Contents

1 PBTypeIntegrated Theory

Built: 11 June 2018
Parent Theories: OMNIType

1.1 Datatypes

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

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

slCommand = PL plCommand | OMNI 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

1.2 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
```

2 ssmPBIntegrated Theory

Built: 11 June 2018

Parent Theories: PBIntegratedDef, ssm

2.1 Theorems

[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]

$\vdash \forall P.$
$(\forall cmd.\ P$ (Name PlatoonLeader says prop cmd)) $\wedge$
$(\forall cmd.\ P$ (Name Omni says prop cmd)) $\wedge$ P TT $\wedge$ P FF $\wedge$
$(\forall v.\ P$ (prop v)) $\wedge$ $(\forall v_1.\ P$ (notf v_1)) $\wedge$
$(\forall v_2\ v_3.\ P$ (v_2 andf v_3)) $\wedge$ $(\forall v_4\ v_5.\ P$ (v_4 orf v_5)) $\wedge$
$(\forall v_6\ v_7.\ P$ (v_6 impf v_7)) $\wedge$ $(\forall v_8\ v_9.\ P$ (v_8 eqf v_9)) $\wedge$
$(\forall v_{10}.\ P$ (v_{10} says TT)) $\wedge$ $(\forall v_{10}.\ P$ (v_{10} says FF)) $\wedge$
$(\forall$ *v133* *v134* $v_{66}.\ P$ (*v133* meet *v134* says prop v_{66})) $\wedge$
$(\forall$ *v135* *v136* $v_{66}.\ P$ (*v135* quoting *v136* says prop v_{66})) $\wedge$
$(\forall v_{10}\ v_{67}.\ P$ (v_{10} says notf v_{67})) $\wedge$
$(\forall v_{10}\ v_{68}\ v_{69}.\ P$ (v_{10} says (v_{68} andf v_{69}))) $\wedge$

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

[PBNS_def]

```
⊢ (PBNS PLAN_PB (exec [SOME (SLc (PL crossLD))]) =
   MOVE_TO_ORP) ∧
  (PBNS MOVE_TO_ORP (exec [SOME (SLc (PL conductORP))]) =
   CONDUCT_ORP) ∧
  (PBNS CONDUCT_ORP (exec [SOME (SLc (PL moveToPB))]) =
   MOVE_TO_PB) ∧
  (PBNS MOVE_TO_PB (exec [SOME (SLc (PL conductPB))]) =
   CONDUCT_PB) ∧
  (PBNS CONDUCT_PB (exec [SOME (SLc (PL completePB))]) =
   COMPLETE_PB) ∧ (PBNS s (trap v0) = s) ∧
  (PBNS s (discard v1) = s)
```

[PBNS_ind]

⊢ $\forall\, P.$
P PLAN_PB (exec [SOME (SLc (PL crossLD))]) $\wedge$
P MOVE_TO_ORP (exec [SOME (SLc (PL conductORP))]) $\wedge$
P CONDUCT_ORP (exec [SOME (SLc (PL moveToPB))]) $\wedge$
P MOVE_TO_PB (exec [SOME (SLc (PL conductPB))]) $\wedge$
P CONDUCT_PB (exec [SOME (SLc (PL completePB))]) $\wedge$
$(\forall\, s\ v_0.\ P\ s\ (\texttt{trap}\ v_0)) \wedge (\forall\, s\ v_1.\ P\ s\ (\texttt{discard}\ v_1)) \wedge$
$(\forall\, v_8.\ P\ v_8\ (\texttt{exec}\ [])) \wedge$
$(\forall\, v_{11}\ v_{10}.\ P\ v_{11}\ (\texttt{exec}\ (\texttt{NONE::}v_{10}))) \wedge$
$(\forall\, v_{16}\ v_{13}\ v_{15}.\ P\ v_{16}\ (\texttt{exec}\ (\texttt{SOME}\ (\texttt{ESCc}\ v_{13})\texttt{::}v_{15}))) \wedge$
P MOVE_TO_ORP (exec [SOME (SLc (PL crossLD))]) $\wedge$

P CONDUCT_ORP (exec [SOME (SLc (PL crossLD))]) ∧
P MOVE_TO_PB (exec [SOME (SLc (PL crossLD))]) ∧
P CONDUCT_PB (exec [SOME (SLc (PL crossLD))]) ∧
P COMPLETE_PB (exec [SOME (SLc (PL crossLD))]) ∧
P PLAN_PB (exec [SOME (SLc (PL conductORP))]) ∧
P CONDUCT_ORP (exec [SOME (SLc (PL conductORP))]) ∧
P MOVE_TO_PB (exec [SOME (SLc (PL conductORP))]) ∧
P CONDUCT_PB (exec [SOME (SLc (PL conductORP))]) ∧
P COMPLETE_PB (exec [SOME (SLc (PL conductORP))]) ∧
P PLAN_PB (exec [SOME (SLc (PL moveToPB))]) ∧
P MOVE_TO_ORP (exec [SOME (SLc (PL moveToPB))]) ∧
P MOVE_TO_PB (exec [SOME (SLc (PL moveToPB))]) ∧
P CONDUCT_PB (exec [SOME (SLc (PL moveToPB))]) ∧
P COMPLETE_PB (exec [SOME (SLc (PL moveToPB))]) ∧
P PLAN_PB (exec [SOME (SLc (PL conductPB))]) ∧
P MOVE_TO_ORP (exec [SOME (SLc (PL conductPB))]) ∧
P CONDUCT_ORP (exec [SOME (SLc (PL conductPB))]) ∧
P CONDUCT_PB (exec [SOME (SLc (PL conductPB))]) ∧
P COMPLETE_PB (exec [SOME (SLc (PL conductPB))]) ∧
P PLAN_PB (exec [SOME (SLc (PL completePB))]) ∧
P MOVE_TO_ORP (exec [SOME (SLc (PL completePB))]) ∧
P CONDUCT_ORP (exec [SOME (SLc (PL completePB))]) ∧
P MOVE_TO_PB (exec [SOME (SLc (PL completePB))]) ∧
P COMPLETE_PB (exec [SOME (SLc (PL completePB))]) ∧
(∀ v_{24}. P v_{24} (exec [SOME (SLc (PL incomplete))])) ∧
(∀ v_{26} v_{25} v_{22} v_{23}.
P v_{26} (exec (SOME (SLc (PL v_{25}))::v_{22}::v_{23}))) ∧
(∀ v_{28} v_{19} v_{27}. P v_{28} (exec (SOME (SLc (OMNI v_{19}))::v_{27}))) ⇒
∀ v v_1. P v v_1

[PBOut_def]

⊢ (PBOut PLAN_PB (exec [SOME (SLc (PL crossLD))]) =
MoveToORP) ∧
(PBOut MOVE_TO_ORP (exec [SOME (SLc (PL conductORP))]) =
ConductORP) ∧
(PBOut CONDUCT_ORP (exec [SOME (SLc (PL moveToPB))]) =
MoveToPB) ∧
(PBOut MOVE_TO_PB (exec [SOME (SLc (PL conductPB))]) =
ConductPB) ∧
(PBOut CONDUCT_PB (exec [SOME (SLc (PL completePB))]) =
CompletePB) ∧ (PBOut s (trap v_0) = unAuthorized) ∧
(PBOut s (discard v_1) = unAuthenticated)

[PBOut_ind]

⊢ ∀ P.
P PLAN_PB (exec [SOME (SLc (PL crossLD))]) ∧
P MOVE_TO_ORP (exec [SOME (SLc (PL conductORP))]) ∧
P CONDUCT_ORP (exec [SOME (SLc (PL moveToPB))]) ∧
P MOVE_TO_PB (exec [SOME (SLc (PL conductPB))]) ∧

P CONDUCT_PB (exec [SOME (SLc (PL completePB))]) ∧
(∀ s v_0. P s (trap v_0)) ∧ (∀ s v_1. P s (discard v_1)) ∧
(∀ v_8. P v_8 (exec [])) ∧
(∀ v_{11} v_{10}. P v_{11} (exec (NONE::v_{10}))) ∧
(∀ v_{16} v_{13} v_{15}. P v_{16} (exec (SOME (ESCc v_{13})::v_{15}))) ∧
P MOVE_TO_ORP (exec [SOME (SLc (PL crossLD))]) ∧
P CONDUCT_ORP (exec [SOME (SLc (PL crossLD))]) ∧
P MOVE_TO_PB (exec [SOME (SLc (PL crossLD))]) ∧
P CONDUCT_PB (exec [SOME (SLc (PL crossLD))]) ∧
P COMPLETE_PB (exec [SOME (SLc (PL crossLD))]) ∧
P PLAN_PB (exec [SOME (SLc (PL conductORP))]) ∧
P CONDUCT_ORP (exec [SOME (SLc (PL conductORP))]) ∧
P MOVE_TO_PB (exec [SOME (SLc (PL conductORP))]) ∧
P CONDUCT_PB (exec [SOME (SLc (PL conductORP))]) ∧
P COMPLETE_PB (exec [SOME (SLc (PL conductORP))]) ∧
P PLAN_PB (exec [SOME (SLc (PL moveToPB))]) ∧
P MOVE_TO_ORP (exec [SOME (SLc (PL moveToPB))]) ∧
P MOVE_TO_PB (exec [SOME (SLc (PL moveToPB))]) ∧
P CONDUCT_PB (exec [SOME (SLc (PL moveToPB))]) ∧
P COMPLETE_PB (exec [SOME (SLc (PL moveToPB))]) ∧
P PLAN_PB (exec [SOME (SLc (PL conductPB))]) ∧
P MOVE_TO_ORP (exec [SOME (SLc (PL conductPB))]) ∧
P CONDUCT_ORP (exec [SOME (SLc (PL conductPB))]) ∧
P CONDUCT_PB (exec [SOME (SLc (PL conductPB))]) ∧
P COMPLETE_PB (exec [SOME (SLc (PL conductPB))]) ∧
P PLAN_PB (exec [SOME (SLc (PL completePB))]) ∧
P MOVE_TO_ORP (exec [SOME (SLc (PL completePB))]) ∧
P CONDUCT_ORP (exec [SOME (SLc (PL completePB))]) ∧
P MOVE_TO_PB (exec [SOME (SLc (PL completePB))]) ∧
P COMPLETE_PB (exec [SOME (SLc (PL completePB))]) ∧
(∀ v_{24}. P v_{24} (exec [SOME (SLc (PL incomplete))])) ∧
(∀ v_{26} v_{25} v_{22} v_{23}.
P v_{26} (exec (SOME (SLc (PL v_{25}))::v_{22}::v_{23}))) ∧
(∀ v_{28} v_{19} v_{27}. P v_{28} (exec (SOME (SLc (OMNI v_{19}))::v_{27}))) ⇒
∀ v v_1. P v v_1

[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_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
```

3 PBIntegratedDef Theory

Built: 11 June 2018

Parent Theories: PBTypeIntegrated, aclfoundation

3.1 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)))]
```

3.2 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)) ∧
```

$$
\begin{array}{l}
(\forall\, v_{12}\ v_{74}\ v_{75}\ xs.\ P\ xs \Rightarrow P\ (v_{12}\ \texttt{says}\ (v_{74}\ \texttt{impf}\ v_{75})::xs)) \wedge \\
(\forall\, v_{12}\ v_{76}\ v_{77}\ xs.\ P\ xs \Rightarrow P\ (v_{12}\ \texttt{says}\ (v_{76}\ \texttt{eqf}\ v_{77})::xs)) \wedge \\
(\forall\, v_{12}\ v_{78}\ v_{79}\ xs.\ P\ xs \Rightarrow P\ (v_{12}\ \texttt{says}\ v_{78}\ \texttt{says}\ v_{79}::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}::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}::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}::xs)) \wedge \\
(\forall\, v_{12}\ v_{87}\ v_{88}\ xs.\ P\ xs \Rightarrow P\ (v_{12}\ \texttt{says}\ v_{87}\ \texttt{domi}\ v_{88}::xs)) \wedge \\
(\forall\, v_{12}\ v_{89}\ v_{90}\ xs.\ P\ xs \Rightarrow P\ (v_{12}\ \texttt{says}\ v_{89}\ \texttt{eqi}\ v_{90}::xs)) \wedge \\
(\forall\, v_{12}\ v_{91}\ v_{92}\ xs.\ P\ xs \Rightarrow P\ (v_{12}\ \texttt{says}\ v_{91}\ \texttt{doms}\ v_{92}::xs)) \wedge \\
(\forall\, v_{12}\ v_{93}\ v_{94}\ xs.\ P\ xs \Rightarrow P\ (v_{12}\ \texttt{says}\ v_{93}\ \texttt{eqs}\ v_{94}::xs)) \wedge \\
(\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
\end{array}
$$

Index