

# 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]

$$\frac{\models Max(\psi) \otimes B : \phi}{X : \psi \models X \otimes B : \phi}$$

## Compositional Verification Principle

$$\frac{\models A : \psi \quad \models \text{Max}(\psi) \otimes B : \phi}{\models A \otimes B : \phi}$$

## Derived Rule

$$\frac{\models A : \psi \quad \models B : \theta \quad \models \text{Max}(\psi) \otimes \text{Max}(\theta) : \phi}{\models A \otimes B : \phi}$$

## 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)

$$\phi ::= p \mid \neg p \mid X \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid [a] \phi \mid \nu X. \phi$$

## Maximal Models $Max(\psi)$

- conditions 1-3 are satisfied
- exponential construction, lazy

# Applet Structure

## Applet $\mathcal{A}$

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

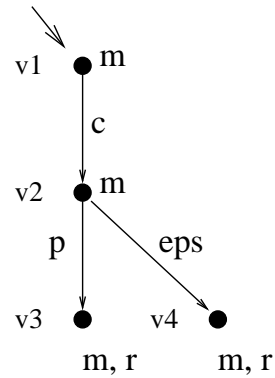
**Maximal Model** for property  $\psi$  is not necessarily a legal applet structure!

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

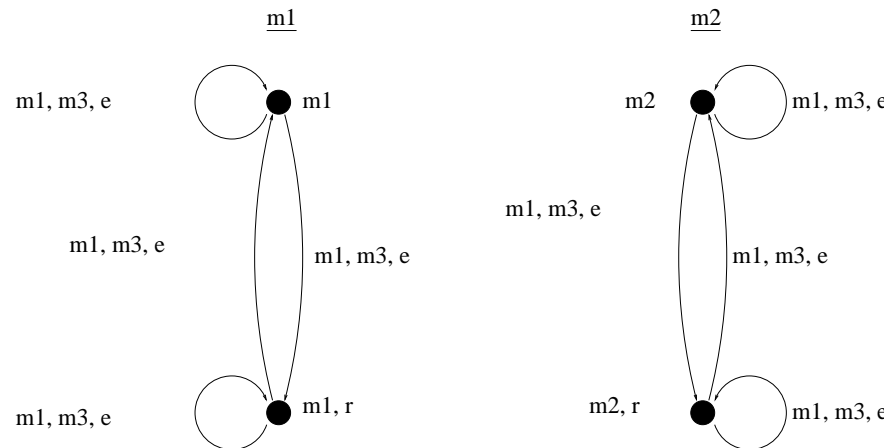
## Maximal Applet $Max_I(\psi)$

- is the maximal model  $Max(\phi_I \wedge \psi)$

# Method Graph: Example



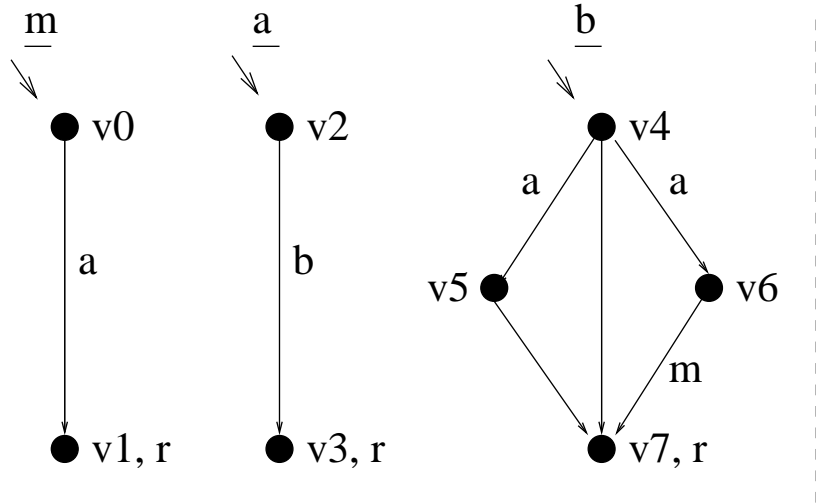
Maximal Applet for interface  $I = (\{m_1, m_2\}, \{m_1, m_3\})$



# Applet Behaviour

- Applet structure  $\mathcal{A}$  induces applet behaviour  $b(\mathcal{A})$ 
  - 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 *not* axiomatizable within the logic!
  - structural simulation implies behavioural simulation

# Applet Behaviour: Example



# Operational Semantics

$$\text{(call)} \quad \frac{m_1, m_2 \in I^+ \quad v_1 \xrightarrow{m_2}_{m_1} v'_1 \quad v_2 \models m_2 \wedge e}{(v_1, \sigma) \xrightarrow{m_1 \text{ call } m_2} (v_2, v'_1 \cdot \sigma)}$$

$$\text{(return)} \quad \frac{m_1, m_2 \in I^+ \quad v_2 \models m_2 \wedge r \quad v_1 \models m_1}{(v_2, v_1 \cdot \sigma) \xrightarrow{m_2 \text{ ret } m_1} (v_1, \sigma)}$$

$$\text{(transfer)} \quad \frac{m \in I^+ \quad v \rightarrow_m v'}{(v, \sigma) \xrightarrow{\varepsilon} (v', \sigma)}$$

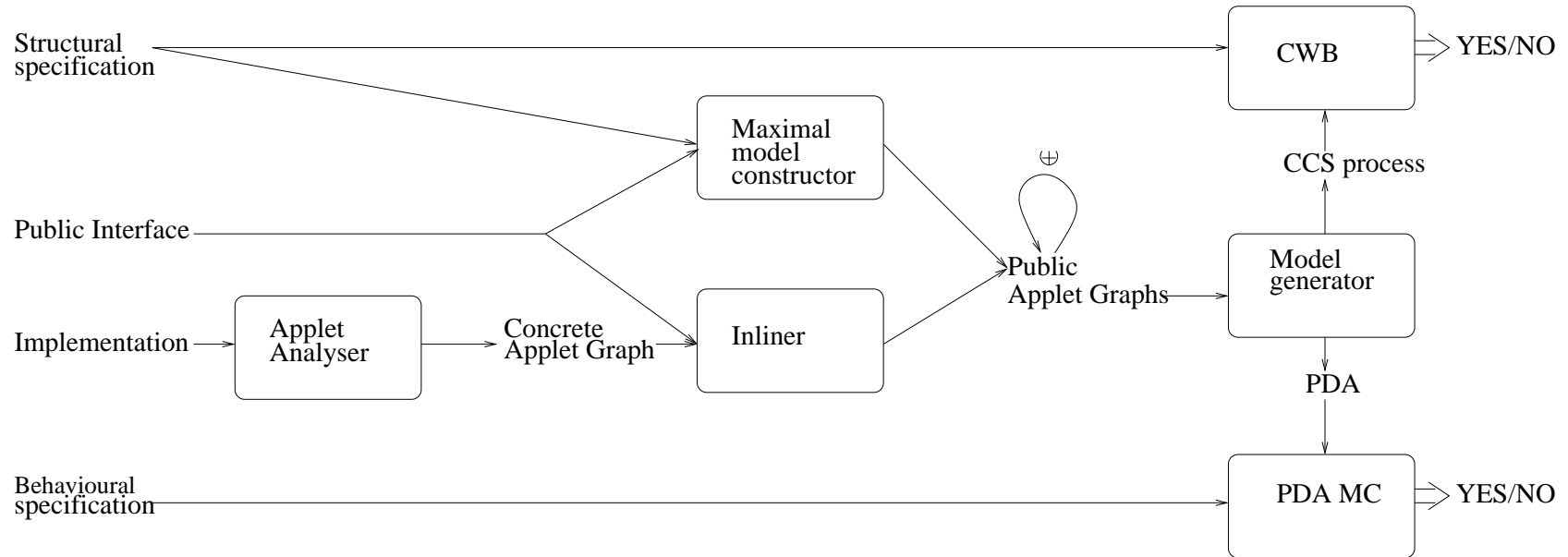
# A Compositional Verification Method

## Compositional Verification Principle

$$\frac{A \models_s \sigma \quad \mathit{Max}_{I_A}(\sigma) \uplus B \models_b \psi}{A \uplus B \models_b \psi} \quad A : I_A$$

1. a) Specify global property  $\psi$  as a **behavioural** property  
b) For applet  $A$ , specify local property  $\sigma$  as a **structural** property
2. Verify the correctness of the property decomposition:
  - a) compute maximal applet  $\mathit{Max}_{I_A}(\sigma)$
  - b) model check  $\mathit{Max}_{I_A}(\sigma) \uplus B \models_b \psi$
3. When implementation of  $A$  available, verify  $A \models_s \sigma$

# 3. Tool Set and Case Study



# 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 **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 **Loyalty.logFull**, no external calls are reachable other than calls to **Purse.getTransaction**
- $(\psi_P)$  from any entry point of **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  $(M, I^- - (I^+ - M))$

### Modified CVP

$$\frac{\alpha_M(\mathcal{A}) \models_s \sigma \quad \text{Max}_{I_{\alpha_M(\mathcal{A})}}(\sigma) \uplus \mathcal{B} \models_b \psi}{\mathcal{A} \uplus \mathcal{B} \models_b^{M \cup I_{\mathcal{B}}^+} \psi}$$

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

# Interface Behaviour

An abstraction on applet behaviour wrt.  $M \subseteq I_{\mathcal{A}}^+$

- 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  $b^M(\mathcal{A})$

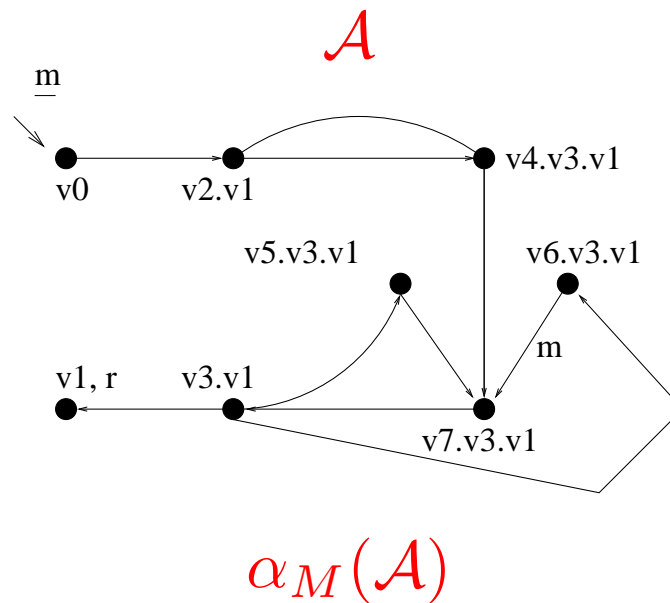
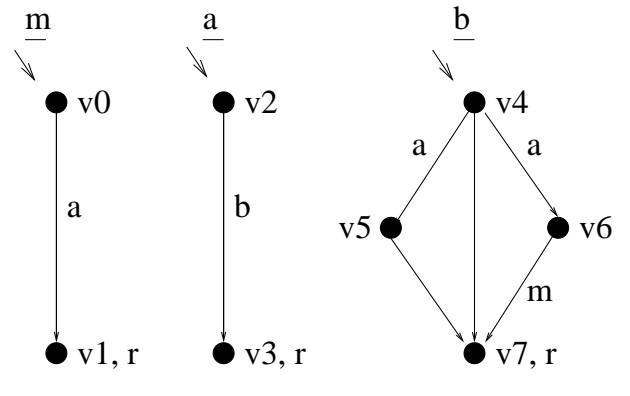
# The Inlining Transformation

**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  $\alpha_M(\mathcal{A})$



- introduces more interface behaviour!

# Simulation Results

**Theorem** Let  $\mathcal{A} : I$  and  $M \subseteq I^+$ .

Then  $b^M(\mathcal{A}) \leq b(\alpha_M(\mathcal{A})) = b^M(\alpha_M(\mathcal{A}))$ .

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

**Theorem** Let  $\mathcal{A} : I$  be last-call recursive, and  $M \subseteq I^+$ .

Then  $b^M(\mathcal{A}) \equiv_w b(\alpha_M(\mathcal{A})) = b^M(\alpha_M(\mathcal{A}))$ .

# Compositional Verification Method

## Modified CVP

$$\frac{\alpha_M(\mathcal{A}) \models_s \sigma \quad \text{Max}_{I_{\alpha_M(\mathcal{A})}}(\sigma) \uplus \mathcal{B} \models_b \psi}{\mathcal{A} \uplus \mathcal{B} \models_b^{M \cup I_{\mathcal{B}}^+} \psi}$$

1. a) Specify global property  $\psi$  as an **interface behavioural** property  
b) For applet  $\mathcal{A}$ , specify local property  $\sigma$  as a **structural** property of  $\alpha_M(\mathcal{A})$
2. Verify the correctness of the property decomposition:
  - a) compute maximal applet  $\text{Max}_{I_{\alpha_M(\mathcal{A})}}(\sigma)$
  - b) model check  $\text{Max}_{I_{\alpha_M(\mathcal{A})}}(\sigma) \uplus \mathcal{B} \models_b \psi$
3. When implementation of  $\mathcal{A}$  available:
  - a) compute  $\alpha_M(\mathcal{A})$
  - b) verify  $\alpha_M(\mathcal{A}) \models_s \sigma$

# 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]

	$\mathcal{M}ax(\sigma_L)$	in [HGSC'04]	$\mathcal{M}ax(\sigma_P)$	in [HGSC'04]
#nodes	8	474	8	2786
#edges	120	277 700	88	603 128
constr. time	0.05 s.	25 min.	0.05 s.	13 hrs.

- 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