# Contents

# 1  ssm1 Theory

**Built:** 16 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]

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

$(a_3 =$
  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*) $\Rightarrow$
$TR'$ $a_0$ $a_1$ $a_2$ $a_3$) $\Rightarrow$
$TR'$ $a_0$ $a_1$ $a_2$ $a_3$)

## 1.3   Theorems

[CFGInterpret_def]
$\vdash$ CFGInterpret ($M$,$Oi$,$Os$)
  (CFG *inputTest stateInterp context* ($x$::*ins*) *state*
   *outStream*) $\iff$
(($M$,$Oi$,$Os$) satList *context* $\wedge$ ($M$,$Oi$,$Os$) sat *x* $\wedge$
($M$,$Oi$,$Os$) sat *stateInterp state*

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

[configuration_one_one]
$\vdash$ $\forall$ $a_0$ $a_1$ $a_2$ $a_3$ $a_4$ $a_5$ $a'_0$ $a'_1$ $a'_2$ $a'_3$ $a'_4$ $a'_5$.
  (CFG $a_0$ $a_1$ $a_2$ $a_3$ $a_4$ $a_5$ = CFG $a'_0$ $a'_1$ $a'_2$ $a'_3$ $a'_4$ $a'_5$) $\iff$
  ($a_0 = a'_0$) $\wedge$ ($a_1 = a'_1$) $\wedge$ ($a_2 = a'_2$) $\wedge$ ($a_3 = a'_3$) $\wedge$
  ($a_4 = a'_4$) $\wedge$ ($a_5 = a'_5$)

[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]

$\vdash \forall\, TR'.$

    ($\forall$ *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*))) ∧
    ($\forall$ *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*))) ∧
    ($\forall$ *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*))) ⇒
    $\forall\, a_0\ a_1\ a_2\ a_3.$ `TR` $a_0\ a_1\ a_2\ a_3$ ⇒ *TR'* $a_0\ a_1\ a_2\ a_3$

[TR_rules]

$\vdash$ ($\forall$ *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` $cmd)$`::`$outs)))$ $\wedge$
      $(\forall\, inputTest$ $NS$ $M$ $Oi$ $Os$ $Out$ $s$ $certList$ $stateInterp$ $x$ $ins$
            $outs$.
          $\neg inputTest$ $x$ $\Rightarrow$
          $TR'$ $(M,Oi,Os)$ `discard`
            `(CFG` $inputTest$ $stateInterp$ $certList$ $(x$`::`$ins)$ $s$ $outs)$
            `(CFG` $inputTest$ $stateInterp$ $certList$ $ins$ `(NS` $s$ `discard)`
                $(Out$ $s$ `discard`::$outs)))$ $\Rightarrow$
      $\forall\, a_0$ $a_1$ $a_2$ $a_3$. `TR` $a_0$ $a_1$ $a_2$ $a_3$ $\Rightarrow$ $TR'$ $a_0$ $a_1$ $a_2$ $a_3$

$[$`TR_trap_cmd_rule`$]$

$\vdash$ $\forall\, inputTest$ $stateInterp$ $certList$ $P$ $cmd$ $ins$ $s$ $outs$.
      $(\forall\, M$ $Oi$ $Os$.
          `CFGInterpret` $(M,Oi,Os)$
            `(CFG` $inputTest$ $stateInterp$ $certList$
                $(P$ `says prop` `(SOME` $cmd)$`::`$ins)$ $s$ $outs)$ $\Rightarrow$
          $(M,Oi,Os)$ `sat prop NONE)` $\Rightarrow$
      $\forall\, NS$ $Out$ $M$ $Oi$ $Os$.
        `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` $cmd)$`::`$outs))$ $\iff$
        $inputTest$ $(P$ `says prop` `(SOME` $cmd))$ $\wedge$
        `CFGInterpret` $(M,Oi,Os)$
          `(CFG` $inputTest$ $stateInterp$ $certList$
              $(P$ `says prop` `(SOME` $cmd)$`::`$ins)$ $s$ $outs)$ $\wedge$
        $(M,Oi,Os)$ `sat prop NONE`

$[$`TRrule0`$]$

$\vdash$ `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` $cmd)$`::`$outs))$ $\iff$
    $inputTest$ $(P$ `says prop` `(SOME` $cmd))$ $\wedge$
    `CFGInterpret` $(M,Oi,Os)$
      `(CFG` $inputTest$ $stateInterp$ $certList$
          $(P$ `says prop` `(SOME` $cmd)$`::`$ins)$ $s$ $outs)$

$[$`TRrule1`$]$

$\vdash$ `TR` $(M,Oi,Os)$ `(trap` $cmd)$
      `(CFG` $inputTest$ $stateInterp$ $certList$

$$(P \text{ says prop (SOME } cmd)::ins) \ s \ outs)$$
$$(\text{CFG } inputTest \ stateInterp \ certList \ ins \ (NS \ s \ (\text{trap } cmd))$$
$$(Out \ s \ (\text{trap } cmd)::outs)) \iff$$
$$inputTest \ (P \text{ says prop (SOME } cmd)) \ \wedge$$
$$\text{CFGInterpret } (M, Oi, Os)$$
$$(\text{CFG } inputTest \ stateInterp \ certList$$
$$(P \text{ says prop (SOME } cmd)::ins) \ s \ outs)$$

[**trType_distinct_clauses**]

$\vdash (\forall a. \ \text{discard} \neq \text{trap } a) \ \wedge \ (\forall a. \ \text{discard} \neq \text{exec } a) \ \wedge$
$\quad \forall a' \ a. \ \text{trap } a \neq \text{exec } a'$

[**trType_one_one**]

$\vdash (\forall a \ a'. \ (\text{trap } a = \text{trap } a') \iff (a = a')) \ \wedge$
$\quad \forall a \ a'. \ (\text{exec } a = \text{exec } a') \iff (a = a')$

# 2 satList Theory

**Built:** 16 April 2019

**Parent Theories:** aclDrules

## 2.1 Definitions

[**satList_def**]

$\vdash \forall M \ Oi \ Os \ formList.$
$\quad (M, Oi, Os) \ \text{satList } formList \iff$
$\quad \text{FOLDR } (\lambda x \ y. \ x \wedge y) \ \text{T (MAP } (\lambda f. \ (M, Oi, Os) \ \text{sat } f) \ formList)$

## 2.2 Theorems

[**satList_conj**]

$\vdash \forall l_1 \ l_2 \ M \ Oi \ Os.$
$\quad (M, Oi, Os) \ \text{satList } l_1 \ \wedge \ (M, Oi, Os) \ \text{satList } l_2 \iff$
$\quad (M, Oi, Os) \ \text{satList } (l_1 \ ++ \ l_2)$

[**satList_CONS**]

$\vdash \forall h \ t \ M \ Oi \ Os.$
$\quad (M, Oi, Os) \ \text{satList } (h::t) \iff$
$\quad (M, Oi, Os) \ \text{sat } h \ \wedge \ (M, Oi, Os) \ \text{satList } t$

[**satList_nil**]

$\vdash (M, Oi, Os) \ \text{satList } []$

# Index