Contents

1   ssm1 Theory

Built: 23 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 =
$\quad (\lambda\, a_0 \;\; a_1 \;\; a_2 \;\; a_3.$
$\qquad \forall\, TR'.$
$\qquad\quad (\forall\, a_0 \;\; a_1 \;\; a_2 \;\; a_3.$
$\qquad\qquad (\exists\, inputTest \;\; P \;\; NS \;\; M \;\; Oi \;\; Os \;\; Out \;\; s \;\; certList \;\; stateInterp$
$\qquad\qquad\quad cmd \;\; ins \;\; outs.$
$\qquad\qquad\quad (a_0 = (M, Oi, Os)) \;\wedge\; (a_1 = \text{exec } cmd) \;\wedge\;$
$\qquad\qquad\quad (a_2 =$
$\qquad\qquad\quad\; \text{CFG } inputTest \;\; stateInterp \;\; certList$
$\qquad\qquad\qquad (P \text{ says prop (SOME } cmd)::ins) \;\; s \;\; outs) \;\wedge\;$
$\qquad\qquad\quad (a_3 =$
$\qquad\qquad\quad\; \text{CFG } inputTest \;\; stateInterp \;\; certList \;\; ins$
$\qquad\qquad\qquad (NS \;\; s \;\; (\text{exec } cmd)) \;\; (Out \;\; s \;\; (\text{exec } cmd)::outs)) \;\wedge\;$
$\qquad\qquad\quad inputTest \;\; (P \text{ says prop (SOME } cmd)) \;\wedge\;$
$\qquad\qquad\quad \text{CFGInterpret } (M, Oi, Os)$
$\qquad\qquad\qquad (\text{CFG } inputTest \;\; stateInterp \;\; certList$
$\qquad\qquad\qquad\quad (P \text{ says prop (SOME } cmd)::ins) \;\; s \;\; outs)) \;\vee\;$
$\qquad\qquad (\exists\, inputTest \;\; P \;\; NS \;\; M \;\; Oi \;\; Os \;\; Out \;\; s \;\; certList \;\; stateInterp$
$\qquad\qquad\quad cmd \;\; ins \;\; outs.$
$\qquad\qquad\quad (a_0 = (M, Oi, Os)) \;\wedge\; (a_1 = \text{trap } cmd) \;\wedge\;$
$\qquad\qquad\quad (a_2 =$
$\qquad\qquad\quad\; \text{CFG } inputTest \;\; stateInterp \;\; certList$
$\qquad\qquad\qquad (P \text{ says prop (SOME } cmd)::ins) \;\; s \;\; outs) \;\wedge\;$

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

[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*))) ⇒
    ∀ 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]

⊢ (∀ *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 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: 23 April 2019
Parent Theories: aclDrules

2.1 Definitions

[satList_def]

⊢ ∀ M Oi Os formList.
    (M,Oi,Os) satList formList ⟺
    FOLDR (λ x y. x ∧ y) T (MAP (λ f. (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