ABSTRACT NON-INTERFERENCE

Parameterizing Non-Interference by Abstract Interpretation

Roberto Giacobazzi and Isabella Mastroeni

Dipartimento di Informatica

Università di Verona

Italy

POPL'04 - Venice, January 15th, 2004

The Problem

- The problem: Protect data confidentiality from erroneous/malicious attacks while data are processed
  - Attack = disclosing properties of confidential data
  - $\Rightarrow$ Access control by declaring data privileges!

The Problem

- The problem: Protect data confidentiality from erroneous/malicious attacks while data are processed

$\Rightarrow$ *Access control methods do not put constraints on how the information is propagated!*

The Problem

- The problem: Protect data confidentiality from erroneous/malicious attacks while data are processed

- Description of the problem:
  - Typing of data (and variables) in *private* (H) and *public* (L);
  - *Non-Interference*: to prevent the results of the computation from leaking even partial information about private inputs!
    - *Explicit flow*: caused by directly passing private data to a public variable: $l := 2 * h$;
    - *Implicit flow*: arise from control structure of the program: **while** h **do** $l := l + 1; h := h - 1$.
  - We consider only *terminating* computations!

The Goal

IT IS ESSENTIAL TO KNOW HOW MUCH AN ATTACKER MAY LEARN FROM A PROGRAM!

- Goal: Automatically generate certificates about secure information flows
  - Design of accurate security polices
  - ⇒ Static program analysis & verification techniques (types,CFA,DFA,...)

The Goal

IT IS ESSENTIAL TO KNOW HOW MUCH AN ATTACKER MAY LEARN FROM A PROGRAM!

- Goal: Automatically generate certificates about secure information flows

- State of the art: Standard non-interference is far too restrictive
  - No sensitive information can be disclosed
  - Any change upon confidential data has not to be revealed by public ones
  - Rigid security policy: L can flow into H but H cannot flow into L [Denning and Denning '77]

The Goal

IT IS ESSENTIAL TO KNOW HOW MUCH AN ATTACKER MAY LEARN FROM A PROGRAM!

- Goal: Automatically generate certificates about secure information flows
- State of the art: Standard non-interference is far too restrictive
- Question: Is there a way to characterize what kind of information flows?
  - Characterize the secrecy degree of a program
  - H can flow into L unless a given property of H is disclosed
  - Weakening standard non-interference
    (a challenge in language-based security [Sabelfeld & Myers ’03])

IDEA: Attackers as Abstract Interpretations

while h **do** $(l := l + 2;\ h := h - 1)$

IDEA: Attackers as Abstract Interpretations

$$\textbf{while } h \textbf{ do } (l := l + 2;\ h := h - 1)$$

- There is an (implicit/absolute) flow from h into l
- The parity of l is not affected by any change of h
- ... no information flow for *parity*!

IDEA: Attackers as Abstract Interpretations

$$\textbf{while } h \textbf{ do } (l := l + 2;\ h := h - 1)$$

- The idea: Abstract non-interference
  - Attackers as program analyzers
  - ⇒ Attackers can analyze I/O behaviour of public data
  - ⇒ Attackers perform “static” program analyses

$\Longrightarrow$ Abstract interpretation is a general method for specifying approximate semantics of programs [Cousot & Cousot ’77]

Attackers are abstract interpretations of program semantics

IDEA: Attackers as Abstract Interpretations

- The idea: Abstract non-interference

- Main results:
  - Generalizing non-interference relatively to the attacker's power
  - Making non-interference parametric on the attacker's point of view
  - Checking abstract non-interference by abstract interpretation
  - Systematic method for deriving attackers for programs by modifying abstractions
  - Abstract Robust Declassification

Related works

Refining security policies by constraining attackers

Characterizing released information

Related works

Refining security policies by constraining attackers

- Complexity:
  - Security levels corresponding to how complex is attacking the program
    [Lowe '02]

Related works

Refining security policies by constraining attackers

- Complexity:
  - Security levels corresponding to how complex is attacking the program
    [Lowe '02]
- Quantitative measure:
  - An absolute (approximate) quantitative evaluation of information leakage (number of statistical tests to disclose properties)
    [Di Pierro et al. '02]

Related works

Characterizing released information

- Quantitative measure:
  - Quantification of the information flowed by information theory [D. Clark et al. ’03]

Related works

Characterizing released information

- Quantitative measure:
  - Quantification of the information flowed by information theory
    [D. Clark et al. ’03]
- Robust declassification:
  - The observational capability of the attacker is characterized by equivalence relations, then the information released is identified and declassified.
    [Zdancewic and Myers ’01]

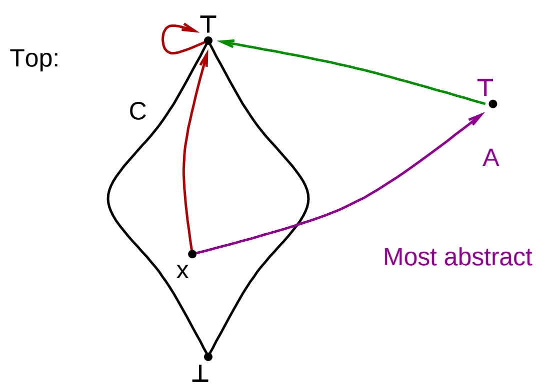
AI: Lattice of Abstractions

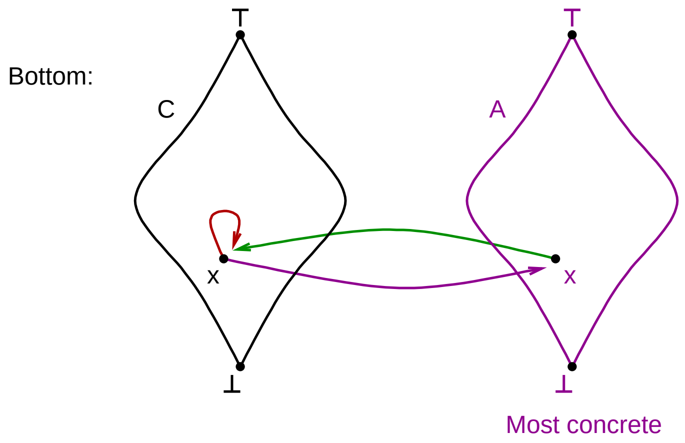
The concrete domain $< C, \leq, \wedge, \vee, \bot, \top >$ [Cousot & Cousot '79]

Lattice of abstract domains $\equiv$ *Abs*(C)

$< \textit{Abs}(C), \sqsubseteq, \sqcap, \sqcup, \top, C >$

$A_1 \sqsubseteq A_2 \quad \Leftrightarrow \quad A_2 \subseteq A_1$ (A_1 more precise than A_2)





Standard non-interference

"One group of users [...] is noninterfering with another group of users if what the first group does [...] has no effect on what the second group of users can see" [Goguen & Meseguer '82]

Standard non-interference

$$\forall l : \mathsf{L}, \forall h_1, h_2 : \mathsf{H}.\ [\![P]\!](h_1, l)^{\mathsf{L}} = [\![P]\!](h_2, l)^{\mathsf{L}}$$

Standard non-interference

Standard non-interference

$$\forall l : \mathsf{L}, \forall h_1, h_2 : \mathsf{H}.\ [\![P]\!](h_1, l)^{\mathsf{L}} = [\![P]\!](h_2, l)^{\mathsf{L}}$$

EXAMPLE:

$$\textbf{while } h \textbf{ do } (l := l + 2;\ h := h - 1).$$

Standard non-interference

Standard non-interference

$$\forall l : \mathsf{L}, \forall h_1, h_2 : \mathsf{H}.\ [\![P]\!](h_1, l)^{\mathsf{L}} = [\![P]\!](h_2, l)^{\mathsf{L}}$$

EXAMPLE:

$$\textbf{while } h \textbf{ do } (l := l + 2;\ h := h - 1).$$

$$h = 0,\ l = 1 \ \rightsquigarrow\ l = 1$$
$$h = 1,\ l = 1 \ \rightsquigarrow\ l = 3$$
$$h = n,\ l = 1 \ \rightsquigarrow\ l = 1 + 2n$$

Standard non-interference

Standard non-interference

$$\forall l : \mathsf{L}, \forall h_1, h_2 : \mathsf{H}.\ [\![P]\!](h_1, l)^{\mathsf{L}} = [\![P]\!](h_2, l)^{\mathsf{L}}$$

EXAMPLE:

$$\textbf{while } h \textbf{ do } (l := l + 2;\ h := h - 1).$$

$$h = 0,\ l = 1 \ \leadsto\ l = 1$$
$$h = 1,\ l = 1 \ \leadsto\ l = 3$$
$$h = n,\ l = 1 \ \leadsto\ l = 1 + 2n$$

If l is unchanged then h is 0!

$\leadsto$ There is an information flow from h into l.

Standard non-interference

Standard non-interference

$$\forall l : \mathsf{L}, \forall h_1, h_2 : \mathsf{H}.\ [\![P]\!](h_1, l)^{\mathsf{L}} = [\![P]\!](h_2, l)^{\mathsf{L}}$$

EXAMPLE:

$$\textbf{while } h \textbf{ do } (l := l + 2;\ h := h - 1).$$

$$h = 0,\ l = 1 \ \leadsto\ l = 1$$
$$h = 1,\ l = 1 \ \leadsto\ l = 3$$
$$h = n,\ l = 1 \ \leadsto\ l = 1 + 2n$$

If l is unchanged then h is 0!

$\leadsto$ There is an information flow from h into l.

$\Rightarrow$ Note that if the input l is even/odd then the output l is even/odd!

Abstracting non-interference I

Standard non-interference

$$\forall l : \mathsf{L}, \forall h_1, h_2 : \mathsf{H}.\ [\![P]\!](h_1, l)^{\mathsf{L}} = [\![P]\!](h_2, l)^{\mathsf{L}}$$

Consider $\alpha, \eta \in \mathit{Abs}(\wp(\mathbb{V}^{\mathsf{L}}))$:

Abstracting non-interference I

Standard non-interference

$$\forall l : \mathsf{L}, \forall h_1, h_2 : \mathsf{H}.\ [\![P]\!](h_1, l)^{\mathsf{L}} = [\![P]\!](h_2, l)^{\mathsf{L}}$$

Consider $\alpha, \eta \in \mathit{Abs}(\wp(\mathbb{V}^{\mathsf{L}}))$:

Narrow (abstract) non-interference $[\eta]P(\alpha)$:

$\eta(l_1) = \eta(l_2) \Rightarrow \alpha([\![P]\!](h_1, l_1)^{\mathsf{L}}) = \alpha([\![P]\!](h_2, l_2)^{\mathsf{L}})$

- No change of H values and η-equivalent L values may affect the α abstraction of L outputs.
- Possible deceptive interference due to η-undistinguished L values!
- The more η is precise the less deceptive interference appears

Abstracting non-interference I

Standard non-interference

$$\forall l : \mathsf{L}, \forall h_1, h_2 : \mathsf{H}.\ [\![P]\!](h_1, l)^{\mathsf{L}} = [\![P]\!](h_2, l)^{\mathsf{L}}$$

Consider $\alpha, \eta \in \mathit{Abs}(\wp(\mathbb{V}^{\mathsf{L}}))$:

Narrow (abstract) non-interference $[\eta]P(\alpha)$:

$\eta(l_1) = \eta(l_2) \Rightarrow \alpha([\![P]\!](h_1, l_1)^{\mathsf{L}}) = \alpha([\![P]\!](h_2, l_2)^{\mathsf{L}})$

EXAMPLE: $[\mathtt{id}]P(\mathit{Par})$

$$P = \ \textbf{while}\ h\ \textbf{do}\ (l := l + 2;\ h := h - 1).$$

$$h = 0,\ l = 1 \rightsquigarrow \mathit{Par}(l) = \text{odd}$$
$$h = 1,\ l = 1 \rightsquigarrow \mathit{Par}(l) = \text{odd}$$
$$h = n,\ l = 1 \rightsquigarrow \mathit{Par}(l) = \text{odd}$$

Abstracting non-interference I

Standard non-interference

$$\forall l : \mathsf{L}, \forall h_1, h_2 : \mathsf{H}.\ [\![P]\!](h_1, l)^{\mathsf{L}} = [\![P]\!](h_2, l)^{\mathsf{L}}$$

Consider $\alpha, \eta \in \mathit{Abs}(\wp(\mathbb{V}^{\mathsf{L}}))$:

Narrow (abstract) non-interference $[\eta]P(\alpha)$:

$\eta(l_1) = \eta(l_2) \Rightarrow \alpha([\![P]\!](h_1, l_1)^{\mathsf{L}}) = \alpha([\![P]\!](h_2, l_2)^{\mathsf{L}})$

EXAMPLE: $[\mathtt{id}]P(\mathit{Par})$

$$P = \textbf{ while } h \textbf{ do } (l := l + 2;\ h := h - 1).$$

$$h = 0,\ l = 1 \rightsquigarrow \mathit{Par}(l) = \text{odd}$$
$$h = 1,\ l = 1 \rightsquigarrow \mathit{Par}(l) = \text{odd}$$
$$h = n,\ l = 1 \rightsquigarrow \mathit{Par}(l) = \text{odd}$$

If l is odd/even then, independently from h, after the execution l is odd/even!
$\rightsquigarrow$ There is not an information flow from h into the parity of l.

Abstracting non-interference I

Standard non-interference

$$\forall l : \mathsf{L}, \forall h_1, h_2 : \mathsf{H}.\ [\![P]\!](h_1, l)^{\mathsf{L}} = [\![P]\!](h_2, l)^{\mathsf{L}}$$

Consider $\alpha, \eta \in \mathit{Abs}(\wp(\mathbb{V}^{\mathsf{L}}))$:

Narrow (abstract) non-interference $[\eta]P(\alpha)$:

$$\eta(l_1) = \eta(l_2) \Rightarrow \alpha([\![P]\!](h_1, l_1)^{\mathsf{L}}) = \alpha([\![P]\!](h_2, l_2)^{\mathsf{L}})$$

EXAMPLE II: $[\mathit{Par}]P(\mathit{Sign})$

$$P = \ l := 2 * l * h^2.$$

$$h = -3,\ l = -2\ (\mathit{Par}(-2) = \text{even}) \ \rightsquigarrow\ \mathit{Sign}(l) = -$$

$$h = 1,\ l = -4\ (\mathit{Par}(-4) = \text{even}) \ \rightsquigarrow\ \mathit{Sign}(l) = -$$

Abstracting non-interference I

Standard non-interference

$$\forall l : \mathsf{L}, \forall h_1, h_2 : \mathsf{H}.\ [\![P]\!](h_1, l)^{\mathsf{L}} = [\![P]\!](h_2, l)^{\mathsf{L}}$$

Consider $\alpha, \eta \in \mathit{Abs}(\wp(\mathbb{V}^{\mathsf{L}}))$:

Narrow (abstract) non-interference $[\eta]P(\alpha)$:

$\eta(l_1) = \eta(l_2) \Rightarrow \alpha([\![P]\!](h_1, l_1)^{\mathsf{L}}) = \alpha([\![P]\!](h_2, l_2)^{\mathsf{L}})$

EXAMPLE II: $[\mathit{Par}]P(\mathit{Sign})$

$$P = \quad l := 2 * l * h^2.$$

$$h = 1,\ l = 4\ (\mathit{Par}(4) = \text{even}) \rightsquigarrow \mathit{Sign}(l) = +$$

$$h = 1,\ l = -4\ (\mathit{Par}(-4) = \text{even}) \rightsquigarrow \mathit{Sign}(l) = -$$

The sign of the output l depends on the sign of the input l!

$\rightsquigarrow$ There is a DECEPTIVE FLOW!

Abstracting non-interference I

Standard non-interference

$$\forall l : \mathsf{L}, \forall h_1, h_2 : \mathsf{H}.\ [\![P]\!](h_1, l)^{\mathsf{L}} = [\![P]\!](h_2, l)^{\mathsf{L}}$$

Consider $\alpha, \eta \in \textit{Abs}(\wp(\mathbb{V}^{\mathsf{L}}))$:

Narrow (abstract) non-interference $[\eta]P(\alpha)$:

$\eta(l_1) = \eta(l_2) \Rightarrow \alpha([\![P]\!](h_1, l_1)^{\mathsf{L}}) = \alpha([\![P]\!](h_2, l_2)^{\mathsf{L}})$

EXAMPLE II: $[\textit{Par}]P(\textit{Sign})$

$$P = \quad l := 2 * l * h^2.$$

$$h = 1,\ l = 4\ (\textit{Par}(4) = \text{even}) \rightsquigarrow \textit{Sign}(l) = +$$

$$h = 1,\ l = -4\ (\textit{Par}(-4) = \text{even}) \rightsquigarrow \textit{Sign}(l) = -$$

The sign of the output l depends on the sign of the input l!

$\rightsquigarrow$ There is a DECEPTIVE FLOW!

$\Rightarrow$ We compute the semantics on the concrete value of the input l!

Abstracting non-interference II

Consider $\alpha, \eta \in \mathit{Abs}(\wp(\mathbb{V}^{L}))$:

Narrow (abstract) non-interference $[\eta]P(\alpha)$:
$\eta(l_1) = \eta(l_2) \Rightarrow \alpha([\![P]\!](h_1, l_1)^{L}) = \alpha([\![P]\!](h_2, l_2)^{L})$

Abstracting non-interference II

Consider $\alpha, \eta \in \mathit{Abs}(\wp(\mathbb{V}^{\mathsf{L}}))$:

Narrow (abstract) non-interference $[\eta]P(\alpha)$:

$$\eta(l_1) = \eta(l_2) \Rightarrow \alpha(\llbracket P \rrbracket(h_1, l_1)^{\mathsf{L}}) = \alpha(\llbracket P \rrbracket(h_2, l_2)^{\mathsf{L}})$$

Abstracting non-interference $(\eta)P(\alpha)$:

$$\eta(l_1) = \eta(l_2) \Rightarrow \alpha(\llbracket P \rrbracket(h_1, \eta(l_1))^{\mathsf{L}}) = \alpha(\llbracket P \rrbracket(h_2, \eta(l_2))^{\mathsf{L}})$$

- No change of H values may affect the α abstraction of L outputs.
- No deceptive interference due to L data

Abstracting non-interference II

Consider $\alpha, \eta \in \mathit{Abs}(\wp(\mathbb{V}^{\mathsf{L}}))$:

Narrow (abstract) non-interference $[\eta]P(\alpha)$:

$\eta(l_1) = \eta(l_2) \Rightarrow \alpha(\llbracket P \rrbracket(h_1, l_1)^{\mathsf{L}}) = \alpha(\llbracket P \rrbracket(h_2, l_2)^{\mathsf{L}})$

Abstracting non-interference $(\eta)P(\alpha)$:

$\eta(l_1) = \eta(l_2) \Rightarrow \alpha(\llbracket P \rrbracket(h_1, \eta(l_1))^{\mathsf{L}}) = \alpha(\llbracket P \rrbracket(h_2, \eta(l_2))^{\mathsf{L}})$

EXAMPLE: $(\mathit{Par})P(\mathit{Sign})$

$$P = \quad l := 2 * l * h^2.$$

$h = -3$, $\mathit{Par}(l) = \text{even}$ $\rightsquigarrow$ $\mathit{Sign}(l) = \text{I don't know}$

$h = 1$, $\mathit{Par}(l) = \text{even}$ $\rightsquigarrow$ $\mathit{Sign}(l) = \text{I don't know}$

Abstracting non-interference II

Consider $\alpha, \eta \in Abs(\wp(\mathbb{V}^{L}))$:

Narrow (abstract) non-interference $[\eta]P(\alpha)$:

$\eta(l_1) = \eta(l_2) \Rightarrow \alpha([\![P]\!](h_1, l_1)^{L}) = \alpha([\![P]\!](h_2, l_2)^{L})$

Abstracting non-interference $(\eta)P(\alpha)$:

$\eta(l_1) = \eta(l_2) \Rightarrow \alpha([\![P]\!](h_1, \eta(l_1))^{L}) = \alpha([\![P]\!](h_2, \eta(l_2))^{L})$

EXAMPLE: $(Par)P(Sign)$

$$P = \quad l := 2 * l * h^2.$$

$h = -3$, $Par(l) =$ even $\rightsquigarrow$ $Sign(l) =$ I don't know

$h = 1$, $Par(l) =$ even $\rightsquigarrow$ $Sign(l) =$ I don't know

$\rightsquigarrow$ There is not an information flow from h into the sign of l.

Abstracting non-interference II

Consider $\alpha, \eta \in Abs(\wp(\mathbb{V}^L))$:

Narrow (abstract) non-interference $[\eta]P(\alpha)$:

$\eta(l_1) = \eta(l_2) \Rightarrow \alpha(\llbracket P \rrbracket(h_1, l_1)^L) = \alpha(\llbracket P \rrbracket(h_2, l_2)^L)$

Abstracting non-interference $(\eta)P(\alpha)$:

$\eta(l_1) = \eta(l_2) \Rightarrow \alpha(\llbracket P \rrbracket(h_1, \eta(l_1))^L) = \alpha(\llbracket P \rrbracket(h_2, \eta(l_2))^L)$

EXAMPLE II: $(\mathtt{id})P(Par)$

$$P = \quad l := l * h^2.$$

$$h = 2,\ l = 1 \ \leadsto\ Par(l) = \text{even}$$
$$h = 3,\ l = 1 \ \leadsto\ Par(l) = \text{odd}$$
$$h = n,\ l = 1 \ \leadsto\ Par(l) = Par(n)$$

Abstracting non-interference II

Consider $\alpha, \eta \in \mathit{Abs}(\wp(\mathbb{V}^{\mathtt{L}}))$:

Narrow (abstract) non-interference $[\eta]P(\alpha)$:

$\eta(l_1) = \eta(l_2) \Rightarrow \alpha([\![P]\!](h_1, l_1)^{\mathtt{L}}) = \alpha([\![P]\!](h_2, l_2)^{\mathtt{L}})$

Abstracting non-interference $(\eta)P(\alpha)$:

$\eta(l_1) = \eta(l_2) \Rightarrow \alpha([\![P]\!](h_1, \eta(l_1))^{\mathtt{L}}) = \alpha([\![P]\!](h_2, \eta(l_2))^{\mathtt{L}})$

EXAMPLE II: $(\mathtt{id})P(\mathit{Par})$

$$P = \quad l := l * h^2.$$

$$h = 2, \ l = 1 \ \rightsquigarrow \ \mathit{Par}(l) = \text{even}$$
$$h = 3, \ l = 1 \ \rightsquigarrow \ \mathit{Par}(l) = \text{odd}$$
$$h = n, \ l = 1 \ \rightsquigarrow \ \mathit{Par}(l) = \mathit{Par}(n)$$

$\rightsquigarrow$ The parity of h is flowing into l!

Abstracting non-interference II

Consider $\alpha, \eta \in \mathit{Abs}(\wp(\mathbb{V}^{\mathsf{L}}))$:

Narrow (abstract) non-interference $[\eta]P(\alpha)$:
$\eta(l_1) = \eta(l_2) \Rightarrow \alpha([\![P]\!](h_1, l_1)^{\mathsf{L}}) = \alpha([\![P]\!](h_2, l_2)^{\mathsf{L}})$

Abstracting non-interference $(\eta)P(\alpha)$:
$\eta(l_1) = \eta(l_2) \Rightarrow \alpha([\![P]\!](h_1, \eta(l_1))^{\mathsf{L}}) = \alpha([\![P]\!](h_2, \eta(l_2))^{\mathsf{L}})$

EXAMPLE II: $(\mathtt{id})P(\mathit{Par})$

$$P = \quad l := l * h^2.$$

$$h = 2,\ l = 1 \ \rightsquigarrow\ \mathit{Par}(l) = \text{even}$$
$$h = 3,\ l = 1 \ \rightsquigarrow\ \mathit{Par}(l) = \text{odd}$$
$$h = n,\ l = 1 \ \rightsquigarrow\ \mathit{Par}(l) = \mathit{Par}(n)$$

$\rightsquigarrow$ The parity of h is flowing into l!

$\Rightarrow$ We are looking for flows from any possible property of h into l!

Abstracting non-interference III

Consider $\alpha, \eta \in Abs(\wp(\mathbb{V}^{\mathsf{L}}))$ and $\phi \in Abs(\wp(\mathbb{V}^{\mathsf{H}}))$:

Abstracting non-interference $(\eta)P(\alpha)$:
$\eta(l_1) = \eta(l_2) \Rightarrow \alpha(\llbracket P \rrbracket(h_1, \eta(l_1))^{\mathsf{L}}) = \alpha(\llbracket P \rrbracket(h_2, \eta(l_2))^{\mathsf{L}})$

Abstracting non-interference III

Consider $\alpha, \eta \in \mathit{Abs}(\wp(\mathbb{V}^{\mathtt{L}}))$ and $\phi \in \mathit{Abs}(\wp(\mathbb{V}^{\mathtt{H}}))$:

Abstracting non-interference $(\eta)P(\alpha)$:

$\eta(l_1)=\eta(l_2) \Rightarrow \alpha([\![P]\!](h_1, \eta(l_1))^{\mathtt{L}}) = \alpha([\![P]\!](h_2, \eta(l_2))^{\mathtt{L}})$

Abstract non-interference $(\eta)P(\phi \rightsquigarrow\!\!\Vert \alpha)$:

$\eta(l_1)=\eta(l_2) \Rightarrow \alpha([\![P]\!](\phi(h_1), \eta(l_1))^{\mathtt{L}}) = \alpha([\![P]\!](\phi(h_2), \eta(l_2))^{\mathtt{L}})$

- No change of ϕ-equivalent $\mathtt{H}$ values may affect the α abstraction of $\mathtt{L}$ outputs.
- No deceptive interference due to $\mathtt{L}$ data;
- ...ϕ does not flow into what α can see on the output

Abstracting non-interference III

Consider $\alpha, \eta \in \mathit{Abs}(\wp(\mathbb{V}^{\mathtt{L}}))$ and $\phi \in \mathit{Abs}(\wp(\mathbb{V}^{\mathtt{H}}))$:

Abstracting non-interference $(\eta)P(\alpha)$:

$\eta(l_1) = \eta(l_2) \Rightarrow \alpha(\llbracket P \rrbracket(h_1, \eta(l_1))^{\mathtt{L}}) = \alpha(\llbracket P \rrbracket(h_2, \eta(l_2))^{\mathtt{L}})$

Abstract non-interference $(\eta)P(\phi \rightsquigarrow\!\!\| \alpha)$:

$\eta(l_1) = \eta(l_2) \Rightarrow \alpha(\llbracket P \rrbracket(\phi(h_1), \eta(l_1))^{\mathtt{L}}) = \alpha(\llbracket P \rrbracket(\phi(h_2), \eta(l_2))^{\mathtt{L}})$

EXAMPLE: $(\mathtt{id})P(\mathit{Sign} \rightsquigarrow\!\!\| \mathit{Par})$

$$P = \quad l := l * h^2.$$

$$\mathit{Sign}(h) = +,\ l = 1 \ \rightsquigarrow\ \mathit{Par}(l) = \text{I don't know}$$
$$\mathit{Sign}(h) = -,\ l = 1 \ \rightsquigarrow\ \mathit{Par}(l) = \text{I don't know}$$

Abstracting non-interference III

Consider $\alpha, \eta \in Abs(\wp(\mathbb{V}^L))$ and $\phi \in Abs(\wp(\mathbb{V}^H))$:

Abstracting non-interference $(\eta)P(\alpha)$:

$\eta(l_1)=\eta(l_2) \Rightarrow \alpha(\llbracket P \rrbracket(h_1,\eta(l_1))^L)=\alpha(\llbracket P \rrbracket(h_2,\eta(l_2))^L)$

Abstract non-interference $(\eta)P(\phi \rightsquigarrow\!\!\!\parallel \alpha)$:

$\eta(l_1)=\eta(l_2) \Rightarrow \alpha(\llbracket P \rrbracket(\phi(h_1),\eta(l_1))^L)=\alpha(\llbracket P \rrbracket(\phi(h_2),\eta(l_2))^L)$

EXAMPLE: $(\mathtt{id})P(\mathit{Sign} \rightsquigarrow\!\!\!\parallel \mathit{Par})$

$$P = \quad l := l * h^2.$$

$\mathit{Sign}(h) = +,\ l = 1 \ \rightsquigarrow\ \mathit{Par}(l) =$ I don't know

$\mathit{Sign}(h) = -,\ l = 1 \ \rightsquigarrow\ \mathit{Par}(l) =$ I don't know

$\rightsquigarrow$ There is not an information flow from the sign of h into the parity of l.

Basic properties

- $[\eta]P(\top)$
- $[\eta]P(\alpha) \;\Rightarrow\; \forall \beta \sqsubseteq \eta.\ [\beta]P(\alpha)$
- $[\eta]P(\alpha) \;\Rightarrow\; \forall \beta \sqsupseteq \alpha.\ [\eta]P(\beta)$
- $\forall i\,.\,[\eta]P(\alpha_i) \;\Rightarrow\; [\eta]P(\bigsqcap_{i \in I} \alpha_i)$

Basic properties

- $[\eta]P(\top)$
- $[\eta]P(\alpha) \;\Rightarrow\; \forall \beta \sqsubseteq \eta.\ [\beta]P(\alpha)$
- $[\eta]P(\alpha) \;\Rightarrow\; \forall \beta \sqsupseteq \alpha.\ [\eta]P(\beta)$
- $\forall i\,.\,[\eta]P(\alpha_i) \;\Rightarrow\; [\eta]P(\bigsqcap_{i \in I} \alpha_i)$
- $(\eta)P(\phi \rightsquigarrow\!\!\!\Vert \top)$
- $(\eta)P(\phi \rightsquigarrow\!\!\!\Vert \alpha) \;\Rightarrow\; \forall \beta \sqsupseteq \alpha.\ (\eta)P(\phi \rightsquigarrow\!\!\!\Vert \beta)$
- $\forall i\,.\,(\eta)P(\phi \rightsquigarrow\!\!\!\Vert \alpha_i) \;\Rightarrow\; (\eta)P(\phi \rightsquigarrow\!\!\!\Vert \bigsqcap_{i \in I} \alpha_i)$

Basic properties

- $[\eta]P(\top)$
- $[\eta]P(\alpha) \;\Rightarrow\; \forall \beta \sqsubseteq \eta.\ [\beta]P(\alpha)$
- $[\eta]P(\alpha) \;\Rightarrow\; \forall \beta \sqsupseteq \alpha.\ [\eta]P(\beta)$
- $\forall i\,.\,[\eta]P(\alpha_i) \;\Rightarrow\; [\eta]P(\bigsqcap_{i \in I} \alpha_i)$
- $(\eta)P(\phi \rightsquigarrow\!\!\| \top)$
- $(\eta)P(\phi \rightsquigarrow\!\!\| \alpha) \;\Rightarrow\; \forall \beta \sqsupseteq \alpha.\ (\eta)P(\phi \rightsquigarrow\!\!\| \beta)$
- $\forall i\,.\,(\eta)P(\phi \rightsquigarrow\!\!\| \alpha_i) \;\Rightarrow\; (\eta)P(\phi \rightsquigarrow\!\!\| \bigsqcap_{i \in I} \alpha_i)$

Standard non-interference: $[\mathtt{id}]P(\mathtt{id}) = (\mathtt{id})P(\mathtt{id} \rightsquigarrow\!\!\| \mathtt{id})$

Basic properties

- $[\eta]P(\top)$
- $[\eta]P(\alpha) \;\Rightarrow\; \forall \beta \sqsubseteq \eta.\ [\beta]P(\alpha)$
- $[\eta]P(\alpha) \;\Rightarrow\; \forall \beta \sqsupseteq \alpha.\ [\eta]P(\beta)$
- $\forall i\,.\,[\eta]P(\alpha_i) \;\Rightarrow\; [\eta]P(\bigsqcap_{i \in I} \alpha_i)$
- $(\eta)P(\phi \rightsquigarrow\!\!\|\, \top)$
- $(\eta)P(\phi \rightsquigarrow\!\!\|\, \alpha) \;\Rightarrow\; \forall \beta \sqsupseteq \alpha.\ (\eta)P(\phi \rightsquigarrow\!\!\|\, \beta)$
- $\forall i\,.\,(\eta)P(\phi \rightsquigarrow\!\!\|\, \alpha_i) \;\Rightarrow\; (\eta)P(\phi \rightsquigarrow\!\!\|\, \bigsqcap_{i \in I} \alpha_i)$

$$\boxed{[\mathtt{id}]P(\mathtt{id}) \Rightarrow (\eta)P(\mathtt{id} \rightsquigarrow\!\!\|\, \alpha) \Rightarrow (\eta)P(\phi \rightsquigarrow\!\!\|\, \alpha)}$$

Basic properties

- $[\eta]P(\top)$
- $[\eta]P(\alpha) \Rightarrow \forall \beta \sqsubseteq \eta.\ [\beta]P(\alpha)$
- $[\eta]P(\alpha) \Rightarrow \forall \beta \sqsupseteq \alpha.\ [\eta]P(\beta)$
- $\forall i\,.\,[\eta]P(\alpha_i) \Rightarrow [\eta]P(\bigsqcap_{i \in I} \alpha_i)$
- $(\eta)P(\phi \rightsquigarrow\!\!\| \top)$
- $(\eta)P(\phi \rightsquigarrow\!\!\| \alpha) \Rightarrow \forall \beta \sqsupseteq \alpha.\ (\eta)P(\phi \rightsquigarrow\!\!\| \beta)$
- $\forall i\,.\,(\eta)P(\phi \rightsquigarrow\!\!\| \alpha_i) \Rightarrow (\eta)P(\phi \rightsquigarrow\!\!\| \bigsqcap_{i \in I} \alpha_i)$

$$\boxed{[\eta]P(\alpha) \Rightarrow (\eta)P(\mathtt{id} \rightsquigarrow\!\!\| \alpha) \Rightarrow (\eta)P(\phi \rightsquigarrow\!\!\| \alpha)}$$

Basic properties

- $[\eta]P(\top)$
- $[\eta]P(\alpha) \;\Rightarrow\; \forall\beta \sqsubseteq \eta.\ [\beta]P(\alpha)$
- $[\eta]P(\alpha) \;\Rightarrow\; \forall\beta \sqsupseteq \alpha.\ [\eta]P(\beta)$
- $\forall i\,.\,[\eta]P(\alpha_i) \;\Rightarrow\; [\eta]P(\bigsqcap_{i\in I} \alpha_i)$
- $(\eta)P(\phi \rightsquigarrow\!\!\| \top)$
- $(\eta)P(\phi \rightsquigarrow\!\!\| \alpha) \;\Rightarrow\; \forall\beta \sqsupseteq \alpha.\ (\eta)P(\phi \rightsquigarrow\!\!\| \beta)$
- $\forall i\,.\,(\eta)P(\phi \rightsquigarrow\!\!\| \alpha_i) \;\Rightarrow\; (\eta)P(\phi \rightsquigarrow\!\!\| \bigsqcap_{i\in I} \alpha_i)$

$[\mathtt{id}]P(\mathtt{id}) \not\Rightarrow [\eta]P(\alpha)$ due to deceptive flows

Deriving output attackers

Abstract interpretation provides advanced methods for designing abstractions (refinement, simplification, compression ...) [Giacobazzi & Ranzato '97]

Designing abstractions = designing attackers

Deriving output attackers

Abstract interpretation provides advanced methods for designing abstractions (refinement, simplification, compression ...) [Giacobazzi & Ranzato '97]

Designing abstractions = designing attackers

$$\Downarrow$$

- Characterize the most concrete α such that $(\eta)P(\phi \rightsquigarrow\!\!\!\!\!\rightarrow\!\!\| \alpha)$

  [The most powerful *output* attacker]

Deriving output attackers

The following theorems hold:

- Consider $\eta \in \mathit{Abs}(\wp(\mathbb{V}^{\mathsf{L}}))$:
  We characterize the function $\lambda\eta.\, [\eta][\![P]\!](\mathtt{id})$ whose result is
  $\bigsqcap \left\{ \beta \,\middle|\, [\eta]P(\beta) \right\}$.

Deriving output attackers

The following theorems hold:

- Consider $\eta \in \mathit{Abs}(\wp(\mathbb{V}^{\mathtt{L}}))$:
  We characterize the function $\lambda\eta.\ [\eta][\![P]\!](\mathtt{id})$ whose result is
  $\bigsqcap\left\{\ \beta \;\middle|\; [\eta]P(\beta)\ \right\}$.
- Consider $\eta \in \mathit{Abs}(\wp(\mathbb{V}^{\mathtt{L}}))$ and $\phi \in \mathit{Abs}(\wp(\mathbb{V}^{\mathtt{H}}))$:
  We characterize the function $\lambda\eta.\ (\eta)[\![P]\!](\phi \rightsquigarrow\!\!\| \mathtt{id})$ whose result is
  $\bigsqcap\left\{\ \beta \;\middle|\; (\eta)P(\phi \rightsquigarrow\!\!\| \beta)\ \right\}$.

Deriving output attackers

The following theorems hold:

- Consider $\eta \in Abs(\wp(\mathbb{V}^{L}))$:
  We characterize the function $\lambda\eta.\ [\eta][\![P]\!](\mathtt{id})$ whose result is
  $\bigsqcap \left\{ \beta \;\middle|\; [\eta]P(\beta) \right\}$.
- Consider $\eta \in Abs(\wp(\mathbb{V}^{L}))$ and $\phi \in Abs(\wp(\mathbb{V}^{H}))$:
  We characterize the function $\lambda\eta.\ (\eta)[\![P]\!](\phi \rightsquigarrow\!\!\| \mathtt{id})$ whose result is
  $\bigsqcap \left\{ \beta \;\middle|\; (\eta)P(\phi \rightsquigarrow\!\!\| \beta) \right\}$.

$\Rightarrow$ This would provide a certificate for security with a fixed input observation.

Deriving canonical attackers

Abstract interpretation provides advanced methods for designing abstractions (refinement, simplification, compression ...) [Giacobazzi & Ranzato '97]

Transforming abstractions = transforming attackers

Deriving canonical attackers

Abstract interpretation provides advanced methods for designing abstractions (refinement, simplification, compression ...) [Giacobazzi & Ranzato '97]

Transforming abstractions = transforming attackers

$\Downarrow$

- Characterize the most concrete δ such that $(\delta)P(\phi \rightsquigarrow\!\!\!\rrbracket \delta)$

  [The most powerful *canonical* attacker]

$\Rightarrow$ This would provide a certificate for security.

Deriving canonical attackers

- $\lambda X.\ [X][\![P]\!](\mathtt{id})$ is monotone on $\mathit{Abs}(\wp(\mathbb{V}^{\mathsf{L}}))$.
- $[\alpha]P(\alpha) \;\Leftrightarrow\; \alpha = [\alpha][\![P]\!](\mathtt{id})$.
- $\mathit{lfp}(\lambda X.\ [X][\![P]\!](\mathtt{id}))$ is the most concrete secure attacker for $\mathtt{P}$ for narrow abstract non-interference.

Deriving canonical attackers

- $\lambda X.\ [X][\![P]\!](\mathtt{id})$ is monotone on $\mathit{Abs}(\wp(\mathbb{V}^{\mathsf{L}}))$.
- $[\alpha]P(\alpha) \;\Leftrightarrow\; \alpha = [\alpha][\![P]\!](\mathtt{id})$.
- $\mathit{lfp}(\lambda X.\ [X][\![P]\!](\mathtt{id}))$ is the most concrete secure attacker for $\mathtt{P}$ for narrow abstract non-interference.

- $(\alpha)P(\phi \rightsquigarrow\!\!\| \alpha) \;\Leftrightarrow\; \alpha = (\alpha)[\![P]\!](\phi \rightsquigarrow\!\!\| \mathtt{id})$
- $\lambda X.\ (X)[\![P]\!](\phi \rightsquigarrow\!\!\| \mathtt{id}) \sqcup X$ is extensive on $\mathit{Abs}(\wp(\mathbb{V}^{\mathsf{L}}))$.
- $\mathit{fix}(\lambda X.\ (X)[\![P]\!](\phi \rightsquigarrow\!\!\| \mathtt{id}) \sqcup X)$ is a secure attacker for $\mathtt{P}$ for abstract non-interference.

Deriving canonical attackers

EXAMPLE:

$$P = \textbf{ while } h \textbf{ do } (l := l * 2;\ h := h - 1)$$

.... we derive a secure attacker $\pi = \Upsilon(\left\{ n\{2\}^{\mathbb{N}} \,\middle|\, n \in 2\mathbb{N}+1 \right\} \cup \{\{0\}\})$:

$$(\pi)[\![P]\!](\mathtt{id} \rightsquigarrow\!\!\| \pi)$$

$$h = 0,\ \pi(l) = 3\{2\}^{\mathbb{N}} \rightsquigarrow \pi(l) = 3\{2\}^{\mathbb{N}}$$
$$h = 2,\ \pi(l) = 3\{2\}^{\mathbb{N}} \rightsquigarrow \pi(l) = 3\{2\}^{\mathbb{N}}$$

$\rightsquigarrow$ In the program l is always multiplied by 2!

Abstract robust declassification

Consider a program $\mathtt{P}$ and its finite computations.

A passive attacker may be able to learn some information by observing the system but, by assumption, that information leakage is allowed by the security policy.

[Zdancewic and Myers 2001]

Abstract robust declassification

Consider a program P and its finite computations.

A passive attacker may be able to learn some information by observing the system but, by assumption, that information leakage is allowed by the security policy.
[Zdancewic and Myers 2001]

- We want to characterize the most concrete *flow-irredundant* property such that $(\eta)P(\phi \rightsquigarrow \alpha)$

  [The maximal amount of information disclosed]

$\Rightarrow$ This would provide a certificate for disclosed secrets.

Abstract robust declassification

Consider the program

$$P = \quad l := l + (h \ \mathtt{mod}\ 3)$$

The transition system is such that $< h, l > \mapsto < h, l + (h \ \mathtt{mod}\ 3) >$.

Consider $\eta(\wp(\mathbb{Z})) = \{\mathbb{Z}, [2, 4], [5, 8], \{5\}, \varnothing\}$ and $\alpha = \mathtt{id}$.

Abstract robust declassification

Consider the program

$$P = \quad l := l + (h \ \mathtt{mod}\ 3)$$

The transition system is such that $< h, l > \mapsto < h, l + (h \ \mathtt{mod}\ 3) >$.
Consider $\eta(\wp(\mathbb{Z})) = \{\mathbb{Z}, [2,4], [5,8], \{5\}, \varnothing\}$ and $\alpha = \mathtt{id}$.

The flow is revealed when h_1 and h_2 differs in the values $h_1 \ \mathtt{mod}\ 3$ and $h_2 \ \mathtt{mod}\ 3$

$$\Rightarrow \phi = \Upsilon(\{3\mathbb{Z}, 3\mathbb{Z}+1, 3\mathbb{Z}+2\})$$

is the maximal amount of information disclosed!

Discussion

- We map security of programs into the lattice of abstract interpretations:
  - systematic methods for designing attackers and certificates
  - security degrees compared in the lattice
  - checking abstract non-interference by static program analysis

Discussion

- We map security of programs into the lattice of abstract interpretations:
  - systematic methods for designing attackers and certificates
  - security degrees compared in the lattice
  - checking abstract non-interference by static program analysis
- Abstract non-interference is a semantics property
  - the method is language independent (as any abstract interpretation)
  - refined semantics may refine security:
    covert channels (termination, non-determinism, synchronization, probabilistic, etc...) is a matter of semantics!

Discussion

- We map security of programs into the lattice of abstract interpretations:
  - systematic methods for designing attackers and certificates
  - security degrees compared in the lattice
  - checking abstract non-interference by static program analysis
- Abstract non-interference is a semantics property
  - the method is language independent (as any abstract interpretation)
  - refined semantics may refine security:
    covert channels (termination, non-determinism, synchronization, probabilistic, etc...) is a matter of semantics!
- How far is any practical application?
  - program slicing may help in checking program secrecy!
  - the common abstraction which is not disclosed for all program slices will not be disclosed by the whole program...