

## Contents

<b>1</b>	<b>ssm1 Theory</b>	<b>3</b>
1.1	Datatypes . . . . .	3
1.2	Definitions . . . . .	3
1.3	Theorems . . . . .	4
<b>2</b>	<b>satList Theory</b>	<b>10</b>
2.1	Definitions . . . . .	10
2.2	Theorems . . . . .	10



# 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

[\[TR\\_def\]](#)

```
⊢ TR =
  (λ a0 a1 a2 a3.
    ∀ TR'.
      (∀ a0 a1 a2 a3.
        (∃ inputTest P NS M Oi Os Out s certList stateInterp
          cmd ins outs.
          (a0 = (M, Oi, Os)) ∧ (a1 = exec cmd) ∧
          (a2 =
            CFG inputTest stateInterp certList
              (P says prop (SOME cmd)::ins) s outs) ∧
          (a3 =
            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)) ∨
        (∃ inputTest P NS M Oi Os Out s certList stateInterp
          cmd ins outs.
          (a0 = (M, Oi, Os)) ∧ (a1 = trap cmd) ∧
          (a2 =
            CFG inputTest stateInterp certList
              (P says prop (SOME cmd)::ins) s outs) ∧
```

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

### 1.3 Theorems

#### [CFGInterpret\_def]

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

#### [CFGInterpret\_ind]

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

#### [configuration\_one\_one]

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

[inst\_distinct\_clauses]

$\vdash \forall a. \text{SOME } a \neq \text{NONE}$

[inst\_one\_one]

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

[TR\_cases]

$\vdash \forall a_0 \ a_1 \ a_2 \ a_3.$

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

**[TR\_discard\_cmd\_rule]**

$$\vdash \text{TR } (M, Oi, Os) \text{ discard}$$

$$\quad (\text{CFG inputTest stateInterp certList } (x::ins) \ s \ outs)$$

$$\quad (\text{CFG inputTest stateInterp certList ins } (NS \ s \ \text{discard}))$$

$$\quad (\text{Out } s \ \text{discard}::outs)) \iff \neg \text{inputTest } x$$
**[TR\_EQ\_rules\_thm]**

$$\vdash (\text{TR } (M, Oi, Os) \ (\text{exec } cmd))$$

$$\quad (\text{CFG inputTest stateInterp certList}$$

$$\quad \quad (P \ \text{says prop } (\text{SOME } cmd)::ins) \ s \ outs)$$

$$\quad (\text{CFG inputTest stateInterp certList ins } (NS \ s \ (\text{exec } cmd)))$$

$$\quad (\text{Out } s \ (\text{exec } cmd)::outs)) \iff$$

$$\text{inputTest } (P \ \text{says prop } (\text{SOME } cmd)) \wedge$$

$$\text{CFGInterpret } (M, Oi, Os)$$

$$\quad (\text{CFG inputTest stateInterp certList}$$

$$\quad \quad (P \ \text{says prop } (\text{SOME } cmd)::ins) \ s \ outs)) \wedge$$

$$(\text{TR } (M, Oi, Os) \ (\text{trap } cmd))$$

$$\quad (\text{CFG inputTest stateInterp certList}$$

$$\quad \quad (P \ \text{says prop } (\text{SOME } cmd)::ins) \ s \ outs)$$

$$\quad (\text{CFG inputTest stateInterp certList ins } (NS \ s \ (\text{trap } cmd)))$$

$$\quad (\text{Out } s \ (\text{trap } cmd)::outs)) \iff$$

$$\text{inputTest } (P \ \text{says prop } (\text{SOME } cmd)) \wedge$$

$$\text{CFGInterpret } (M, Oi, Os)$$

$$\quad (\text{CFG inputTest stateInterp certList}$$

$$\quad \quad (P \ \text{says prop } (\text{SOME } cmd)::ins) \ s \ outs)) \wedge$$

$$(\text{TR } (M, Oi, Os) \ \text{discard})$$

$$\quad (\text{CFG inputTest stateInterp certList } (x::ins) \ s \ outs)$$

$$\quad (\text{CFG inputTest stateInterp certList ins } (NS \ s \ \text{discard}))$$

$$\quad (\text{Out } s \ \text{discard}::outs)) \iff \neg \text{inputTest } x$$
**[TR\_exec\_cmd\_rule]**

$$\vdash \forall \text{inputTest certList stateInterp } P \ cmd \ ins \ s \ outs.$$

$$\quad (\forall M \ Oi \ Os.$$

$$\quad \quad \text{CFGInterpret } (M, Oi, Os)$$

$$\quad \quad (\text{CFG inputTest stateInterp certList}$$

$$\quad \quad \quad (P \ \text{says prop } (\text{SOME } cmd)::ins) \ s \ outs) \Rightarrow$$

$$\quad \quad (M, Oi, Os) \ \text{sat prop } (\text{SOME } cmd)) \Rightarrow$$

$$\forall NS \ Out \ M \ Oi \ Os.$$

$$\quad \text{TR } (M, Oi, Os) \ (\text{exec } cmd)$$

$$\quad \quad (\text{CFG inputTest stateInterp certList}$$

$$\quad \quad \quad (P \ \text{says prop } (\text{SOME } cmd)::ins) \ s \ outs)$$

$$\quad \quad (\text{CFG inputTest stateInterp certList ins}$$

$$\quad \quad \quad (NS \ s \ (\text{exec } cmd)) \ (\text{Out } s \ (\text{exec } cmd)::outs)) \iff$$

$$\text{inputTest } (P \ \text{says prop } (\text{SOME } cmd)) \wedge$$

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<sub>0</sub> a<sub>1</sub> a<sub>2</sub> a<sub>3</sub>. TR a<sub>0</sub> a<sub>1</sub> a<sub>2</sub> a<sub>3</sub> ⇒ TR' a<sub>0</sub> a<sub>1</sub> a<sub>2</sub> a<sub>3</sub>

[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

```



$$\begin{aligned}
& (P \text{ says prop (SOME cmd)::ins) } s \text{ outs} \\
& (\text{CFG inputTest stateInterp certList ins} \\
& \quad (\text{NS } s \text{ (trap cmd)) (Out } s \text{ (trap cmd)::outs))) } \wedge \\
& (\forall \text{inputTest NS } M \text{ } O_i \text{ } O_s \text{ Out } s \text{ certList stateInterp } x \text{ ins} \\
& \quad \text{outs.} \\
& \neg \text{inputTest } x \Rightarrow \\
& \text{TR}' (M, O_i, O_s) \text{ discard} \\
& (\text{CFG inputTest stateInterp certList (x::ins) } s \text{ outs} \\
& \quad (\text{CFG inputTest stateInterp certList ins (NS } s \text{ discard)} \\
& \quad \quad (\text{Out } s \text{ discard::outs}))) \Rightarrow \\
& \forall a_0 \ a_1 \ a_2 \ a_3. \text{ TR } a_0 \ a_1 \ a_2 \ a_3 \Rightarrow \text{TR}' a_0 \ a_1 \ a_2 \ a_3
\end{aligned}$$
**[TR\_trap\_cmd\_rule]**

$$\begin{aligned}
& \vdash \forall \text{inputTest stateInterp certList } P \text{ cmd ins } s \text{ outs.} \\
& \quad (\forall M \text{ } O_i \text{ } O_s. \\
& \quad \quad \text{CFGInterpret } (M, O_i, O_s) \\
& \quad \quad (\text{CFG inputTest stateInterp certList} \\
& \quad \quad \quad (P \text{ says prop (SOME cmd)::ins) } s \text{ outs}) \Rightarrow \\
& \quad \quad (M, O_i, O_s) \text{ sat prop NONE}) \Rightarrow \\
& \forall \text{NS Out } M \text{ } O_i \text{ } O_s. \\
& \quad \text{TR } (M, O_i, O_s) \text{ (trap cmd)} \\
& \quad (\text{CFG inputTest stateInterp certList} \\
& \quad \quad (P \text{ says prop (SOME cmd)::ins) } s \text{ outs}) \\
& \quad (\text{CFG inputTest stateInterp certList ins} \\
& \quad \quad (\text{NS } s \text{ (trap cmd)) (Out } s \text{ (trap cmd)::outs})) \iff \\
& \text{inputTest } (P \text{ says prop (SOME cmd))} \wedge \\
& \text{CFGInterpret } (M, O_i, O_s) \\
& \quad (\text{CFG inputTest stateInterp certList} \\
& \quad \quad (P \text{ says prop (SOME cmd)::ins) } s \text{ outs}) \wedge \\
& (M, O_i, O_s) \text{ sat prop NONE}
\end{aligned}$$
**[TRrule0]**

$$\begin{aligned}
& \vdash \text{TR } (M, O_i, O_s) \text{ (exec cmd)} \\
& \quad (\text{CFG inputTest stateInterp certList} \\
& \quad \quad (P \text{ says prop (SOME cmd)::ins) } s \text{ outs}) \\
& \quad (\text{CFG inputTest stateInterp certList ins (NS } s \text{ (exec cmd))} \\
& \quad \quad (\text{Out } s \text{ (exec cmd)::outs})) \iff \\
& \text{inputTest } (P \text{ says prop (SOME cmd))} \wedge \\
& \text{CFGInterpret } (M, O_i, O_s) \\
& \quad (\text{CFG inputTest stateInterp certList} \\
& \quad \quad (P \text{ says prop (SOME cmd)::ins) } s \text{ outs})
\end{aligned}$$
**[TRrule1]**

$$\begin{aligned}
& \vdash \text{TR } (M, O_i, O_s) \text{ (trap cmd)} \\
& \quad (\text{CFG inputTest stateInterp certList}
\end{aligned}$$

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

[trType\_distinct\_clauses]

$$\begin{aligned}
 & \vdash (\forall a. \text{discard} \neq \text{trap } a) \wedge (\forall a. \text{discard} \neq \text{exec } a) \wedge \\
 & \forall a' a. \text{trap } a \neq \text{exec } a'
 \end{aligned}$$

[trType\_one\_one]

$$\begin{aligned}
 & \vdash (\forall a \ a'. (\text{trap } a = \text{trap } a') \iff (a = a')) \wedge \\
 & \forall a \ a'. (\text{exec } a = \text{exec } a') \iff (a = a')
 \end{aligned}$$

## 2 satList Theory

**Built:** 09 April 2019

**Parent Theories:** aclDrules

### 2.1 Definitions

[satList\_def]

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

### 2.2 Theorems

[satList\_conj]

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

[satList\_CONS]

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

[satList\_nil]

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

## Index

### **satList Theory**, 10

Definitions, 10

satList\_def, 10

Theorems, 10

satList\_conj, 10

satList\_CONS, 10

satList\_nil, 10

### **ssm1 Theory**, 3

Datatypes, 3

Definitions, 3

TR\_def, 3

Theorems, 4

CFGInterpret\_def, 4

CFGInterpret\_ind, 4

configuration\_one\_one, 4

inst\_distinct\_clauses, 5

inst\_one\_one, 5

TR\_cases, 5

TR\_discard\_cmd\_rule, 6

TR\_EQ\_rules\_thm, 6

TR\_exec\_cmd\_rule, 6

TR\_ind, 7

TR\_rules, 7

TR\_strongind, 8

TR\_trap\_cmd\_rule, 9

TRrule0, 9

TRrule1, 9

trType\_distinct\_clauses, 10

trType\_one\_one, 10