Contents

1 ssm1 Theory

Built: 09 April 2019
Parent Theories: satList

1.1 Datatypes

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

inst = SOME 'command | NONE

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

1.2 Definitions

[TR_def]

$$
\begin{array}{l}
\vdash \texttt{TR} = \\
\quad (\lambda\, a_0\ a_1\ a_2\ a_3.\\
\qquad \forall\, TR'.\\
\qquad\quad (\forall\, a_0\ a_1\ a_2\ a_3.\\
\qquad\qquad (\exists\, inputTest\ P\ NS\ M\ Oi\ Os\ Out\ s\ certList\ stateInterp\\
\qquad\qquad\quad cmd\ ins\ outs.\\
\qquad\qquad\quad (a_0 = (M, Oi, Os)) \wedge (a_1 = \texttt{exec}\ cmd) \wedge\\
\qquad\qquad\quad (a_2 =\\
\qquad\qquad\quad\ \texttt{CFG}\ inputTest\ stateInterp\ certList\\
\qquad\qquad\quad\ \ (P\ \texttt{says prop (SOME}\ cmd\texttt{)::}ins)\ s\ outs) \wedge\\
\qquad\qquad\quad (a_3 =\\
\qquad\qquad\quad\ \texttt{CFG}\ inputTest\ stateInterp\ certList\ ins\\
\qquad\qquad\quad\ \ (NS\ s\ \texttt{(exec}\ cmd\texttt{))}\ (Out\ s\ \texttt{(exec}\ cmd\texttt{)::}outs)) \wedge\\
\qquad\qquad\quad inputTest\ (P\ \texttt{says prop (SOME}\ cmd\texttt{))} \wedge\\
\qquad\qquad\quad \texttt{CFGInterpret}\ (M, Oi, Os)\\
\qquad\qquad\quad\ \ (\texttt{CFG}\ inputTest\ stateInterp\ certList\\
\qquad\qquad\quad\ \ \ (P\ \texttt{says prop (SOME}\ cmd\texttt{)::}ins)\ s\ outs)) \vee\\
\qquad\qquad (\exists\, inputTest\ P\ NS\ M\ Oi\ Os\ Out\ s\ certList\ stateInterp\\
\qquad\qquad\quad cmd\ ins\ outs.\\
\qquad\qquad\quad (a_0 = (M, Oi, Os)) \wedge (a_1 = \texttt{trap}\ cmd) \wedge\\
\qquad\qquad\quad (a_2 =\\
\qquad\qquad\quad\ \texttt{CFG}\ inputTest\ stateInterp\ certList\\
\qquad\qquad\quad\ \ (P\ \texttt{says prop (SOME}\ cmd\texttt{)::}ins)\ s\ outs) \wedge
\end{array}
$$

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

1.3 Theorems

[CFGInterpret_def]

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

[CFGInterpret_ind]

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

[configuration_one_one]

$$
\begin{array}{l}
\vdash \forall a_0\ a_1\ a_2\ a_3\ a_4\ a_5\ a'_0\ a'_1\ a'_2\ a'_3\ a'_4\ a'_5. \\
\quad (\texttt{CFG}\ a_0\ a_1\ a_2\ a_3\ a_4\ a_5 = \texttt{CFG}\ a'_0\ a'_1\ a'_2\ a'_3\ a'_4\ a'_5) \iff \\
\quad (a_0 = a'_0) \wedge (a_1 = a'_1) \wedge (a_2 = a'_2) \wedge (a_3 = a'_3) \wedge \\
\quad (a_4 = a'_4) \wedge (a_5 = a'_5)
\end{array}
$$

[inst_distinct_clauses]

$\vdash \forall a.$ SOME $a \neq$ NONE

[inst_one_one]

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

[TR_cases]

$\vdash \forall a_0\ a_1\ a_2\ a_3.$
TR $a_0\ a_1\ a_2\ a_3 \iff$
($\exists$ *inputTest P NS M Oi Os Out s certList stateInterp cmd ins outs*.
(a_0 = (M, Oi, Os)) $\wedge$ (a_1 = exec *cmd*) $\wedge$
(a_2 =
CFG *inputTest stateInterp certList*
(*P* says prop (SOME *cmd*)::*ins*) *s outs*) $\wedge$
(a_3 =
CFG *inputTest stateInterp certList ins*
(*NS s* (exec *cmd*)) (*Out s* (exec *cmd*)::*outs*)) $\wedge$
inputTest (*P* says prop (SOME *cmd*)) $\wedge$
CFGInterpret (M, Oi, Os)
(CFG *inputTest stateInterp certList*
(*P* says prop (SOME *cmd*)::*ins*) *s outs*)) $\vee$
($\exists$ *inputTest P NS M Oi Os Out s certList stateInterp cmd ins outs*.
(a_0 = (M, Oi, Os)) $\wedge$ (a_1 = trap *cmd*) $\wedge$
(a_2 =
CFG *inputTest stateInterp certList*
(*P* says prop (SOME *cmd*)::*ins*) *s outs*) $\wedge$
(a_3 =
CFG *inputTest stateInterp certList ins*
(*NS s* (trap *cmd*)) (*Out s* (trap *cmd*)::*outs*)) $\wedge$
inputTest (*P* says prop (SOME *cmd*)) $\wedge$
CFGInterpret (M, Oi, Os)
(CFG *inputTest stateInterp certList*
(*P* says prop (SOME *cmd*)::*ins*) *s outs*)) $\vee$
$\exists$ *inputTest NS M Oi Os Out s certList stateInterp cmd x ins outs*.
(a_0 = (M, Oi, Os)) $\wedge$ (a_1 = discard) $\wedge$
(a_2 =
CFG *inputTest stateInterp certList* (*x*::*ins*) *s outs*) $\wedge$
(a_3 =
CFG *inputTest stateInterp certList ins* (*NS s* discard)
(*Out s* discard::*outs*)) $\wedge$ $\neg$*inputTest x*

[TR_discard_cmd_rule]

```
⊢ TR (M,Oi,Os) discard
    (CFG inputTest stateInterp certList (x::ins) s outs)
    (CFG inputTest stateInterp certList ins (NS s discard)
       (Out s discard::outs)) ⟺ ¬inputTest x
```

[TR_EQ_rules_thm]

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

[TR_exec_cmd_rule]

```
⊢ ∀inputTest certList stateInterp P cmd ins s outs.
    (∀M Oi Os.
       CFGInterpret (M,Oi,Os)
         (CFG inputTest stateInterp certList
            (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 inputTest stateInterp certList
           (P says prop (SOME cmd)::ins) s outs)
        (CFG inputTest stateInterp certList ins
           (NS s (exec cmd)) (Out s (exec cmd)::outs)) ⟺
      inputTest (P says prop (SOME cmd)) ∧
```

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

[TR_ind]

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

[TR_rules]

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

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

[TR_strongind]

```
 ⊢ ∀ TR'.
     (∀ inputTest P NS M Oi Os Out s certList stateInterp cmd ins
        outs.
        inputTest (P says prop (SOME cmd)) ∧
        CFGInterpret (M, Oi, Os)
          (CFG inputTest stateInterp certList
             (P says prop (SOME cmd)::ins) s outs) ⇒
        TR' (M, Oi, Os) (exec cmd)
          (CFG inputTest stateInterp certList
             (P says prop (SOME cmd)::ins) s outs)
          (CFG inputTest stateInterp certList ins
             (NS s (exec cmd)) (Out s (exec cmd)::outs))) ∧
     (∀ inputTest P NS M Oi Os Out s certList stateInterp cmd ins
        outs.
        inputTest (P says prop (SOME cmd)) ∧
        CFGInterpret (M, Oi, Os)
          (CFG inputTest stateInterp certList
             (P says prop (SOME cmd)::ins) s outs) ⇒
        TR' (M, Oi, Os) (trap cmd)
          (CFG inputTest stateInterp certList
```

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

[TR_trap_cmd_rule]

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

[TRrule0]

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

[TRrule1]

```
 ⊢ TR (M,Oi,Os) (trap cmd)
     (CFG inputTest stateInterp certList
```

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

[trType_distinct_clauses]

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

[trType_one_one]

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

2 satList Theory

Built: 09 April 2019

Parent Theories: aclDrules

2.1 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)
```

2.2 Theorems

[satList_conj]

```
⊢ ∀l1 l2 M Oi Os.
    (M,Oi,Os) satList l1 ∧ (M,Oi,Os) satList l2 ⟺
    (M,Oi,Os) satList (l1 ++ l2)
```

[satList_CONS]

```
⊢ ∀h t M Oi Os.
    (M,Oi,Os) satList (h::t) ⟺
    (M,Oi,Os) sat h ∧ (M,Oi,Os) satList t
```

[satList_nil]

```
⊢ (M,Oi,Os) satList []
```

Index