Rightarrow P\ (v_{12}\ \texttt{says FF::}xs)) \wedge$
$(\forall\, \mathit{v134}\ xs.\ P\ xs \Rightarrow P\ (\texttt{Name}\ \mathit{v134}\ \texttt{says prop NONE::}xs)) \wedge$
$(\forall\, \mathit{v144}\ xs.$
$\quad P\ xs \Rightarrow$
$\quad P\ (\texttt{Name PlatoonLeader says prop (SOME}\ \mathit{v144}\texttt{)::}xs)) \wedge$
$(\forall\, \mathit{v146}\ xs.$
$\quad P\ xs \Rightarrow$
$\quad P$
$\quad\quad (\texttt{Name PlatoonSergeant says prop (SOME (ESCc}\ \mathit{v146}\texttt{))::}$
$\quad\quad\quad xs)) \wedge$
$(\forall\, \mathit{v150}\ xs.$
$\quad P\ xs \Rightarrow$
$\quad P$
$\quad\quad (\texttt{Name PlatoonSergeant says}$
$\quad\quad\ \texttt{prop (SOME (SLc (PL}\ \mathit{v150}\texttt{)))::}xs)) \wedge$
$(\forall\, \mathit{v135}\ \mathit{v136}\ v_{68}\ xs.$
$\quad P\ xs \Rightarrow P\ (\mathit{v135}\ \texttt{meet}\ \mathit{v136}\ \texttt{says prop}\ v_{68}\texttt{::}xs)) \wedge$
$(\forall\, \mathit{v137}\ \mathit{v138}\ v_{68}\ xs.$
$\quad P\ xs \Rightarrow P\ (\mathit{v137}\ \texttt{quoting}\ \mathit{v138}\ \texttt{says prop}\ v_{68}\texttt{::}xs)) \wedge$
$(\forall\, v_{12}\ v_{69}\ xs.\ P\ xs \Rightarrow P\ (v_{12}\ \texttt{says notf}\ v_{69}\texttt{::}xs)) \wedge$
$(\forall\, v_{12}\ v_{70}\ v_{71}\ xs.\ P\ xs \Rightarrow P\ (v_{12}\ \texttt{says (}v_{70}\ \texttt{andf}\ v_{71}\texttt{)::}xs)) \wedge$
$(\forall\, v_{12}\ v_{72}\ v_{73}\ xs.\ P\ xs \Rightarrow P\ (v_{12}\ \texttt{says (}v_{72}\ \texttt{orf}\ v_{73}\texttt{)::}xs)) \wedge$
$(\forall\, v_{12}\ v_{74}\ v_{75}\ xs.\ P\ xs \Rightarrow P\ (v_{12}\ \texttt{says (}v_{74}\ \texttt{impf}\ v_{75}\texttt{)::}xs)) \wedge$
$(\forall\, v_{12}\ v_{76}\ v_{77}\ xs.\ P\ xs \Rightarrow P\ (v_{12}\ \texttt{says (}v_{76}\ \texttt{eqf}\ v_{77}\texttt{)::}xs)) \wedge$
$(\forall\, v_{12}\ v_{78}\ v_{79}\ xs.\ P\ xs \Rightarrow P\ (v_{12}\ \texttt{says}\ v_{78}\ \texttt{says}\ v_{79}\texttt{::}xs)) \wedge$
$(\forall\, v_{12}\ v_{80}\ v_{81}\ xs.$
$\quad P\ xs \Rightarrow P\ (v_{12}\ \texttt{says}\ v_{80}\ \texttt{speaks\_for}\ v_{81}\texttt{::}xs)) \wedge$
$(\forall\, v_{12}\ v_{82}\ v_{83}\ xs.$
$\quad P\ xs \Rightarrow P\ (v_{12}\ \texttt{says}\ v_{82}\ \texttt{controls}\ v_{83}\texttt{::}xs)) \wedge$
$(\forall\, v_{12}\ v_{84}\ v_{85}\ v_{86}\ xs.$
$\quad P\ xs \Rightarrow P\ (v_{12}\ \texttt{says reps}\ v_{84}\ v_{85}\ v_{86}\texttt{::}xs)) \wedge$
$(\forall\, v_{12}\ v_{87}\ v_{88}\ xs.\ P\ xs \Rightarrow P\ (v_{12}\ \texttt{says}\ v_{87}\ \texttt{domi}\ v_{88}\texttt{::}xs)) \wedge$
$(\forall\, v_{12}\ v_{89}\ v_{90}\ xs.\ P\ xs \Rightarrow P\ (v_{12}\ \texttt{says}\ v_{89}\ \texttt{eqi}\ v_{90}\texttt{::}xs)) \wedge$
$(\forall\, v_{12}\ v_{91}\ v_{92}\ xs.\ P\ xs \Rightarrow P\ (v_{12}\ \texttt{says}\ v_{91}\ \texttt{doms}\ v_{92}\texttt{::}xs)) \wedge$
$(\forall\, v_{12}\ v_{93}\ v_{94}\ xs.\ P\ xs \Rightarrow P\ (v_{12}\ \texttt{says}\ v_{93}\ \texttt{eqs}\ v_{94}\texttt{::}xs)) \wedge$
$(\forall\, v_{12}\ v_{95}\ v_{96}\ xs.\ P\ xs \Rightarrow P\ (v_{12}\ \texttt{says}\ v_{95}\ \texttt{eqn}\ v_{96}\texttt{::}xs)) \wedge$
$(\forall\, v_{12}\ v_{97}\ v_{98}\ xs.\ P\ xs \Rightarrow P\ (v_{12}\ \texttt{says}\ v_{97}\ \texttt{lte}\ v_{98}\texttt{::}xs)) \wedge$
$(\forall\, v_{12}\ v_{99}\ \mathit{v100}\ xs.\ P\ xs \Rightarrow P\ (v_{12}\ \texttt{says}\ v_{99}\ \texttt{lt}\ \mathit{v100}\texttt{::}xs)) \wedge$
$(\forall\, v_{14}\ v_{15}\ xs.\ P\ xs \Rightarrow P\ (v_{14}\ \texttt{speaks\_for}\ v_{15}\texttt{::}xs)) \wedge$
$(\forall\, v_{16}\ v_{17}\ xs.\ P\ xs \Rightarrow P\ (v_{16}\ \texttt{controls}\ v_{17}\texttt{::}xs)) \wedge$
$(\forall\, v_{18}\ v_{19}\ v_{20}\ xs.\ P\ xs \Rightarrow P\ (\texttt{reps}\ v_{18}\ v_{19}\ v_{20}\texttt{::}xs)) \wedge$
$(\forall\, v_{21}\ v_{22}\ xs.\ P\ xs \Rightarrow P\ (v_{21}\ \texttt{domi}\ v_{22}\texttt{::}xs)) \wedge$

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

[getRecon_ind]

$\vdash \forall P.$
P [] $\wedge$
$(\forall xs.$
P
(Name PlatoonLeader says
prop (SOME (SLc (PL recon)))::xs)) $\wedge$
$(\forall xs.\ P\ xs \Rightarrow P$ (TT::xs)) $\wedge$ $(\forall xs.\ P\ xs \Rightarrow P$ (FF::xs)) $\wedge$
$(\forall v_2\ xs.\ P\ xs \Rightarrow P$ (prop v_2::xs)) $\wedge$
$(\forall v_3\ xs.\ P\ xs \Rightarrow P$ (notf v_3::xs)) $\wedge$
$(\forall v_4\ v_5\ xs.\ P\ xs \Rightarrow P$ (v_4 andf v_5::xs)) $\wedge$
$(\forall v_6\ v_7\ xs.\ P\ xs \Rightarrow P$ (v_6 orf v_7::xs)) $\wedge$
$(\forall v_8\ v_9\ xs.\ P\ xs \Rightarrow P$ (v_8 impf v_9::xs)) $\wedge$
$(\forall v_{10}\ v_{11}\ xs.\ P\ xs \Rightarrow P$ (v_{10} eqf v_{11}::xs)) $\wedge$
$(\forall v_{12}\ xs.\ P\ xs \Rightarrow P$ (v_{12} says TT::xs)) $\wedge$
$(\forall v_{12}\ xs.\ P\ xs \Rightarrow P$ (v_{12} says FF::xs)) $\wedge$
$(\forall v134\ xs.\ P\ xs \Rightarrow P$ (Name $v134$ says prop NONE::xs)) $\wedge$
$(\forall v146\ xs.$
$P\ xs \Rightarrow$
P
(Name PlatoonLeader says prop (SOME (ESCc $v146$))::
xs)) $\wedge$
$(\forall xs.$
$P\ xs \Rightarrow$
P
(Name PlatoonLeader says
prop (SOME (SLc (PL receiveMission)))::xs)) $\wedge$
$(\forall xs.$
$P\ xs \Rightarrow$
P
(Name PlatoonLeader says
prop (SOME (SLc (PL warno)))::xs)) $\wedge$
$(\forall xs.$
$P\ xs \Rightarrow$
P
(Name PlatoonLeader says
prop (SOME (SLc (PL tentativePlan)))::xs)) $\wedge$
$(\forall xs.$
$P\ xs \Rightarrow$
P
(Name PlatoonLeader says
prop (SOME (SLc (PL report1)))::xs)) $\wedge$
$(\forall xs.$
$P\ xs \Rightarrow$
P
(Name PlatoonLeader says
prop (SOME (SLc (PL completePlan)))::xs)) $\wedge$
$(\forall xs.$
$P\ xs \Rightarrow$

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* 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 (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 v68 v136 v135 .
   getReport (v135 meet v136 says prop v68::xs) =
   getReport xs) ∧
(∀ xs v68 v138 v137 .
   getReport (v137 quoting v138 says prop v68::xs) =
   getReport xs) ∧
(∀ xs v69 v12 .
   getReport (v12 says notf v69::xs) = getReport xs) ∧
(∀ xs v71 v70 v12 .
   getReport (v12 says (v70 andf v71)::xs) = getReport xs) ∧
(∀ xs v73 v72 v12 .
   getReport (v12 says (v72 orf v73)::xs) = getReport xs) ∧
(∀ xs v75 v74 v12 .
   getReport (v12 says (v74 impf v75)::xs) = getReport xs) ∧
(∀ xs v77 v76 v12 .
   getReport (v12 says (v76 eqf v77)::xs) = getReport xs) ∧
(∀ xs v79 v78 v12 .
   getReport (v12 says v78 says v79::xs) = getReport xs) ∧
(∀ xs v81 v80 v12 .
   getReport (v12 says v80 speaks_for v81::xs) =
   getReport xs) ∧
(∀ xs v83 v82 v12 .
   getReport (v12 says v82 controls v83::xs) =
   getReport xs) ∧
(∀ xs v86 v85 v84 v12 .
   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
```

$$
\begin{array}{l}
\quad \texttt{prop (SOME (SLc (PSG } v151\texttt{)))::}xs\texttt{))} \wedge \\
(\forall\, v144\ xs. \\
\quad P\ xs \Rightarrow \\
\quad P\ \texttt{(Name PlatoonSergeant says prop (SOME } v144\texttt{)::}xs\texttt{))} \wedge \\
(\forall\, v135\ v136\ v_{68}\ xs. \\
\quad P\ xs \Rightarrow P\ (v135\ \texttt{meet}\ v136\ \texttt{says prop}\ v_{68}\texttt{::}xs\texttt{))} \wedge \\
(\forall\, v137\ v138\ v_{68}\ xs. \\
\quad P\ xs \Rightarrow P\ (v137\ \texttt{quoting}\ v138\ \texttt{says prop}\ v_{68}\texttt{::}xs\texttt{))} \wedge \\
(\forall\, v_{12}\ v_{69}\ xs.\ P\ xs \Rightarrow P\ (v_{12}\ \texttt{says notf}\ v_{69}\texttt{::}xs\texttt{))} \wedge \\
(\forall\, v_{12}\ v_{70}\ v_{71}\ xs.\ P\ xs \Rightarrow P\ (v_{12}\ \texttt{says (}v_{70}\ \texttt{andf}\ v_{71}\texttt{)::}xs\texttt{))} \wedge \\
(\forall\, v_{12}\ v_{72}\ v_{73}\ xs.\ P\ xs \Rightarrow P\ (v_{12}\ \texttt{says (}v_{72}\ \texttt{orf}\ v_{73}\texttt{)::}xs\texttt{))} \wedge \\
(\forall\, v_{12}\ v_{74}\ v_{75}\ xs.\ P\ xs \Rightarrow P\ (v_{12}\ \texttt{says (}v_{74}\ \texttt{impf}\ v_{75}\texttt{)::}xs\texttt{))} \wedge \\
(\forall\, v_{12}\ v_{76}\ v_{77}\ xs.\ P\ xs \Rightarrow P\ (v_{12}\ \texttt{says (}v_{76}\ \texttt{eqf}\ v_{77}\texttt{)::}xs\texttt{))} \wedge \\
(\forall\, v_{12}\ v_{78}\ v_{79}\ xs.\ P\ xs \Rightarrow P\ (v_{12}\ \texttt{says}\ v_{78}\ \texttt{says}\ v_{79}\texttt{::}xs\texttt{))} \wedge \\
(\forall\, v_{12}\ v_{80}\ v_{81}\ xs. \\
\quad P\ xs \Rightarrow P\ (v_{12}\ \texttt{says}\ v_{80}\ \texttt{speaks\_for}\ v_{81}\texttt{::}xs\texttt{))} \wedge \\
(\forall\, v_{12}\ v_{82}\ v_{83}\ xs. \\
\quad P\ xs \Rightarrow P\ (v_{12}\ \texttt{says}\ v_{82}\ \texttt{controls}\ v_{83}\texttt{::}xs\texttt{))} \wedge \\
(\forall\, v_{12}\ v_{84}\ v_{85}\ v_{86}\ xs. \\
\quad P\ xs \Rightarrow P\ (v_{12}\ \texttt{says reps}\ v_{84}\ v_{85}\ v_{86}\texttt{::}xs\texttt{))} \wedge \\
(\forall\, v_{12}\ v_{87}\ v_{88}\ xs.\ P\ xs \Rightarrow P\ (v_{12}\ \texttt{says}\ v_{87}\ \texttt{domi}\ v_{88}\texttt{::}xs\texttt{))} \wedge \\
(\forall\, v_{12}\ v_{89}\ v_{90}\ xs.\ P\ xs \Rightarrow P\ (v_{12}\ \texttt{says}\ v_{89}\ \texttt{eqi}\ v_{90}\texttt{::}xs\texttt{))} \wedge \\
(\forall\, v_{12}\ v_{91}\ v_{92}\ xs.\ P\ xs \Rightarrow P\ (v_{12}\ \texttt{says}\ v_{91}\ \texttt{doms}\ v_{92}\texttt{::}xs\texttt{))} \wedge \\
(\forall\, v_{12}\ v_{93}\ v_{94}\ xs.\ P\ xs \Rightarrow P\ (v_{12}\ \texttt{says}\ v_{93}\ \texttt{eqs}\ v_{94}\texttt{::}xs\texttt{))} \wedge \\
(\forall\, v_{12}\ v_{95}\ v_{96}\ xs.\ P\ xs \Rightarrow P\ (v_{12}\ \texttt{says}\ v_{95}\ \texttt{eqn}\ v_{96}\texttt{::}xs\texttt{))} \wedge \\
(\forall\, v_{12}\ v_{97}\ v_{98}\ xs.\ P\ xs \Rightarrow P\ (v_{12}\ \texttt{says}\ v_{97}\ \texttt{lte}\ v_{98}\texttt{::}xs\texttt{))} \wedge \\
(\forall\, v_{12}\ v_{99}\ v100\ xs.\ P\ xs \Rightarrow P\ (v_{12}\ \texttt{says}\ v_{99}\ \texttt{lt}\ v100\texttt{::}xs\texttt{))} \wedge \\
(\forall\, v_{14}\ v_{15}\ xs.\ P\ xs \Rightarrow P\ (v_{14}\ \texttt{speaks\_for}\ v_{15}\texttt{::}xs\texttt{))} \wedge \\
(\forall\, v_{16}\ v_{17}\ xs.\ P\ xs \Rightarrow P\ (v_{16}\ \texttt{controls}\ v_{17}\texttt{::}xs\texttt{))} \wedge \\
(\forall\, v_{18}\ v_{19}\ v_{20}\ xs.\ P\ xs \Rightarrow P\ \texttt{(reps}\ v_{18}\ v_{19}\ v_{20}\texttt{::}xs\texttt{))} \wedge \\
(\forall\, v_{21}\ v_{22}\ xs.\ P\ xs \Rightarrow P\ (v_{21}\ \texttt{domi}\ v_{22}\texttt{::}xs\texttt{))} \wedge \\
(\forall\, v_{23}\ v_{24}\ xs.\ P\ xs \Rightarrow P\ (v_{23}\ \texttt{eqi}\ v_{24}\texttt{::}xs\texttt{))} \wedge \\
(\forall\, v_{25}\ v_{26}\ xs.\ P\ xs \Rightarrow P\ (v_{25}\ \texttt{doms}\ v_{26}\texttt{::}xs\texttt{))} \wedge \\
(\forall\, v_{27}\ v_{28}\ xs.\ P\ xs \Rightarrow P\ (v_{27}\ \texttt{eqs}\ v_{28}\texttt{::}xs\texttt{))} \wedge \\
(\forall\, v_{29}\ v_{30}\ xs.\ P\ xs \Rightarrow P\ (v_{29}\ \texttt{eqn}\ v_{30}\texttt{::}xs\texttt{))} \wedge \\
(\forall\, v_{31}\ v_{32}\ xs.\ P\ xs \Rightarrow P\ (v_{31}\ \texttt{lte}\ v_{32}\texttt{::}xs\texttt{))} \wedge \\
(\forall\, v_{33}\ v_{34}\ xs.\ P\ xs \Rightarrow P\ (v_{33}\ \texttt{lt}\ v_{34}\texttt{::}xs\texttt{))} \Rightarrow \\
\forall\, v.\ P\ v
\end{array}
$$

Index