Compositional Verification of Sequential Programs with Procedures

Dilian Gurov
Theory Group, KTH CSC

Joint work with:

Marieke Huisman, Christoph Sprenger

Talk at KTH CSC
March 13, 2006
Overview

1. Motivation

2. Framework for Compositional Verification
   (a) Models, Simulation and Logic
   (b) Maximal Models
   (c) Program Model
   (d) Compositional Verification Method

3. Tool Set and Case Study

4. Interface Abstraction

5. Conclusions
1. Motivation: Smart Cards and Security

Smart cards
- store privacy-sensitive data
- require strong guarantees of security: formal verification

Multiple interacting applets (e.g. JavaCard applets)
- communication via method invocation over shared interfaces
- example: electronic purse applet and several loyalties

Post-issuance loading
- ability to load new applets after the card has been issued to the user
- requires compositional verification
Compositional Verification

Compositional Verification Principle

\[
\frac{\models A : \psi \quad X : \psi \models X \otimes B : \phi}{\models A \otimes B : \phi}
\]

premises: local property of \( A \) and correctness of decomposition

Scenarios for secure post-issuance loading

1. card issuer specifies \( \phi \) and \( \psi \) and checks property decomposition;
   pre-load check of \( \models A : \psi \)

2. card issuer provides only \( \phi \), applet provider specifies \( \psi \);
   pre-load check of \( \models A : \psi \) and property decomposition
Maximal Models

In certain setups: simulation preorder $\preceq$ on components

1. property preserving

2. preserved by composition $\otimes$

3. for any formula $\psi$, the set of models for $\psi$ has a maximal element $Max(\psi)$ wrt. the preorder: maximal model

Maximal Model Principle [Grumberg & Long ’94]

\[
\models Max(\psi) \otimes B : \phi \\
X : \psi \models X \otimes B : \phi
\]
Compositional Verification Principle

\[ \vdash A : \psi \quad \vdash Max(\psi) \otimes B : \phi \]
\[ \vdash A \otimes B : \phi \]

Derived Rule

\[ \vdash A : \psi \quad \vdash B : \theta \quad \vdash Max(\psi) \otimes Max(\theta) : \phi \]
\[ \vdash A \otimes B : \phi \]
2. The Framework: Models, Simulation and Logic

Components Applets

- control–flow
- structure, induces behaviour
- unified treatment: model

Model $\mathcal{M} = (S, L, A, \rightarrow, \lambda)$

- Labelled transition system $\mathcal{T} = (S, L, \rightarrow)$
- set of atomic propositions $A$
- valuation $\lambda : S \rightarrow \mathcal{P}(A)$
Simulation Preorder \( \leq \)

- standard, on states

Simulation Logic modal logic with:

- box modalities (only)

- greatest fixed–point recursion (only)

\[
\phi ::= p \mid \neg p \mid X \mid \phi_1 \land \phi_2 \mid \phi_1 \lor \phi_2 \mid [a] \phi \mid \nu X.\phi
\]

Maximal Models \( \text{Max}(\psi) \)

- conditions 1-3 are satisfied

- exponential construction, lazy
Applet Structure

Applet $A$

- control–flow graph represented as model
- applet composition $\oplus$
- structural simulation and logic (safety properties)

Maximal Model for property $\psi$ is not necessarily a legal applet structure!

- interface $I = (I^+, I^-)$ of provided and required methods
- formula $\phi_I$ axiomatizing applets with interface $I$

Maximal Applet $Max_I(\psi)$

- is the maximal model $Max(\phi_I \land \psi)$
Method Graph: Example

Maximal Applet for interface $I = (\{m_1, m_2\}, \{m_1, m_3\})$
Applet Behaviour

• Applet structure \( A \) induces applet behaviour \( b(A) \)
  - configurations: pairs \( (v, \sigma) \) of control point and call stack
  - labels: \( \varepsilon, m_1 \text{ call } m_2, m_2 \text{ ret } m_1 \)
  - transitions: standard, pushdown automaton
  - valuation: as of the control point

• Behavioural simulation and logic
  - applet interaction properties (safety properties)

• Maximal models
  - applet behaviour is \textit{not} axiomatizable within the logic!
  - structural simulation implies behavioural simulation
Applet Behaviour: Example
Operational Semantics

(call)

\[
\frac{m_1, m_2 \in I^+ \quad v_1 \xrightarrow{m_2} v_1' \quad v_2 \models m_2 \land e}{(v_1, \sigma) \xrightarrow{m_1 \text{call } m_2} (v_2, v_1' \cdot \sigma)}
\]

(return)

\[
\frac{m_1, m_2 \in I^+ \quad v_2 \models m_2 \land r \quad v_1 \models m_1}{(v_2, v_1 \cdot \sigma) \xrightarrow{m_2 \text{ ret } m_1} (v_1, \sigma)}
\]

(transfer)

\[
\frac{m \in I^+ \quad v \xrightarrow{m} v'}{(v, \sigma) \xrightarrow{\varepsilon} (v', \sigma)}
\]
A Compositional Verification Method

Compositional Verification Principle

\[
\begin{align*}
\mathcal{A} & \models_s \sigma \\
\text{Max}_{I_{\mathcal{A}}} (\sigma) \cup \mathcal{B} & \models_b \psi \\
\mathcal{A} \cup \mathcal{B} & \models_b \psi
\end{align*}
\]

\[\mathcal{A} : I_{\mathcal{A}}\]

1. a) Specify global property \( \psi \) as a behavioural property
   b) For applet \( \mathcal{A} \), specify local property \( \sigma \) as a structural property

2. Verify the correctness of the property decomposition:
   a) compute maximal applet \( \text{Max}_{I_{\mathcal{A}}} (\sigma) \)
   b) model check \( \text{Max}_{I_{\mathcal{A}}} (\sigma) \cup \mathcal{B} \models_b \psi \)

3. When implementation of \( \mathcal{A} \) available, verify \( \mathcal{A} \models_s \sigma \)
3. Tool Set and Case Study

[Diagram showing tool set and case study process]

- Structural specification
- Public Interface
- Implementation
- Behavioural specification

Applet Analyser → Concrete Applet Graph → Applet Graphs → Public Interface

Maximal model constructor → Inliner

CWB → YES/NO

CCS process

Model generator

PDA

PDA MC → YES/NO

KTH CSC, March 13, 2006
Gemplus Electronic Purse Case Study

Purpose

- “benchmark” to evaluate formal methods for JavaCard applications
- provided by Gemplus in context of Verificard project

Structure

- one electronic purse applet and several loyalty applets
- communicate by method invocation via shared interfaces

Properties

- general: absence of illicit control flow in applet interactions
- in particular: no access to service which has not been paid for
Purse Case Study – Context

Situation

- purse maintains a log table of credit and debit transactions
- loyalties use log information to determine loyalty points
- purse provides logFull service to warn loyalties before overwriting log

Bad scenario featuring loyalties AirFrance and RentACar

- loyalty AirFrance subscribed to logFull service (and has paid for it!)
- AirFrance.logFull calls partner loyalty RentACar (e.g. to ask for balance)
- RentACar, not subscribed to logFull, deduces log might be full, asks purse for log table contents, thus circumventing payment for logFull
Purse Case Study – Specification

Global Behavioural Property

$(\phi)$ a call to \texttt{Loyalty.logFull} does not trigger calls to any other loyalties
(neither directly nor indirectly via purse)

Local Structural Properties

$(\psi_L)$ from any entry point of \texttt{Loyalty.logFull}, no external calls are reachable other than calls to \texttt{Purse.getTransaction}

$(\psi_P)$ from any entry point of \texttt{Purse.getTransaction}, no external calls are reachable
Purse Case Study – Verification Tasks

Property Decomposition

- **extract** loyalty and purse interfaces: $I_L$ and $I_P$ (SOOT-based tool)
- **generate maximal applets**: $\theta_{I_L}(\psi_L)$ and $\theta_{I_P}(\psi_P)$ (own tool)
- **PDS model checking**: $\theta_{I_L}(\psi_L) \uplus \theta_{I_P}(\psi_P) \models b \phi$ (Alfred or Moped)

Local Properties

- **extract** applet control graphs: $L$ and $P$ (SOOT-based tool)
- **finite-state model checking**: $L \models_s \psi_L$ and $P \models_s \psi_P$ (CWB)
Main Shortcomings

1. Requires knowledge of the complete interface, but in a truly compositional setting we can only assume knowledge of the names of the public methods.

2. Interfaces are significantly larger than public ones, which is critical for the applicability of the (exponential) maximal model construction.
4. Interface Abstraction

Public and Private Methods  \( M \subseteq I_{\mathcal{A}}^+ \) a set of public methods

Transformation transforms applet \( \mathcal{A} \) with interface \((I^+, I^-)\) into a simulating applet \( \alpha_M(\mathcal{A}) \) with interface \((M, I^- \cup (I^+ \setminus M))\)

Modified CVP

\[
\begin{align*}
\alpha_M(\mathcal{A}) & \models_s \sigma \\
\text{Max}_{I_{\alpha_M(\mathcal{A})}}(\sigma) \cup \mathcal{B} & \models_b \psi \\
\mathcal{A} \uplus \mathcal{B} & \models_{b_{M \cup I_{\mathcal{B}}^+}} \psi
\end{align*}
\]

- simulation: w.r.t. public behaviour, or interface behaviour
- transformation: inlining of private methods
Interface Behaviour

An abstraction on applet behaviour \( \text{wrt. } M \subseteq I_A^+ \)

- keep configurations unchanged

- relabel configurations
  - current control is in the top–most public method of \( v \cdot \sigma \)

- relabel transitions accordingly
  - configuration–dependent relabelling

- denoted \( b^M(\mathcal{A}) \)
The Inlining Transformation

Inlining  replace method call by method body

- need to: guarantee termination, prove simulation

Transformation  For each (public) method \( m \in M \)

- execute \( m \) so that:
  - label local calls and returns by \( \varepsilon \)
  - treat calls to public methods as local transfer, but keep label
  - replace recursion by iteration

- result denoted \( \alpha_M(A) \)
• introduces more interface behaviour!
Simulation Results

**Theorem** Let \( A : I \) and \( M \subseteq I^+ \).

Then \( b^M(A) \leq b(\alpha_M(A)) = b^M(\alpha_M(A)) \).

**Last-call recursion** call edges are followed by transfer edges only

**Theorem** Let \( A : I \) be last-call recursive, and \( M \subseteq I^+ \).

Then \( b^M(A) \equiv_w b(\alpha_M(A)) = b^M(\alpha_M(A)) \).
Compositional Verification Method

Modified CVP

\[
\alpha_M(A) \models_s \sigma \quad \text{Max}_{I\alpha_M(A)}(\sigma) \cup B \models_b \psi \\
A \cup B \models_{M \cup I^+_B} \psi
\]

1. a) Specify global property \( \psi \) as an interface behavioural property
   b) For applet \( A \), specify local property \( \sigma \) as a structural property of \( \alpha_M(A) \)

2. Verify the correctness of the property decomposition:
   a) compute maximal applet \( \text{Max}_{I\alpha_M(A)}(\sigma) \)
   b) model check \( \text{Max}_{I\alpha_M(A)}(\sigma) \cup B \models_b \psi \)

3. When implementation of \( A \) available:
   a) compute \( \alpha_M(A) \)
   b) verify \( \alpha_M(A) \models_s \sigma \)

KTH CSC, March 13, 2006
Practical Impact of Inlining

- Knowledge of public interfaces suffices for applying the verification method.
- Reconsider the case study from [Huisman, Gurov, Sprenger, Chugunov: FASE’04]

<table>
<thead>
<tr>
<th></th>
<th>$\max(\sigma_L)$ in [HGSC’04]</th>
<th>$\max(\sigma_P)$ in [HGSC’04]</th>
</tr>
</thead>
<tbody>
<tr>
<td>#nodes</td>
<td>8</td>
<td>474</td>
</tr>
<tr>
<td>#edges</td>
<td>120</td>
<td>277 700</td>
</tr>
<tr>
<td>constr. time</td>
<td>0.05 s.</td>
<td>25 min.</td>
</tr>
</tbody>
</table>

- Some natural structural properties are only expressible as properties of the inlined applet.
5. Conclusion

We presented

- An algorithmic method:
  - for compositional verification
  - of control–flow based safety properties
  - of structure and behaviour
  - of sequential programs with procedures

- Refinement:
  - discriminating between public and private methods
  - interface abstraction: in–lining of private methods
Current work

- refining the program model: exceptions, multi-threading
- maximal models for behavioural properties

Publications

- Verification Framework: MemoCode’04
- Case Study: FASE’04 (win)
- Interface Abstraction: SEFM’05
- Journal Version: JIC
New Slide

- blah