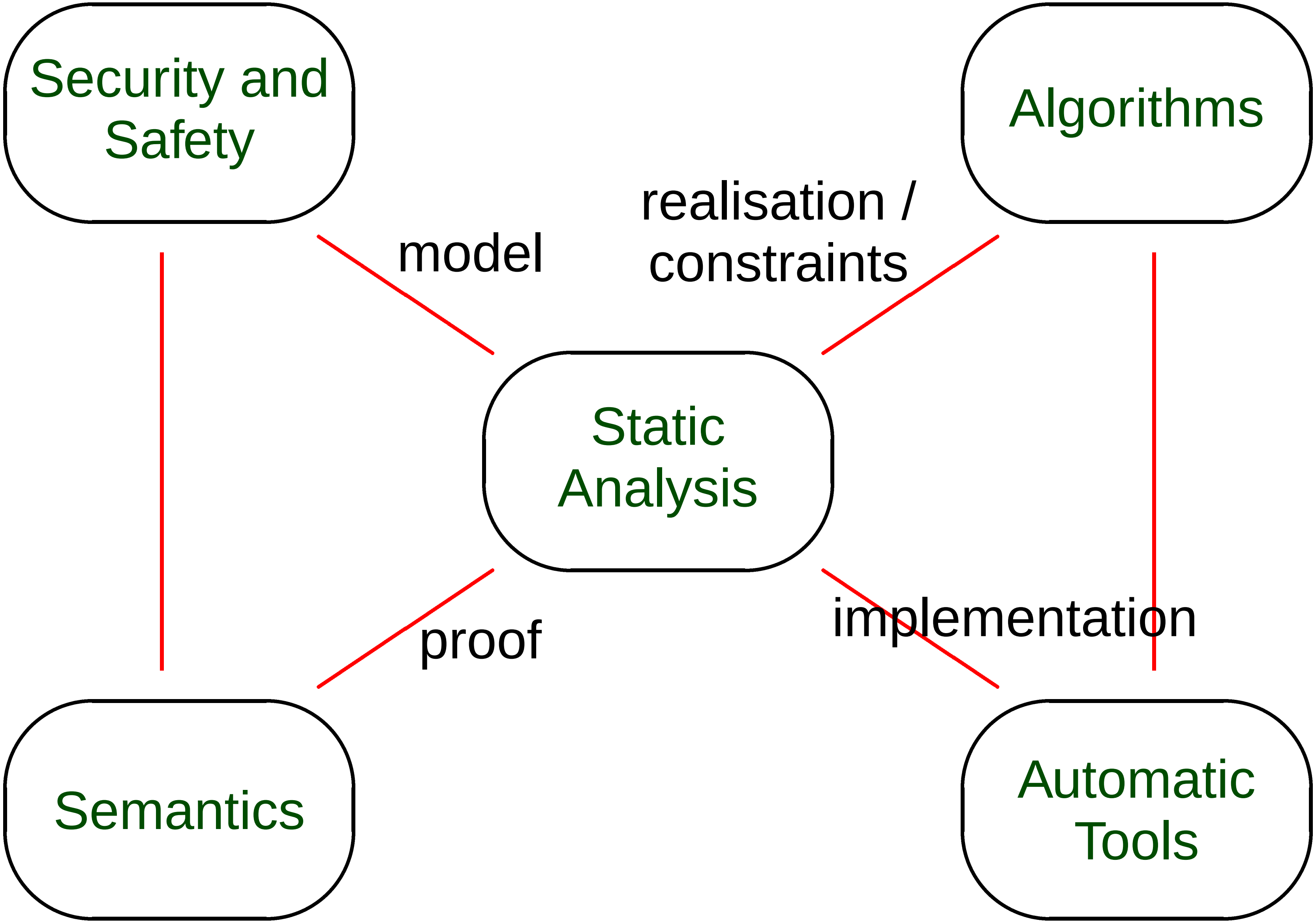
Security through Static Analysis

Chris Hankin, Imperial College

Thanks to: SecSafe partners, David Clark and Sebastian Hunt

Overview

- SecSafe objectives
- Carmel and Security
- Flow Logic
- Information Flow
- CFA for Carmel
- Conclusions


Security and Safety
Algorithms
model
realisation / constraints
Static Analysis
proof
implementation
Semantics
Automatic Tools

A subset of Javacard (Carmel) - Motivation:

- hiding of uninteresting language and JCVM details
- focus on salient features
- reduction of specification and development effort
- (almost) direct translation from JCVM language

$\Rightarrow$ the essence [JCVML*e*]

JCVM language	Carmel
185 low-level instructions	30 high-level instructions
AID, tokens, offsets	names

- Memory allocation control
  - *Dynamic memory allocation must be bounded.*
  - *No memory must be allocated after personalization.*
- Information flow control
  - *Given types of information must not flow outside the applet.*
- Service control
  - *Given program points must be executable only if given conditions are satisfied.*
- Error prediction
  - *No exception must reach the toplevel except ISOExceptions.*

Flow logic:
a multi-paradigmatic approach to static analysis

- Specification oriented
- Semantics based *not* semantics directed
- Integrates state-of-the-art from abstract interpretation and data flow analysis
- Multi-paradigmatic: functional,imperative,concurrent . . .

Information Flow for Algol-like Languages

- Information Flow Analysis
  - Prevent flow from *high* to *low*
- Flow Logic specification
  - Simple imperative language
  - Idealised Algol
- Extended with probabilistic constructs

Following Denning, it is possible to categorise information flows into direct vs indirect and explicit vs implicit flows.

Indirect flows: transitive flows (a flow from x to y followed by a flow from y to z implies a flow from x to z)

Direct explicit flows: arise from assignments; for example, $x := y + z$ causes explicit information flows from both y and z to x.

Direct implicit flows:

- *Local flows* arise from guards in conditionals:

$$\texttt{if} \quad x \quad \texttt{then} \quad y := z \quad \texttt{else} \quad y := w.$$

- *Global flows* arise from guards in while loops

$$x := y; (\texttt{while} \quad w \quad \texttt{do} \quad x := z); \cdots.$$

We will illustrate the approach for a simple imperative language:

$$S \in \mathbf{Statement},\ C \in \textbf{Command}$$
$$\ell \in \textbf{Lab},\ x \in \mathbf{Ide}$$
$$a \in \textbf{Arith-exp},\ b \in \textbf{Bool-exp}$$

$$\begin{array}{lcl} S & ::= & C^{\ell} \\ C & ::= & \texttt{skip} \mid x := a \mid S_1; S_2 \mid \\ & & \texttt{if}\ b\ \texttt{then}\ S_1\ \texttt{else}\ S_2 \mid \texttt{while}\ b\ \texttt{do}\ S \mid \\ & & \texttt{new}\ x.S \end{array}$$

$$\frac{}{\texttt{skip}^{\ell}, \sigma \Downarrow \texttt{void}, \sigma}$$

$$\frac{a, \sigma \Downarrow v, \sigma}{(x \mathbin{:=} a)^{\ell}, \sigma \Downarrow \texttt{void}, \sigma[x \mapsto v]}$$

$$\frac{S_1, \sigma \Downarrow \texttt{void}, \sigma' \qquad S_2, \sigma' \Downarrow \texttt{void}, \sigma''}{(S_1 ; S_2)^{\ell}, \sigma \Downarrow \texttt{void}, \sigma''}$$

$$\frac{b, \sigma \Downarrow 1, \sigma \quad S_1, \sigma \Downarrow \texttt{void}, \sigma'}{(\texttt{if}\ b\ \texttt{then}\ S_1\ \texttt{else}\ S_2)^{\ell}, \sigma \Downarrow \texttt{void}, \sigma'}$$

$$\frac{b, \sigma \Downarrow 0, \sigma \quad S_2, \sigma \Downarrow \texttt{void}, \sigma'}{(\texttt{if}\ b\ \texttt{then}\ S_1\ \texttt{else}\ S_2)^{\ell}, \sigma \Downarrow \texttt{void}, \sigma'}$$

$$\frac{b, \sigma \Downarrow 0, \sigma}{(\texttt{while}\ b\ \texttt{do}\ S)^{\ell}, \sigma \Downarrow \texttt{void}, \sigma}$$

$$\frac{b, \sigma \Downarrow 1, \sigma \quad S, \sigma \Downarrow \texttt{void}, \sigma' \quad (\texttt{while}\ b\ \texttt{do}\ S)^{\ell}, \sigma' \Downarrow \texttt{void}, \sigma''}{(\texttt{while}\ b\ \texttt{do}\ S)^{\ell}, \sigma \Downarrow \texttt{void}, \sigma''}$$

$$\frac{S, \sigma[x \mapsto 0] \Downarrow \texttt{void}, \sigma'}{(\texttt{new}\ x.\ S)^{\ell}, \sigma \Downarrow \texttt{void}, \sigma'[x \mapsto \sigma x]}$$

We write

$$(\widehat{\mathsf{X}}, \widehat{\mathsf{G}}, \widehat{\mathsf{D}}) \models S$$

when $(\widehat{\mathsf{X}}, \widehat{\mathsf{G}}, \widehat{\mathsf{D}})$ is an acceptable Information Flow Analysis of the statement S.

$$\widehat{\mathsf{X}} \in \mathbf{Assign} = \mathbf{Lab} \to \mathcal{P}(\mathbf{Ide})$$

$$\widehat{\mathsf{G}} \in \mathbf{Global} = \mathbf{Lab} \to \mathcal{P}(\widehat{\mathbf{Ide}})$$

$$\widehat{\mathsf{D}} \in \mathbf{Dep} = \mathbf{Lab} \to \mathcal{P}(\widehat{\mathbf{Ide}} \times \widehat{\mathbf{Ide}})$$

where $\widehat{\mathbf{Ide}} = \mathbf{Ide} \cup \{\bullet\}$.

We use $;$ for relational composition, thus: $x\ R\,;S\ z$ iff $\exists y.\ x\ R\ y\ S\ z$. We also overload this notation to allow the 'composition' of a set with a relation, thus:

$Y\,;R \stackrel{\mathrm{def}}{=} \{z \mid \exists y \in Y.\ y\ R\ z\}$.

We use the notation $f\backslash x$ to restrict the range of a partial function, thus: $(f\backslash x)(y)$ is undefined if $x = y$ and is $f(y)$ otherwise. We apply the same notation to binary relations:

$R\backslash x \stackrel{\mathrm{def}}{=} \{(y,z) \in R \mid y \neq x\}$.

Where convenient, we treat $D(\ell)$ as a function of type $\widehat{\mathbf{Ide}} \to \mathcal{P}(\widehat{\mathbf{Ide}})$. In particular, we use a 'function update' notation on relations thus: $R[x \mapsto Y] \stackrel{\mathrm{def}}{=} R\backslash x \cup \{x\} \times Y$.

$$(\widehat{\mathsf{X}}, \widehat{\mathsf{G}}, \widehat{\mathsf{D}}) \models \texttt{skip}^{\ell} \text{ iff } \widehat{\mathsf{D}}(\ell) \supseteq \mathrm{Id}$$

$$\begin{aligned}
(\widehat{\mathsf{X}}, \widehat{\mathsf{G}}, \widehat{\mathsf{D}}) &\models (x := a)^{\ell} \\
\text{iff} \quad & \widehat{\mathsf{X}}(\ell) \supseteq \{x\} \wedge \\
& \widehat{\mathsf{D}}(\ell) \supseteq \mathrm{Id}[x \mapsto \mathrm{FV}(a)]
\end{aligned}$$

$$\begin{aligned}
(\widehat{\mathsf{X}}, \widehat{\mathsf{G}}, \widehat{\mathsf{D}}) &\models (C_1^{\ell_1}; C_2^{\ell_2})^{\ell} \\
\text{iff} \quad & (\widehat{\mathsf{X}}, \widehat{\mathsf{G}}, \widehat{\mathsf{D}}) \models C_1^{\ell_1} \wedge (\widehat{\mathsf{X}}, \widehat{\mathsf{G}}, \widehat{\mathsf{D}}) \models C_2^{\ell_2} \wedge \\
& \widehat{\mathsf{X}}(\ell) \supseteq \widehat{\mathsf{X}}(\ell_1) \cup \widehat{\mathsf{X}}(\ell_2) \wedge \\
& \widehat{\mathsf{G}}(\ell) \supseteq \widehat{\mathsf{G}}(\ell_1) \cup \widehat{\mathsf{G}}(\ell_2) ; \widehat{\mathsf{D}}(\ell_1) \wedge \\
& \widehat{\mathsf{D}}(\ell) \supseteq \widehat{\mathsf{D}}(\ell_2) ; \widehat{\mathsf{D}}(\ell_1)
\end{aligned}$$

$$
\begin{aligned}
(\widehat{\mathsf{X}}, \widehat{\mathsf{G}}, \widehat{\mathsf{D}}) &\models (\texttt{if}\ b\ \texttt{then}\ C_1^{\ell_1}\ \texttt{else}\ C_2^{\ell_2})^{\ell} \\
\text{iff} \quad & (\widehat{\mathsf{X}}, \widehat{\mathsf{G}}, \widehat{\mathsf{D}}) \models C_1^{\ell_1} \wedge (\widehat{\mathsf{X}}, \widehat{\mathsf{G}}, \widehat{\mathsf{D}}) \models C_2^{\ell_2} \wedge \\
& \widehat{\mathsf{X}}(\ell) \supseteq \widehat{\mathsf{X}}(\ell_1) \cup \widehat{\mathsf{X}}(\ell_2) \wedge \\
& \widehat{\mathsf{G}}(\ell) \supseteq \widehat{\mathsf{G}}(\ell_1) \cup \widehat{\mathsf{G}}(\ell_2) \wedge \\
& (\bullet \in \widehat{\mathsf{G}}(\ell) \Rightarrow \widehat{\mathsf{G}}(\ell) \supseteq \mathrm{FV}(b)) \wedge \\
& \widehat{\mathsf{D}}(\ell) \supseteq \widehat{\mathsf{D}}(\ell_1) \cup \widehat{\mathsf{D}}(\ell_2) \wedge \\
& \widehat{\mathsf{D}}(\ell) \supseteq \widehat{\mathsf{X}}(\ell) \times \mathrm{FV}(b)
\end{aligned}
$$

$$(\widehat{\mathsf{X}}, \widehat{\mathsf{G}}, \widehat{\mathsf{D}}) \models (\mathtt{while}\ b\ \mathtt{do}\ C^{\ell_1})^{\ell}$$

$$\begin{aligned} \text{iff} \quad & (\widehat{\mathsf{X}}, \widehat{\mathsf{G}}, \widehat{\mathsf{D}}) \models C^{\ell_1} \wedge \\ & \widehat{\mathsf{X}}(\ell) \supseteq \widehat{\mathsf{X}}(\ell_1) \wedge \\ & \widehat{\mathsf{G}}(\ell) \supseteq \{\bullet\} \cup \mathrm{FV}(b) \cup \widehat{\mathsf{G}}(\ell_1) \cup \widehat{\mathsf{G}}(\ell)\,;\widehat{\mathsf{D}}(\ell_1) \wedge \\ & \widehat{\mathsf{D}}(\ell) \supseteq \mathrm{Id} \cup \widehat{\mathsf{D}}(\ell)\,;\widehat{\mathsf{D}}(\ell_1) \wedge \\ & \widehat{\mathsf{D}}(\ell) \supseteq \widehat{\mathsf{X}}(\ell) \times \mathrm{FV}(b) \end{aligned}$$

$$(\widehat{\mathsf{X}}, \widehat{\mathsf{G}}, \widehat{\mathsf{D}}) \models (\mathtt{new}\ x.\ C^{\ell_1})^{\ell}$$

$$\begin{aligned} \text{iff} \quad & (\widehat{\mathsf{X}}, \widehat{\mathsf{G}}, \widehat{\mathsf{D}}) \models C^{\ell_1} \wedge \\ & \widehat{\mathsf{X}}(\ell) \supseteq \widehat{\mathsf{X}}(\ell_1) \backslash \{x\} \wedge \\ & \widehat{\mathsf{G}}(\ell) \supseteq \widehat{\mathsf{G}}(\ell_1) \backslash \{x\} \wedge \\ & \widehat{\mathsf{D}}(\ell) \supseteq \widehat{\mathsf{D}}(\ell_1) \backslash \{x\} \cup \{(x, x)\} \end{aligned}$$

We are concerned with three aspects of correctness:

- First, that the analysis is well-defined.
- Second, that the analysis results are a proper abstraction of the semantics.
- Third, that every program has an acceptable information flow analysis and that the constraints have solutions.

Having analysed a program, C^ℓ, we determine that there is a breach of security if either

- $H \cap \widehat{\mathsf{G}}(\ell) \neq \emptyset$, or
- $\exists x \in L. \exists y \in H. x \widehat{\mathsf{D}}(\ell) y$

First we consider an example:

$$
\begin{array}{l}
(((\,\texttt{while}\ (x < 3)\\
\quad\ \texttt{do}\ (((\,\texttt{if}\ (p = g)\\
\qquad\qquad\quad \texttt{then}\ (f := 1)^{l_{10}}\\
\qquad\qquad\quad \texttt{else}\ (f := 0)^{l_{11}})^{l_8};\\
\qquad\qquad\ (x := x + 1)^{l_9})^{l_6};\\
\qquad\quad\ (g := g + 10)^{l_7})^{l_5}\ )^{l_3};\\
\quad (f := 2)^{l_4}\ )^{l_1};\\
\ \ (x := 0)^{l_2}\ )^{l_0}
\end{array}
$$

The analysis of this program produces a set of constraints to be solved:

$$
\begin{aligned}
\widehat{\mathsf{X}}(\ell_0) \supseteq\ & \widehat{\mathsf{X}}(\ell_1) \cup \widehat{\mathsf{X}}(\ell_2) \ \wedge\ \widehat{\mathsf{G}}(\ell_0) \supseteq \widehat{\mathsf{G}}(\ell_1) \cup \widehat{\mathsf{G}}(\ell_2)\,;\widehat{\mathsf{D}}(\ell_1) \\
& \wedge \widehat{\mathsf{D}}(\ell_0) \supseteq \widehat{\mathsf{D}}(\ell_2)\,;\widehat{\mathsf{D}}(\ell_1) \\
\widehat{\mathsf{X}}(\ell_1) \supseteq\ & \widehat{\mathsf{X}}(\ell_3) \cup \widehat{\mathsf{X}}(\ell_4) \ \wedge\ \widehat{\mathsf{G}}(\ell_1) \supseteq \widehat{\mathsf{G}}(\ell_3) \cup \widehat{\mathsf{G}}(\ell_4)\,;\widehat{\mathsf{D}}(\ell_3) \\
& \wedge \widehat{\mathsf{D}}(\ell_1) \supseteq \widehat{\mathsf{D}}(\ell_4)\,;\widehat{\mathsf{D}}(\ell_3) \\
\widehat{\mathsf{X}}(\ell_2) \supseteq\ & \{x\} \ \wedge\ \widehat{\mathsf{D}}(\ell_2) \supseteq \mathrm{Id}[x \mapsto \emptyset] \\
& \vdots
\end{aligned}
$$

Iterating over these constraints beginning from $\widehat{\mathsf{X}} = \lambda x.\emptyset$, $\widehat{\mathsf{G}} = \lambda x.\emptyset$, and $\widehat{\mathsf{D}} = \lambda x.\emptyset$ to a fixed point giving the least solution yields:

$$\begin{array}{rcl} \widehat{\mathsf{X}}(\ell_0) & = & \{f, x, g\} \\ \widehat{\mathsf{G}}(\ell_0) & = & \{\bullet, x\} \\ \widehat{\mathsf{D}}(\ell_0) & = & \{(p, p), (g, g), (g, x)\} \end{array}$$

which satisfy the security criteria for the while language (cf type-based approaches).

We now return to the correctness ...

The specification of the analysis is essentially defining the relation:

$$\models \; : (\mathbf{Assign}\times\mathbf{Global}\times\mathbf{Dep}\times\mathbf{Statement}) \to \{\mathtt{true}, \mathtt{false}\}$$

$$\mathcal{Q} :$$
$$((\mathbf{Assign} \times \mathbf{Global} \times \mathbf{Dep} \times \mathbf{Statement}) \to \{\mathtt{true}, \mathtt{false}\}) \to$$
$$((\mathbf{Assign} \times \mathbf{Global} \times \mathbf{Dep} \times \mathbf{Statement}) \to \{\mathtt{true}, \mathtt{false}\})$$

$$Q_1 \sqsubseteq Q_2 \Leftrightarrow \;\; \forall(\widehat{\mathsf{X}}, \widehat{\mathsf{G}}, \widehat{\mathsf{D}}, S) :$$
$$(Q_1(\widehat{\mathsf{X}}, \widehat{\mathsf{G}}, \widehat{\mathsf{D}}, S) = \mathtt{true}) \Rightarrow (Q_2(\widehat{\mathsf{X}}, \widehat{\mathsf{G}}, \widehat{\mathsf{D}}, S) = \mathtt{true})$$

Given a set of variables X, we write $\sigma_1 \sim_X \sigma_2$ to mean that the two stores agree on all $x \in X$:

$$\sigma_1 \sim_X \sigma_2 \Leftrightarrow \forall x \in X.\ \sigma_1(x) = \sigma_2(x)$$

Clearly, $\sim_X$ is an equivalence relation for any choice of X.

We sometimes write $\sim_x$ to mean $\sim_{\{x\}}$.

Assignment Freedom

Suppose $(\widehat{\mathsf{X}}, \widehat{\mathsf{G}}, \widehat{\mathsf{D}}) \models C^\ell$ and let $X' = \{x \in \mathbf{Ide} \mid x \notin \widehat{\mathsf{X}}(\ell)\}$. Then:

1. if $C^\ell, \sigma \Downarrow \mathtt{void}, \sigma'$ then $\sigma' \sim_{X'} \sigma$
2. if $x \notin \widehat{\mathsf{X}}(\ell)$ then $x\ \widehat{\mathsf{D}}(\ell)\ x$

Proof: Part 1 by induction on the height of the derivation.

Part 2 by structural induction.

Store Independence

Suppose $(\widehat{\mathsf{X}}, \widehat{\mathsf{G}}, \widehat{\mathsf{D}}) \models C^\ell$, then, for all x:

$$\begin{array}{l} \text{if } (\sigma_1 \sim_{D_x} \sigma_2) \\ \text{then} \\ \quad \text{if } (C^\ell, \sigma_1 \Downarrow \mathtt{void}, \sigma_1' \wedge C^\ell, \sigma_2 \Downarrow \mathtt{void}, \sigma_2') \\ \quad \text{then } \sigma_1' \sim_x \sigma_2' \end{array}$$

where $D_x = \widehat{\mathsf{D}}(\ell)(x)$.

Proof: Proof is by induction on the height of the first derivation.

Termination Independence

Suppose $(\widehat{\mathsf{X}}, \widehat{\mathsf{G}}, \widehat{\mathsf{D}}) \models C^\ell$. Then:

1. if $\bullet \notin \widehat{\mathsf{G}}(\ell)$ then $C^\ell, \sigma \Downarrow$ for all σ.
2. if $\sigma_1 \sim_{\widehat{\mathsf{G}}(\ell)} \sigma_2$ then $C^\ell, \sigma_1 \Downarrow \;\Leftrightarrow C^\ell, \sigma_2 \Downarrow$

Proof: Part 1 is by structural induction. Part 2 is by induction on the height of the derivation.

Existence of solutions

For all $S \in \mathbf{Statement}$ the set $\{(\widehat{\mathsf{X}}, \widehat{\mathsf{G}}, \widehat{\mathsf{D}}) \mid (\widehat{\mathsf{X}}, \widehat{\mathsf{G}}, \widehat{\mathsf{D}}) \models S\}$ is a Moore family.

Recall that a subset Y of a complete lattice $L = (L, \sqsubseteq)$ is a Moore family if and only if $\sqcap Y' \in Y$ for all $Y' \subseteq Y$.

An immediate corollary of our result is that there is always an acceptable information flow analysis for a statement and that, moreover, there is a least analysis.

We now return to Carmel.

The main issue is that we have to deal with method invocation. This essentially means that we need an inter-procedural information flow analysis.

But . . . which methods are being invoked?

Flow Logic for Carmel (Rene Rydhof Hansen)

- Control Flow Analysis for Carmel
- Proved correct wrt. semantics
- Extensions for exceptions, ownership (firewall) etc.
- Basis for prototype implementation

$\hat{K}$ tracks values of static fields for each class, $\hat{H}$ tracks values of instance fields for individual objects, $\hat{L}$ is the local heap and $\hat{S}$ is the abstract operand stack. Judgements are of the form:

$$(\hat{K}, \hat{H}, \hat{L}, \hat{S}) \models \texttt{addr} : \texttt{instr}$$

- Analysing push:

$$\begin{array}{ll} & (\hat{K}, \hat{H}, \hat{L}, \hat{S}) \models (m_0, pc_0) : \texttt{push}\ t\ v \\ \text{iff} & \{v\} :: \hat{S}(m_0, pc_0) \sqsubseteq \hat{S}(m_0, pc_0 + 1) \\ & \hat{L}(m_0, pc_0) \sqsubseteq \hat{L}(m_0, pc_0 + 1) \end{array}$$

- Analysing `invokevirtual`:

$$
\begin{array}{l}
(\hat{K}, \hat{H}, \hat{L}, \hat{S}) \models (m_0, pc_0) : \texttt{invokevirtual}\ m \text{ iff} \\
\quad A_1 :: \cdots :: A_{|m|} :: B :: X \triangleleft \hat{S}(m_0, pc_0) : \\
\quad \forall (\mathrm{Ref}\sigma) \in B : \\
\qquad m_v = \mathsf{methodLookup}(m.\mathrm{id}, \sigma) \\
\qquad \{(\mathrm{Ref}\sigma)\} :: A_1 :: \cdots :: A_{|m|} \sqsubseteq \hat{L}(m_v, 1)[0..|m|] \\
\qquad T :: Y \triangleleft \hat{S}(m_v, \mathsf{END}_{m_v}) : \\
\qquad\quad T :: X \sqsubseteq \hat{S}(m_0, pc_0 + 1) \\
\quad \hat{L}(m_0, pc_0) \sqsubseteq \hat{L}(m_0, pc_0 + 1)
\end{array}
$$

The information flow analysis of `invokevirtual` might look something like:

$$
\begin{aligned}
&(\widehat{\mathsf{X}}, \widehat{\mathsf{G}}, \widehat{\mathsf{D}}) \models_{(\hat{K},\hat{H},\hat{L},\hat{S})} (m_0, pc_0) : \texttt{invokevirtual}\ m \text{ iff} \\
&A_1 :: \cdots :: A_{|m|} :: B :: X \lhd \hat{S}(m_0, pc_0) : \\
&\forall (\mathrm{Ref}\,\sigma) \in B : \\
&\quad m_v = \mathsf{methodLookup}(m.\mathrm{id}, \sigma) \\
&\quad \widehat{\mathsf{D}}(m_0, pc_0)^+ \subseteq \widehat{\mathsf{D}}(m_v, 1) \\
&\quad \widehat{\mathsf{X}}(m_v, \mathsf{END}_{m_v}) \subseteq \widehat{\mathsf{X}}(m_0, pc_0 + 1) \\
&\quad \widehat{\mathsf{G}}(m_v, \mathsf{END}_{m_v}) \subseteq \widehat{\mathsf{G}}(m_0, pc_0 + 1) \\
&\quad \widehat{\mathsf{D}}(m_v, \mathsf{END}_{m_v})^- \subseteq \widehat{\mathsf{D}}(m_0, pc_0 + 1)
\end{aligned}
$$

Conclusions

We have seen:

- SecSafe objectives
- Flow Logic
- Information Flow Analysis

It remains to:

- develop the Infromation Flow Logic for Carmel
- to develop other security analyses