Compositional Verification of Secure Applet Interactions

Marieke Huisman
INRIA Sophia Antipolis, France
Marieke.Huisman@sophia.inria.fr

Joint work with:

Dilian Gurov
SICS, Sweden
dilian@sics.se

Gilles Barthe
INRIA Sophia Antipolis, France
Gilles.Barthe@sophia.inria.fr

Introduction

- Growing interest in verification of (control flow based) security properties
- New generation of multi-application smart cards: post-issuance loading of applets
- Decomposition of properties desirable (also useful for management of verification)

A framework for compositional verification

- Program model
- Temporal logic specification language + high level patterns
- Proof system to show correctness of decomposition

Background

- VerifiCard project: reasoning about JavaCard
- Representation of program as call-transfer graph (Jensen, Le Métayer, Thorn, 1999)
- Context Free Processes: efficient model checking algorithms (polynomial in the size of the CFP) (Burkart & Steffen, 1992, Esparza & Schwoon, 2001)
- Compositional proof systems for modal μ-calculus (Dam & Gurov, 1998)

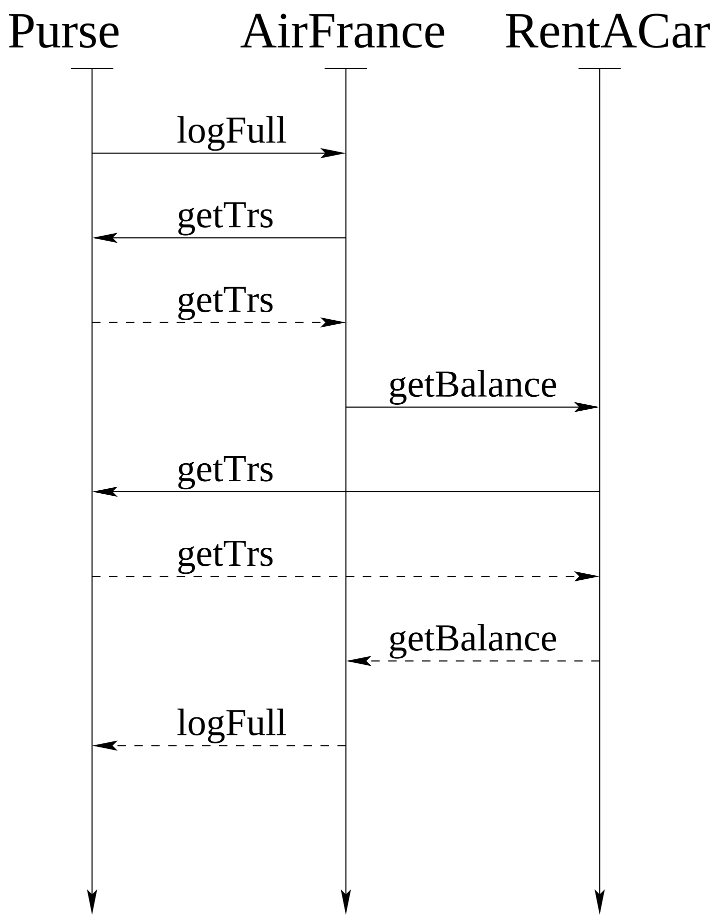
A framework for compositional verification

- Program model
- Temporal logic specification language + high level patterns
- Proof system to show correctness of decomposition

Running example: electronic purse

- Three applets: purse + 2 loyalties (AirFrance, RentACar)
- Purse keeps log table of transactions
- Loyalties can subscribe to `logFull` service
- AirFrance subscribed, RentACar not subscribed
- `logFull`: AirFrance asks purse for transactions, and RentACar for balance
- RentACar deduces that log table is full

Electronic purse: bad scenario



The program model

Compositional program model

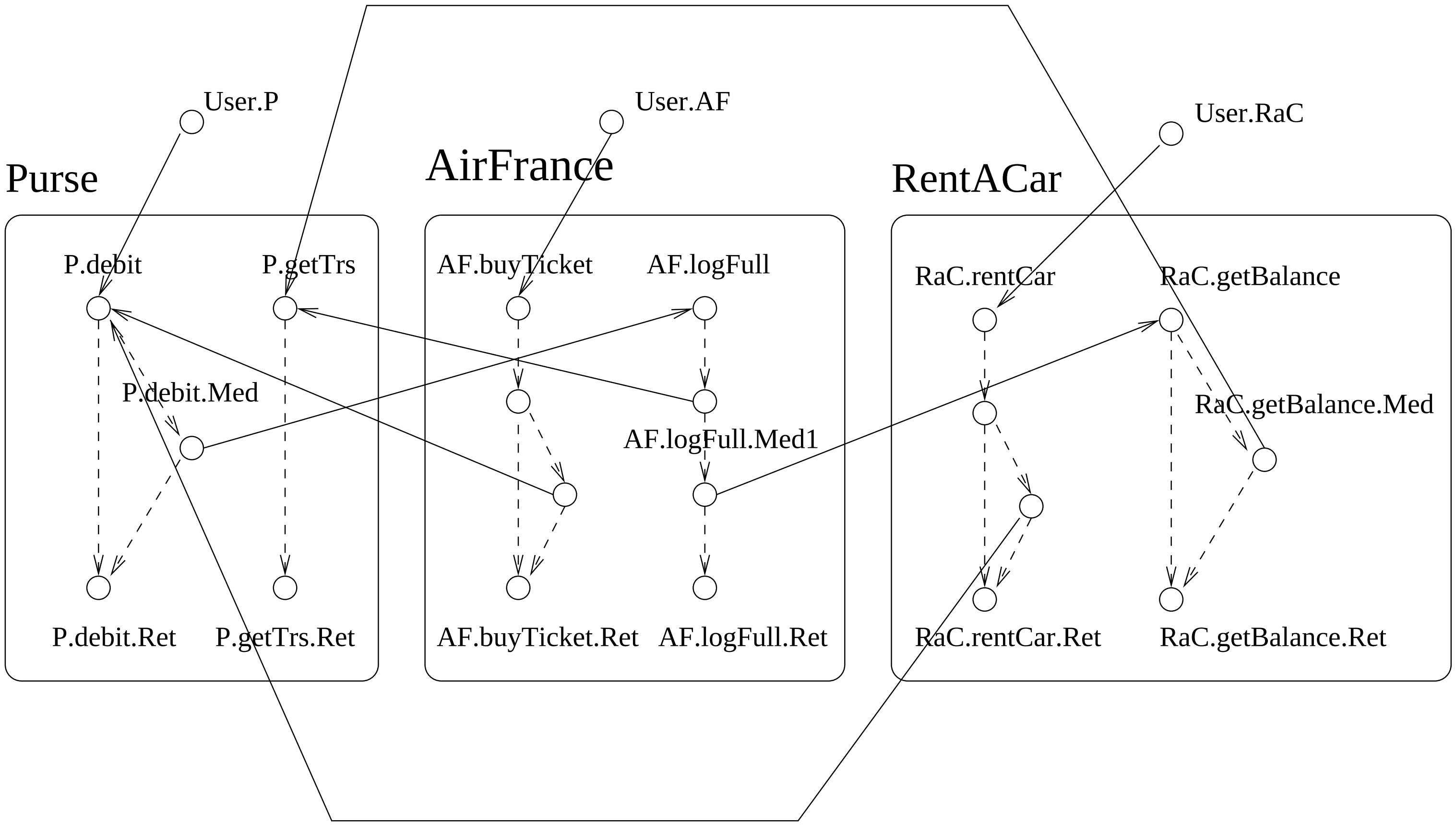
- Program: set of applets, communicating via method invocations and returns
- External program points
- Applet: transfer graph + call graph
- Each applet: local control stack
- Global state: well-formed set of local states
- Transition rules induce operational semantics

Formally

$$\mathcal{M} \triangleq (A, V; \textbf{app}, \textbf{ret}; \rightarrow^{T}, \rightarrow^{C})$$

- A a set of applets
- vertices V are program points
- $\textbf{app} : V \rightharpoonup A$ attributes vertices to applets
- set of return vertices identified by **ret**
- program $=$ transfer graph $\rightarrow^{T}$ $+$ call graph $\rightarrow^{C}$, distributed over different applets

Example compositional model for electronic purse



Operational semantics

- Two sets of transition rules:
  - Applet transition rules
  - Transition rules to combine behaviour of sets of applets
- Transition labels denote method invocations and returns
  - Perfect actions: internal flow or method invocation/return local to applet set (call, ret)
  - Imperfect actions: method invocation/return over applet set boundaries (call?, call!, ret?, ret!)
- Imperfect actions can synchronise, forming perfect actions

Local state per applet

Every applet has local state: list of pairs, representing the control stack

Example: local states in the Purse

When `AF.logFull.Med1` has been reached:

- $\mathtt{Purse}.\pi = \langle \mathtt{User.P}, \mathtt{P.debit.Med} \rangle \cdot \langle \mathtt{P.debit.Med}, \mathtt{AF.logFull} \rangle$
- $\mathtt{AirFrance}.\pi = \langle \mathtt{P.debit.Med}, \mathtt{AF.logFull.Med1} \rangle$
- $\mathtt{RentACar}.\pi = \epsilon$

States formally

- An applet state of $\mathcal{M}$ is a pair $a.\pi$, where $a \in A$ and $\pi \in (V \times V)^*$
- A program state of $\mathcal{M}$ is a wellformed collection $a_1.\pi_1 \,|\, a_2.\pi_2 \,|\, \ldots \,|\, a_n.\pi_n$ of applet states.

Active

- $a.\pi$ is active iff the last program point in π is local to applet a
- A program state is called active iff it contains an active applet state.

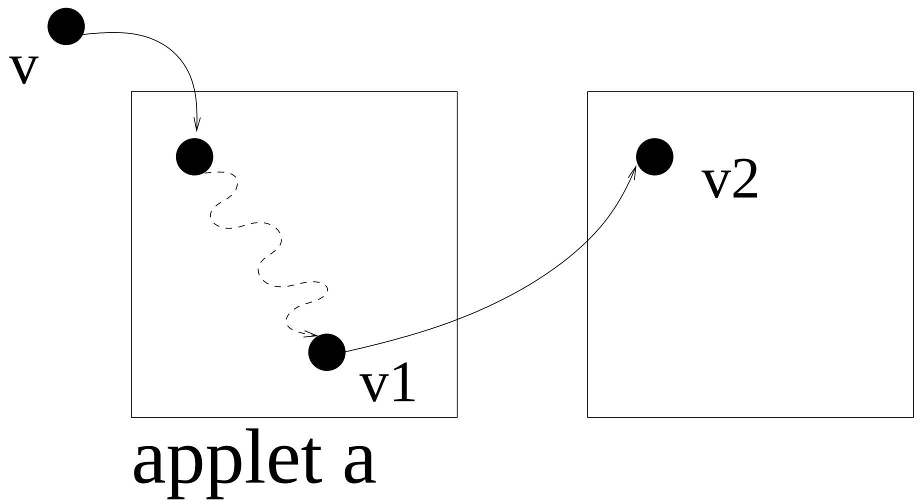
Wellformedness condition on program states

A program state is wellformed iff

- each applet name is mentioned at most once
- at most one applet is active

Example applet transition rule: send call

$$[\textsf{send call}] \quad \frac{v_1 \rightarrow^C v_2 \qquad \mathbf{loc}_a\, v_1 \qquad \neg\mathbf{loc}_a\, v_2}{a.\pi \cdot \langle v, v_1\rangle \xrightarrow{v_1\ \textsf{call!}\ v_2} a.\pi \cdot \langle v, v_1\rangle \cdot \langle v_1, v_2\rangle}$$



Applet transition rules

$$[\text{local call}] \quad \frac{v_1 \to^C v_2 \qquad \mathbf{loc}_a\, v_1 \qquad \mathbf{loc}_a\, v_2}{a.\pi \cdot \langle v, v_1 \rangle \xrightarrow{v_1 \text{ call } v_2} a.\pi \cdot \langle v, v_1, \cdot \rangle \, \langle v_1, v_2 \rangle}$$

$$[\text{local return}] \quad \frac{v_1 \to^T v_2 \qquad \mathbf{loc}_a\, v_1 \qquad \mathbf{loc}_a\, v_3 \qquad \mathbf{ret}\, v_3}{a.\pi \cdot \langle v, v_1 \rangle \cdot \langle v_1, v_3 \rangle \xrightarrow{v_3 \text{ ret } v_1} a.\pi \cdot \langle v, v_2 \rangle}$$

$$[\text{local transfer}] \quad \frac{v_1 \to^T v_2 \qquad \mathbf{loc}_a\, v_1 \qquad v_1 \not\to^C}{a.\pi \cdot \langle v, v_1 \rangle \longrightarrow a.\pi \cdot \langle v, v_2 \rangle}$$

$$\text{[send call]}\quad \frac{v_1 \to^{C} v_2 \qquad \mathbf{loc}_a\, v_1 \qquad \neg\mathbf{loc}_a\, v_2}{a.\pi \cdot \langle v, v_1\rangle \xrightarrow{v_1\ \mathsf{call!}\ v_2} a.\pi \cdot \langle v, v_1\rangle \cdot \langle v_1, v_2\rangle}$$

$$\text{[receive call]}\quad \frac{v_1 \to^{C} v_2 \qquad \neg\mathbf{loc}_a\, v_1 \qquad \mathbf{loc}_a\, v_2 \qquad \neg\mathsf{active} a}{a.\pi \xrightarrow{v_1\ \mathsf{call?}\ v_2} a.\pi \cdot \langle v_1, v_2\rangle}$$

$$\text{[send return]}\quad \frac{\neg\mathbf{loc}_a\, v_1 \qquad \mathbf{loc}_a\, v_2 \qquad \mathsf{return}\ \ v_2}{a.\pi \cdot \langle v_1, v_2\rangle \xrightarrow{v_2\ \mathsf{ret!}\ v_1} a.\pi}$$

$$\text{[receive return]}\quad \frac{v_1 \to^{T} v_2 \qquad \mathbf{loc}_a\, v_1 \qquad \neg\mathbf{loc}_a\, v_3 \qquad \alpha(v_3) = \alpha(v_4)}{a.\pi \cdot \langle v, v_1\rangle \cdot \langle v_1, v_3\rangle \xrightarrow{v_4\ \mathsf{ret?}\ v_1} a.\pi \cdot \langle v, v_2\rangle}$$

Local trace AirFrance applet

`AirFrance:`

$$
\begin{aligned}
\epsilon \quad & \xrightarrow{\texttt{P.debit.Med}\ \text{call?}\ \texttt{AF.logFull}} \\
\langle \texttt{P.debit.Med}, \texttt{AF.logFull} \rangle \quad & \longrightarrow \\
\langle \texttt{P.debit.Med}, \texttt{AF.logFull.Med1} \rangle \quad & \xrightarrow{\texttt{AF.logFull.Med1}\ \text{call!}\ \texttt{P.getTrs}} \\
\langle \texttt{P.debit.Med}, \texttt{AF.logFull.Med1} \rangle \cdot \\
\langle \texttt{AF.logFull.Med1}, \texttt{P.getTrs} \rangle \quad & \xrightarrow{\texttt{P.getTrs.Ret}\ \text{ret?}\ \texttt{AF.logFull.Med1}} \\
\langle \texttt{P.debit.Med}, \texttt{AF.logFull.Med2} \rangle \quad & \xrightarrow{\texttt{AF.logFull.Med2}\ \text{call!}\ \texttt{RaC.getBalance}} \\
\ldots
\end{aligned}
$$

Transition rules for composite states

$$[\text{tau}] \quad \frac{\mathcal{A}_1 \xrightarrow{\tau} \mathcal{A}'_1}{\mathcal{A}_1 \mid \mathcal{A}_2 \xrightarrow{\tau} \mathcal{A}'_1 \mid \mathcal{A}_2}$$

$$[\text{synchro}] \quad \frac{\mathcal{A}_1 \xrightarrow{v_1\ l?\ v_2} \mathcal{A}'_1 \qquad \mathcal{A}_2 \xrightarrow{v_1\ l!\ v_2} \mathcal{A}'_2}{\mathcal{A}_1 \mid \mathcal{A}_2 \xrightarrow{v_1\ l\ v_2} \mathcal{A}'_1 \mid \mathcal{A}'_2}\ \ell \in \mathsf{call}, \mathsf{ret}$$

$$[\text{propagation}] \quad \frac{\mathcal{A}_1 \xrightarrow{v_1\ l?!\ v_2} \mathcal{A}'_1 \qquad \neg\mathbf{loc}_{\mathcal{A}_2}\, v_1 \qquad \neg\mathbf{loc}_{\mathcal{A}_2}\, v_2}{\mathcal{A}_1 \mid \mathcal{A}_2 \xrightarrow{v_1\ l?!\ v_2} \mathcal{A}'_1 \mid \mathcal{A}_2}$$

Fragment of the global trace

$$\mathtt{P}.\epsilon \mid \mathtt{AF}.\epsilon \mid \mathtt{RaC}.\epsilon \xrightarrow{\mathtt{User.P}\ \mathsf{call?}\ \mathtt{P.debit}}$$

$$\mathtt{P}.\langle \mathtt{User.P}, \mathtt{P.debit}\rangle \mid \mathtt{AF}.\epsilon \mid \mathtt{RaC}.\epsilon \longrightarrow$$

$$\mathtt{P}.\langle \mathtt{User.P}, \mathtt{P.debit.Med}\rangle \mid \mathtt{AF}.\epsilon \mid \mathtt{RaC}.\epsilon \xrightarrow{\mathtt{P.debit.Med}\ \mathsf{call}\ \mathtt{AF.logFull}}$$

$$\begin{array}{r} \mathtt{P}.\langle \mathtt{User.P}, \mathtt{P.debit.Med}\rangle \cdot \langle \mathtt{P.debit.Med}, \mathtt{AF.logFull}\rangle \mid \\ \mathtt{AF}.\langle \mathtt{P.debit.Med}, \mathtt{AF.logFull}\rangle \mid \\ \mathtt{RaC}.\epsilon \end{array} \longrightarrow$$

$$\ldots \xrightarrow{\mathtt{P.debit.Ret}\ \mathsf{ret!}\ \mathtt{User.P}}$$

$$\mathtt{P}.\epsilon \mid \mathtt{AF}.\epsilon \mid \mathtt{RaC}.\epsilon$$

The properties

Behavioural applet properties

- m_1 never triggers m_2 (confidentiality, integrity)
- m_1 never triggers m_2 when it triggers m_3 (exclusion)
- m_1 only triggers m_2 through m_3 (segregation of duty)
- m_1 never triggers m_2 before m_3 (authentication)

Specifying the electronic purse system

Invocation of `logFull` in `AirFrance` applet by `Purse`
should not trigger
a call from `RentACar` to `getTrs` (to ask the transactions) in `Purse`:

$$\textsc{Spec} \triangleq \textsc{Within}\ \texttt{AirFrance.logFull}\ \textsc{CannotCall}\ \texttt{RentACar}\ \texttt{Purse.getTrs}$$

Formulae

$$\phi ::= \quad \sigma \;\mid\; \mathsf{active} \;\mid\; \neg\phi \;\mid\; \phi \wedge \phi \;\mid\; \forall x.\phi \;\mid\; [\alpha]\,\phi \;\mid\; X \;\mid\; \nu X.\phi$$

with standard semantics, and:

$$\begin{aligned} \|\mathsf{active}\|_{\rho}^{\mathcal{M}} &\triangleq \{s \in \mathcal{S}_{\mathcal{M}} \mid s \text{ is active}\} \\ \|\sigma\|_{\rho}^{\mathcal{M}} &\triangleq \text{if } [\![\sigma]\!]_{\rho}^{\mathcal{M}} \text{ then } \mathcal{S}_{\mathcal{M}} \text{ else } \emptyset \end{aligned}$$

Atomic formulae

$$\sigma ::= t = t \mid \mathsf{return}\ v \mid \mathsf{local}_{\mathcal{A}}\, v$$

with semantics

$$\begin{aligned}
[\![t_1 = t_2]\!]^{\mathcal{M}}_{\rho} &\triangleq t_1\rho = t_2\rho \\
[\![\mathsf{return}\ v]\!]^{\mathcal{M}}_{\rho} &\triangleq \textbf{ret}(v\rho) \\
[\![\mathsf{local}_{\mathcal{A}}\, v]\!]^{\mathcal{M}}_{\rho} &\triangleq \textbf{app}(v\rho) \text{ occurs in } \mathcal{A}\rho
\end{aligned}$$

Specification patterns

$$
\begin{aligned}
&\textsc{Always}\ \phi \ \triangleq \\
&\quad \nu X.\ \phi \ \wedge\ [\tau]\, X \\
&\qquad\qquad \wedge\ \ \forall v_1.\ \forall v_2.\quad [v_1\ \mathsf{call}\ v_2]\, X \\
&\qquad\qquad\qquad\qquad\qquad \wedge\quad [v_1\ \mathsf{call?}\ v_2]\, X \\
&\qquad\qquad\qquad\qquad\qquad \wedge\quad [v_1\ \mathsf{call!}\ v_2]\, X \\
&\qquad\qquad\qquad\qquad\qquad \wedge\quad [v_1\ \mathsf{ret}\ v_2]\, X \\
&\qquad\qquad\qquad\qquad\qquad \wedge\quad [v_1\ \mathsf{ret?}\ v_2]\, X \\
&\qquad\qquad\qquad\qquad\qquad \wedge\quad [v_1\ \mathsf{ret!}\ v_2]\, X
\end{aligned}
$$

$$
\begin{aligned}
&\textsc{Within}\ v\ \ \phi \ \triangleq \\
&\quad \forall v_1.\quad [v_1\ \mathsf{call}\ v]\ \textsc{Always}_{-\mathsf{ret}\ v_1}\phi \\
&\qquad\ \wedge\quad [v_1\ \mathsf{call?}\ v]\ \textsc{Always}_{-\mathsf{ret!}\ v_1}\phi
\end{aligned}
$$

$\textsc{Always}_{-\lambda\ v}\ \phi$ is $\textsc{Always}\ \phi$ with $[v_1 \lambda\ v_2]\, X$ replaced by $[v_1 \lambda\ v_2]\,((v_2 = v) \vee X)$.

Specifying the individual applets

$$
\begin{array}{lcl}
\textsc{Spec}_{\mathsf{P}} & \triangleq & \mathsf{local}_{\texttt{Purse}}\,\texttt{Purse.getTrs} \wedge \\
& & \textsc{Always}\ \ (\textsc{Within}\ \texttt{Purse.getTrs} \\
& & \qquad\qquad\qquad (\textsc{CallsExtOnly}\ \emptyset)) \\[1em]
\textsc{Spec}_{\mathsf{AF}} & \triangleq & \mathsf{local}_{\texttt{AirFrance}}\,\texttt{AirFrance.logFull} \wedge \\
& & \textsc{Always}\ \ (\textsc{Within}\ \texttt{AirFrance.logFull} \\
& & \qquad\qquad\qquad (\textsc{CallsExtOnly}\ \texttt{Purse.getTrs}, \\
& & \qquad\qquad\qquad\qquad \texttt{RentACar.getBalance})) \\[1em]
\textsc{Spec}_{\mathsf{RaC}} & \triangleq & \mathsf{local}_{\texttt{RentACar}}\,\texttt{RentACar.getBalance} \wedge \\
& & \textsc{Always}\ \ (\textsc{Within}\ \texttt{RentACar.getBalance} \\
& & \qquad\qquad\qquad (\textsc{CallsExtOnly}\ \emptyset))
\end{array}
$$

Correctness of the decomposition

- Parameterize specifications:
  - replace concrete applet names by applet variables
  - replace concrete method names by program point variables
- Prove:

$$
\begin{array}{r}
a_P.\pi_P : \textsc{Spec}_{\mathsf{P}}, \\
a_{AF}.\pi_{AF} : \textsc{Spec}_{\mathsf{AF}}, \\
a_{RaC}.\pi_{RaC} : \textsc{Spec}_{\mathsf{RaC}}
\end{array}
\quad \vdash \; a_P.\pi_P \,\big|\, a_{AF}.\pi_{AF} \,\big|\, a_{RaC}.\pi_{RaC} : \textsc{Spec}
$$

The proof system

Proof system

- Proof rules for different kind of assertions
  - atomic formula assertion σ
  - satisfaction assertion $\mathcal{A} : \phi$
  - transition assertion $\mathcal{A}_1 \xrightarrow{\alpha} \mathcal{A}_2$
  - transfer-edge assertion $v \rightarrow^T v'$
  - call-edge assertion $v \rightarrow^C v'$
- Fixed points handled with discharge condition (Dam/Gurov)
- Modal operators introduce transition assertions, transition rules introduce edge assertions

Sequents

$$\gamma_1, \ldots, \gamma_n \vdash \delta_1, \ldots, \delta_n$$

has intuitive meaning

$$\gamma_1 \wedge \ldots \wedge \gamma_n \Rightarrow \delta_1 \vee \ldots \vee \delta_n$$

Proof system – Active rules

$$(\mathsf{ActL})\quad \frac{\Gamma,\ \Pi = \pi \cdot \langle v_1, v_2\rangle\ ,\ \mathsf{local}_a\, v_2\ \vdash\ \Delta}{\Gamma,\ a.\Pi : \mathsf{active}\ \vdash\ \Delta}\ \text{fresh}\ \pi, v_1, v_2$$

$$(\mathsf{ActR})\quad \frac{\Gamma\ \vdash\ \Pi = \Pi' \cdot \langle v_1, v_2\rangle\ ,\ \Delta \qquad \Gamma\ \vdash\ \mathsf{local}_a\, v_2,\ \Delta}{\Gamma\ \vdash\ a.\Pi : \mathsf{active},\ \Delta}$$

Proof system – Box rules

$$(\mathsf{BoxL})\quad \frac{\Gamma \vdash \mathcal{A} \xrightarrow{\alpha} \mathcal{A}',\ \Delta \qquad \Gamma,\ \mathcal{A}' : \phi \vdash \Delta}{\Gamma,\ \mathcal{A} : [\alpha]\,\phi \vdash \Delta}$$

$$(\mathsf{BoxR})\quad \frac{\Gamma, \mathcal{A} \xrightarrow{\alpha} \mathcal{X} \vdash \mathcal{X} : \phi,\ \Delta}{\Gamma \vdash \mathcal{A} : [\alpha]\,\phi,\ \Delta}\ \text{fresh } \mathcal{X}$$

Proof system – Transition rules

$$
(\mathsf{SendCallR})\quad \frac{\begin{array}{l} \Gamma \vdash v_1 \to^{C} v_2,\ \Delta \\ \Gamma \vdash \mathsf{local}_a\, v_1,\ \Delta \\ \Gamma,\ \mathsf{local}_a\, v_2 \vdash \Delta \\ \Gamma \vdash \Pi = \pi \cdot \langle v, v_1 \rangle\ ,\ \Delta \end{array}}{\Gamma \vdash a.\Pi \xrightarrow{v_1\ \mathsf{call!}\ v_2} a.(\Pi \cdot \langle v_1, v_2 \rangle),\ \Delta}
$$

$$
(\mathsf{SendCallL})\quad \frac{\begin{array}{l} \Gamma[a.(\Pi \cdot \langle v_1, v_2 \rangle)/\mathcal{X}], \\ \qquad \Pi = \pi \cdot \langle v, v_1 \rangle\ , \\ \quad v_1 \to^{C} v_2,\ \mathsf{local}_a\, v_1 \quad \vdash \mathsf{local}_a\, v_2,\ \Delta[a.(\Pi \cdot \langle v_1, v_2 \rangle)/\mathcal{X}] \end{array}}{\Gamma,\ a.\Pi \xrightarrow{v_1\ \mathsf{call!}\ v_2} \mathcal{X} \vdash \Delta}\ \mathsf{fresh}\ v, \pi
$$

Proof system - Composite transition rules

$$
(\mathsf{ComPerfL})\quad
\frac{
\begin{array}{c}
\Gamma[(\mathcal{Y}\,|\,\mathcal{A}_2)/\mathcal{X}],\ \mathcal{A}_1 \xrightarrow{v_1 l v_2} \mathcal{Y},\ \mathsf{wf}(\mathcal{A}_1\,|\,\mathcal{A}_2),\ \mathsf{wf}(\mathcal{Y}\,|\,\mathcal{A}_2)\ \vdash\ \mathsf{local}_{\mathcal{A}_2}\, v_1,\ \mathsf{local}_{\mathcal{A}_2}\, v_2,\ \Delta[(\mathcal{Y}\,|\,\mathcal{A}_2)/\mathcal{X}] \\
\Gamma[(\mathcal{A}_1\,|\,\mathcal{Y})/\mathcal{X}],\ \mathcal{A}_2 \xrightarrow{v_1 l v_2} \mathcal{Y},\ \mathsf{wf}(\mathcal{A}_1\,|\,\mathcal{A}_2),\ \mathsf{wf}(\mathcal{A}_1\,|\,\mathcal{Y})\ \vdash\ \mathsf{local}_{\mathcal{A}_1}\, v_1,\ \mathsf{local}_{\mathcal{A}_1}\, v_2,\ \Delta[(\mathcal{A}_1\,|\,\mathcal{Y})/\mathcal{X}] \\
\Gamma[(\mathcal{X}_1\,|\,\mathcal{X}_2)/\mathcal{X}],\ \mathcal{A}_1 \xrightarrow{v_1 l! v_2} \mathcal{X}_1,\ \mathcal{A}_2 \xrightarrow{v_1 l? v_2} \mathcal{X}_2,\ \mathsf{wf}(\mathcal{A}_1\,|\,\mathcal{A}_2),\ \mathsf{wf}(\mathcal{X}_1\,|\,\mathcal{X}_2)\ \vdash\ \Delta[(\mathcal{X}_1\,|\,\mathcal{X}_2)/\mathcal{X}] \\
\Gamma[(\mathcal{X}_1\,|\,\mathcal{X}_2)/\mathcal{X}],\ \mathcal{A}_1 \xrightarrow{v_1 l? v_2} \mathcal{X}_1,\ \mathcal{A}_2 \xrightarrow{v_1 l! v_2} \mathcal{X}_2,\ \mathsf{wf}(\mathcal{A}_1\,|\,\mathcal{A}_2),\ \mathsf{wf}(\mathcal{X}_1\,|\,\mathcal{X}_2)\ \vdash\ \Delta[(\mathcal{X}_1\,|\,\mathcal{X}_2)/\mathcal{X}]
\end{array}
}{
\Gamma,\ \mathcal{A}_1\,|\,\mathcal{A}_2 \xrightarrow{v_1 l v_2} \mathcal{X}\ \vdash\ \Delta
}\ \mathsf{fresh}\ \mathcal{X}_1, \mathcal{X}_2, \mathcal{Y}
$$

Correctness

- Soundness:
  - Formalisation of program model and operational semantics
  - Formalisation of proof system
  - Proof rules sound *w.r.t.* the underlying semantics
  - All in PVS
- Completeness: future work

Example verification

The method

Prove

$$\begin{array}{r} a_P.\pi_P : \textsc{Spec}_{\mathsf{P}}, \\ a_{AF}.\pi_{AF} : \textsc{Spec}_{\mathsf{AF}}, \\ a_{RaC}.\pi_{RaC} : \textsc{Spec}_{\mathsf{RaC}} \end{array} \quad \vdash \; a_P.\pi_P \,|\, a_{AF}.\pi_{AF} \,|\, a_{RaC}.\pi_{RaC} : \textsc{Spec}$$

by

- symbolic execution to compute symbolic next states
- identification of impossible states
- loop detection to close off branches in proof

Some abbreviations

$$\textsc{Spec} \triangleq \textsc{Within}\ \texttt{AirFrance.logFull}\ \textsc{Spec}'$$

$$\textsc{Spec}' \triangleq \textsc{CannotCall}\ \texttt{RentACar}\ \texttt{Purse.getTrs}$$

$$\textsc{Spec}_{\mathsf{P}} \triangleq \mathsf{local}_{\texttt{Purse}}\,\texttt{Purse.getTrs}\ \wedge\ \textsc{Spec}'_{\mathsf{P}}$$

$$\textsc{Spec}'_{\mathsf{P}} \triangleq \textsc{Always}\ (\textsc{Within}\ \texttt{Purse.getTrs}\ (\textsc{CallsExtOnly}\ \emptyset))$$

Some abbreviations - 2

$$\textsc{Spec}_{\mathsf{AF}} \triangleq \mathsf{local}_{\mathtt{AirFrance}}\, \mathtt{AirFrance.logFull} \wedge \textsc{Spec}'_{\mathsf{AF}}$$

$$\textsc{Spec}'_{\mathsf{AF}} \triangleq \textsc{Always}\ (\textsc{Within}\ \mathtt{AirFrance.logFull}\ \textsc{Spec}''_{\mathsf{AF}})$$

$$\textsc{Spec}''_{\mathsf{AF}} \triangleq \textsc{CallsExtOnly}\ \mathtt{Purse.getTrs},\ \mathtt{RentACar.getBalance}$$

$$\textsc{Spec}_{\mathsf{RaC}} \triangleq \mathsf{local}_{\mathtt{RentACar}}\, \mathtt{RentACar.getBalance} \wedge \textsc{Spec}'_{\mathsf{RaC}}$$

$$\textsc{Spec}'_{\mathsf{RaC}} \triangleq \textsc{Always}\ (\textsc{Within}\ \mathtt{RentACar.getBalance}(\textsc{CallsExtOnly}\ \emptyset))$$

Step 1

- Unfold pattern
- Apply logical rules to outermost logical connectives of $\textsc{Spec}$
- Two subgoals: focus on first

$$\begin{array}{l}\mathsf{local}_{a_P}\, v_{GT},\ \mathsf{local}_{a_{AF}}\, v_{LF},\ \mathsf{local}_{a_{RaC}}\, v_{GB},\\ \quad a_P.\pi_P : \textsc{Spec}'_{\mathsf{P}},\ a_{AF}.\pi_{AF} : \textsc{Spec}'_{\mathsf{AF}},\ a_{RaC}.\pi_{RaC} : \textsc{Spec}'_{\mathsf{RaC}}\\ \quad \vdash\ a_P.\pi_P \,|\, a_{AF}.\pi_{AF} \,|\, a_{RaC}.\pi_{RaC} : [v_1\ \mathsf{call}\ v_{LF}]\ \ \textsc{Always}_{-\mathsf{ret}\ v_1}\textsc{Spec}'\end{array}$$

Apply rule BoxR and left transition rules

- Result: nine sequents – nine different ways in which a perfect call action can come about

- Four subgoals discarded by Id: a_{AF} is not involved in communication

- Two subgoals assume a_{AF} sends a call to an external v_{LF}, discarded by SendCallL and Id.

- Three subgoals consider the different ways of producing a perfect call to vertex v_{LF}

v_{LF} is invoked by a local call

$$
\begin{array}{r}
a_P.\pi_P : \textsc{Spec}'_{\mathsf{P}}, \\
a_{AF}.\pi_{AF} : \textsc{Spec}'_{\mathsf{AF}}, \\
a_{RaC}.\pi_{RaC} : \textsc{Spec}'_{\mathsf{RaC}}, \\
a_{AF}.\pi_{AF} \xrightarrow{v_1 \text{ call } v_{LF}} \mathcal{X}, \\
\mathsf{wf}(a_P.\pi_P \,|\, a_{AF}.\pi_{AF} \,|\, a_{RaC}.\pi_{RaC}), \\
\mathsf{wf}(a_P.\pi_P \,|\, \mathcal{X} \,|\, a_{RaC}.\pi_{RaC}), \\
\mathsf{local}_{a_P}\, v_{GT}, \\
\mathsf{local}_{a_{AF}}\, v_{LF}, \\
\mathsf{local}_{a_{RaC}}\, v_{GB} \quad \vdash \\
a_P.\pi_P \,|\, \mathcal{X} \,|\, a_{RaC}.\pi_{RaC} : \textsc{Always}_{-\mathsf{ret}\ v_1} \textsc{Spec}'
\end{array}
$$

Deriving assumptions about $\mathcal{X}$

- Apply l.h.s. rules
- Apply BoxL to box-formula labeled v_1 call v_{LF}.
- First subgoal: show $a_{AF}.\pi_{AF} \xrightarrow{v_1 \text{ call } v_{LF}} \mathcal{X}$ by Id
- Second subgoal:

$$
\begin{array}{l}
\mathsf{local}_{a_P}\, v_{GT},\ \mathsf{local}_{a_{AF}}\, v_{LF},\ \mathsf{local}_{a_{RaC}}\, v_{GB},\\
a_P.\pi_P : \textsc{Spec}'_{\mathsf{P}},\ \ \mathcal{X} : \textsc{Always}_{-\mathsf{ret}\ v_1}\textsc{Spec}''_{\mathsf{AF}},\ \ a_{RaC}.\pi_{RaC} : \textsc{Spec}'_{\mathsf{RaC}},\\
a_{AF}.\pi_{AF} \xrightarrow{v_1 \text{ call } v_{LF}} \mathcal{X},\\
\mathsf{wf}(a_P.\pi_P \,|\, a_{AF}.\pi_{AF} \,|\, a_{RaC}.\pi_{RaC}),\ \ \mathsf{wf}(a_P.\pi_P \,|\, \mathcal{X} \,|\, a_{RaC}.\pi_{RaC})\\
\vdash\ a_P.\pi_P \,|\, \mathcal{X} \,|\, a_{RaC}.\pi_{RaC} : \textsc{Always}_{-\mathsf{ret}\ v_1}\textsc{Spec}'
\end{array}
$$

Eliminate transition assertion by LocCallL

$$\begin{aligned}
\mathsf{local}_{a_P}\, v_{GT},\\
\mathsf{local}_{a_{AF}}\, v_{LF},\\
\mathsf{local}_{a_{AF}}\, v_1,\\
\mathsf{local}_{a_{RaC}}\, v_{GB},\\
a_P.\pi_P : \textsc{Spec}'_{\mathsf{P}},\\
a_{AF}.\pi_{AF} \cdot \langle v_1, v_{LF}\rangle : \textsc{Always}_{-\mathsf{ret}\ v_1}\textsc{Spec}''_{\mathsf{AF}},\\
a_{RaC}.\pi_{RaC} : \textsc{Spec}'_{\mathsf{RaC}},\\
\pi_{AF} = \pi \cdot \langle v, v_1\rangle\ ,\\
v_1 \to^{C} v_{LF},\\
\mathsf{wf}(a_P.\pi_P \,|\, a_{AF}.\pi_{AF} \,|\, a_{RaC}.\pi_{RaC}),\\
\mathsf{wf}(a_P.\pi_P \,|\, a_{AF}.\pi_{AF} \cdot \langle v_1, v_{LF}\rangle \,|\, a_{RaC}.\pi_{RaC}) \quad \vdash
\end{aligned}$$

$$a_P.\pi_P \,|\, a_{AF}.\pi_{AF} \cdot \langle v_1, v_{LF}\rangle \,|\, a_{RaC}.\pi_{RaC} : \textsc{Always}_{-\mathsf{ret}\ v_1}\textsc{Spec}'$$

Analysis of symbolic next-state

- Computed next-state:

$$a_P.\pi_P \,|\, a_{AF}.\pi_{AF} \cdot \langle v_1, v_{LF}\rangle \,|\, a_{RaC}.\pi_{RaC}$$

- Show in this symbolic next-state formula $\textsc{Always}_{-\mathsf{ret}\ v_1}\textsc{Spec}'$ holds:
  - In this state $\textsc{Spec}'$ is true: trivial, because of wellformedness.
  - In all possible next states within the call (not labeled v ret v_1) $\textsc{Always}_{-\mathsf{ret}\ v_1}\textsc{Spec}'$ holds.
    External call: only two possible destinations of call v_{GT} or v_{GB}
    Continue with these states
    Other cases: loop and discharge

Ongoing and future work

- Maximal graph construction - completeness for a fragment of the logic
- Concurrency
- Keep notion of method in program model
- Extension of Erlang Verification Tool
- Verification of individual applets
- Case studies