Contents

1 ssm11 Theory

Built: 10 June 2018

Parent Theories: satList

1.1 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
```

1.2 Definitions

[TR_def]

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

1.3 Theorems

[CFGInterpret_def]

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

[CFGInterpret_ind]

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

[configuration_one_one]

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

[order_distinct_clauses]

```
⊢ ∀a. SOME a ≠ NONE
```

[order_one_one]

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

[TR_cases]

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

[TR_discard_cmd_rule]

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

[TR_EQ_rules_thm]

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

[TR_exec_cmd_rule]

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

```
        (CFG authenticationTest stateInterp securityContext
           (P says prop (SOME cmd)::ins) s outs) ∧
      (M,Oi,Os) sat prop (SOME cmd)
```

[TR_ind]

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

[TR_rules]

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

[TR_strongind]

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

```
        TR′ (M,Oi,Os) (discard cmd)
          (CFG authenticationTest stateInterp securityContext
             (x::ins) s outs)
          (CFG authenticationTest stateInterp securityContext
             ins (NS s (discard cmd))
             (Out s (discard cmd)::outs))) ⇒
     ∀ a0 a1 a2 a3. TR a0 a1 a2 a3 ⇒ TR′ a0 a1 a2 a3
```

[TR_trap_cmd_rule]

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

[TRrule0]

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

[TRrule1]

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

[trType_distinct_clauses]

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

[trType_one_one]

```
⊢ (∀ a a′. (discard a = discard a′) ⇔ (a = a′)) ∧
  (∀ a a′. (trap a = trap a′) ⇔ (a = a′)) ∧
  ∀ a a′. (exec a = exec a′) ⇔ (a = a′)
```

2 ssm Theory

Built: 10 June 2018

Parent Theories: satList

2.1 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
```

2.2 Definitions

[authenticationTest_def]

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

[commandList_def]

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

[inputList_def]

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

[propCommandList_def]

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

[TR_def]

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

2.3 Theorems

[CFGInterpret_def]

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

[CFGInterpret_ind]

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

[configuration_one_one]

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

[extractCommand_def]

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

[extractCommand_ind]

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

$(\forall v145\ v115\ v116.\ P'\ (v145\ \texttt{says}\ v115\ \texttt{eqs}\ v116)) \land$
$(\forall v146\ v119\ v120.\ P'\ (v146\ \texttt{says}\ v119\ \texttt{eqn}\ v120)) \land$
$(\forall v147\ v123\ v124.\ P'\ (v147\ \texttt{says}\ v123\ \texttt{lte}\ v124)) \land$
$(\forall v148\ v127\ v128.\ P'\ (v148\ \texttt{says}\ v127\ \texttt{lt}\ v128)) \land$
$(\forall v_{24}\ v_{25}.\ P'\ (v_{24}\ \texttt{speaks\_for}\ v_{25})) \land$
$(\forall v_{28}\ v_{29}.\ P'\ (v_{28}\ \texttt{controls}\ v_{29})) \land$
$(\forall v_{33}\ v_{34}\ v_{35}.\ P'\ (\texttt{reps}\ v_{33}\ v_{34}\ v_{35})) \land$
$(\forall v_{38}\ v_{39}.\ P'\ (v_{38}\ \texttt{domi}\ v_{39})) \land$
$(\forall v_{42}\ v_{43}.\ P'\ (v_{42}\ \texttt{eqi}\ v_{43})) \land$
$(\forall v_{46}\ v_{47}.\ P'\ (v_{46}\ \texttt{doms}\ v_{47})) \land$
$(\forall v_{50}\ v_{51}.\ P'\ (v_{50}\ \texttt{eqs}\ v_{51})) \land$
$(\forall v_{54}\ v_{55}.\ P'\ (v_{54}\ \texttt{eqn}\ v_{55})) \land$
$(\forall v_{58}\ v_{59}.\ P'\ (v_{58}\ \texttt{lte}\ v_{59})) \land$
$(\forall v_{62}\ v_{63}.\ P'\ (v_{62}\ \texttt{lt}\ v_{63})) \Rightarrow$
$\forall v.\ P'\ v$

[extractInput_def]

$\vdash \texttt{extractInput}\ (P\ \texttt{says}\ \texttt{prop}\ x) = x$

[extractInput_ind]

$\vdash \forall P'.$
$(\forall P\ x.\ P'\ (P\ \texttt{says}\ \texttt{prop}\ x)) \land P'\ \texttt{TT} \land P'\ \texttt{FF} \land$
$(\forall v_1.\ P'\ (\texttt{prop}\ v_1)) \land (\forall v_3.\ P'\ (\texttt{notf}\ v_3)) \land$
$(\forall v_6\ v_7.\ P'\ (v_6\ \texttt{andf}\ v_7)) \land (\forall v_{10}\ v_{11}.\ P'\ (v_{10}\ \texttt{orf}\ v_{11})) \land$
$(\forall v_{14}\ v_{15}.\ P'\ (v_{14}\ \texttt{impf}\ v_{15})) \land$
$(\forall v_{18}\ v_{19}.\ P'\ (v_{18}\ \texttt{eqf}\ v_{19})) \land (\forall v129.\ P'\ (v129\ \texttt{says}\ \texttt{TT})) \land$
$(\forall v130.\ P'\ (v130\ \texttt{says}\ \texttt{FF})) \land$
$(\forall v131\ v_{66}.\ P'\ (v131\ \texttt{says}\ \texttt{notf}\ v_{66})) \land$
$(\forall v132\ v_{69}\ v_{70}.\ P'\ (v132\ \texttt{says}\ (v_{69}\ \texttt{andf}\ v_{70}))) \land$
$(\forall v133\ v_{73}\ v_{74}.\ P'\ (v133\ \texttt{says}\ (v_{73}\ \texttt{orf}\ v_{74}))) \land$
$(\forall v134\ v_{77}\ v_{78}.\ P'\ (v134\ \texttt{says}\ (v_{77}\ \texttt{impf}\ v_{78}))) \land$
$(\forall v135\ v_{81}\ v_{82}.\ P'\ (v135\ \texttt{says}\ (v_{81}\ \texttt{eqf}\ v_{82}))) \land$
$(\forall v136\ v_{85}\ v_{86}.\ P'\ (v136\ \texttt{says}\ v_{85}\ \texttt{says}\ v_{86})) \land$
$(\forall v137\ v_{89}\ v_{90}.\ P'\ (v137\ \texttt{says}\ v_{89}\ \texttt{speaks\_for}\ v_{90})) \land$
$(\forall v138\ v_{93}\ v_{94}.\ P'\ (v138\ \texttt{says}\ v_{93}\ \texttt{controls}\ v_{94})) \land$
$(\forall v139\ v_{98}\ v_{99}\ v100.\ P'\ (v139\ \texttt{says}\ \texttt{reps}\ v_{98}\ v_{99}\ v100)) \land$
$(\forall v140\ v103\ v104.\ P'\ (v140\ \texttt{says}\ v103\ \texttt{domi}\ v104)) \land$
$(\forall v141\ v107\ v108.\ P'\ (v141\ \texttt{says}\ v107\ \texttt{eqi}\ v108)) \land$
$(\forall v142\ v111\ v112.\ P'\ (v142\ \texttt{says}\ v111\ \texttt{doms}\ v112)) \land$
$(\forall v143\ v115\ v116.\ P'\ (v143\ \texttt{says}\ v115\ \texttt{eqs}\ v116)) \land$
$(\forall v144\ v119\ v120.\ P'\ (v144\ \texttt{says}\ v119\ \texttt{eqn}\ v120)) \land$
$(\forall v145\ v123\ v124.\ P'\ (v145\ \texttt{says}\ v123\ \texttt{lte}\ v124)) \land$
$(\forall v146\ v127\ v128.\ P'\ (v146\ \texttt{says}\ v127\ \texttt{lt}\ v128)) \land$
$(\forall v_{24}\ v_{25}.\ P'\ (v_{24}\ \texttt{speaks\_for}\ v_{25})) \land$
$(\forall v_{28}\ v_{29}.\ P'\ (v_{28}\ \texttt{controls}\ v_{29})) \land$
$(\forall v_{33}\ v_{34}\ v_{35}.\ P'\ (\texttt{reps}\ v_{33}\ v_{34}\ v_{35})) \land$
$(\forall v_{38}\ v_{39}.\ P'\ (v_{38}\ \texttt{domi}\ v_{39})) \land$
$(\forall v_{42}\ v_{43}.\ P'\ (v_{42}\ \texttt{eqi}\ v_{43})) \land$
$(\forall v_{46}\ v_{47}.\ P'\ (v_{46}\ \texttt{doms}\ v_{47})) \land$

$(\forall v_{50}\ v_{51}.\ P'\ (v_{50}\ \texttt{eqs}\ v_{51}))\ \land$
$(\forall v_{54}\ v_{55}.\ P'\ (v_{54}\ \texttt{eqn}\ v_{55}))\ \land$
$(\forall v_{58}\ v_{59}.\ P'\ (v_{58}\ \texttt{lte}\ v_{59}))\ \land$
$(\forall v_{62}\ v_{63}.\ P'\ (v_{62}\ \texttt{lt}\ v_{63}))\ \Rightarrow$
$\forall v.\ P'\ v$

[extractPropCommand_def]

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

[extractPropCommand_ind]

$\vdash \forall P'.$
$(\forall P\ \mathit{cmd}.\ P'\ (P\ \texttt{says prop}\ (\texttt{SOME}\ \mathit{cmd})))\ \land\ P'\ \texttt{TT}\ \land\ P'\ \texttt{FF}\ \land$
$(\forall v_{1}.\ P'\ (\texttt{prop}\ v_{1}))\ \land\ (\forall v_{3}.\ P'\ (\texttt{notf}\ v_{3}))\ \land$
$(\forall v_{6}\ v_{7}.\ P'\ (v_{6}\ \texttt{andf}\ v_{7}))\ \land\ (\forall v_{10}\ v_{11}.\ P'\ (v_{10}\ \texttt{orf}\ v_{11}))\ \land$
$(\forall v_{14}\ v_{15}.\ P'\ (v_{14}\ \texttt{impf}\ v_{15}))\ \land$
$(\forall v_{18}\ v_{19}.\ P'\ (v_{18}\ \texttt{eqf}\ v_{19}))\ \land\ (\forall \mathit{v129}.\ P'\ (\mathit{v129}\ \texttt{says TT}))\ \land$
$(\forall \mathit{v130}.\ P'\ (\mathit{v130}\ \texttt{says FF}))\ \land$
$(\forall \mathit{v132}.\ P'\ (\mathit{v132}\ \texttt{says prop NONE}))\ \land$
$(\forall \mathit{v133}\ v_{66}.\ P'\ (\mathit{v133}\ \texttt{says notf}\ v_{66}))\ \land$
$(\forall \mathit{v134}\ v_{69}\ v_{70}.\ P'\ (\mathit{v134}\ \texttt{says}\ (v_{69}\ \texttt{andf}\ v_{70})))\ \land$
$(\forall \mathit{v135}\ v_{73}\ v_{74}.\ P'\ (\mathit{v135}\ \texttt{says}\ (v_{73}\ \texttt{orf}\ v_{74})))\ \land$
$(\forall \mathit{v136}\ v_{77}\ v_{78}.\ P'\ (\mathit{v136}\ \texttt{says}\ (v_{77}\ \texttt{impf}\ v_{78})))\ \land$
$(\forall \mathit{v137}\ v_{81}\ v_{82}.\ P'\ (\mathit{v137}\ \texttt{says}\ (v_{81}\ \texttt{eqf}\ v_{82})))\ \land$
$(\forall \mathit{v138}\ v_{85}\ v_{86}.\ P'\ (\mathit{v138}\ \texttt{says}\ v_{85}\ \texttt{says}\ v_{86}))\ \land$
$(\forall \mathit{v139}\ v_{89}\ v_{90}.\ P'\ (\mathit{v139}\ \texttt{says}\ v_{89}\ \texttt{speaks\_for}\ v_{90}))\ \land$
$(\forall \mathit{v140}\ v_{93}\ v_{94}.\ P'\ (\mathit{v140}\ \texttt{says}\ v_{93}\ \texttt{controls}\ v_{94}))\ \land$
$(\forall \mathit{v141}\ v_{98}\ v_{99}\ \mathit{v100}.\ P'\ (\mathit{v141}\ \texttt{says reps}\ v_{98}\ v_{99}\ \mathit{v100}))\ \land$
$(\forall \mathit{v142}\ \mathit{v103}\ \mathit{v104}.\ P'\ (\mathit{v142}\ \texttt{says}\ \mathit{v103}\ \texttt{domi}\ \mathit{v104}))\ \land$
$(\forall \mathit{v143}\ \mathit{v107}\ \mathit{v108}.\ P'\ (\mathit{v143}\ \texttt{says}\ \mathit{v107}\ \texttt{eqi}\ \mathit{v108}))\ \land$
$(\forall \mathit{v144}\ \mathit{v111}\ \mathit{v112}.\ P'\ (\mathit{v144}\ \texttt{says}\ \mathit{v111}\ \texttt{doms}\ \mathit{v112}))\ \land$
$(\forall \mathit{v145}\ \mathit{v115}\ \mathit{v116}.\ P'\ (\mathit{v145}\ \texttt{says}\ \mathit{v115}\ \texttt{eqs}\ \mathit{v116}))\ \land$
$(\forall \mathit{v146}\ \mathit{v119}\ \mathit{v120}.\ P'\ (\mathit{v146}\ \texttt{says}\ \mathit{v119}\ \texttt{eqn}\ \mathit{v120}))\ \land$
$(\forall \mathit{v147}\ \mathit{v123}\ \mathit{v124}.\ P'\ (\mathit{v147}\ \texttt{says}\ \mathit{v123}\ \texttt{lte}\ \mathit{v124}))\ \land$
$(\forall \mathit{v148}\ \mathit{v127}\ \mathit{v128}.\ P'\ (\mathit{v148}\ \texttt{says}\ \mathit{v127}\ \texttt{lt}\ \mathit{v128}))\ \land$
$(\forall v_{24}\ v_{25}.\ P'\ (v_{24}\ \texttt{speaks\_for}\ v_{25}))\ \land$
$(\forall v_{28}\ v_{29}.\ P'\ (v_{28}\ \texttt{controls}\ v_{29}))\ \land$
$(\forall v_{33}\ v_{34}\ v_{35}.\ P'\ (\texttt{reps}\ v_{33}\ v_{34}\ v_{35}))\ \land$
$(\forall v_{38}\ v_{39}.\ P'\ (v_{38}\ \texttt{domi}\ v_{39}))\ \land$
$(\forall v_{42}\ v_{43}.\ P'\ (v_{42}\ \texttt{eqi}\ v_{43}))\ \land$
$(\forall v_{46}\ v_{47}.\ P'\ (v_{46}\ \texttt{doms}\ v_{47}))\ \land$
$(\forall v_{50}\ v_{51}.\ P'\ (v_{50}\ \texttt{eqs}\ v_{51}))\ \land$
$(\forall v_{54}\ v_{55}.\ P'\ (v_{54}\ \texttt{eqn}\ v_{55}))\ \land$
$(\forall v_{58}\ v_{59}.\ P'\ (v_{58}\ \texttt{lte}\ v_{59}))\ \land$
$(\forall v_{62}\ v_{63}.\ P'\ (v_{62}\ \texttt{lt}\ v_{63}))\ \Rightarrow$
$\forall v.\ P'\ v$

[TR_cases]

$\vdash \forall a_{0}\ a_{1}\ a_{2}\ a_{3}.$
$\texttt{TR}\ a_{0}\ a_{1}\ a_{2}\ a_{3}\ \iff$

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

[TR_discard_cmd_rule]

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

[TR_EQ_rules_thm]

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

[TR_exec_cmd_rule]

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

[TR_ind]

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

[TR_rules]

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

[TR_strongind]

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

[TR_trap_cmd_rule]

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

[TRrule0]

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

[TRrule1]

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

[trType_distinct_clauses]

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

[trType_one_one]

```
⊢ (∀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 satList Theory

Built: 10 June 2018

Parent Theories: aclDrules

3.1 Definitions

[satList_def]

```
⊢ ∀M Oi Os formList.
    (M,Oi,Os) satList formList ⇔
    FOLDR (λx y. x ∧ y) T (MAP (λf. (M,Oi,Os) sat f) formList)
```

3.2 Theorems

[satList_conj]

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

[satList_CONS]

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

[satList_nil]

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

Index