An Isabelle/HOL Formalization of the Modular Assembly Kit for Security Properties

Oliver Bračevac, Richard Gay, Sylvia Grewe,
Heiko Mantel, Henning Sudbrock, Markus Tasch

Abstract

The "Modular Assembly Kit for Security Properties" (MAKS) is a framework for both the definition and verification of possibilistic information-flow security properties at the specification-level. MAKS supports the uniform representation of a wide range of possibilistic information-flow properties and provides support for the verification of such properties via unwinding results and compositionality results. We provide a formalization of this framework in Isabelle/HOL.

Contents

1 Introduction

This is a formalization of the Modular Assembly Kit for Security Properties (MAKS) [2, 3] in its version from [3]. We provide a more detailed explanation on how key concepts of MAKS are formalized in Isabelle/HOL in [1].

2 Basic Definitions

In the following, we define the notion of prefixes and the notion of projection. These definitions are preliminaries for the remaining parts of the Isabelle/HOL formalization of MAKS.

theory *Prefix*
imports *Main*
begin

definition *prefix* :: $'e\ list \Rightarrow 'e\ list \Rightarrow bool$ (**infixl** $\preceq$ *100*)
where
$(l1 \preceq l2) \equiv (\exists l3.\ l1\ @\ l3 = l2)$

definition *prefixclosed* :: $('e\ list)\ set \Rightarrow bool$
where
$prefixclosed\ tr \equiv (\forall l1 \in tr.\ \forall l2.\ l2 \preceq l1 \longrightarrow l2 \in tr)$

lemma *empty-prefix-of-all*: $[] \preceq l$
⟨*proof*⟩

lemma *empty-trace-contained*: $[\![\ prefixclosed\ tr\ ;\ tr \neq \{\}\ ]\!] \Longrightarrow [] \in tr$
⟨*proof*⟩

lemma *transitive-prefix*: $[\![\ l1 \preceq l2\ ;\ l2 \preceq l3\ ]\!] \Longrightarrow l1 \preceq l3$
⟨*proof*⟩

end
theory *Projection*
imports *Main*
begin

definition *projection*:: $'e\ list \Rightarrow 'e\ set \Rightarrow 'e\ list$ (**infixl** $\upharpoonright$ *100*)
where
$l \upharpoonright E \equiv filter\ (\lambda x\ .\ x \in E)\ l$

lemma *projection-on-union*:
$l \upharpoonright Y = [] \Longrightarrow l \upharpoonright (X \cup Y) = l \upharpoonright X$
⟨*proof*⟩

lemma *projection-on-empty-trace*: $[] \upharpoonright X = []$ ⟨*proof*⟩

lemma *projection-to-emptyset-is-empty-trace*: $l \upharpoonright \{\} = []$ ⟨*proof*⟩

lemma *projection-idempotent*: $l \upharpoonright X = (l \upharpoonright X) \upharpoonright X$ ⟨*proof*⟩

lemma *projection-empty-implies-absence-of-events*: $l \upharpoonright X = [] \Longrightarrow X \cap (set\ l) = \{\}$
⟨*proof*⟩

lemma *disjoint-projection*: $X \cap Y = \{\} \Longrightarrow (l \upharpoonright X) \upharpoonright Y = []$
⟨*proof*⟩

lemma *projection-concatenation-commute*:
$(l1\ @\ l2) \upharpoonright X = (l1 \upharpoonright X)\ @\ (l2 \upharpoonright X)$
⟨*proof*⟩

lemma *projection-subset-eq-from-superset-eq*:
$((xs \upharpoonright (X \cup Y)) = (ys \upharpoonright (X \cup Y))) \Longrightarrow ((xs \upharpoonright X) = (ys \upharpoonright X))$
(**is** $(?L1 = ?L2) \Longrightarrow (?L3 = ?L4)$)
⟨*proof*⟩

lemma *list-subset-iff-projection-neutral*: $(set\ l \subseteq X) = ((l \upharpoonright X) = l)$
(**is** $?A = ?B$)
⟨*proof*⟩

lemma *projection-split-last*: $Suc\ n = length\ (\tau \upharpoonright X) \Longrightarrow$
$\exists\ \beta\ x\ \alpha.\ (x \in X \wedge \tau = \beta\ @\ [x]\ @\ \alpha \wedge \alpha \upharpoonright X = [] \wedge n = length\ ((\beta\ @\ \alpha) \upharpoonright X))$
⟨*proof*⟩

lemma *projection-rev-commute*:
$rev\ (l \upharpoonright X) = (rev\ l) \upharpoonright X$
⟨*proof*⟩

lemma *projection-split-first*: $[\![\ (\tau \upharpoonright X) = x\ \#\ xs\ ]\!] \Longrightarrow \exists\ \alpha\ \beta.\ (\tau = \alpha\ @\ [x]\ @\ \beta \wedge \alpha \upharpoonright X = [])$
⟨*proof*⟩

lemma *projection-split-first-with-suffix*:
$[\![\ (\tau \upharpoonright X) = x\ \#\ xs\ ]\!] \Longrightarrow \exists\ \alpha\ \beta.\ (\tau = \alpha\ @\ [x]\ @\ \beta \wedge \alpha \upharpoonright X = [] \wedge \beta \upharpoonright X = xs)$
⟨*proof*⟩

lemma *projection-split-arbitrary-element*:
$[\![\tau \upharpoonright X = (\alpha @ [x] @ \beta) \upharpoonright X;\ x \in X]\!]$
$\Longrightarrow \exists\ \alpha'\ \beta'.\ (\tau = \alpha' @ [x] @ \beta' \wedge \alpha' \upharpoonright X = \alpha \upharpoonright X \wedge \beta' \upharpoonright X = \beta \upharpoonright X)$
⟨*proof*⟩

lemma *projection-on-intersection*: $l \upharpoonright X = [] \Longrightarrow l \upharpoonright (X \cap Y) = []$
(**is** $?L1 = [] \Longrightarrow ?L2 = []$)
⟨*proof*⟩

lemma *projection-on-subset*: $[\![\ Y \subseteq X;\ l \upharpoonright X = []\ ]\!] \Longrightarrow l \upharpoonright Y = []$
⟨*proof*⟩

lemma *projection-on-subset2*: $[\![\ set\ l \subseteq L;\ l \upharpoonright X' = [];\ X \cap L \subseteq X'\ ]\!] \Longrightarrow l \upharpoonright X = []$
⟨*proof*⟩

lemma *non-empty-projection-on-subset*: $X \subseteq Y \wedge l_1 \upharpoonright Y = l_2 \upharpoonright Y \Longrightarrow l_1 \upharpoonright X = l_2 \upharpoonright X$
⟨*proof*⟩

lemma *projection-intersection-neutral*: $(set\ l \subseteq X) \Longrightarrow (l \upharpoonright (X \cap Y) = l \upharpoonright Y)$
⟨*proof*⟩

lemma *projection-commute*:
$(l \upharpoonright X) \upharpoonright Y = (l \upharpoonright Y) \upharpoonright X$
⟨*proof*⟩

lemma *projection-subset-elim*: $Y \subseteq X \Longrightarrow (l \upharpoonright X) \upharpoonright Y = l \upharpoonright Y$
⟨*proof*⟩

lemma *projection-sequence*: $(xs \upharpoonright X) \upharpoonright Y = (xs \upharpoonright (X \cap Y))$
⟨*proof*⟩

```
fun merge :: 'e set ⇒ 'e set ⇒ 'e list ⇒ 'e list ⇒ 'e list
where
merge A B [] t2 = t2 |
merge A B t1 [] = t1 |
merge A B (e1 # t1') (e2 # t2') = (if e1 = e2 then
                                      e1 # (merge A B t1' t2')
                                    else (if e1 ∈ (A ∩ B) then
                                           e2 # (merge A B (e1 # t1') t2')
                                         else e1 # (merge A B t1' (e2 # t2'))))
```

lemma *merge-property*: $[\![$set t1 $\subseteq$ A; set t2 $\subseteq$ B; t1 $\upharpoonright$ B = t2 $\upharpoonright$ A $]\!]$
$\Longrightarrow$ *let* t = (*merge* A B t1 t2) *in* (t $\upharpoonright$ A = t1 $\wedge$ t $\upharpoonright$ B = t2 $\wedge$ *set* t $\subseteq$ ((*set* t1) $\cup$ (*set* t2)))
$\langle$*proof*$\rangle$

end

3 System Specification

3.1 Event Systems

We define the system model of event systems as well as the parallel composition operator for event systems provided as part of MAKS in [3].

theory *EventSystems*
imports *../Basics/Prefix ../Basics/Projection*
begin

record *'e ES-rec* =
  E-ES :: *'e set*
  I-ES :: *'e set*
  O-ES :: *'e set*
  Tr-ES :: (*'e list*) *set*

abbreviation *ESrecEES* :: *'e ES-rec* ⇒ *'e set*
(*E*_ [*1000*] *1000*)
where
E_{ES} ≡ (*E-ES ES*)

abbreviation *ESrecIES* :: *'e ES-rec* ⇒ *'e set*
(*I*_ [*1000*] *1000*)
where
I_{ES} ≡ (*I-ES ES*)

abbreviation *ESrecOES* :: *'e ES-rec* ⇒ *'e set*
(*O*_ [*1000*] *1000*)
where
O_{ES} ≡ (*O-ES ES*)

abbreviation *ESrecTrES* :: *'e ES-rec* ⇒ (*'e list*) *set*
(*Tr*_ [*1000*] *1000*)
where
Tr_{ES} ≡ (*Tr-ES ES*)

definition *es-inputs-are-events* :: *'e ES-rec* ⇒ *bool*
where
es-inputs-are-events ES ≡ $I_{ES} \subseteq E_{ES}$

definition *es-outputs-are-events* :: $'e$ *ES-rec* $\Rightarrow$ *bool*
where
es-outputs-are-events ES $\equiv O_{ES} \subseteq E_{ES}$

definition *es-inputs-outputs-disjoint* :: $'e$ *ES-rec* $\Rightarrow$ *bool*
where
es-inputs-outputs-disjoint ES $\equiv I_{ES} \cap O_{ES} = \{\}$

definition *traces-contain-events* :: $'e$ *ES-rec* $\Rightarrow$ *bool*
where
traces-contain-events ES $\equiv \forall\, l \in Tr_{ES}.\ \forall\, e \in (set\ l).\ e \in E_{ES}$

definition *traces-prefixclosed* :: $'e$ *ES-rec* $\Rightarrow$ *bool*
where
traces-prefixclosed ES $\equiv$ *prefixclosed* Tr_{ES}

definition *ES-valid* :: $'e$ *ES-rec* $\Rightarrow$ *bool*
where
ES-valid ES $\equiv$
es-inputs-are-events ES $\wedge$ *es-outputs-are-events ES*
$\wedge$ *es-inputs-outputs-disjoint ES* $\wedge$ *traces-contain-events ES*
$\wedge$ *traces-prefixclosed ES*

definition *total* :: $'e$ *ES-rec* $\Rightarrow$ $'e$ *set* $\Rightarrow$ *bool*
where
total ES E $\equiv E \subseteq E_{ES} \wedge (\forall\, \tau \in Tr_{ES}.\ \forall\, e \in E.\ \tau\ @\ [e] \in Tr_{ES})$

lemma *totality*: $[\![$ *total ES E*; $t \in Tr_{ES}$; *set* $t' \subseteq E$ $]\!] \Longrightarrow t\ @\ t' \in Tr_{ES}$
⟨*proof*⟩

definition *composeES* :: $'e$ *ES-rec* $\Rightarrow$ $'e$ *ES-rec* $\Rightarrow$ $'e$ *ES-rec*
where
composeES ES1 ES2 $\equiv$
⦇
E-ES $= E_{ES1} \cup E_{ES2}$,
I-ES $= (I_{ES1} - O_{ES2}) \cup (I_{ES2} - O_{ES1})$,
O-ES $= (O_{ES1} - I_{ES2}) \cup (O_{ES2} - I_{ES1})$,
Tr-ES $= \{\tau\ .\ (\tau \upharpoonright E_{ES1}) \in Tr_{ES1} \wedge (\tau \upharpoonright E_{ES2}) \in Tr_{ES2}$
$\wedge\ (set\ \tau \subseteq E_{ES1} \cup E_{ES2})\}$
⦈

abbreviation *composeESAbbrv* :: $'e$ *ES-rec* $\Rightarrow$ $'e$ *ES-rec* $\Rightarrow$ $'e$ *ES-rec*
(- ‖ -[*1000*] *1000*)
where
ES1 ‖ *ES2* $\equiv$ (*composeES ES1 ES2*)

definition *composable* :: *'e ES-rec* ⇒ *'e ES-rec* ⇒ *bool*
where
composable ES1 ES2 ≡ $(E_{ES1} \cap E_{ES2}) \subseteq ((O_{ES1} \cap I_{ES2}) \cup (O_{ES2} \cap I_{ES1}))$

lemma *composeES-yields-ES*:
  ⟦ *ES-valid ES1*; *ES-valid ES2* ⟧ ⟹ *ES-valid* (*ES1* ∥ *ES2*)
  ⟨*proof*⟩

end

3.2 State-Event Systems

We define the system model of state-event systems as well as the translation from state-event systems to event systems provided as part of MAKS in [3]. State-event systems are the basis for the unwinding theorems that we prove later in this entry.

theory *StateEventSystems*
imports *EventSystems*
begin

record (*'s*, *'e*) *SES-rec* =
  S-SES :: *'s set*
  s0-SES :: *'s*
  E-SES :: *'e set*
  I-SES :: *'e set*
  O-SES :: *'e set*
  T-SES :: *'s* ⇒ *'e* ⇀ *'s*

abbreviation *SESrecSSES* :: (*'s*, *'e*) *SES-rec* ⇒ *'s set*
(*S*_ [*1000*] *1000*)
where
S_{SES} ≡ (*S-SES SES*)

abbreviation *SESrecs0SES* :: (*'s*, *'e*) *SES-rec* ⇒ *'s*
(*s0*_ [*1000*] *1000*)
where
$s0_{SES}$ ≡ (*s0-SES SES*)

abbreviation *SESrecESES* :: (*'s*, *'e*) *SES-rec* ⇒ *'e set*
(*E*_ [*1000*] *1000*)
where
E_{SES} ≡ (*E-SES SES*)

abbreviation *SESrecISES* :: (*'s*, *'e*) *SES-rec* ⇒ *'e set*
(*I*_ [*1000*] *1000*)
where
I_{SES} ≡ (*I-SES SES*)

abbreviation *SESrecOSES* :: ('s, 'e) *SES-rec* ⇒ 'e *set*
(*O*_ [*1000*] *1000*)
where
O_{SES} ≡ (*O-SES SES*)

abbreviation *SESrecTSES* :: ('s, 'e) *SES-rec* ⇒ ('s ⇒ 'e ⇀ 's)
(*T*_ [*1000*] *1000*)
where
T_{SES} ≡ (*T-SES SES*)

abbreviation *TSESpred* :: 's ⇒ 'e ⇒ ('s, 'e) *SES-rec* ⇒ 's ⇒ *bool*
(_ _⟶_ _ [*100*,*100*,*100*,*100*] *100*)
where
$s\ e\longrightarrow_{SES}\ s' \equiv (T_{SES}\ s\ e = Some\ s')$

definition *s0-is-state* :: ('s, 'e) *SES-rec* ⇒ *bool*
where
s0-is-state SES ≡ $s0_{SES} \in S_{SES}$

definition *ses-inputs-are-events* :: ('s, 'e) *SES-rec* ⇒ *bool*
where
ses-inputs-are-events SES ≡ $I_{SES} \subseteq E_{SES}$

definition *ses-outputs-are-events* :: ('s, 'e) *SES-rec* ⇒ *bool*
where
ses-outputs-are-events SES ≡ $O_{SES} \subseteq E_{SES}$

definition *ses-inputs-outputs-disjoint* :: ('s, 'e) *SES-rec* ⇒ *bool*
where
ses-inputs-outputs-disjoint SES ≡ $I_{SES} \cap O_{SES} = \{\}$

definition *correct-transition-relation* :: ('s, 'e) *SES-rec* ⇒ *bool*
where
correct-transition-relation SES ≡
$\forall x\ y\ z.\ x\ y\longrightarrow_{SES} z \longrightarrow ((x \in S_{SES}) \wedge (y \in E_{SES}) \wedge (z \in S_{SES}))$

definition *SES-valid* :: ('s, 'e) *SES-rec* ⇒ *bool*
where
SES-valid SES ≡
  s0-is-state SES ∧ *ses-inputs-are-events SES*
  ∧ *ses-outputs-are-events SES* ∧ *ses-inputs-outputs-disjoint SES* ∧
  correct-transition-relation SES

primrec *path* :: ('s, 'e) *SES-rec* ⇒ 's ⇒ 'e *list* ⇀ 's
where
path-empt: *path SES s1* [] = (*Some s1*) |

path-nonempt: *path SES s1* $(e \# t) =$
$(if\ (\exists s2.\ s1\ e\longrightarrow_{SES}\ s2)$
$then\ (path\ SES\ (the\ (T_{SES}\ s1\ e))\ t)$
$else\ None)$

abbreviation *pathpred* :: $'s \Rightarrow 'e\ list \Rightarrow ('s, 'e)\ SES\text{-}rec \Rightarrow 's \Rightarrow bool$
$(\text{- -}\Longrightarrow_{\text{-}}\text{ - }[100, 100, 100, 100]\ 100)$
where
$s\ t\Longrightarrow_{SES}\ s' \equiv path\ SES\ s\ t = Some\ s'$

definition *reachable* :: $('s, 'e)\ SES\text{-}rec \Rightarrow 's \Rightarrow bool$
where
$reachable\ SES\ s \equiv (\exists t.\ s0_{SES}\ t\Longrightarrow_{SES}\ s)$

definition *enabled* :: $('s, 'e)\ SES\text{-}rec \Rightarrow 's \Rightarrow 'e\ list \Rightarrow bool$
where
$enabled\ SES\ s\ t \equiv (\exists s'.\ s\ t\Longrightarrow_{SES}\ s')$

definition *possible-traces* :: $('s, 'e)\ SES\text{-}rec \Rightarrow ('e\ list)\ set$
where
$possible\text{-}traces\ SES \equiv \{t.\ (enabled\ SES\ s0_{SES}\ t)\}$

definition *induceES* :: $('s, 'e)\ SES\text{-}rec \Rightarrow 'e\ ES\text{-}rec$
where
$induceES\ SES \equiv$
$(\!|$
$E\text{-}ES = E_{SES},$
$I\text{-}ES = I_{SES},$
$O\text{-}ES = O_{SES},$
$Tr\text{-}ES = possible\text{-}traces\ SES$
$|\!)$

lemma *none-remains-none* : $\bigwedge s\ e.\ (path\ SES\ s\ t) = None$
$\Longrightarrow (path\ SES\ s\ (t\ @\ [e])) = None$
⟨*proof*⟩

lemma *path-trans-single-neg*: $\bigwedge s1.\ [\![s1\ t\Longrightarrow_{SES}\ s2;\ \neg\ (s2\ e\longrightarrow_{SES}\ sn)]\!]$
$\Longrightarrow \neg\ (s1\ (t\ @\ [e])\Longrightarrow_{SES}\ sn)$
⟨*proof*⟩

lemma *path-split-single*: $s1\ (t@[e])\Longrightarrow_{SES}\ sn$
$\Longrightarrow \exists s'.\ s1\ t\Longrightarrow_{SES}\ s' \wedge s'\ e\longrightarrow_{SES}\ sn$
⟨*proof*⟩

lemma *path-trans-single*: $\bigwedge s.$ ⟦ $s\ t \Longrightarrow_{SES} s'$; $s'\ e \longrightarrow_{SES} sn$ ⟧
$\Longrightarrow s\ (t\ @\ [e]) \Longrightarrow_{SES} sn$
⟨*proof*⟩

lemma *path-split*: $\bigwedge sn.$ ⟦ $s1\ (t1\ @\ t2) \Longrightarrow_{SES} sn$ ⟧
$\Longrightarrow (\exists s2.\ (s1\ t1 \Longrightarrow_{SES} s2 \wedge s2\ t2 \Longrightarrow_{SES} sn))$
⟨*proof*⟩

lemma *path-trans*:
$\bigwedge sn.$ ⟦ $s1\ l1 \Longrightarrow_{SES} s2$; $s2\ l2 \Longrightarrow_{SES} sn$ ⟧ $\Longrightarrow s1\ (l1\ @\ l2) \Longrightarrow_{SES} sn$
⟨*proof*⟩

lemma *enabledPrefixSingle* : ⟦ *enabled SES s* $(t@[e])$ ⟧ $\Longrightarrow$ *enabled SES s t*
⟨*proof*⟩

lemma *enabledPrefix* : ⟦ *enabled SES s* $(t1\ @\ t2)$ ⟧ $\Longrightarrow$ *enabled SES s t1*
⟨*proof*⟩

lemma *enabledPrefixSingleFinalStep* : ⟦ *enabled SES s* $(t@[e])$ ⟧ $\Longrightarrow \exists\ t'\ t''.\ t'\ e \longrightarrow_{SES} t''$
⟨*proof*⟩

lemma *induceES-yields-ES*:
SES-valid SES $\Longrightarrow$ *ES-valid* (*induceES SES*)
⟨*proof*⟩

end

4 Security Specification

4.1 Views & Flow Policies

We define views, flow policies and how views can be derived from a given flow policy.

theory *Views*
imports *Main*
begin

record *'e V-rec* =

V :: $'e$ *set*
N :: $'e$ *set*
C :: $'e$ *set*

abbreviation *VrecV* :: $'e$ *V-rec* $\Rightarrow$ $'e$ *set*
$(V_{-}\ [100]\ 1000)$
where
$V_v \equiv (V\ v)$

abbreviation *VrecN* :: $'e$ *V-rec* $\Rightarrow$ $'e$ *set*
$(N_{-}\ [100]\ 1000)$
where
$N_v \equiv (N\ v)$

abbreviation *VrecC* :: $'e$ *V-rec* $\Rightarrow$ $'e$ *set*
$(C_{-}\ [100]\ 1000)$
where
$C_v \equiv (C\ v)$

definition *VN-disjoint* :: $'e$ *V-rec* $\Rightarrow$ *bool*
where
VN-disjoint $v \equiv V_v \cap N_v = \{\}$

definition *VC-disjoint* :: $'e$ *V-rec* $\Rightarrow$ *bool*
where
VC-disjoint $v \equiv V_v \cap C_v = \{\}$

definition *NC-disjoint* :: $'e$ *V-rec* $\Rightarrow$ *bool*
where
NC-disjoint $v \equiv N_v \cap C_v = \{\}$

definition *V-valid* :: $'e$ *V-rec* $\Rightarrow$ *bool*
where
V-valid $v \equiv$ *VN-disjoint* $v \wedge$ *VC-disjoint* $v \wedge$ *NC-disjoint* v

definition *isViewOn* :: $'e$ *V-rec* $\Rightarrow$ $'e$ *set* $\Rightarrow$ *bool*
where
isViewOn $\mathcal{V}$ $E \equiv$ *V-valid* $\mathcal{V} \wedge V_{\mathcal{V}} \cup N_{\mathcal{V}} \cup C_{\mathcal{V}} = E$

end
theory *FlowPolicies*
imports *Views*
begin

record $'domain$ *FlowPolicy-rec* =
D :: $'domain$ *set*
v-rel :: ($'domain \times 'domain$) *set*

n-rel :: (*'domain* × *'domain*) *set*
c-rel :: (*'domain* × *'domain*) *set*

definition *FlowPolicy* :: *'domain FlowPolicy-rec* ⇒ *bool*
where
FlowPolicy fp ≡
((*v-rel fp*) ∪ (*n-rel fp*) ∪ (*c-rel fp*) = ((*D fp*) × (*D fp*)))
∧ (*v-rel fp*) ∩ (*n-rel fp*) = {}
∧ (*v-rel fp*) ∩ (*c-rel fp*) = {}
∧ (*n-rel fp*) ∩ (*c-rel fp*) = {}
∧ (∀ *d* ∈ (*D fp*). (*d*, *d*) ∈ (*v-rel fp*))

type-synonym (*'e*, *'domain*) *dom-type* = *'e* ⇀ *'domain*

definition *dom* :: (*'e*, *'domain*) *dom-type* ⇒ *'domain set* ⇒ *'e set* ⇒ *bool*
where
dom domas dset es ≡
(∀ *e*. ∀ *d*. ((*domas e* = *Some d*) ⟶ (*e* ∈ *es* ∧ *d* ∈ *dset*)))

definition *view-dom* :: *'domain FlowPolicy-rec* ⇒ *'domain* ⇒ (*'e*, *'domain*) *dom-type* ⇒ *'e V-rec*
where
view-dom fp d domas ≡
⦇ *V* = {*e*. ∃ *d'*. (*domas e* = *Some d'* ∧ (*d'*, *d*) ∈ (*v-rel fp*))},
N = {*e*. ∃ *d'*. (*domas e* = *Some d'* ∧ (*d'*, *d*) ∈ (*n-rel fp*))},
C = {*e*. ∃ *d'*. (*domas e* = *Some d'* ∧ (*d'*, *d*) ∈ (*c-rel fp*))} ⦈

end

4.2 Basic Security Predicates

We define all 14 basic security predicates provided as part of MAKS in [3].

theory *BasicSecurityPredicates*
imports *Views ../Basics/Projection*
begin

definition *areTracesOver* :: (*'e list*) *set* ⇒ *'e set* ⇒ *bool*
where
areTracesOver Tr E ≡
∀ τ ∈ *Tr*. (*set* τ) ⊆ *E*

type-synonym *'e BSP* = *'e V-rec* ⇒ ((*'e list*) *set*) ⇒ *bool*

definition *BSP-valid* :: *'e BSP* ⇒ *bool*

where
BSP-valid bsp $\equiv$
  $\forall \mathcal{V}\ Tr\ E.\ (\ isViewOn\ \mathcal{V}\ E \wedge areTracesOver\ Tr\ E\ )$
  $\longrightarrow (\exists\ Tr'.\ Tr' \supseteq Tr\ \wedge bsp\ \mathcal{V}\ Tr')$

definition R :: $'e$ *BSP*
where
$R\ \mathcal{V}\ Tr \equiv$
  $\forall \tau{\in}Tr.\ \exists \tau'{\in}Tr.\ \tau' \upharpoonright C_{\mathcal{V}} = [] \wedge \tau' \upharpoonright V_{\mathcal{V}} = \tau \upharpoonright V_{\mathcal{V}}$

lemma *BSP-valid-R*: *BSP-valid R*
⟨*proof*⟩

definition D :: $'e$ *BSP*
where
$D\ \mathcal{V}\ Tr \equiv$
  $\forall \alpha\ \beta.\ \forall c{\in}C_{\mathcal{V}}.\ ((\beta\ @\ [c]\ @\ \alpha) \in Tr \wedge \alpha \upharpoonright C_{\mathcal{V}} = [])$
  $\longrightarrow (\exists \alpha'\ \beta'.\ ((\beta'\ @\ \alpha') \in Tr \wedge \alpha' \upharpoonright V_{\mathcal{V}} = \alpha \upharpoonright V_{\mathcal{V}} \wedge \alpha' \upharpoonright C_{\mathcal{V}} = []$
  $\wedge \beta' \upharpoonright (V_{\mathcal{V}} \cup C_{\mathcal{V}}) = \beta \upharpoonright (V_{\mathcal{V}} \cup C_{\mathcal{V}})))$

lemma *BSP-valid-D*: *BSP-valid D*
⟨*proof*⟩

definition I :: $'e$ *BSP*
where
$I\ \mathcal{V}\ Tr \equiv$
  $\forall \alpha\ \beta.\ \forall c{\in}C_{\mathcal{V}}.\ ((\beta\ @\ \alpha) \in Tr \wedge \alpha \upharpoonright C_{\mathcal{V}} = [])$
  $\longrightarrow (\exists \alpha'\ \beta'.\ ((\beta'\ @\ [c]\ @\ \alpha') \in Tr \wedge \alpha' \upharpoonright V_{\mathcal{V}} = \alpha \upharpoonright V_{\mathcal{V}} \wedge \alpha' \upharpoonright C_{\mathcal{V}} = []$
  $\wedge \beta' \upharpoonright (V_{\mathcal{V}} \cup C_{\mathcal{V}}) = \beta \upharpoonright (V_{\mathcal{V}} \cup C_{\mathcal{V}})))$

lemma *BSP-valid-I*: *BSP-valid I*
⟨*proof*⟩

type-synonym $'e$ *Rho* = $'e$ *V-rec* $\Rightarrow$ $'e$ *set*

definition
Adm :: $'e$ *V-rec* $\Rightarrow$ $'e$ *Rho* $\Rightarrow$ ($'e$ *list*) *set* $\Rightarrow$ $'e$ *list* $\Rightarrow$ $'e$ $\Rightarrow$ *bool*
where
$Adm\ \mathcal{V}\ \varrho\ Tr\ \beta\ e \equiv$
  $\exists \gamma.\ ((\gamma\ @\ [e]) \in Tr \wedge \gamma \upharpoonright (\varrho\ \mathcal{V}) = \beta \upharpoonright (\varrho\ \mathcal{V}))$

definition IA :: $'e$ *Rho* $\Rightarrow$ $'e$ *BSP*
where
$IA\ \varrho\ \mathcal{V}\ Tr \equiv$
  $\forall \alpha\ \beta.\ \forall c{\in}C_{\mathcal{V}}.\ ((\beta\ @\ \alpha) \in Tr \wedge \alpha \upharpoonright C_{\mathcal{V}} = [] \wedge (Adm\ \mathcal{V}\ \varrho\ Tr\ \beta\ c))$
  $\longrightarrow (\exists\ \alpha'\ \beta'.\ ((\beta'\ @\ [c]\ @\ \alpha') \in Tr) \wedge \alpha' \upharpoonright V_{\mathcal{V}} = \alpha \upharpoonright V_{\mathcal{V}}$

$\wedge\ \alpha'\restriction C_{\mathcal{V}} = [] \wedge \beta'\restriction(V_{\mathcal{V}} \cup C_{\mathcal{V}}) = \beta\restriction(V_{\mathcal{V}} \cup C_{\mathcal{V}}))$

lemma *BSP-valid-IA*: *BSP-valid* (*IA* ϱ)
⟨*proof*⟩

definition *BSD* :: *'e BSP*
where
BSD $\mathcal{V}$ *Tr* ≡
$\forall \alpha\ \beta.\ \forall c{\in}C_{\mathcal{V}}.\ ((\beta\ @\ [c]\ @\ \alpha) \in Tr \wedge \alpha\restriction C_{\mathcal{V}} = [])$
$\longrightarrow (\exists \alpha'.\ ((\beta\ @\ \alpha') \in Tr \wedge \alpha'\restriction V_{\mathcal{V}} = \alpha\restriction V_{\mathcal{V}} \wedge \alpha'\restriction C_{\mathcal{V}} = []))$

lemma *BSP-valid-BSD*: *BSP-valid BSD*
⟨*proof*⟩

definition *BSI* :: *'e BSP*
where
BSI $\mathcal{V}$ *Tr* ≡
$\forall \alpha\ \beta.\ \forall c{\in}C_{\mathcal{V}}.\ ((\beta\ @\ \alpha) \in Tr \wedge \alpha\restriction C_{\mathcal{V}} = [])$
$\longrightarrow (\exists \alpha'.\ ((\beta\ @\ [c]\ @\ \alpha') \in Tr \wedge \alpha'\restriction V_{\mathcal{V}} = \alpha\restriction V_{\mathcal{V}} \wedge \alpha'\restriction C_{\mathcal{V}} = []))$

lemma *BSP-valid-BSI*: *BSP-valid BSI*
⟨*proof*⟩

definition *BSIA* :: *'e Rho* ⇒ *'e BSP*
where
BSIA ϱ $\mathcal{V}$ *Tr* ≡
$\forall \alpha\ \beta.\ \forall c{\in}C_{\mathcal{V}}.\ ((\beta\ @\ \alpha) \in Tr \wedge \alpha\restriction C_{\mathcal{V}} = [] \wedge (Adm\ \mathcal{V}\ \varrho\ Tr\ \beta\ c))$
$\longrightarrow (\exists \alpha'.\ ((\beta\ @\ [c]\ @\ \alpha') \in Tr \wedge \alpha'\restriction V_{\mathcal{V}} = \alpha\restriction V_{\mathcal{V}} \wedge \alpha'\restriction C_{\mathcal{V}} = []))$

lemma *BSP-valid-BSIA*: *BSP-valid* (*BSIA* ϱ)
⟨*proof*⟩

record *'e Gamma* =
Nabla :: *'e set*
Delta :: *'e set*
Upsilon :: *'e set*

abbreviation *GammaNabla* :: *'e Gamma* ⇒ *'e set*
(∇- [*100*] *1000*)
where
$\nabla_{\Gamma} \equiv (Nabla\ \Gamma)$

abbreviation *GammaDelta* :: *'e Gamma* ⇒ *'e set*
(Δ- [*100*] *1000*)
where
$\Delta_{\Gamma} \equiv (Delta\ \Gamma)$

abbreviation *GammaUpsilon* :: $'e$ *Gamma* $\Rightarrow$ $'e$ *set*
($\Upsilon_{\text{-}}$ [*100*] *1000*)
where
$\Upsilon_\Gamma \equiv (\mathit{Upsilon}\ \Gamma)$

definition *FCD* :: $'e$ *Gamma* $\Rightarrow$ $'e$ *BSP*
where
$\mathit{FCD}\ \Gamma\ \mathcal{V}\ \mathit{Tr} \equiv$
$\quad \forall \alpha\ \beta.\ \forall c{\in}(C_\mathcal{V} \cap \Upsilon_\Gamma).\ \forall v{\in}(V_\mathcal{V} \cap \nabla_\Gamma).$
$\quad\quad ((\beta\ @\ [c,v]\ @\ \alpha) \in \mathit{Tr} \wedge \alpha \upharpoonright C_\mathcal{V} = [])$
$\quad\quad\quad \longrightarrow (\exists \alpha'.\ \exists \delta'.\ (\mathit{set}\ \delta') \subseteq (N_\mathcal{V} \cap \Delta_\Gamma)$
$\quad\quad\quad\quad \wedge ((\beta\ @\ \delta'\ @\ [v]\ @\ \alpha') \in \mathit{Tr}$
$\quad\quad\quad\quad \wedge \alpha' \upharpoonright V_\mathcal{V} = \alpha \upharpoonright V_\mathcal{V} \wedge \alpha' \upharpoonright C_\mathcal{V} = []))$

lemma *BSP-valid-FCD*: *BSP-valid* (*FCD* Γ)
⟨*proof*⟩

definition *FCI* :: $'e$ *Gamma* $\Rightarrow$ $'e$ *BSP*
where
$\mathit{FCI}\ \Gamma\ \mathcal{V}\ \mathit{Tr} \equiv$
$\quad \forall \alpha\ \beta.\ \forall c{\in}(C_\mathcal{V} \cap \Upsilon_\Gamma).\ \forall v{\in}(V_\mathcal{V} \cap \nabla_\Gamma).$
$\quad\quad ((\beta\ @\ [v]\ @\ \alpha) \in \mathit{Tr} \wedge \alpha \upharpoonright C_\mathcal{V} = [])$
$\quad\quad\quad \longrightarrow (\exists \alpha'.\ \exists \delta'.\ (\mathit{set}\ \delta') \subseteq (N_\mathcal{V} \cap \Delta_\Gamma)$
$\quad\quad\quad\quad \wedge ((\beta\ @\ [c]\ @\ \delta'\ @\ [v]\ @\ \alpha') \in \mathit{Tr}$
$\quad\quad\quad\quad \wedge \alpha' \upharpoonright V_\mathcal{V} = \alpha \upharpoonright V_\mathcal{V} \wedge \alpha' \upharpoonright C_\mathcal{V} = []))$

lemma *BSP-valid-FCI*: *BSP-valid* (*FCI* Γ)
⟨*proof*⟩

definition *FCIA* :: $'e$ *Rho* $\Rightarrow$ $'e$ *Gamma* $\Rightarrow$ $'e$ *BSP*
where
$\mathit{FCIA}\ \varrho\ \Gamma\ \mathcal{V}\ \mathit{Tr} \equiv$
$\quad \forall \alpha\ \beta.\ \forall c{\in}(C_\mathcal{V} \cap \Upsilon_\Gamma).\ \forall v{\in}(V_\mathcal{V} \cap \nabla_\Gamma).$
$\quad\quad ((\beta\ @\ [v]\ @\ \alpha) \in \mathit{Tr} \wedge \alpha \upharpoonright C_\mathcal{V} = [] \wedge (\mathit{Adm}\ \mathcal{V}\ \varrho\ \mathit{Tr}\ \beta\ c))$
$\quad\quad\quad \longrightarrow (\exists \alpha'.\ \exists \delta'.\ (\mathit{set}\ \delta') \subseteq (N_\mathcal{V} \cap \Delta_\Gamma)$
$\quad\quad\quad\quad \wedge ((\beta\ @\ [c]\ @\ \delta'\ @\ [v]\ @\ \alpha') \in \mathit{Tr}$
$\quad\quad\quad\quad \wedge \alpha' \upharpoonright V_\mathcal{V} = \alpha \upharpoonright V_\mathcal{V} \wedge \alpha' \upharpoonright C_\mathcal{V} = []))$

lemma *BSP-valid-FCIA*: *BSP-valid* (*FCIA* ϱ Γ)
⟨*proof*⟩

definition *SR* :: $'e$ *BSP*
where
$\mathit{SR}\ \mathcal{V}\ \mathit{Tr} \equiv \forall \tau{\in}\mathit{Tr}.\ \tau \upharpoonright (V_\mathcal{V} \cup N_\mathcal{V}) \in \mathit{Tr}$

lemma *BSP-valid SR*

$\langle proof \rangle$

definition *SD* :: *'e BSP*
where
$SD\ \mathcal{V}\ Tr \equiv$
$\forall \alpha\ \beta.\ \forall c{\in}C_{\mathcal{V}}.\ ((\beta\ @\ [c]\ @\ \alpha) \in Tr \land \alpha{\upharpoonright}C_{\mathcal{V}} = []) \longrightarrow \beta\ @\ \alpha \in Tr$

lemma *BSP-valid SD*
$\langle proof \rangle$

definition *SI* :: *'e BSP*
where
$SI\ \mathcal{V}\ Tr \equiv$
$\forall \alpha\ \beta.\ \forall c{\in}C_{\mathcal{V}}.\ ((\beta\ @\ \alpha) \in Tr \land \alpha \upharpoonright C_{\mathcal{V}} = []) \longrightarrow \beta\ @\ [c]\ @\ \alpha \in Tr$

lemma *BSP-valid SI*
$\langle proof \rangle$

definition *SIA* :: *'e Rho* ⇒ *'e BSP*
where
$SIA\ \varrho\ \mathcal{V}\ Tr \equiv$
$\forall \alpha\ \beta.\ \forall c{\in}C_{\mathcal{V}}.\ ((\beta\ @\ \alpha) \in Tr \land \alpha \upharpoonright C_{\mathcal{V}} = [] \land (Adm\ \mathcal{V}\ \varrho\ Tr\ \beta\ c))$
$\longrightarrow (\beta\ @\ [c]\ @\ \alpha) \in Tr$

lemma *BSP-valid* (*SIA* ϱ)
$\langle proof \rangle$

end

4.3 Information-Flow Properties

We define the notion of information-flow properties from [3].

theory *InformationFlowProperties*
imports *BasicSecurityPredicates*
begin

type-synonym *'e SP* = (*'e BSP*) *set*

type-synonym *'e IFP-type* = (*'e V-rec set*) × *'e SP*

definition *IFP-valid* :: *'e set* ⇒ *'e IFP-type* ⇒ *bool*
where
IFP-valid E ifp ≡
$\forall \mathcal{V} \in (fst\ ifp).\ isViewOn\ \mathcal{V}\ E$
$\land (\forall BSP \in (snd\ ifp).\ BSP\text{-}valid\ BSP)$

definition *IFPIsSatisfied* :: *'e IFP-type* ⇒ (*'e list*) *set* ⇒ *bool*
where
IFPIsSatisfied ifp Tr ≡
  ∀ 𝒱∈(*fst ifp*). ∀ *BSP*∈(*snd ifp*). *BSP* 𝒱 *Tr*

end

4.4 Property Library

We define the representations of several possibilistic information-flow properties from the literature that are provided as part of MAKS in [3].

theory *PropertyLibrary*
imports *InformationFlowProperties ../SystemSpecification/EventSystems ../Verification/Basics/BSPTaxonomy*
begin

definition
HighInputsConfidential :: *'e set* ⇒ *'e set* ⇒ *'e set* ⇒ *'e V-rec*
where
HighInputsConfidential L H IE ≡ (| *V*=*L*, *N*=*H*−*IE*, *C*=*H* ∩ *IE* |)

definition *HighConfidential* :: *'e set* ⇒ *'e set* ⇒ *'e V-rec*
where
HighConfidential L H ≡ (| *V*=*L*, *N*={}, *C*=*H* |)

fun *interleaving* :: *'e list* ⇒ *'e list* ⇒ (*'e list*) *set*
where
interleaving t1 [] = {*t1*} |
interleaving [] *t2* = {*t2*} |
interleaving (*e1* # *t1*) (*e2* # *t2*) =
  {*t*. (∃ *t'*. *t*=(*e1* # *t'*) ∧ *t'* ∈ *interleaving t1* (*e2* #*t2*))}
  ∪ {*t*. (∃ *t'*. *t*=(*e2* # *t'*) ∧ *t'* ∈ *interleaving* (*e1* # *t1*) *t2*)}

definition *GNI* :: *'e set* ⇒ *'e set* ⇒ *'e set* ⇒ *'e IFP-type*
where
GNI L H IE ≡ ({*HighInputsConfidential L H IE*}, {*BSD*, *BSI*})

lemma *GNI-valid*: *L* ∩ *H* = {} ⟹ *IFP-valid* (*L* ∪ *H*) (*GNI L H IE*)
  ⟨*proof*⟩

definition *litGNI* :: *'e set* ⇒ *'e set* ⇒ *'e set* ⇒ (*'e list*) *set* ⇒ *bool*
where
litGNI L H IE Tr ≡
  ∀ *t1 t2 t3*.

$t1 \ @ \ t2 \in Tr \land t3 \upharpoonright (L \cup (H - IE)) = t2 \upharpoonright (L \cup (H - IE))$
$\longrightarrow (\exists \ t4. \ t1 \ @ \ t4 \in Tr \land t4 \upharpoonright (L \cup (H \cap IE)) = t3 \upharpoonright (L \cup (H \cap IE)))$

definition *IBGNI* :: $'e$ *set* $\Rightarrow$ $'e$ *set* $\Rightarrow$ $'e$ *set* $\Rightarrow$ $'e$ *IFP-type*
where *IBGNI L H IE* $\equiv$ ({*HighInputsConfidential L H IE*}, {*D*, *I*})

lemma *IBGNI-valid*: $L \cap H = \{\} \Longrightarrow$ *IFP-valid* $(L \cup H)$ (*IBGNI L H IE*)
⟨*proof*⟩

definition
litIBGNI :: $'e$ *set* $\Rightarrow$ $'e$ *set* $\Rightarrow$ $'e$ *set* $\Rightarrow$ ($'e$ *list*) *set* $\Rightarrow$ *bool*
where
litIBGNI L H IE Tr $\equiv$
$\forall \ \tau\text{-}l \in Tr. \ \forall \ t\text{-}hi \ t.$
$(set \ t\text{-}hi) \subseteq (H \cap IE) \ \land t \in interleaving \ t\text{-}hi \ (\tau\text{-}l \upharpoonright L)$
$\longrightarrow (\exists \ \tau' \in Tr. \ \tau' \upharpoonright (L \cup (H \cap IE)) = t)$

definition *FC* :: $'e$ *set* $\Rightarrow$ $'e$ *set* $\Rightarrow$ $'e$ *set* $\Rightarrow$ $'e$ *IFP-type*
where
FC L H IE $\equiv$
({*HighInputsConfidential L H IE*},
{*BSD*, *BSI*, (*FCD* ⦇ *Nabla*=*IE*, *Delta*={}, *Upsilon*=*IE* ⦈),
(*FCI* ⦇ *Nabla*=*IE*, *Delta*={}, *Upsilon*=*IE* ⦈)})

lemma *FC-valid*: $L \cap H = \{\} \Longrightarrow$ *IFP-valid* $(L \cup H)$ (*FC L H IE*)
⟨*proof*⟩

definition *litFC* :: $'e$ *set* $\Rightarrow$ $'e$ *set* $\Rightarrow$ $'e$ *set* $\Rightarrow$ ($'e$ *list*) *set* $\Rightarrow$ *bool*
where
litFC L H IE Tr $\equiv$
$\forall t\text{-}1 \ t\text{-}2. \ \forall \ hi \in (H \cap IE).$
(
$(\forall \ li \in (L \cap IE).$
$t\text{-}1 \ @ \ [li] \ @ \ t\text{-}2 \in Tr \land t\text{-}2 \upharpoonright (H \cap IE) = []$
$\longrightarrow (\exists \ t\text{-}3. \ t\text{-}1 \ @ \ [hi] \ @ \ [li] \ @ \ t\text{-}3 \in Tr$
$\land t\text{-}3 \upharpoonright L = t\text{-}2 \upharpoonright L \land t\text{-}3 \upharpoonright (H \cap IE) = [] \ ))$
$\land (t\text{-}1 \ @ \ t\text{-}2 \in Tr \land t\text{-}2 \upharpoonright (H \cap IE) = []$
$\longrightarrow (\exists \ t\text{-}3. \ t\text{-}1 \ @ \ [hi] \ \ @ \ t\text{-}3 \in Tr$
$\land t\text{-}3 \upharpoonright L = t\text{-}2 \upharpoonright L \land t\text{-}3 \upharpoonright (H \cap IE) = [] \ ))$
$\land (\forall \ li \in (L \cap IE).$
$t\text{-}1 \ @ \ [hi] \ @ \ [li] \ @ \ t\text{-}2 \in Tr \land t\text{-}2 \upharpoonright (H \cap IE) = []$
$\longrightarrow (\exists \ t\text{-}3. \ t\text{-}1 \ @ \ [li] \ @ \ t\text{-}3 \in Tr$
$\land t\text{-}3 \upharpoonright L = t\text{-}2 \upharpoonright L \land t\text{-}3 \upharpoonright (H \cap IE) = [] \ ))$
$\land (t\text{-}1 \ @ \ [hi] \ \ @ \ t\text{-}2 \in Tr \land t\text{-}2 \upharpoonright (H \cap IE) = []$
$\longrightarrow (\exists \ t\text{-}3. \ t\text{-}1 \ \ @ \ t\text{-}3 \in Tr$
$\land t\text{-}3 \upharpoonright L = t\text{-}2 \upharpoonright L \land t\text{-}3 \upharpoonright (H \cap IE) = [] \ ))$

)

definition *NDO* :: $'e\ set \Rightarrow 'e\ set \Rightarrow 'e\ set \Rightarrow 'e\ IFP\text{-}type$
where
$NDO\ UI\ L\ H \equiv$
$(\ \{HighConfidential\ L\ H\},\ \{BSD,\ (BSIA\ (\lambda\ \mathcal{V}.\ C_{\mathcal{V}} \cup (V_{\mathcal{V}} \cap UI)))\})$

lemma *NDO-valid*: $L \cap H = \{\} \Longrightarrow IFP\text{-}valid\ (L \cup H)\ (NDO\ UI\ L\ H)$
⟨*proof*⟩

definition *litNDO* :: $'e\ set \Rightarrow 'e\ set \Rightarrow 'e\ set \Rightarrow ('e\ list)\ set \Rightarrow bool$
where
$litNDO\ UI\ L\ H\ Tr \equiv$
$\forall \tau\text{-}l \in Tr.\ \forall\ \tau\text{-}hlui \in Tr.\ \forall\ t.$
$t{\upharpoonright}L = \tau\text{-}l{\upharpoonright}L \wedge t{\upharpoonright}(H \cup (L \cap UI)) = \tau\text{-}hlui{\upharpoonright}(H \cup (L \cap UI)) \longrightarrow t \in Tr$

definition *NF* :: $'e\ set \Rightarrow 'e\ set \Rightarrow 'e\ IFP\text{-}type$
where
$NF\ L\ H \equiv (\ \{HighConfidential\ L\ H\},\ \{R\})$

lemma *NF-valid*: $L \cap H = \{\} \Longrightarrow IFP\text{-}valid\ (L \cup H)\ (NF\ L\ H)$
⟨*proof*⟩

definition *litNF* :: $'e\ set \Rightarrow 'e\ set \Rightarrow ('e\ list)\ set \Rightarrow bool$
where
$litNF\ L\ H\ Tr \equiv \forall \tau \in Tr.\ \tau \upharpoonright L \in Tr$

definition *GNF* :: $'e\ set \Rightarrow 'e\ set \Rightarrow 'e\ set \Rightarrow 'e\ IFP\text{-}type$
where
$GNF\ L\ H\ IE \equiv (\ \{HighInputsConfidential\ L\ H\ IE\},\ \{R\})$

lemma *GNF-valid*: $L \cap H = \{\} \Longrightarrow IFP\text{-}valid\ (L \cup H)\ (GNF\ L\ H\ IE)$
⟨*proof*⟩

definition *litGNF* :: $'e\ set \Rightarrow 'e\ set \Rightarrow 'e\ set \Rightarrow ('e\ list)\ set \Rightarrow bool$
where
$litGNF\ L\ H\ IE\ Tr \equiv$
$\forall \tau \in Tr.\ \exists \tau' \in Tr.\ \tau'{\upharpoonright}\ (H \cap IE) = [] \wedge \tau'{\upharpoonright}\ L = \tau \upharpoonright L$

definition *SEP* :: $'e\ set \Rightarrow 'e\ set \Rightarrow 'e\ IFP\text{-}type$

where
$SEP\ L\ H \equiv (\ \{HighConfidential\ L\ H\},\ \{BSD,\ (BSIA\ (\lambda\ \mathcal{V}.\ C_{\mathcal{V}}))\})$

lemma *SEP-valid*: $L \cap H = \{\} \Longrightarrow IFP\text{-}valid\ (L \cup H)\ (SEP\ L\ H)$
⟨*proof*⟩

definition *litSEP* :: $'e\ set \Rightarrow\ 'e\ set \Rightarrow (\ 'e\ list)\ set \Rightarrow bool$
where
$litSEP\ L\ H\ Tr \equiv$
$\forall \tau\text{-}l \in Tr.\ \forall\ \tau\text{-}h \in Tr.$
$interleaving\ (\tau\text{-}l \upharpoonright L)\ (\tau\text{-}h \upharpoonright H) \subseteq \{\tau \in Tr\ .\ \tau \upharpoonright L = \tau\text{-}l \upharpoonright L\}$

definition *PSP* :: $'e\ set \Rightarrow\ 'e\ set \Rightarrow\ 'e\ IFP\text{-}type$
where
$PSP\ L\ H \equiv$
$(\ \{HighConfidential\ L\ H\},\ \{BSD,\ (BSIA\ (\lambda\ \mathcal{V}.\ C_{\mathcal{V}} \cup N_{\mathcal{V}} \cup V_{\mathcal{V}}))\})$

lemma *PSP-valid*: $L \cap H = \{\} \Longrightarrow IFP\text{-}valid\ (L \cup H)\ (PSP\ L\ H)$
⟨*proof*⟩

definition *litPSP* :: $'e\ set \Rightarrow\ 'e\ set \Rightarrow (\ 'e\ list)\ set \Rightarrow bool$
where
$litPSP\ L\ H\ Tr \equiv$
$(\forall \tau \in Tr.\ \tau \upharpoonright L \in Tr)$
$\wedge\ (\forall\ \alpha\ \beta.\ (\beta\ @\ \alpha) \in Tr \wedge (\alpha \upharpoonright H) = []$
$\longrightarrow (\forall\ h \in H.\ \beta\ @\ [h] \in Tr \longrightarrow \beta\ @\ [h]\ @\ \alpha \in Tr))$

end

5 Verification

5.1 Basic Definitions

We define when an event system and a state-event system are secure given an information-flow property.

theory *SecureSystems*
imports *../../SystemSpecification/StateEventSystems*
../../SecuritySpecification/InformationFlowProperties
begin

locale *SecureESIFP* =
fixes *ES* :: $'e\ ES\text{-}rec$
and *IFP* :: $'e\ IFP\text{-}type$

assumes *validES*: *ES-valid ES*
and *validIFPES*: $IFP\text{-}valid\ E_{ES}\ IFP$

context *SecureESIFP*
begin

definition *ES-sat-IFP* :: *bool*
where
ES-sat-IFP ≡ *IFPIsSatisfied IFP* Tr_{ES}

end

locale *SecureSESIFP* =
fixes *SES* :: ('*s*, '*e*) *SES-rec*
and *IFP* :: '*e IFP-type*

assumes *validSES*: *SES-valid SES*
and *validIFPSES*: *IFP-valid* E_{SES} *IFP*

sublocale *SecureSESIFP* ⊆ *SecureESIFP induceES SES IFP*
⟨*proof*⟩

context *SecureSESIFP*
begin

abbreviation *SES-sat-IFP*
where
SES-sat-IFP ≡ *ES-sat-IFP*

end

end

5.2 Taxonomy Results

We prove the taxonomy results from [3].

theory *BSPTaxonomy*
imports ../../*SystemSpecification*/*EventSystems*
../../*SecuritySpecification*/*BasicSecurityPredicates*
begin

locale *BSPTaxonomyDifferentCorrections* =
fixes *ES* :: '*e ES-rec*

and $\mathcal{V}$ :: $'e$ *V-rec*

assumes *validES*: *ES-valid ES*
and *VIsViewOnE*: *isViewOn* $\mathcal{V}$ E_{ES}

locale *BSPTaxonomyDifferentViews* =
fixes *ES* :: $'e$ *ES-rec*
and $\mathcal{V}_1$ :: $'e$ *V-rec*
and $\mathcal{V}_2$ :: $'e$ *V-rec*

assumes *validES*: *ES-valid ES*
and $\mathcal{V}_1$*IsViewOnE*: *isViewOn* $\mathcal{V}_1$ E_{ES}
and $\mathcal{V}_2$*IsViewOnE*: *isViewOn* $\mathcal{V}_2$ E_{ES}

locale *BSPTaxonomyDifferentViewsFirstDim*= *BSPTaxonomyDifferentViews* +
assumes *V2-subset-V1*: $V_{\mathcal{V}_2} \subseteq V_{\mathcal{V}_1}$
and *N2-supset-N1*: $N_{\mathcal{V}_2} \supseteq N_{\mathcal{V}_1}$
and *C2-subset-C1*: $C_{\mathcal{V}_2} \subseteq C_{\mathcal{V}_1}$

sublocale *BSPTaxonomyDifferentViewsFirstDim* ⊆ *BSPTaxonomyDifferentViews*
⟨*proof*⟩

locale *BSPTaxonomyDifferentViewsSecondDim*= *BSPTaxonomyDifferentViews* +
assumes *V2-subset-V1*: $V_{\mathcal{V}_2} \subseteq V_{\mathcal{V}_1}$
and *N2-supset-N1*: $N_{\mathcal{V}_2} \supseteq N_{\mathcal{V}_1}$
and *C2-equals-C1*: $C_{\mathcal{V}_2} = C_{\mathcal{V}_1}$

sublocale *BSPTaxonomyDifferentViewsSecondDim* ⊆ *BSPTaxonomyDifferentViews*
⟨*proof*⟩

context *BSPTaxonomyDifferentCorrections*
begin

lemma *SR-implies-R*:
$SR\ \mathcal{V}\ Tr_{ES} \Longrightarrow R\ \mathcal{V}\ Tr_{ES}$
⟨*proof*⟩

lemma *SD-implies-BSD* :
$(SD\ \mathcal{V}\ Tr_{ES}) \Longrightarrow BSD\ \mathcal{V}\ Tr_{ES}$
⟨*proof*⟩

lemma *BSD-implies-D*:
$BSD\ \mathcal{V}\ Tr_{ES} \Longrightarrow D\ \mathcal{V}\ Tr_{ES}$
⟨*proof*⟩

lemma *SD-implies-SR*:
$SD\ \mathcal{V}\ Tr_{ES} \Longrightarrow SR\ \mathcal{V}\ Tr_{ES}$

⟨*proof*⟩

lemma *D-implies-R*:
$D\ \mathcal{V}\ Tr_{ES} \Longrightarrow R\ \mathcal{V}\ Tr_{ES}$
⟨*proof*⟩

lemma *SR-implies-R-for-modified-view* :
$[\![SR\ \mathcal{V}\ Tr_{ES};\ \mathcal{V}' = (\!|\ V = V_{\mathcal{V}} \cup N_{\mathcal{V}}\ ,\ N = \{\}\ ,\ C = C_{\mathcal{V}}\ |\!)]\!] \Longrightarrow R\ \mathcal{V}'\ Tr_{ES}$
⟨*proof*⟩

lemma *R-implies-SR-for-modified-view* :
$[\![R\ \mathcal{V}'\ Tr_{ES};\ \mathcal{V}' = (\!|\ V = V_{\mathcal{V}} \cup N_{\mathcal{V}}\ ,\ N = \{\}\ ,\ C = C_{\mathcal{V}}\ |\!)]\!] \Longrightarrow SR\ \mathcal{V}\ Tr_{ES}$
⟨*proof*⟩

lemma *SD-implies-BSD-for-modified-view* :
$[\![SD\ \mathcal{V}\ Tr_{ES};\ \mathcal{V}' = (\!|\ V = V_{\mathcal{V}} \cup N_{\mathcal{V}}\ ,\ N = \{\}\ ,\ C = C_{\mathcal{V}}\ |\!)]\!] \Longrightarrow BSD\ \mathcal{V}'\ Tr_{ES}$
⟨*proof*⟩

lemma *BSD-implies-SD-for-modified-view* :
$[\![BSD\ \mathcal{V}'\ Tr_{ES};\ \mathcal{V}' = (\!|\ V = V_{\mathcal{V}} \cup N_{\mathcal{V}}\ ,\ N = \{\}\ ,\ C = C_{\mathcal{V}}\ |\!)]\!] \Longrightarrow SD\ \mathcal{V}\ Tr_{ES}$
⟨*proof*⟩

lemma *SD-implies-FCD*:
$(SD\ \mathcal{V}\ Tr_{ES}) \Longrightarrow FCD\ \Gamma\ \mathcal{V}\ Tr_{ES}$
⟨*proof*⟩

lemma *SI-implies-BSI* :
$(SI\ \mathcal{V}\ Tr_{ES}) \Longrightarrow BSI\ \mathcal{V}\ Tr_{ES}$
⟨*proof*⟩

lemma *BSI-implies-I*:
$(BSI\ \mathcal{V}\ Tr_{ES}) \Longrightarrow (I\ \mathcal{V}\ Tr_{ES})$
⟨*proof*⟩

lemma *SIA-implies-BSIA*:
$(SIA\ \varrho\ \mathcal{V}\ Tr_{ES}) \Longrightarrow (BSIA\ \varrho\ \mathcal{V}\ Tr_{ES})$
⟨*proof*⟩

lemma *BSIA-implies-IA*:
$(BSIA\ \varrho\ \mathcal{V}\ Tr_{ES}) \Longrightarrow (IA\ \varrho\ \mathcal{V}\ Tr_{ES})$
⟨*proof*⟩

lemma *SI-implies-SIA*:
$SI\ \mathcal{V}\ Tr_{ES} \Longrightarrow SIA\ \varrho\ \mathcal{V}\ Tr_{ES}$
⟨*proof*⟩

lemma *BSI-implies-BSIA*:
$BSI\ \mathcal{V}\ Tr_{ES} \Longrightarrow BSIA\ \varrho\ \mathcal{V}\ Tr_{ES}$
⟨*proof*⟩

lemma *I-implies-IA*:
$I\ \mathcal{V}\ Tr_{ES} \Longrightarrow IA\ \varrho\ \mathcal{V}\ Tr_{ES}$
⟨*proof*⟩

lemma *SI-implies-BSI-for-modified-view* :
$[\![SI\ \mathcal{V}\ Tr_{ES};\ \mathcal{V}' = (\!|\ V = V_{\mathcal{V}} \cup N_{\mathcal{V}}\ ,\ N = \{\}\ ,\ C = C_{\mathcal{V}}\ |\!)]\!] \Longrightarrow BSI\ \mathcal{V}'\ Tr_{ES}$
⟨*proof*⟩

lemma *BSI-implies-SI-for-modified-view* :
$[\![BSI\ \mathcal{V}'\ Tr_{ES};\ \mathcal{V}' = (\!|\ V = V_{\mathcal{V}} \cup N_{\mathcal{V}}\ ,\ N = \{\}\ ,\ C = C_{\mathcal{V}}\ |\!)]\!] \Longrightarrow SI\ \mathcal{V}\ Tr_{ES}$
⟨*proof*⟩

lemma *SIA-implies-BSIA-for-modified-view* :
$[\![SIA\ \varrho\ \mathcal{V}\ Tr_{ES};\ \mathcal{V}' = (\!|\ V = V_{\mathcal{V}} \cup N_{\mathcal{V}}\ ,\ N = \{\}\ ,\ C = C_{\mathcal{V}}\ |\!)\ ;\ \varrho\ \mathcal{V} = \varrho'\ \mathcal{V}']\!] \Longrightarrow BSIA\ \varrho'\ \mathcal{V}'\ Tr_{ES}$
⟨*proof*⟩

lemma *BSIA-implies-SIA-for-modified-view* :
$[\![BSIA\ \varrho'\ \mathcal{V}'\ Tr_{ES};\ \mathcal{V}' = (\!|\ V = V_{\mathcal{V}} \cup N_{\mathcal{V}}\ ,\ N = \{\}\ ,\ C = C_{\mathcal{V}}\ |\!);\ \varrho\ \mathcal{V} = \varrho'\ \mathcal{V}']\!] \Longrightarrow SIA\ \varrho\ \mathcal{V}\ Tr_{ES}$
⟨*proof*⟩
end

lemma *Adm-implies-Adm-for-modified-rho*:
$[\![\ Adm\ \mathcal{V}_2\ \varrho_2\ Tr\ \alpha\ e;\varrho_2(\mathcal{V}_2) \supseteq \varrho_1(\mathcal{V}_1)]\!] \Longrightarrow Adm\ \mathcal{V}_1\ \varrho_1\ Tr\ \alpha\ e$
⟨*proof*⟩

context *BSPTaxonomyDifferentCorrections*
begin

lemma *SI-implies-FCI*:
$(SI\ \mathcal{V}\ Tr_{ES}) \Longrightarrow FCI\ \Gamma\ \mathcal{V}\ Tr_{ES}$
⟨*proof*⟩

lemma *SIA-implies-FCIA*:
$(SIA\ \varrho\ \mathcal{V}\ Tr_{ES}) \Longrightarrow FCIA\ \varrho\ \Gamma\ \mathcal{V}\ Tr_{ES}$

⟨*proof*⟩

lemma *FCI-implies-FCIA*:
$(FCI\ \Gamma\ \mathcal{V}\ Tr_{ES}) \Longrightarrow FCIA\ \varrho\ \Gamma\ \mathcal{V}\ Tr_{ES}$
⟨*proof*⟩

lemma *Trivially-fulfilled-SR-C-empty*:
$C_{\mathcal{V}} = \{\} \Longrightarrow SR\ \mathcal{V}\ Tr_{ES}$
⟨*proof*⟩

lemma *Trivially-fulfilled-R-C-empty*:
$C_{\mathcal{V}} = \{\} \Longrightarrow R\ \mathcal{V}\ Tr_{ES}$
⟨*proof*⟩

lemma *Trivially-fulfilled-SD-C-empty*:
$C_{\mathcal{V}} = \{\} \Longrightarrow SD\ \mathcal{V}\ Tr_{ES}$
⟨*proof*⟩

lemma *Trivially-fulfilled-BSD-C-empty*:
$C_{\mathcal{V}} = \{\} \Longrightarrow BSD\ \mathcal{V}\ Tr_{ES}$
⟨*proof*⟩

lemma *Trivially-fulfilled-D-C-empty*:
$C_{\mathcal{V}} = \{\} \Longrightarrow D\ \mathcal{V}\ Tr_{ES}$
⟨*proof*⟩

lemma *Trivially-fulfilled-FCD-C-empty*:
$C_{\mathcal{V}} = \{\} \Longrightarrow FCD\ \Gamma\ \mathcal{V}\ Tr_{ES}$
⟨*proof*⟩

lemma *Trivially-fullfilled-R-V-empty*:
$V_{\mathcal{V}}=\{\} \Longrightarrow R\ \mathcal{V}\ Tr_{ES}$
⟨*proof*⟩

lemma *Trivially-fulfilled-BSD-V-empty*:
$V_{\mathcal{V}} = \{\} \Longrightarrow BSD\ \mathcal{V}\ Tr_{ES}$
⟨*proof*⟩

lemma *Trivially-fulfilled-D-V-empty*:
$V_{\mathcal{V}} = \{\} \Longrightarrow D\ \mathcal{V}\ Tr_{ES}$
⟨*proof*⟩

lemma *Trivially-fulfilled-FCD-V-empty*:
$V_{\mathcal{V}} = \{\} \Longrightarrow FCD\ \Gamma\ \mathcal{V}\ Tr_{ES}$
⟨*proof*⟩

lemma *Trivially-fulfilled-FCD-Nabla-Υ-empty*:
$[\![\nabla_{\Gamma}=\{\} \vee \Upsilon_{\Gamma}=\{\}]\!] \Longrightarrow FCD\ \Gamma\ \mathcal{V}\ Tr_{ES}$

⟨*proof*⟩

lemma *Trivially-fulfilled-FCD-N-subseteq-Δ-and-BSD*:
$[\![N_{\mathcal{V}} \subseteq \Delta_{\Gamma};\ BSD\ \mathcal{V}\ Tr_{ES}]\!] \Longrightarrow FCD\ \Gamma\ \mathcal{V}\ Tr_{ES}$
⟨*proof*⟩

lemma *Trivially-fulfilled-SI-C-empty*:
$C_{\mathcal{V}} = \{\} \Longrightarrow SI\ \mathcal{V}\ Tr_{ES}$
⟨*proof*⟩

lemma *Trivially-fulfilled-BSI-C-empty*:
$C_{\mathcal{V}} = \{\} \Longrightarrow BSI\ \mathcal{V}\ Tr_{ES}$
⟨*proof*⟩

lemma *Trivially-fulfilled-I-C-empty*:
$C_{\mathcal{V}} = \{\} \Longrightarrow I\ \mathcal{V}\ Tr_{ES}$
⟨*proof*⟩

lemma *Trivially-fulfilled-FCI-C-empty*:
$C_{\mathcal{V}} = \{\} \Longrightarrow FCI\ \Gamma\ \mathcal{V}\ Tr_{ES}$
⟨*proof*⟩

lemma *Trivially-fulfilled-SIA-C-empty*:
$C_{\mathcal{V}} = \{\} \Longrightarrow SIA\ \varrho\ \mathcal{V}\ Tr_{ES}$
⟨*proof*⟩

lemma *Trivially-fulfilled-BSIA-C-empty*:
$C_{\mathcal{V}} = \{\} \Longrightarrow BSIA\ \varrho\ \mathcal{V}\ Tr_{ES}$
⟨*proof*⟩

lemma *Trivially-fulfilled-IA-C-empty*:
$C_{\mathcal{V}} = \{\} \Longrightarrow IA\ \varrho\ \mathcal{V}\ Tr_{ES}$
⟨*proof*⟩

lemma *Trivially-fulfilled-FCIA-C-empty*:
$C_{\mathcal{V}} = \{\} \Longrightarrow FCIA\ \Gamma\ \varrho\ \mathcal{V}\ Tr_{ES}$
⟨*proof*⟩

lemma *Trivially-fulfilled-FCI-V-empty*:
$V_{\mathcal{V}} = \{\} \Longrightarrow FCI\ \Gamma\ \mathcal{V}\ Tr_{ES}$
⟨*proof*⟩

lemma *Trivially-fulfilled-FCIA-V-empty*:
$V_{\mathcal{V}} = \{\} \Longrightarrow FCIA\ \varrho\ \Gamma\ \mathcal{V}\ Tr_{ES}$
⟨*proof*⟩

lemma *Trivially-fulfilled-BSIA-V-empty-rho-subseteq-C-N*:
$[\![V_{\mathcal{V}} = \{\};\ \varrho\ \mathcal{V} \supseteq (C_{\mathcal{V}} \cup N_{\mathcal{V}})\ ]\!] \Longrightarrow BSIA\ \varrho\ \ \mathcal{V}\ Tr_{ES}$
⟨*proof*⟩

lemma *Trivially-fulfilled-IA-V-empty-rho-subseteq-C-N*:

$[\![V_{\mathcal{V}} = \{\}; \varrho\ \mathcal{V} \supseteq (C_{\mathcal{V}} \cup N_{\mathcal{V}})\ ]\!] \Longrightarrow IA\ \varrho\ \ \mathcal{V}\ \ Tr_{ES}$
⟨*proof*⟩

lemma *Trivially-fulfilled-BSI-V-empty-total-ES-C*:
$[\![V_{\mathcal{V}} = \{\};\ total\ ES\ C_{\mathcal{V}}\ ]\!] \Longrightarrow BSI\ \mathcal{V}\ \ Tr_{ES}$
⟨*proof*⟩

lemma *Trivially-fulfilled-I-V-empty-total-ES-C*:
$[\![V_{\mathcal{V}} = \{\};\ total\ ES\ C_{\mathcal{V}}\ ]\!] \Longrightarrow I\ \mathcal{V}\ \ Tr_{ES}$
⟨*proof*⟩

lemma *Trivially-fulfilled-FCI-Nabla-Υ-empty*:
$[\![\nabla_{\Gamma}=\{\} \vee \Upsilon_{\Gamma}=\{\}]\!]\Longrightarrow FCI\ \Gamma\ \mathcal{V}\ \ Tr_{ES}$
⟨*proof*⟩

lemma *Trivially-fulfilled-FCIA-Nabla-Υ-empty*:
$[\![\nabla_{\Gamma}=\{\} \vee \Upsilon_{\Gamma}=\{\}]\!]\Longrightarrow FCIA\ \varrho\ \Gamma\ \mathcal{V}\ \ Tr_{ES}$
⟨*proof*⟩

lemma *Trivially-fulfilled-FCI-N-subseteq-Δ-and-BSI*:
$[\![N_{\mathcal{V}} \subseteq \Delta_{\Gamma};\ BSI\ \mathcal{V}\ \ Tr_{ES}]\!] \Longrightarrow FCI\ \Gamma\ \mathcal{V}\ \ Tr_{ES}$
⟨*proof*⟩

lemma *Trivially-fulfilled-FCIA-N-subseteq-Δ-and-BSIA*:
$[\![N_{\mathcal{V}} \subseteq \Delta_{\Gamma};\ BSIA\ \varrho\ \mathcal{V}\ \ Tr_{ES}]\!] \Longrightarrow FCIA\ \varrho\ \Gamma\ \mathcal{V}\ \ Tr_{ES}$
⟨*proof*⟩

end

context *BSPTaxonomyDifferentViewsFirstDim*
begin

lemma *R-implies-R-for-modified-view*:
$R\ \mathcal{V}_1\ \ Tr_{ES} \Longrightarrow R\ \mathcal{V}_2\ \ Tr_{ES}$
⟨*proof*⟩

lemma *BSD-implies-BSD-for-modified-view*:
$BSD\ \mathcal{V}_1\ \ Tr_{ES}\Longrightarrow BSD\ \mathcal{V}_2\ \ Tr_{ES}$
⟨*proof*⟩

lemma *D-implies-D-for-modified-view*:
$D\ \mathcal{V}_1\ \ Tr_{ES} \Longrightarrow D\ \mathcal{V}_2\ \ Tr_{ES}$
⟨*proof*⟩
end

context *BSPTaxonomyDifferentViewsSecondDim*
begin

lemma *FCD-implies-FCD-for-modified-view-gamma*:
$[\![FCD\ \Gamma_1\ \mathcal{V}_1\ \ Tr_{ES};$
$\quad V_{\mathcal{V}_2}\cap\nabla_{\Gamma_2} \subseteq\ \ V_{\mathcal{V}_1}\cap\nabla_{\Gamma_1};\ \ N_{\mathcal{V}_2}\cap\Delta_{\Gamma_2} \supseteq\ \ N_{\mathcal{V}_1}\cap\Delta_{\Gamma_1};\ \ C_{\mathcal{V}_2}\cap\Upsilon_{\Gamma_2} \subseteq\ \ C_{\mathcal{V}_1}\cap\Upsilon_{\Gamma_1}\ ]\!]$

$\Longrightarrow$ *FCD* Γ_2 $\mathcal{V}_2$ Tr_{ES}
⟨*proof*⟩

lemma *SI-implies-SI-for-modified-view* :
SI $\mathcal{V}_1$ $Tr_{ES} \Longrightarrow$ *SI* $\mathcal{V}_2$ Tr_{ES}
⟨*proof*⟩

lemma *BSI-implies-BSI-for-modified-view* :
BSI $\mathcal{V}_1$ $Tr_{ES} \Longrightarrow$ *BSI* $\mathcal{V}_2$ Tr_{ES}
⟨*proof*⟩

lemma *I-implies-I-for-modified-view* :
I $\mathcal{V}_1$ $Tr_{ES} \Longrightarrow$ *I* $\mathcal{V}_2$ Tr_{ES}
⟨*proof*⟩

lemma *SIA-implies-SIA-for-modified-view* :
$[\![$*SIA* ϱ_1 $\mathcal{V}_1$ Tr_{ES}; $\varrho_2(\mathcal{V}_2) \supseteq \varrho_1(\mathcal{V}_1)$ $]\!] \Longrightarrow$ *SIA* ϱ_2 $\mathcal{V}_2$ Tr_{ES}
⟨*proof*⟩

lemma *BSIA-implies-BSIA-for-modified-view* :
$[\![$*BSIA* ϱ_1 $\mathcal{V}_1$ Tr_{ES}; $\varrho_2(\mathcal{V}_2) \supseteq \varrho_1(\mathcal{V}_1)$ $]\!] \Longrightarrow$ *BSIA* ϱ_2 $\mathcal{V}_2$ Tr_{ES}
⟨*proof*⟩

lemma *IA-implies-IA-for-modified-view* :
$[\![$*IA* ϱ_1 $\mathcal{V}_1$ Tr_{ES}; $\varrho_2(\mathcal{V}_2) \supseteq \varrho_1(\mathcal{V}_1)$ $]\!] \Longrightarrow$ *IA* ϱ_2 $\mathcal{V}_2$ Tr_{ES}
⟨*proof*⟩

lemma *FCI-implies-FCI-for-modified-view-gamma*:
$[\![$*FCI* Γ_1 $\mathcal{V}_1$ Tr_{ES};
$V_{\mathcal{V}_2} \cap \nabla_{\Gamma_2} \subseteq V_{\mathcal{V}_1} \cap \nabla_{\Gamma_1}$; $N_{\mathcal{V}_2} \cap \Delta_{\Gamma_2} \supseteq N_{\mathcal{V}_1} \cap \Delta_{\Gamma_1}$; $C_{\mathcal{V}_2} \cap \Upsilon_{\Gamma_2} \subseteq C_{\mathcal{V}_1} \cap \Upsilon_{\Gamma_1}$ $]\!]$
$\Longrightarrow$ *FCI* Γ_2 $\mathcal{V}_2$ Tr_{ES}
⟨*proof*⟩

lemma *FCIA-implies-FCIA-for-modified-view-rho-gamma*:
$[\![$*FCIA* ϱ_1 Γ_1 $\mathcal{V}_1$ Tr_{ES}; $\varrho_2(\mathcal{V}_2) \supseteq \varrho_1(\mathcal{V}_1)$;
$V_{\mathcal{V}_2} \cap \nabla_{\Gamma_2} \subseteq V_{\mathcal{V}_1} \cap \nabla_{\Gamma_1}$; $N_{\mathcal{V}_2} \cap \Delta_{\Gamma_2} \supseteq N_{\mathcal{V}_1} \cap \Delta_{\Gamma_1}$; $C_{\mathcal{V}_2} \cap \Upsilon_{\Gamma_2} \subseteq C_{\mathcal{V}_1} \cap \Upsilon_{\Gamma_1}$ $]\!]$
$\Longrightarrow$ *FCIA* ϱ_2 Γ_2 $\mathcal{V}_2$ Tr_{ES}
⟨*proof*⟩
end

end

5.3 Unwinding

We define the unwinding conditions provided in [3] and prove the unwinding theorems from [3] that use these unwinding conditions.

5.3.1 Unwinding Conditions

theory *UnwindingConditions*
imports *../Basics/BSPTaxonomy*
../../SystemSpecification/StateEventSystems
begin

locale *Unwinding* =
fixes *SES* :: ('s, 'e) *SES-rec*
and $\mathcal{V}$ :: 'e *V-rec*

assumes *validSES*: *SES-valid SES*
and *validVU*: *isViewOn* $\mathcal{V}$ E_{SES}

sublocale *Unwinding* ⊆ *BSPTaxonomyDifferentCorrections induceES SES* $\mathcal{V}$
⟨*proof*⟩

context *Unwinding*
begin

definition *osc* :: 's *rel* ⇒ *bool*
where
$osc\ ur \equiv$
$\forall s1 \in S_{SES}.\ \forall s1' \in S_{SES}.\ \forall s2' \in S_{SES}.\ \forall e \in (E_{SES} - C_{\mathcal{V}}).$
$(reachable\ SES\ s1 \wedge reachable\ SES\ s1'$
$\wedge\ s1'\ e\longrightarrow_{SES}\ s2' \wedge (s1', s1) \in ur)$
$\longrightarrow (\exists s2 \in S_{SES}.\ \exists \delta.\ \delta \upharpoonright C_{\mathcal{V}} = [] \wedge \delta \upharpoonright V_{\mathcal{V}} = [e] \upharpoonright V_{\mathcal{V}}$
$\wedge\ s1\ \delta\Longrightarrow_{SES}\ s2 \wedge (s2', s2) \in ur)$

definition *lrf* :: 's *rel* ⇒ *bool*
where
$lrf\ ur \equiv$
$\forall s \in S_{SES}.\ \forall s' \in S_{SES}.\ \forall c \in C_{\mathcal{V}}.$
$((reachable\ SES\ s \wedge s\ c\longrightarrow_{SES}\ s') \longrightarrow (s', s) \in ur)$

definition *lrb* :: 's *rel* ⇒ *bool*
where
$lrb\ ur \equiv \forall s \in S_{SES}.\ \forall c \in C_{\mathcal{V}}.$
$(reachable\ SES\ s \longrightarrow (\exists s' \in S_{SES}.\ (s\ c\longrightarrow_{SES}\ s' \wedge ((s, s') \in ur))))$

definition *fcrf* :: $'e$ *Gamma* $\Rightarrow$ $'s$ *rel* $\Rightarrow$ *bool*
where
fcrf Γ *ur* $\equiv$
$\forall c \in (C_{\mathcal{V}} \cap \Upsilon_{\Gamma}).\ \forall v \in (V_{\mathcal{V}} \cap \nabla_{\Gamma}).\ \forall s \in S_{SES}.\ \forall s' \in S_{SES}.$
$((reachable\ SES\ s \wedge s\ ([c]\ @\ [v]) \Longrightarrow_{SES} s')$
$\longrightarrow (\exists s'' \in S_{SES}.\ \exists \delta.\ (\forall d \in (set\ \delta).\ d \in (N_{\mathcal{V}} \cap \Delta_{\Gamma})) \wedge$
$s\ (\delta\ @\ [v]) \Longrightarrow_{SES} s'' \wedge (s', s'') \in ur))$

definition *fcrb* :: $'e$ *Gamma* $\Rightarrow$ $'s$ *rel* $\Rightarrow$ *bool*
where
fcrb Γ *ur* $\equiv$
$\forall c \in (C_{\mathcal{V}} \cap \Upsilon_{\Gamma}).\ \forall v \in (V_{\mathcal{V}} \cap \nabla_{\Gamma}).\ \forall s \in S_{SES}.\ \forall s'' \in S_{SES}.$
$((reachable\ SES\ s \wedge s\ v \longrightarrow_{SES} s'')$
$\longrightarrow (\exists s' \in S_{SES}.\ \exists \delta.\ (\forall d \in (set\ \delta).\ d \in (N_{\mathcal{V}} \cap \Delta_{\Gamma})) \wedge$
$s\ ([c]\ @\ \delta\ @\ [v]) \Longrightarrow_{SES} s' \wedge (s'', s') \in ur))$

definition *En* :: $'e$ *Rho* $\Rightarrow$ $'s$ $\Rightarrow$ $'e$ $\Rightarrow$ *bool*
where
En ϱ *s e* $\equiv$
$\exists \beta\ \gamma.\ \exists s' \in S_{SES}.\ \exists s'' \in S_{SES}.$
$s0_{SES}\ \beta \Longrightarrow_{SES} s \wedge (\gamma \upharpoonright (\varrho\ \mathcal{V}) = \beta \upharpoonright (\varrho\ \mathcal{V}))$
$\wedge\ s0_{SES}\ \gamma \Longrightarrow_{SES} s' \wedge s'\ e \longrightarrow_{SES} s''$

definition *lrbe* :: $'e$ *Rho* $\Rightarrow$ $'s$ *rel* $\Rightarrow$ *bool*
where
lrbe ϱ *ur* $\equiv$
$\forall s \in S_{SES}.\ \forall c \in C_{\mathcal{V}}\ .$
$((reachable\ SES\ s \wedge (En\ \varrho\ s\ c))$
$\longrightarrow (\exists s' \in S_{SES}.\ (s\ c \longrightarrow_{SES} s' \wedge (s, s') \in ur)))$

definition *fcrbe* :: $'e$ *Gamma* $\Rightarrow$ $'e$ *Rho* $\Rightarrow$ $'s$ *rel* $\Rightarrow$ *bool*
where
fcrbe Γ ϱ *ur* $\equiv$
$\forall c \in (C_{\mathcal{V}} \cap \Upsilon_{\Gamma}).\ \forall v \in (V_{\mathcal{V}} \cap \nabla_{\Gamma}).\ \forall s \in S_{SES}.\ \forall s'' \in S_{SES}.$
$((reachable\ SES\ s \wedge s\ v \longrightarrow_{SES} s'' \wedge (En\ \varrho\ s\ c))$
$\longrightarrow (\exists s' \in S_{SES}.\ \exists \delta.\ (\forall d \in (set\ \delta).\ d \in (N_{\mathcal{V}} \cap \Delta_{\Gamma})) \wedge$
$s\ ([c]\ @\ \delta\ @\ [v]) \Longrightarrow_{SES} s' \wedge (s'', s') \in ur))$

end

end

5.3.2 Auxiliary Results

theory *AuxiliaryLemmas*
imports *UnwindingConditions*
begin

context *Unwinding*
begin

lemma *osc-property*:
$\bigwedge s1\ s1'.\ [\![\ osc\ ur;\ s1 \in S_{SES};\ s1' \in S_{SES};\ \alpha \upharpoonright C_{\mathcal{V}} = [];$
$reachable\ SES\ s1;\ reachable\ SES\ s1';\ enabled\ SES\ s1'\ \alpha;\ (s1',\ s1) \in ur\ ]\!]$
$\Longrightarrow (\exists\,\alpha'.\ \alpha' \upharpoonright C_{\mathcal{V}} = [] \wedge \alpha' \upharpoonright V_{\mathcal{V}} = \alpha \upharpoonright V_{\mathcal{V}} \wedge enabled\ SES\ s1\ \alpha')$
⟨*proof*⟩

lemma *path-state-closure*: $[\![\ s\ \tau\Longrightarrow_{SES}\ s';\ s \in S_{SES}\ ]\!] \Longrightarrow s' \in S_{SES}$
(**is** $[\![\ ?P\ s\ \tau\ s';\ ?S\ s\ SES\ ]\!] \Longrightarrow\ ?S\ s'\ SES\ $)
⟨*proof*⟩

theorem *En-to-Adm*:
$[\![\ reachable\ SES\ s;\ En\ \varrho\ s\ e]\!]$
$\Longrightarrow \exists\,\beta.\ (\ s0_{SES}\ \beta\Longrightarrow_{SES}\ s \wedge Adm\ \mathcal{V}\ \varrho\ Tr_{(induceES\ SES)}\ \beta\ e\ )$
⟨*proof*⟩

theorem *Adm-to-En*:
$[\![\ \beta \in Tr_{(induceES\ SES)};\ Adm\ \mathcal{V}\ \varrho\ Tr_{(induceES\ SES)}\ \beta\ e\ ]\!]$
$\Longrightarrow \exists\,s \in S_{SES}.\ (s0_{SES}\ \beta\Longrightarrow_{SES}\ s \wedge En\ \varrho\ s\ e)$
⟨*proof*⟩

lemma *state-from-induceES-trace*:
$[\![\ \ (\beta\ @\ \alpha) \in Tr_{(induceES\ SES)}\ ]\!]$
$\Longrightarrow \exists\,s \in S_{SES}.\ s0_{SES}\ \beta\Longrightarrow_{SES}\ s \wedge enabled\ SES\ s\ \alpha \wedge reachable\ SES\ s$
⟨*proof*⟩

lemma *path-split2*:$s0_{SES}\ (\beta\ @\ \alpha)\Longrightarrow_{SES}\ s$
$\Longrightarrow \exists\,s' \in S_{SES}.\ (\ s0_{SES}\ \beta\Longrightarrow_{SES}\ s' \wedge s'\ \alpha\Longrightarrow_{SES}\ s \wedge reachable\ SES\ s'\ )$
⟨*proof*⟩

lemma *path-split-single2*:
$s0_{SES}\ (\beta\ @\ [x])\Longrightarrow_{SES}\ s$
$\Longrightarrow \exists\,s' \in S_{SES}.\ (\ s0_{SES}\ \beta\Longrightarrow_{SES}\ s' \wedge s'\ x\longrightarrow_{SES}\ s \wedge reachable\ SES\ s'\ )$
⟨*proof*⟩

lemma *modified-view-valid*: $isViewOn\ (\!|\ V = (V_{\mathcal{V}} \cup N_{\mathcal{V}}),\ N = \{\},\ C = C_{\mathcal{V}}|\!)\ E_{SES}$

⟨*proof*⟩

end

end

5.3.3 Unwinding Theorems

theory *UnwindingResults*
imports *AuxiliaryLemmas*
begin

context *Unwinding*
begin
theorem *unwinding-theorem-BSD*:
$[\![$ *lrf ur*; *osc ur* $]\!] \Longrightarrow$ *BSD* $\mathcal{V}$ $Tr_{(induceES\ SES)}$
⟨*proof*⟩

theorem *unwinding-theorem-BSI*:
$[\![$ *lrb ur*; *osc ur* $]\!] \Longrightarrow$ *BSI* $\mathcal{V}$ $Tr_{(induceES\ SES)}$
⟨*proof*⟩

theorem *unwinding-theorem-BSIA*:
$[\![$ *lrbe* ϱ *ur*; *osc ur* $]\!] \Longrightarrow$ *BSIA* ϱ $\mathcal{V}$ $Tr_{(induceES\ SES)}$
⟨*proof*⟩

theorem *unwinding-theorem-FCD*:
$[\![$ *fcrf* Γ *ur*; *osc ur* $]\!] \Longrightarrow$ *FCD* Γ $\mathcal{V}$ $Tr_{(induceES\ SES)}$
⟨*proof*⟩

theorem *unwinding-theorem-FCI*:
$[\![$ *fcrb* Γ *ur*; *osc ur* $]\!] \Longrightarrow$ *FCI* Γ $\mathcal{V}$ $Tr_{(induceES\ SES)}$
⟨*proof*⟩

theorem *unwinding-theorem-FCIA*:
$[\![$ *fcrbe* Γ ϱ *ur*; *osc ur* $]\!] \Longrightarrow$ *FCIA* ϱ Γ $\mathcal{V}$ $Tr_{(induceES\ SES)}$
⟨*proof*⟩

theorem *unwinding-theorem-SD*:
$[\![$ $\mathcal{V}' = (\!|\ V = (V_{\mathcal{V}} \cup N_{\mathcal{V}}),\ N = \{\},\ C = C_{\mathcal{V}}\ |\!)$;
  Unwinding.lrf SES $\mathcal{V}'$ *ur*; *Unwinding.osc SES* $\mathcal{V}'$ *ur* $]\!]$
  $\Longrightarrow$ *SD* $\mathcal{V}$ $Tr_{(induceES\ SES)}$
⟨*proof*⟩

theorem *unwinding-theorem-SI*:
$[\![$ $\mathcal{V}' = (\!|\ V = (V_{\mathcal{V}} \cup N_{\mathcal{V}}),\ N = \{\},\ C = C_{\mathcal{V}}\ |\!)$;
  Unwinding.lrb SES $\mathcal{V}'$ *ur*; *Unwinding.osc SES* $\mathcal{V}'$ *ur* $]\!]$
  $\Longrightarrow$ *SI* $\mathcal{V}$ $Tr_{(induceES\ SES)}$
⟨*proof*⟩

theorem *unwinding-theorem-SIA*:
$[\![\ \mathcal{V}' = (\!|\ V = (V_{\mathcal{V}} \cup N_{\mathcal{V}}),\ N = \{\},\ C = C_{\mathcal{V}}\ |\!);\ \varrho\ \mathcal{V} = \varrho\ \mathcal{V}';$
$\mathit{Unwinding.lrbe}\ \mathit{SES}\ \mathcal{V}'\ \varrho\ \mathit{ur};\ \mathit{Unwinding.osc}\ \mathit{SES}\ \mathcal{V}'\ \mathit{ur}\ ]\!]$
$\Longrightarrow \mathit{SIA}\ \varrho\ \mathcal{V}\ \mathit{Tr}_{(\mathit{induceES}\ \mathit{SES})}$
⟨*proof*⟩

theorem *unwinding-theorem-SR*:
$[\![\ \mathcal{V}' = (\!|\ V = (V_{\mathcal{V}} \cup N_{\mathcal{V}}),\ N = \{\},\ C = C_{\mathcal{V}}\ |\!);$
$\mathit{Unwinding.lrf}\ \mathit{SES}\ \mathcal{V}'\ \mathit{ur};\ \mathit{Unwinding.osc}\ \mathit{SES}\ \mathcal{V}'\ \mathit{ur}\ ]\!]$
$\Longrightarrow \mathit{SR}\ \mathcal{V}\ \mathit{Tr}_{(\mathit{induceES}\ \mathit{SES})}$
⟨*proof*⟩

theorem *unwinding-theorem-D*:
$[\![\ \mathit{lrf}\ \mathit{ur};\ \mathit{osc}\ \mathit{ur}\ ]\!] \Longrightarrow D\ \mathcal{V}\ \mathit{Tr}_{(\mathit{induceES}\ \mathit{SES})}$
⟨*proof*⟩

theorem *unwinding-theorem-I*:
$[\![\ \mathit{lrb}\ \mathit{ur};\ \mathit{osc}\ \mathit{ur}\ ]\!] \Longrightarrow I\ \mathcal{V}\ \mathit{Tr}_{(\mathit{induceES}\ \mathit{SES})}$
⟨*proof*⟩

theorem *unwinding-theorem-IA*:
$[\![\ \mathit{lrbe}\ \varrho\ \mathit{ur};\ \mathit{osc}\ \mathit{ur}\ ]\!] \Longrightarrow \mathit{IA}\ \varrho\ \mathcal{V}\ \mathit{Tr}_{(\mathit{induceES}\ \mathit{SES})}$
⟨*proof*⟩

theorem *unwinding-theorem-R*:
$[\![\ \mathit{lrf}\ \mathit{ur};\ \mathit{osc}\ \mathit{ur}\ ]\!] \Longrightarrow R\ \mathcal{V}\ (\mathit{Tr}_{(\mathit{induceES}\ \mathit{SES})})$
⟨*proof*⟩

end

end

5.4 Compositionality

We prove the compositionality results from [3].

5.4.1 Auxiliary Definitions & Results

theory *CompositionBase*
imports *../Basics/BSPTaxonomy*
begin

definition
properSeparationOfViews ::
$'e\ \mathit{ES\text{-}rec} \Rightarrow\ 'e\ \mathit{ES\text{-}rec} \Rightarrow\ 'e\ \mathit{V\text{-}rec} \Rightarrow\ 'e\ \mathit{V\text{-}rec} \Rightarrow\ 'e\ \mathit{V\text{-}rec} \Rightarrow \mathit{bool}$
where
$\mathit{properSeparationOfViews}\ \mathit{ES1}\ \mathit{ES2}\ \mathcal{V}\ \mathcal{V}1\ \mathcal{V}2 \equiv$
$V_{\mathcal{V}} \cap E_{ES1} = V_{\mathcal{V}1}$
$\wedge\ V_{\mathcal{V}} \cap E_{ES2} = V_{\mathcal{V}2}$
$\wedge\ C_{\mathcal{V}} \cap E_{ES1} \subseteq C_{\mathcal{V}1}$
$\wedge\ C_{\mathcal{V}} \cap E_{ES2} \subseteq C_{\mathcal{V}2}$

$\wedge\ N_{\mathcal{V}1} \cap N_{\mathcal{V}2} = \{\}$

definition
wellBehavedComposition ::
$'e$ *ES-rec* $\Rightarrow$ $'e$ *ES-rec* $\Rightarrow$ $'e$ *V-rec* $\Rightarrow$ $'e$ *V-rec* $\Rightarrow$ $'e$ *V-rec* $\Rightarrow$ *bool*
where
wellBehavedComposition ES1 ES2 $\mathcal{V}$ $\mathcal{V}1$ $\mathcal{V}2$ $\equiv$
$(\ N_{\mathcal{V}1} \cap E_{ES2} = \{\} \wedge N_{\mathcal{V}2} \cap E_{ES1} = \{\}\ )$
$\vee\ (\exists\, \varrho 1.\ (\ N_{\mathcal{V}1} \cap E_{ES2} = \{\} \wedge$ *total ES1* $(C_{\mathcal{V}1} \cap N_{\mathcal{V}2})$
$\wedge$ *BSIA* $\varrho 1$ $\mathcal{V}1$ Tr_{ES1}))
$\vee\ (\exists\, \varrho 2.\ (\ N_{\mathcal{V}2} \cap E_{ES1} = \{\} \wedge$ *total ES2* $(C_{\mathcal{V}2} \cap N_{\mathcal{V}1})$
$\wedge$ *BSIA* $\varrho 2$ $\mathcal{V}2$ Tr_{ES2}))
$\vee\ (\exists\, \varrho 1\ \varrho 2\ \Gamma 1\ \Gamma 2.$ (
$\nabla_{\Gamma 1} \subseteq E_{ES1} \wedge \Delta_{\Gamma 1} \subseteq E_{ES1} \wedge \Upsilon_{\Gamma 1} \subseteq E_{ES1}$
$\wedge\ \nabla_{\Gamma 2} \subseteq E_{ES2} \wedge \Delta_{\Gamma 2} \subseteq E_{ES2} \wedge \Upsilon_{\Gamma 2} \subseteq E_{ES2}$
$\wedge$ *BSIA* $\varrho 1$ $\mathcal{V}1$ Tr_{ES1} $\wedge$ *BSIA* $\varrho 2$ $\mathcal{V}2$ Tr_{ES2}
$\wedge$ *total ES1* $(C_{\mathcal{V}1} \cap N_{\mathcal{V}2}) \wedge$ *total ES2* $(C_{\mathcal{V}2} \cap N_{\mathcal{V}1})$
$\wedge$ *FCIA* $\varrho 1$ $\Gamma 1$ $\mathcal{V}1$ Tr_{ES1} $\wedge$ *FCIA* $\varrho 2$ $\Gamma 2$ $\mathcal{V}2$ Tr_{ES2}
$\wedge\ V_{\mathcal{V}1} \cap V_{\mathcal{V}2} \subseteq \nabla_{\Gamma 1} \cup \nabla_{\Gamma 2}$
$\wedge\ C_{\mathcal{V}1} \cap N_{\mathcal{V}2} \subseteq \Upsilon_{\Gamma 1} \wedge C_{\mathcal{V}2} \cap N_{\mathcal{V}1} \subseteq \Upsilon_{\Gamma 2}$
$\wedge\ N_{\mathcal{V}1} \cap \Delta_{\Gamma 1} \cap E_{ES2} = \{\} \wedge N_{\mathcal{V}2} \cap \Delta_{\Gamma 2} \cap E_{ES1} = \{\}$))

locale *Compositionality* =
fixes *ES1* :: $'e$ *ES-rec*
and *ES2* :: $'e$ *ES-rec*
and $\mathcal{V}$ :: $'e$ *V-rec*
and $\mathcal{V}1$ :: $'e$ *V-rec*
and $\mathcal{V}2$ :: $'e$ *V-rec*

assumes *validES1*: *ES-valid ES1*
and *validES2*: *ES-valid ES2*
and *composableES1ES2*: *composable ES1 ES2*

and *validVC*: *isViewOn* $\mathcal{V}$ $(E_{(ES1\ \parallel\ ES2)})$
and *validV1*: *isViewOn* $\mathcal{V}1$ E_{ES1}
and *validV2*: *isViewOn* $\mathcal{V}2$ E_{ES2}

and *propSepViews*: *properSeparationOfViews ES1 ES2* $\mathcal{V}$ $\mathcal{V}1$ $\mathcal{V}2$

and *well-behaved-composition*: *wellBehavedComposition ES1 ES2* $\mathcal{V}$ $\mathcal{V}1$ $\mathcal{V}2$

sublocale *Compositionality* $\subseteq$ *BSPTaxonomyDifferentCorrections ES1* $\parallel$ *ES2* $\mathcal{V}$
$\langle$*proof*$\rangle$

context *Compositionality*
begin

lemma *Vv-is-Vv1-union-Vv2*: $V_{\mathcal{V}} = V_{\mathcal{V}1} \cup V_{\mathcal{V}2}$
⟨*proof*⟩

lemma *disjoint-Nv1-Vv2*: $N_{\mathcal{V}1} \cap V_{\mathcal{V}2} = \{\}$
⟨*proof*⟩

lemma *disjoint-Nv2-Vv1*: $N_{\mathcal{V}2} \cap V_{\mathcal{V}1} = \{\}$
⟨*proof*⟩

lemma *merge-property'*: $[\![$ *set t1* $\subseteq E_{ES1}$; *set t2* $\subseteq E_{ES2}$;
$t1 \upharpoonright E_{ES2} = t2 \upharpoonright E_{ES1}$; $t1 \upharpoonright V_{\mathcal{V}} = []$; $t2 \upharpoonright V_{\mathcal{V}} = []$;
$t1 \upharpoonright C_{\mathcal{V}} = []$; $t2 \upharpoonright C_{\mathcal{V}} = []$ $]\!]$
$\Longrightarrow \exists\ t.\ (t \upharpoonright E_{ES1} = t1 \wedge t \upharpoonright E_{ES2} = t2 \wedge t \upharpoonright V_{\mathcal{V}} = [] \wedge t \upharpoonright C_{\mathcal{V}} = [] \wedge$ *set* $t \subseteq (E_{ES1} \cup E_{ES2}))$
⟨*proof*⟩

lemma *Nv1-union-Nv2-subsetof-Nv*: $N_{\mathcal{V}1} \cup N_{\mathcal{V}2} \subseteq N_{\mathcal{V}}$
⟨*proof*⟩

end

end
theory *CompositionSupport*
imports *CompositionBase*
begin

locale *CompositionSupport* =
fixes *ESi* :: *'e ES-rec*
and $\mathcal{V}$ :: *'e V-rec*
and $\mathcal{V}i$ :: *'e V-rec*

assumes *validESi*: *ES-valid ESi*

and *validVi*: *isViewOn* $\mathcal{V}i$ E_{ESi}
and *Vv-inter-Ei-is-Vvi*: $V_{\mathcal{V}} \cap E_{ESi} = V_{\mathcal{V}i}$
and *Cv-inter-Ei-subsetof-Cvi*: $C_{\mathcal{V}} \cap E_{ESi} \subseteq C_{\mathcal{V}i}$

context *CompositionSupport*
begin

lemma *BSD-in-subsystem*:
$\llbracket\ c \in C_{\mathcal{V}};\ ((\beta\ @\ [c]\ @\ \alpha) \upharpoonright E_{ESi}) \in Tr_{ESi}\ ;\ BSD\ \mathcal{V}i\ Tr_{ESi}\ \rrbracket$
$\Longrightarrow \exists\,\alpha\text{-}i'.\ (\ ((\beta \upharpoonright E_{ESi})\ @\ \alpha\text{-}i') \in Tr_{ESi}$
$\wedge\ (\alpha\text{-}i' \upharpoonright V_{\mathcal{V}i}) = (\alpha \upharpoonright V_{\mathcal{V}i}) \wedge \alpha\text{-}i' \upharpoonright C_{\mathcal{V}i} = []\ )$
⟨*proof*⟩

lemma *BSD-in-subsystem2*:
$\llbracket\ ((\beta\ @\ \alpha) \upharpoonright E_{ESi}) \in Tr_{ESi}\ ;\ BSD\ \mathcal{V}i\ Tr_{ESi}\ \rrbracket$
$\Longrightarrow \exists\ \alpha\text{-}i'.\ (\ ((\beta \upharpoonright E_{ESi})\ @\ \alpha\text{-}i') \in Tr_{ESi} \wedge (\alpha\text{-}i' \upharpoonright V_{\mathcal{V}i}) = (\alpha \upharpoonright V_{\mathcal{V}i}) \wedge \alpha\text{-}i' \upharpoonright C_{\mathcal{V}i} = []\ )$
⟨*proof*⟩

end

end

5.4.2 Generalized Zipping Lemma

theory *GeneralizedZippingLemma*
imports *CompositionBase*
begin

context *Compositionality*
begin

lemma *generalized-zipping-lemma1*: $\llbracket\ N_{\mathcal{V}1} \cap E_{ES2} = \{\};\ N_{\mathcal{V}2} \cap E_{ES1} = \{\}\ \rrbracket \Longrightarrow$
$\forall\ \tau\ lambda\ t1\ t2.\ (\ (\ set\ \tau \subseteq E_{(ES1\ \|\ ES2)} \wedge set\ lambda \subseteq V_{\mathcal{V}} \wedge set\ t1 \subseteq E_{ES1} \wedge set\ t2 \subseteq E_{ES2}$
$\wedge\ ((\tau \upharpoonright E_{ES1})\ @\ t1) \in Tr_{ES1} \wedge ((\tau \upharpoonright E_{ES2})\ @\ t2) \in Tr_{ES2} \wedge (lambda \upharpoonright E_{ES1}) = (t1 \upharpoonright V_{\mathcal{V}})$
$\wedge\ (lambda \upharpoonright E_{ES2}) = (t2 \upharpoonright V_{\mathcal{V}}) \wedge (t1 \upharpoonright C_{\mathcal{V}1}) = [] \wedge (t2 \upharpoonright C_{\mathcal{V}2}) = [])$
$\longrightarrow (\exists\ t.\ ((\tau\ @\ t) \in Tr_{(ES1\ \|\ ES2)} \wedge (t \upharpoonright V_{\mathcal{V}}) = lambda \wedge (t \upharpoonright C_{\mathcal{V}}) = []))\ )$
⟨*proof*⟩

lemma *generalized-zipping-lemma2*: $\llbracket\ N_{\mathcal{V}1} \cap E_{ES2} = \{\};\ total\ ES1\ (C_{\mathcal{V}1} \cap N_{\mathcal{V}2});\ BSIA\ \varrho 1\ \mathcal{V}1\ Tr_{ES1}\ \rrbracket$
$\Longrightarrow$
$\forall\ \tau\ lambda\ t1\ t2.\ (\ (\ set\ \tau \subseteq (E_{(ES1\ \|\ ES2)}) \wedge set\ lambda \subseteq V_{\mathcal{V}} \wedge set\ t1 \subseteq E_{ES1} \wedge set\ t2 \subseteq E_{ES2}$
$\wedge\ ((\tau \upharpoonright E_{ES1})\ @\ t1) \in Tr_{ES1} \wedge ((\tau \upharpoonright E_{ES2})\ @\ t2) \in Tr_{ES2}$
$\wedge\ (lambda \upharpoonright E_{ES1}) = (t1 \upharpoonright V_{\mathcal{V}}) \wedge (lambda \upharpoonright E_{ES2}) = (t2 \upharpoonright V_{\mathcal{V}})$
$\wedge\ (t1 \upharpoonright C_{\mathcal{V}1}) = [] \wedge (t2 \upharpoonright C_{\mathcal{V}2}) = [])$
$\longrightarrow (\exists\ t.\ ((\tau\ @\ t) \in (Tr_{(ES1\ \|\ ES2)}) \wedge (t \upharpoonright V_{\mathcal{V}}) = lambda \wedge (t \upharpoonright C_{\mathcal{V}}) = []))\ )$
⟨*proof*⟩

lemma *generalized-zipping-lemma3*: $\llbracket\ N_{\mathcal{V}2} \cap E_{ES1} = \{\};\ total\ ES2\ (C_{\mathcal{V}2} \cap N_{\mathcal{V}1});\ BSIA\ \varrho 2\ \mathcal{V}2\ Tr_{ES2}\ \rrbracket$
$\Longrightarrow$
$\forall\ \tau\ lambda\ t1\ t2.\ (\ (\ set\ \tau \subseteq E_{(ES1\ \|\ ES2)} \wedge set\ lambda \subseteq V_{\mathcal{V}} \wedge set\ t1 \subseteq E_{ES1} \wedge set\ t2 \subseteq E_{ES2}$
$\wedge\ ((\tau \upharpoonright E_{ES1})\ @\ t1) \in Tr_{ES1} \wedge ((\tau \upharpoonright E_{ES2})\ @\ t2) \in Tr_{ES2}$

$\wedge$ (*lambda* $\upharpoonright$ E_{ES1}) = (*t1* $\upharpoonright$ $V_{\mathcal{V}}$) $\wedge$ (*lambda* $\upharpoonright$ E_{ES2}) = (*t2* $\upharpoonright$ $V_{\mathcal{V}}$)
$\wedge$ (*t1* $\upharpoonright$ $C_{\mathcal{V}1}$) = [] $\wedge$ (*t2* $\upharpoonright$ $C_{\mathcal{V}2}$) = [])
$\longrightarrow$ ($\exists$ *t*. ((τ @ *t*) $\in$ $Tr_{(ES1 \parallel ES2)}$ $\wedge$ (*t* $\upharpoonright$ $V_{\mathcal{V}}$) = *lambda* $\wedge$ (*t* $\upharpoonright$ $C_{\mathcal{V}}$) = [])))
⟨*proof*⟩

lemma *generalized-zipping-lemma4*:
⟦ $\nabla_{\Gamma 1} \subseteq E_{ES1}$; $\Delta_{\Gamma 1} \subseteq E_{ES1}$; $\Upsilon_{\Gamma 1} \subseteq E_{ES1}$; $\nabla_{\Gamma 2} \subseteq E_{ES2}$; $\Delta_{\Gamma 2} \subseteq E_{ES2}$; $\Upsilon_{\Gamma 2} \subseteq E_{ES2}$;
BSIA $\varrho 1$ $\mathcal{V}1$ Tr_{ES1}; *BSIA* $\varrho 2$ $\mathcal{V}2$ Tr_{ES2}; *total ES1* ($C_{\mathcal{V}1} \cap N_{\mathcal{V}2}$); *total ES2* ($C_{\mathcal{V}2} \cap N_{\mathcal{V}1}$);
FCIA $\varrho 1$ $\Gamma 1$ $\mathcal{V}1$ Tr_{ES1}; *FCIA* $\varrho 2$ $\Gamma 2$ $\mathcal{V}2$ Tr_{ES2}; $V_{\mathcal{V}1} \cap V_{\mathcal{V}2} \subseteq \nabla_{\Gamma 1} \cup \nabla_{\Gamma 2}$;
$C_{\mathcal{V}1} \cap N_{\mathcal{V}2} \subseteq \Upsilon_{\Gamma 1}$; $C_{\mathcal{V}2} \cap N_{\mathcal{V}1} \subseteq \Upsilon_{\Gamma 2}$;
$N_{\mathcal{V}1} \cap \Delta_{\Gamma 1} \cap E_{ES2} = \{\}$; $N_{\mathcal{V}2} \cap \Delta_{\Gamma 2} \cap E_{ES1} = \{\}$ ⟧ $\Longrightarrow$
$\forall$ τ *lambda t1 t2*. ((*set* $\tau \subseteq$ ($E_{(ES1 \parallel ES2)}$) $\wedge$ *set lambda* $\subseteq V_{\mathcal{V}}$ $\wedge$ *set t1* $\subseteq E_{ES1}$
$\wedge$ *set t2* $\subseteq E_{ES2}$ $\wedge$ (($\tau \upharpoonright E_{ES1}$) @ *t1*) $\in Tr_{ES1}$ $\wedge$ (($\tau \upharpoonright E_{ES2}$) @ *t2*) $\in Tr_{ES2}$
$\wedge$ (*lambda* $\upharpoonright$ E_{ES1}) = (*t1* $\upharpoonright$ $V_{\mathcal{V}}$) $\wedge$ (*lambda* $\upharpoonright$ E_{ES2}) = (*t2* $\upharpoonright$ $V_{\mathcal{V}}$)
$\wedge$ (*t1* $\upharpoonright$ $C_{\mathcal{V}1}$) = [] $\wedge$ (*t2* $\upharpoonright$ $C_{\mathcal{V}2}$) = [])
$\longrightarrow$ ($\exists$ *t*. ((τ @ *t*) $\in$ ($Tr_{(ES1 \parallel ES2)}$) $\wedge$ (*t* $\upharpoonright$ $V_{\mathcal{V}}$) = *lambda* $\wedge$ (*t* $\upharpoonright$ $C_{\mathcal{V}}$) = [])))
⟨*proof*⟩

lemma *generalized-zipping-lemma*:
$\forall$ τ *lambda t1 t2*. ((*set* $\tau \subseteq E_{(ES1 \parallel ES2)}$
$\wedge$ *set lambda* $\subseteq V_{\mathcal{V}}$ $\wedge$ *set t1* $\subseteq E_{ES1}$ $\wedge$ *set t2* $\subseteq E_{ES2}$
$\wedge$ (($\tau \upharpoonright E_{ES1}$) @ *t1*) $\in Tr_{ES1}$ $\wedge$ (($\tau \upharpoonright E_{ES2}$) @ *t2*) $\in Tr_{ES2}$
$\wedge$ (*lambda* $\upharpoonright$ E_{ES1}) = (*t1* $\upharpoonright$ $V_{\mathcal{V}}$) $\wedge$ (*lambda* $\upharpoonright$ E_{ES2}) = (*t2* $\upharpoonright$ $V_{\mathcal{V}}$)
$\wedge$ (*t1* $\upharpoonright$ $C_{\mathcal{V}1}$) = [] $\wedge$ (*t2* $\upharpoonright$ $C_{\mathcal{V}2}$) = [])
$\longrightarrow$ ($\exists$ *t*. ((τ @ *t*) $\in$ $Tr_{(ES1 \parallel ES2)}$ $\wedge$ (*t* $\upharpoonright$ $V_{\mathcal{V}}$) = *lambda* $\wedge$ (*t* $\upharpoonright$ $C_{\mathcal{V}}$) = [])))
⟨*proof*⟩

end

end

5.4.3 Compositionality Results

theory *CompositionalityResults*
imports *GeneralizedZippingLemma CompositionSupport*
begin

context *Compositionality*
begin

theorem *compositionality-BSD*:
⟦ *BSD* $\mathcal{V}1$ Tr_{ES1}; *BSD* $\mathcal{V}2$ Tr_{ES2} ⟧ $\Longrightarrow$ *BSD* $\mathcal{V}$ $Tr_{(ES1 \parallel ES2)}$
⟨*proof*⟩

theorem *compositionality-BSI*:
⟦ *BSD* $\mathcal{V}1$ Tr_{ES1}; *BSD* $\mathcal{V}2$ Tr_{ES2}; *BSI* $\mathcal{V}1$ Tr_{ES1}; *BSI* $\mathcal{V}2$ Tr_{ES2} ⟧

$\Longrightarrow$ *BSI* $\mathcal{V}$ $Tr_{(ES1 \parallel ES2)}$
⟨*proof*⟩

theorem *compositionality-BSIA*:
$[\![$ *BSD* $\mathcal{V}1$ Tr_{ES1}; *BSD* $\mathcal{V}2$ Tr_{ES2}; *BSIA* $\varrho 1$ $\mathcal{V}1$ Tr_{ES1}; *BSIA* $\varrho 2$ $\mathcal{V}2$ Tr_{ES2};
$(\varrho 1\ \mathcal{V}1) \subseteq (\varrho\ \mathcal{V}) \cap E_{ES1}$; $(\varrho 2\ \mathcal{V}2) \subseteq (\varrho\ \mathcal{V}) \cap E_{ES2}$ $]\!]$
$\Longrightarrow$ *BSIA* ϱ $\mathcal{V}$ $(Tr_{(ES1 \parallel ES2)})$
⟨*proof*⟩

theorem *compositionality-FCD*:
$[\![$ *BSD* $\mathcal{V}1$ Tr_{ES1}; *BSD* $\mathcal{V}2$ Tr_{ES2};
$\nabla_{\Gamma} \cap E_{ES1} \subseteq \nabla_{\Gamma 1}$; $\nabla_{\Gamma} \cap E_{ES2} \subseteq \nabla_{\Gamma 2}$;
$\Upsilon_{\Gamma} \cap E_{ES1} \subseteq \Upsilon_{\Gamma 1}$; $\Upsilon_{\Gamma} \cap E_{ES2} \subseteq \Upsilon_{\Gamma 2}$;
$(\ \Delta_{\Gamma 1} \cap N_{\mathcal{V}1} \cup \Delta_{\Gamma 2} \cap N_{\mathcal{V}2}\ ) \subseteq \Delta_{\Gamma}$;
$N_{\mathcal{V}1} \cap \Delta_{\Gamma 1} \cap E_{ES2} = \{\}$; $N_{\mathcal{V}2} \cap \Delta_{\Gamma 2} \cap E_{ES1} = \{\}$;
FCD $\Gamma 1$ $\mathcal{V}1$ Tr_{ES1}; *FCD* $\Gamma 2$ $\mathcal{V}2$ Tr_{ES2} $]\!]$
$\Longrightarrow$ *FCD* Γ $\mathcal{V}$ $(Tr_{(ES1 \parallel ES2)})$
⟨*proof*⟩

theorem *compositionality-FCI*:
$[\![$ *BSD* $\mathcal{V}1$ Tr_{ES1}; *BSD* $\mathcal{V}2$ Tr_{ES2}; *BSIA* $\varrho 1$ $\mathcal{V}1$ Tr_{ES1}; *BSIA* $\varrho 2$ $\mathcal{V}2$ Tr_{ES2};
total ES1 $(C_{\mathcal{V}1} \cap \Upsilon_{\Gamma 1})$; *total ES2* $(C_{\mathcal{V}2} \cap \Upsilon_{\Gamma 2})$;
$\nabla_{\Gamma} \cap E_{ES1} \subseteq \nabla_{\Gamma 1}$; $\nabla_{\Gamma} \cap E_{ES2} \subseteq \nabla_{\Gamma 2}$;
$\Upsilon_{\Gamma} \cap E_{ES1} \subseteq \Upsilon_{\Gamma 1}$; $\Upsilon_{\Gamma} \cap E_{ES2} \subseteq \Upsilon_{\Gamma 2}$;
$(\ \Delta_{\Gamma 1} \cap N_{\mathcal{V}1} \cup \Delta_{\Gamma 2} \cap N_{\mathcal{V}2}\ ) \subseteq \Delta_{\Gamma}$;
$(N_{\mathcal{V}1} \cap \Delta_{\Gamma 1} \cap E_{ES2} = \{\} \wedge N_{\mathcal{V}2} \cap \Delta_{\Gamma 2} \cap E_{ES1} \subseteq \Upsilon_{\Gamma 1})$
$\vee$ $(\ N_{\mathcal{V}2} \cap \Delta_{\Gamma 2} \cap E_{ES1} = \{\} \wedge N_{\mathcal{V}1} \cap \Delta_{\Gamma 1} \cap E_{ES2} \subseteq \Upsilon_{\Gamma 2})$;
FCI $\Gamma 1$ $\mathcal{V}1$ Tr_{ES1}; *FCI* $\Gamma 2$ $\mathcal{V}2$ Tr_{ES2} $]\!]$
$\Longrightarrow$ *FCI* Γ $\mathcal{V}$ $(Tr_{(ES1 \parallel ES2)})$
⟨*proof*⟩

theorem *compositionality-FCIA*:
$[\![$ *BSD* $\mathcal{V}1$ Tr_{ES1}; *BSD* $\mathcal{V}2$ Tr_{ES2}; *BSIA* $\varrho 1$ $\mathcal{V}1$ Tr_{ES1}; *BSIA* $\varrho 2$ $\mathcal{V}2$ Tr_{ES2};
$(\varrho 1\ \mathcal{V}1) \subseteq (\varrho\ \mathcal{V}) \cap E_{ES1}$; $(\varrho 2\ \mathcal{V}2) \subseteq (\varrho\ \mathcal{V}) \cap E_{ES2}$;
total ES1 $(C_{\mathcal{V}1} \cap \Upsilon_{\Gamma 1} \cap N_{\mathcal{V}2} \cap \Delta_{\Gamma 2})$; *total ES2* $(C_{\mathcal{V}2} \cap \Upsilon_{\Gamma 2} \cap N_{\mathcal{V}1} \cap \Delta_{\Gamma 1})$;
$\nabla_{\Gamma} \cap E_{ES1} \subseteq \nabla_{\Gamma 1}$; $\nabla_{\Gamma} \cap E_{ES2} \subseteq \nabla_{\Gamma 2}$;
$\Upsilon_{\Gamma} \cap E_{ES1} \subseteq \Upsilon_{\Gamma 1}$; $\Upsilon_{\Gamma} \cap E_{ES2} \subseteq \Upsilon_{\Gamma 2}$;
$(\ \Delta_{\Gamma 1} \cap N_{\mathcal{V}1} \cup \Delta_{\Gamma 2} \cap N_{\mathcal{V}2}\ ) \subseteq \Delta_{\Gamma}$;
$(N_{\mathcal{V}1} \cap \Delta_{\Gamma 1} \cap E_{ES2} = \{\} \wedge N_{\mathcal{V}2} \cap \Delta_{\Gamma 2} \cap E_{ES1} \subseteq \Upsilon_{\Gamma 1})$
$\vee$ $(\ N_{\mathcal{V}2} \cap \Delta_{\Gamma 2} \cap E_{ES1} = \{\} \wedge N_{\mathcal{V}1} \cap \Delta_{\Gamma 1} \cap E_{ES2} \subseteq \Upsilon_{\Gamma 2})$;
FCIA $\varrho 1$ $\Gamma 1$ $\mathcal{V}1$ Tr_{ES1}; *FCIA* $\varrho 2$ $\Gamma 2$ $\mathcal{V}2$ Tr_{ES2} $]\!]$
$\Longrightarrow$ *FCIA* ϱ Γ $\mathcal{V}$ $(Tr_{(ES1 \parallel ES2)})$
⟨*proof*⟩

theorem *compositionality-R*:

$[\![R\ \mathcal{V}1\ Tr_{ES1};\ R\ \mathcal{V}2\ Tr_{ES2}\ ]\!] \Longrightarrow R\ \mathcal{V}\ (Tr_{(ES1\ \|\ ES2)})$
⟨*proof*⟩

end

locale *CompositionalityStrictBSPs* = *Compositionality* +

assumes *N$\mathcal{V}$-inter-E1-is-N$\mathcal{V}$1*: $N_{\mathcal{V}} \cap E_{ES1} = N_{\mathcal{V}1}$
and *N$\mathcal{V}$-inter-E2-is-N$\mathcal{V}$2*: $N_{\mathcal{V}} \cap E_{ES2} = N_{\mathcal{V}2}$

sublocale *CompositionalityStrictBSPs* ⊆ *Compositionality*
⟨*proof*⟩

context *CompositionalityStrictBSPs*
begin

theorem *compositionality-SR*:
$[\![SR\ \mathcal{V}1\ Tr_{ES1};\ SR\ \mathcal{V}2\ Tr_{ES2}\ ]\!] \Longrightarrow SR\ \mathcal{V}\ (Tr_{(ES1\ \|\ ES2)})$
⟨*proof*⟩

theorem *compositionality-SD*:
$[\![SD\ \mathcal{V}1\ Tr_{ES1};\ SD\ \mathcal{V}2\ Tr_{ES2}\ ]\!] \Longrightarrow SD\ \mathcal{V}\ (Tr_{(ES1\ \|\ ES2)})$
⟨*proof*⟩

theorem *compositionality-SI*:
$[\![SD\ \mathcal{V}1\ Tr_{ES1};\ SD\ \mathcal{V}2\ Tr_{ES2};\ SI\ \mathcal{V}1\ Tr_{ES1};\ SI\ \mathcal{V}2\ Tr_{ES2}\ ]\!]$
$\Longrightarrow SI\ \mathcal{V}\ (Tr_{(ES1\ \|\ ES2)})$
⟨*proof*⟩

theorem *compositionality-SIA*:
$[\![SD\ \mathcal{V}1\ Tr_{ES1};\ SD\ \mathcal{V}2\ Tr_{ES2};\ SIA\ \varrho1\ \mathcal{V}1\ Tr_{ES1};\ SIA\ \varrho2\ \mathcal{V}2\ Tr_{ES2};$
$(\varrho1\ \mathcal{V}1) \subseteq (\varrho\ \mathcal{V}) \cap E_{ES1};\ (\varrho2\ \mathcal{V}2) \subseteq (\varrho\ \mathcal{V}) \cap E_{ES2}\ ]\!]$
$\Longrightarrow SIA\ \varrho\ \mathcal{V}\ (Tr_{(ES1\ \|\ ES2)})$
⟨*proof*⟩
end

end

Acknowledgments

This work was partially funded by the DFG (German Research Foundation) under the projects FM-SecEng (MA 3326/1-2, MA 3326/1-3) and RSCP (MA 3326/4-3).

References

[1] S. Grewe, H. Mantel, M. Tasch, R. Gay, and H. Sudbrock. I-MAKS – A Framework for Information-Flow Security in Isabelle/HOL. Technical Report TUD-CS-2018-0056, TU Darmstadt, 2018.

[2] H. Mantel. Possibilistic Definitions of Security – An Assembly Kit. In *Proceedings of the 13th IEEE Computer Security Foundations Workshop (CSFW)*, pages 185–199, 2000.

[3] H. Mantel. *A Uniform Framework for the Formal Specification and Verification of Information Flow Security*. PhD thesis, Saarland University, Saarbrücken, Germany, 2003.