Contents

1 ConductORPType Theory

Built: 10 June 2018
Parent Theories: indexedLists, patternMatches

1.1 Datatypes

plCommand = secure | withdraw | complete | plIncomplete

psgCommand = actionsIn | psgIncomplete

slCommand = PL plCommand | PSG psgCommand

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

slState = CONDUCT_ORP | SECURE | ACTIONS_IN | WITHDRAW
| COMPLETE

stateRole = PlatoonLeader | PlatoonSergeant

1.2 Theorems

[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'

[slCommand_one_one]

⊢ (∀ a a'. (PL a = PL a') ⟺ (a = a')) ∧
∀ a a'. (PSG a = PSG 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

[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

2 ssmConductORP Theory

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

2.1 Definitions

[secContextConductORP_def]

⊢ ∀ *plcmd psgcmd incomplete*.
secContextConductORP *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]

[ssmConductORPStateInterp_def]

⊢ ∀ *slState*. ssmConductORPStateInterp *slState* = TT

2.2 Theorems

[authTestConductORP_cmd_reject_lemma]

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

[authTestConductORP_def]

⊢ (authTestConductORP (Name PlatoonLeader says prop *cmd*) ⟺
T) ∧
(authTestConductORP (Name PlatoonSergeant says prop *cmd*) ⟺
T) ∧ (authTestConductORP TT ⟺ F) ∧
(authTestConductORP FF ⟺ F) ∧
(authTestConductORP (prop *v*) ⟺ F) ∧
(authTestConductORP (notf v_1) ⟺ F) ∧
(authTestConductORP (v_2 andf v_3) ⟺ F) ∧
(authTestConductORP (v_4 orf v_5) ⟺ F) ∧

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

[authTestConductORP_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})) ∧

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

[conductORPNS_def]

$\vdash$ (conductORPNS CONDUCT_ORP (exec (PL secure)) = SECURE) $\wedge$
(conductORPNS CONDUCT_ORP (exec (PL plIncomplete)) =
CONDUCT_ORP) $\wedge$
(conductORPNS SECURE (exec (PSG actionsIn)) = ACTIONS_IN) $\wedge$
(conductORPNS SECURE (exec (PSG psgIncomplete)) = SECURE) $\wedge$
(conductORPNS ACTIONS_IN (exec (PL withdraw)) = WITHDRAW) $\wedge$
(conductORPNS ACTIONS_IN (exec (PL plIncomplete)) =
ACTIONS_IN) $\wedge$
(conductORPNS WITHDRAW (exec (PL complete)) = COMPLETE) $\wedge$
(conductORPNS WITHDRAW (exec (PL plIncomplete)) = WITHDRAW) $\wedge$
(conductORPNS s (trap (PL cmd')) = s) $\wedge$
(conductORPNS s (trap (PSG cmd)) = s) $\wedge$
(conductORPNS s (discard (PL cmd')) = s) $\wedge$
(conductORPNS s (discard (PSG cmd)) = s)

[conductORPNS_ind]

$\vdash \forall P.$
P CONDUCT_ORP (exec (PL secure)) $\wedge$
P CONDUCT_ORP (exec (PL plIncomplete)) $\wedge$
P SECURE (exec (PSG actionsIn)) $\wedge$
P SECURE (exec (PSG psgIncomplete)) $\wedge$
P ACTIONS_IN (exec (PL withdraw)) $\wedge$
P ACTIONS_IN (exec (PL plIncomplete)) $\wedge$
P WITHDRAW (exec (PL complete)) $\wedge$
P WITHDRAW (exec (PL plIncomplete)) $\wedge$
$(\forall\, s\ cmd.\ P\ s$ (trap (PL cmd))) $\wedge$
$(\forall\, s\ cmd.\ P\ s$ (trap (PSG cmd))) $\wedge$
$(\forall\, s\ cmd.\ P\ s$ (discard (PL cmd))) $\wedge$

```
     (∀ s cmd. P s (discard (PSG cmd))) ∧
     P CONDUCT_ORP (exec (PL withdraw)) ∧
     P CONDUCT_ORP (exec (PL complete)) ∧
     (∀ v11. P CONDUCT_ORP (exec (PSG v11))) ∧
     (∀ v13. P SECURE (exec (PL v13))) ∧
     P ACTIONS_IN (exec (PL secure)) ∧
     P ACTIONS_IN (exec (PL complete)) ∧
     (∀ v17. P ACTIONS_IN (exec (PSG v17))) ∧
     P WITHDRAW (exec (PL secure)) ∧
     P WITHDRAW (exec (PL withdraw)) ∧
     (∀ v20. P WITHDRAW (exec (PSG v20))) ∧
     (∀ v21. P COMPLETE (exec v21)) ⇒
     ∀ v v1. P v v1
```

[conductORPOut_def]

```
⊢ (conductORPOut CONDUCT_ORP (exec (PL secure)) = Secure) ∧
  (conductORPOut CONDUCT_ORP (exec (PL plIncomplete)) =
   ConductORP) ∧
  (conductORPOut SECURE (exec (PSG actionsIn)) = ActionsIn) ∧
  (conductORPOut SECURE (exec (PSG psgIncomplete)) = Secure) ∧
  (conductORPOut ACTIONS_IN (exec (PL withdraw)) = Withdraw) ∧
  (conductORPOut ACTIONS_IN (exec (PL plIncomplete)) =
   ActionsIn) ∧
  (conductORPOut WITHDRAW (exec (PL complete)) = Complete) ∧
  (conductORPOut WITHDRAW (exec (PL plIncomplete)) =
   Withdraw) ∧
  (conductORPOut s (trap (PL cmd')) = unAuthorized) ∧
  (conductORPOut s (trap (PSG cmd)) = unAuthorized) ∧
  (conductORPOut s (discard (PL cmd')) = unAuthenticated) ∧
  (conductORPOut s (discard (PSG cmd)) = unAuthenticated)
```

[conductORPOut_ind]

```
⊢ ∀ P.
     P CONDUCT_ORP (exec (PL secure)) ∧
     P CONDUCT_ORP (exec (PL plIncomplete)) ∧
     P SECURE (exec (PSG actionsIn)) ∧
     P SECURE (exec (PSG psgIncomplete)) ∧
     P ACTIONS_IN (exec (PL withdraw)) ∧
     P ACTIONS_IN (exec (PL plIncomplete)) ∧
     P WITHDRAW (exec (PL complete)) ∧
     P WITHDRAW (exec (PL plIncomplete)) ∧
     (∀ 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_ORP (exec (PL withdraw)) ∧
     P CONDUCT_ORP (exec (PL complete)) ∧
     (∀ v11. P CONDUCT_ORP (exec (PSG v11))) ∧
     (∀ v13. P SECURE (exec (PL v13))) ∧
```

```
     P ACTIONS_IN (exec (PL secure)) ∧
     P ACTIONS_IN (exec (PL complete)) ∧
     (∀ v17. P ACTIONS_IN (exec (PSG v17))) ∧
     P WITHDRAW (exec (PL secure)) ∧
     P WITHDRAW (exec (PL withdraw)) ∧
     (∀ v20. P WITHDRAW (exec (PSG v20))) ∧
     (∀ v21. P COMPLETE (exec v21)) ⇒
     ∀ v v1. P v v1
```

[PlatoonLeader_exec_plCommand_justified_thm]

```
 ⊢ ∀ NS Out M Oi Os.
     TR (M,Oi,Os) (exec (SLc (PL plCommand)))
       (CFG authTestConductORP ssmConductORPStateInterp
          (secContextConductORP plCommand psgCommand incomplete)
          (Name PlatoonLeader says
           prop (SOME (SLc (PL plCommand)))::ins) s outs)
       (CFG authTestConductORP ssmConductORPStateInterp
          (secContextConductORP plCommand psgCommand incomplete)
          ins (NS s (exec (SLc (PL plCommand))))
          (Out s (exec (SLc (PL plCommand)))::outs)) ⟺
     authTestConductORP
       (Name PlatoonLeader says
        prop (SOME (SLc (PL plCommand)))) ∧
     CFGInterpret (M,Oi,Os)
       (CFG authTestConductORP ssmConductORPStateInterp
          (secContextConductORP plCommand psgCommand incomplete)
          (Name PlatoonLeader says
           prop (SOME (SLc (PL plCommand)))::ins) s outs) ∧
     (M,Oi,Os) sat prop (SOME (SLc (PL plCommand)))
```

[PlatoonLeader_plCommand_lemma]

```
 ⊢ CFGInterpret (M,Oi,Os)
     (CFG authTestConductORP ssmConductORPStateInterp
        (secContextConductORP 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_psgCommand_justified_thm]

```
 ⊢ ∀ NS Out M Oi Os.
     TR (M,Oi,Os) (exec (SLc (PSG psgCommand)))
       (CFG authTestConductORP ssmConductORPStateInterp
          (secContextConductORP plCommand psgCommand incomplete)
          (Name PlatoonSergeant says
           prop (SOME (SLc (PSG psgCommand)))::ins) s outs)
       (CFG authTestConductORP ssmConductORPStateInterp
          (secContextConductORP plCommand psgCommand incomplete)
          ins (NS s (exec (SLc (PSG psgCommand))))
```

```
        (Out s (exec (SLc (PSG psgCommand)))::outs)) ⟺
    authTestConductORP
      (Name PlatoonSergeant says
       prop (SOME (SLc (PSG psgCommand)))) ∧
    CFGInterpret (M,Oi,Os)
      (CFG authTestConductORP ssmConductORPStateInterp
         (secContextConductORP plCommand psgCommand incomplete)
         (Name PlatoonSergeant says
          prop (SOME (SLc (PSG psgCommand)))::ins) s outs) ∧
    (M,Oi,Os) sat prop (SOME (SLc (PSG psgCommand)))
```

[PlatoonSergeant_psgCommand_lemma]

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

Index