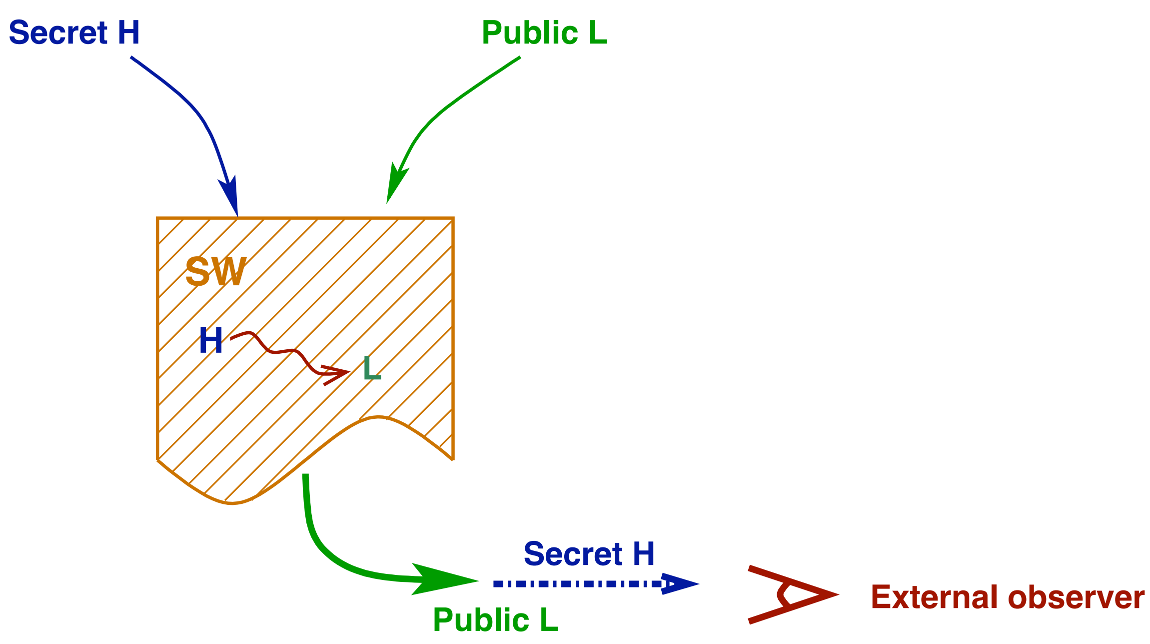
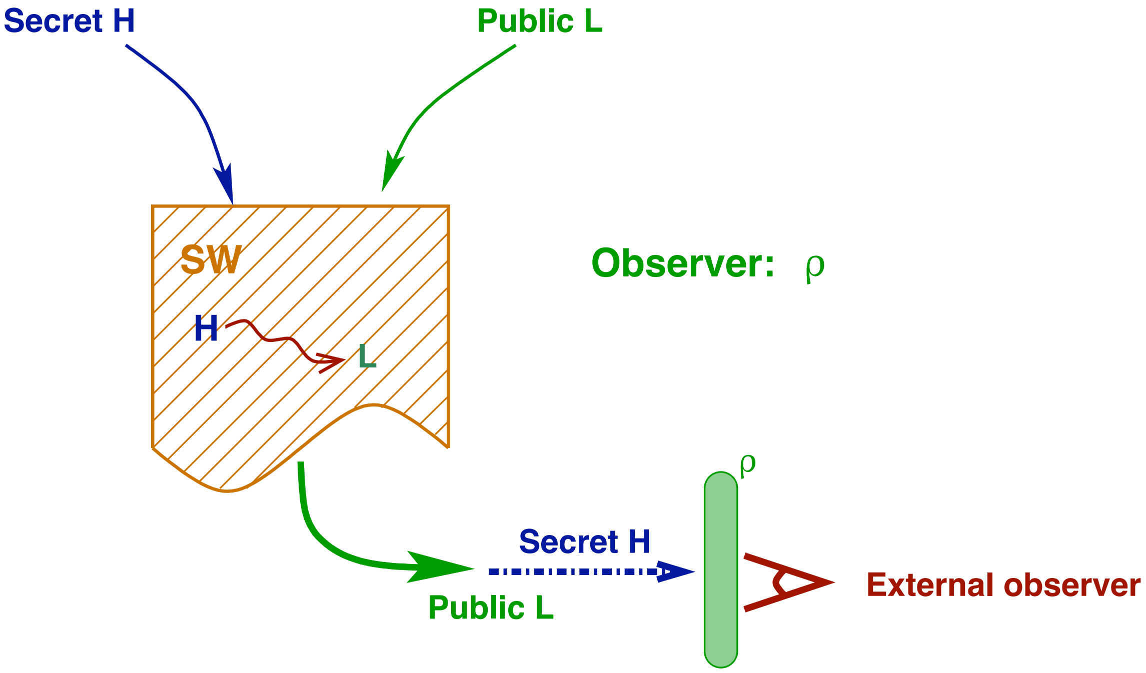
Language-based Security
Abstract Non-Interference

Isabella Mastroeni

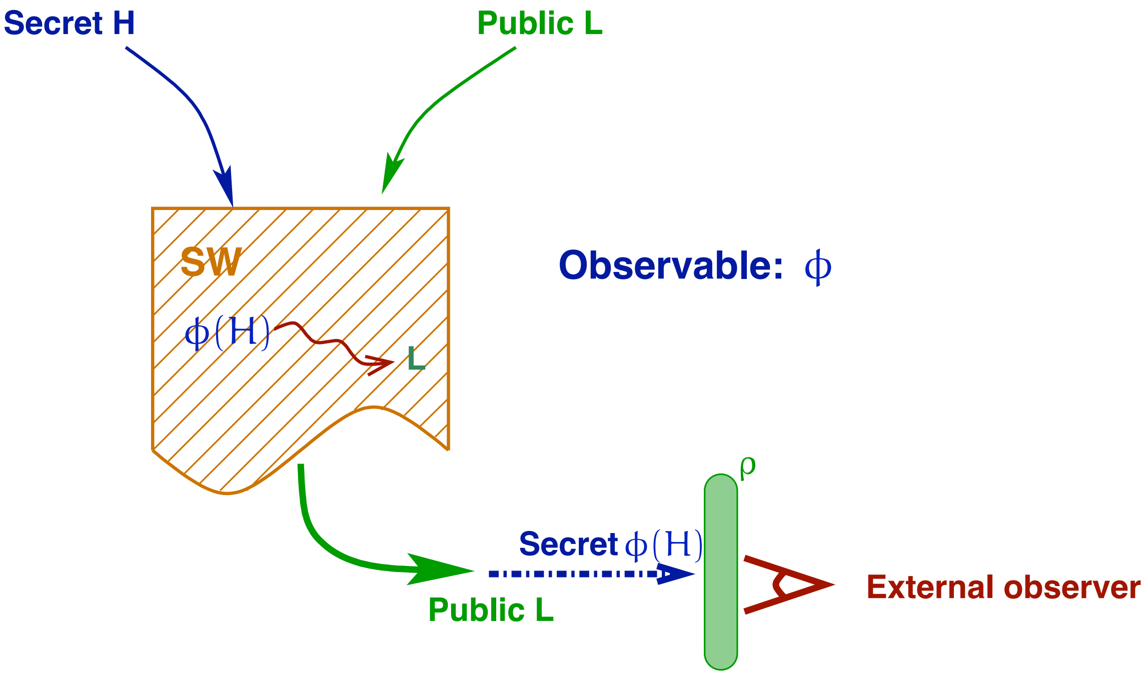
Our Idea



Our Idea

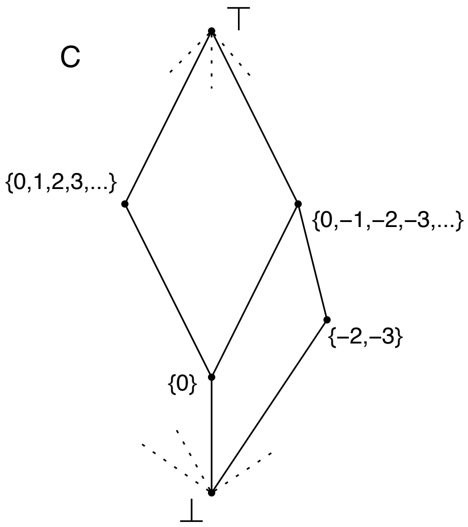


Our Idea



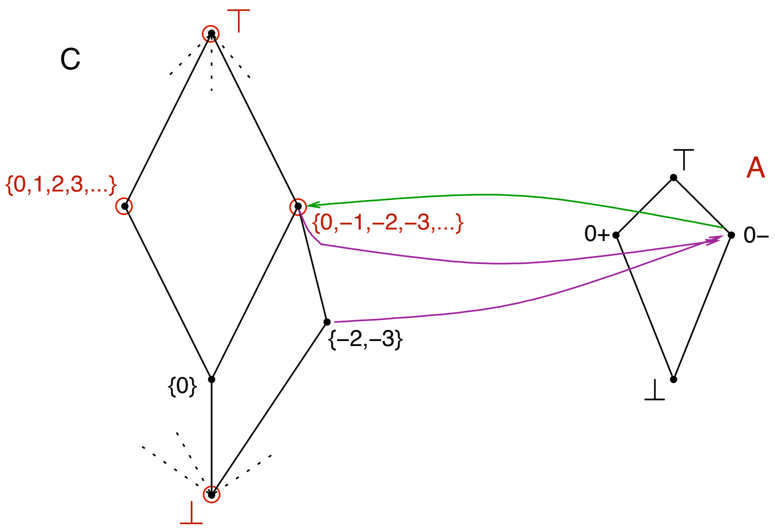
Abstract Interpretation

Consider $C = \wp(\mathbb{Z})$: [Cousot & Cousot'77]



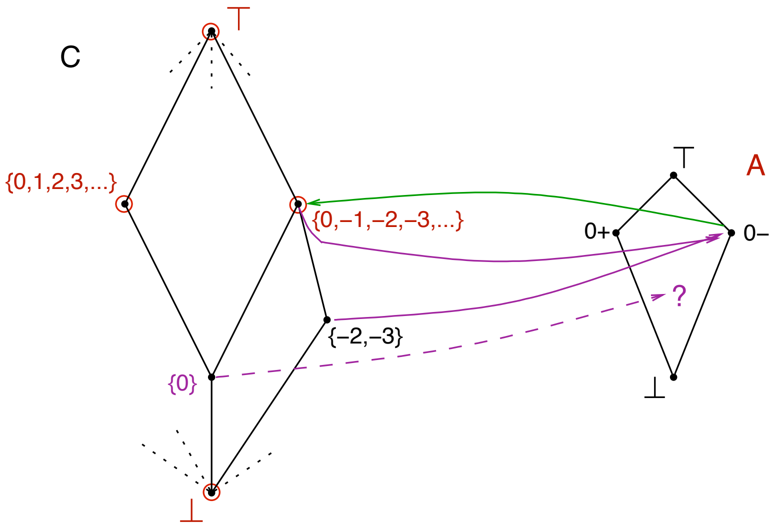
Abstract Interpretation

Consider $C = \wp(\mathbb{Z})$: [Cousot & Cousot'77]



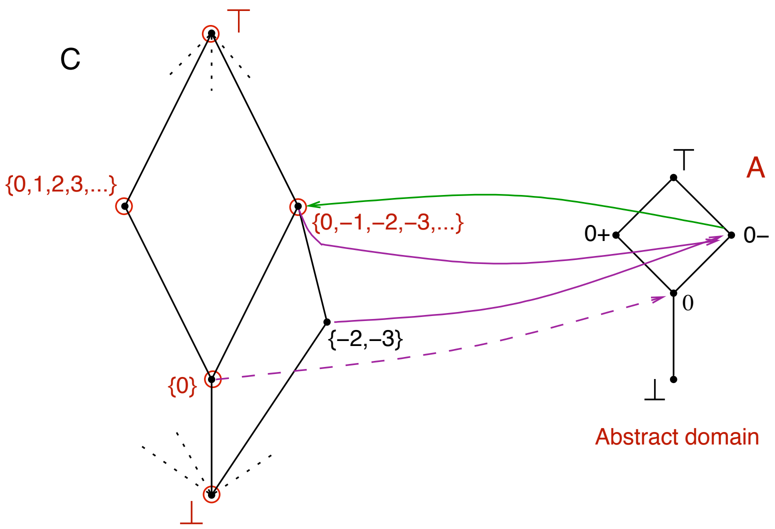
Abstract Interpretation

Consider $\mathrm{C} = \wp(\mathbb{Z})$: [Cousot & Cousot'77]



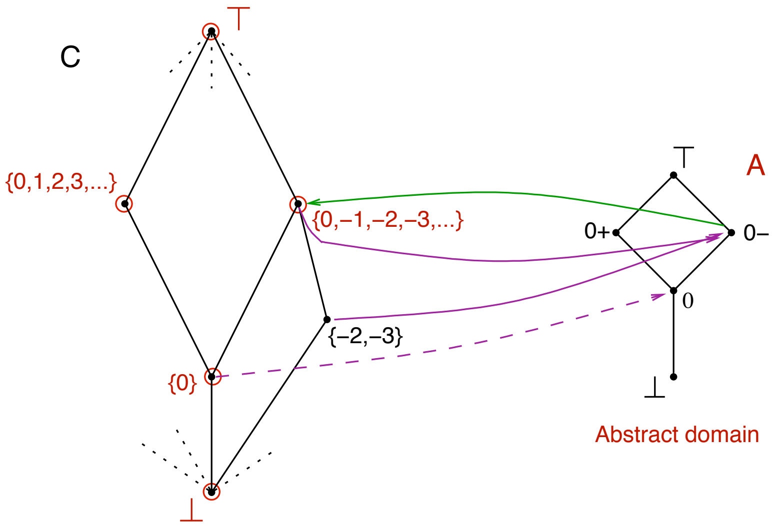
Abstract Interpretation

Consider $\mathrm{C} = \wp(\mathbb{Z})$: [Cousot & Cousot'77]



Abstract Interpretation

Consider $C = \wp(\mathbb{Z})$: [Cousot & Cousot'77]

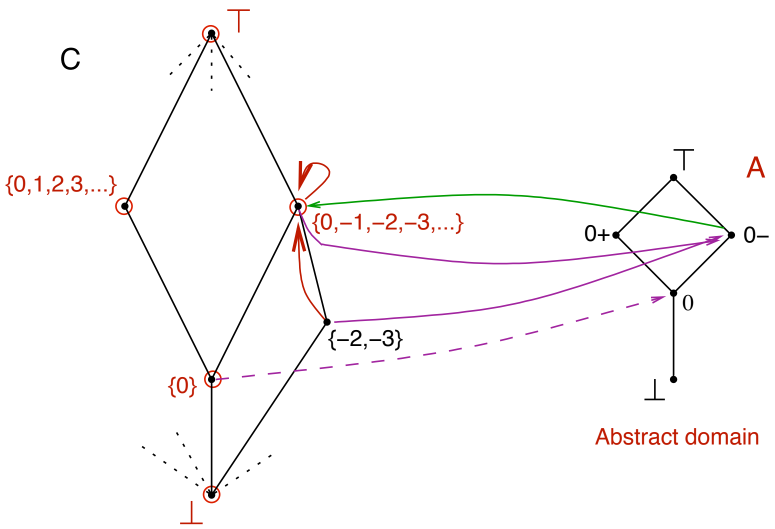


α, γ monotone, $\alpha(x) \leq y \Leftrightarrow x \leq \gamma(y)$, $\alpha\gamma(y) = y$, $\gamma\alpha(x) \geq x$

$\gamma(x) = \bigvee \left\{ y \,\middle|\, \alpha(y) \leq x \right\} \stackrel{\text{def}}{=} \alpha^{+}(x)$ and $\alpha(x) = \bigwedge \left\{ y \,\middle|\, x \leq \gamma(y) \right\} \stackrel{\text{def}}{=} \gamma^{-}(x)$

Abstract Interpretation

Consider $C = \wp(\mathbb{Z})$: [Cousot & Cousot'77]



$\gamma\alpha$ monotone, $\gamma\alpha(x) \geq x$, $\gamma\alpha(\gamma\alpha(x)) = \gamma\alpha(x)$

$\Rightarrow$ Upper closure operator.

Abstract Interpretation

Consider the complete lattice $< C, \leq, \wedge, \vee, \bot, \top >$, $A_i \in uco(C)$

Lattice of Abstract Domains $\equiv$ Lattice *uco*

$A \equiv \rho(C)$

$< uco(C), \sqsubseteq, \sqcap, \sqcup, \lambda x.\ \top, \lambda x.\ x >$

Abstract Interpretation

Consider the complete lattice $< C, \leq, \wedge, \vee, \bot, \top >$, $A_i \in uco(C)$

Lattice of Abstract Domains $\equiv$ Lattice *uco*

$A \equiv \rho(C)$

$< uco(C), \sqsubseteq, \sqcap, \sqcup, \lambda x.\ \top, \lambda x.\ x >$

$$A_1 \sqsubseteq A_2 \quad \Leftrightarrow \quad A_2 \subseteq A_1$$

Abstract Interpretation

Consider the complete lattice $< C, \leq, \wedge, \vee, \perp, \top >$, $A_i \in uco(C)$

Lattice of Abstract Domains $\equiv$ Lattice *uco*
A $\equiv \rho(C)$
$< uco(C), \sqsubseteq, \sqcap, \sqcup, \lambda x.\, \top, \lambda x.\, x >$

$$A_1 \sqsubseteq A_2 \quad \Leftrightarrow \quad A_2 \subseteq A_1$$

$$\sqcap_i A_i = \mathcal{M}(\cup_i A_i)$$

Abstract Interpretation

Consider the complete lattice $< C, \leq, \wedge, \vee, \perp, \top >$, $A_i \in uco(C)$

Lattice of Abstract Domains $\equiv$ Lattice *uco*
A $\equiv \rho(C)$
$< uco(C), \sqsubseteq, \sqcap, \sqcup, \lambda x.\, \top, \lambda x.\, x >$

$$A_1 \sqsubseteq A_2 \quad \Leftrightarrow \quad A_2 \subseteq A_1$$
$$\sqcap_i A_i = \mathcal{M}(\cup_i A_i)$$

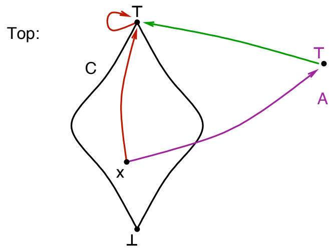
$$\sqcup_i A_i = \cap_i A_i$$

Abstract Interpretation

Consider the complete lattice $< C, \leq, \wedge, \vee, \bot, \top >$, $A_i \in uco(C)$

Lattice of Abstract Domains $\equiv$ Lattice *uco*
$A \equiv \rho(C)$
$< uco(C), \sqsubseteq, \sqcap, \sqcup, \lambda x.\ \top, \lambda x.\ x >$

$$A_1 \sqsubseteq A_2 \quad \Leftrightarrow \quad A_2 \subseteq A_1$$
$$\sqcap_i A_i = \mathcal{M}(\cup_i A_i)$$
$$\sqcup_i A_i = \cap_i A_i$$

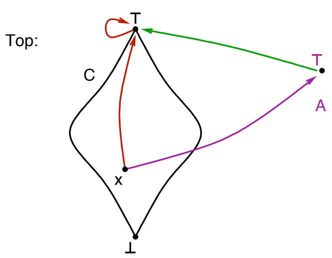


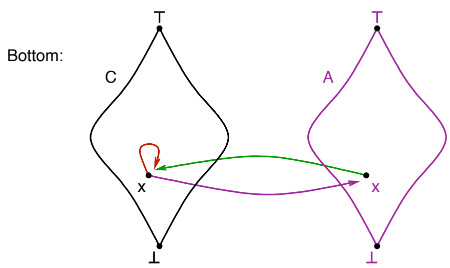
Abstract Interpretation

Consider the complete lattice $< C, \leq, \wedge, \vee, \bot, \top >$, $A_i \in uco(C)$

Lattice of Abstract Domains $\equiv$ Lattice *uco*
$A \equiv \rho(C)$
$< uco(C), \sqsubseteq, \sqcap, \sqcup, \lambda x.\ \top, \lambda x.\ x >$

$$A_1 \sqsubseteq A_2 \quad \Leftrightarrow \quad A_2 \subseteq A_1$$
$$\sqcap_i A_i = \mathcal{M}(\cup_i A_i)$$
$$\sqcup_i A_i = \cap_i A_i$$

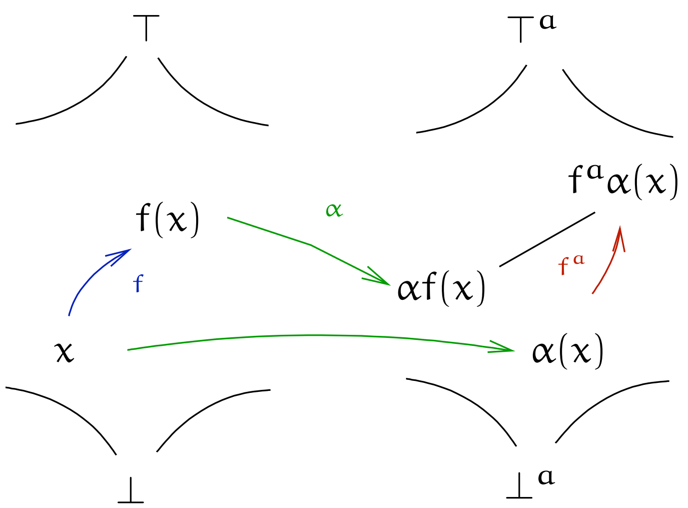




Abstract domain completeness

Let $< A, \alpha, \gamma, C >$ a galois insertion. [Cousot & Cousot '77,'79]
$f : C \longrightarrow C$, $f^a = \alpha \circ f \circ \gamma : A \longrightarrow A$ (b.c.a. of f) and $\rho = \gamma \circ \alpha$

ρ correct for f



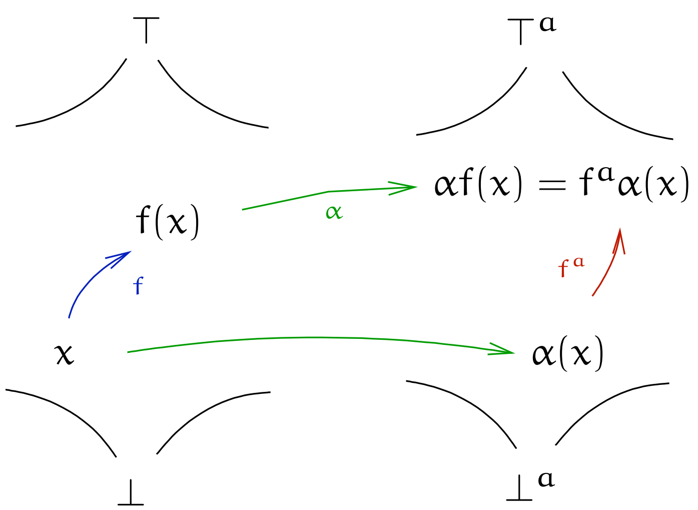
Abstract domain completeness

Let $< A, \alpha, \gamma, C >$ a galois insertion. [Cousot & Cousot '77,'79]
$f : C \longrightarrow C$, $f^a = \alpha \circ f \circ \gamma : A \longrightarrow A$ (b.c.a. of f) and $\rho = \gamma \circ \alpha$

ρ complete for f

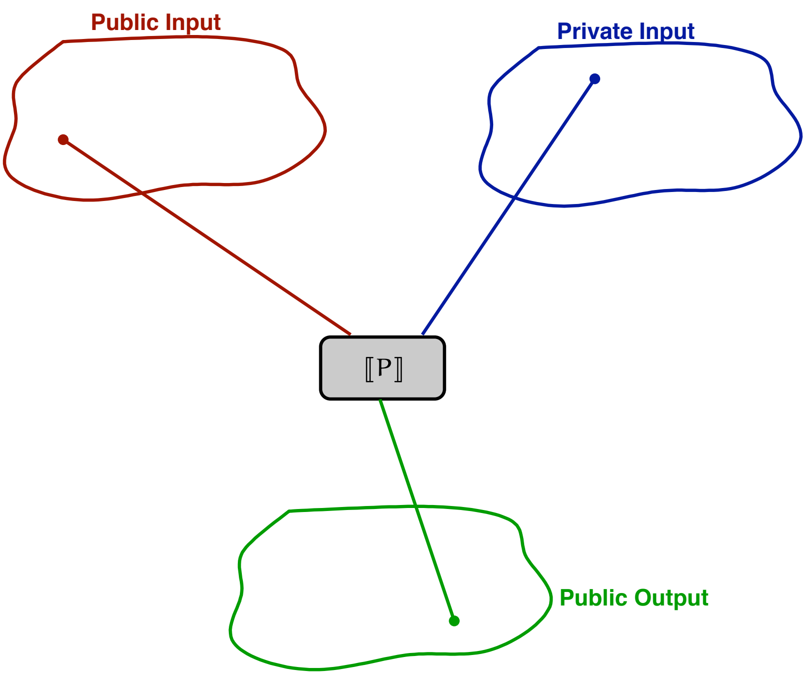
|||

$\rho f \rho = \rho f$



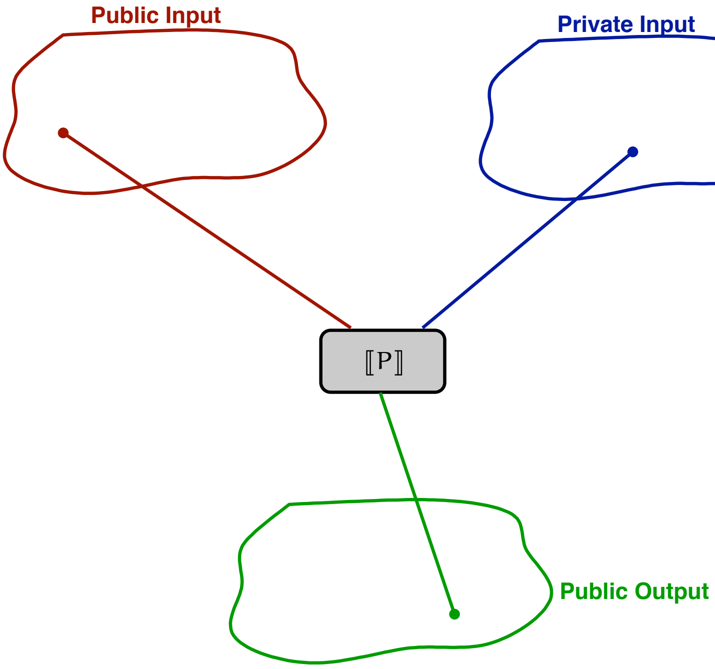
DEFINING ABSTRACT NON-INTERFERENCE

Standard Non-Interference



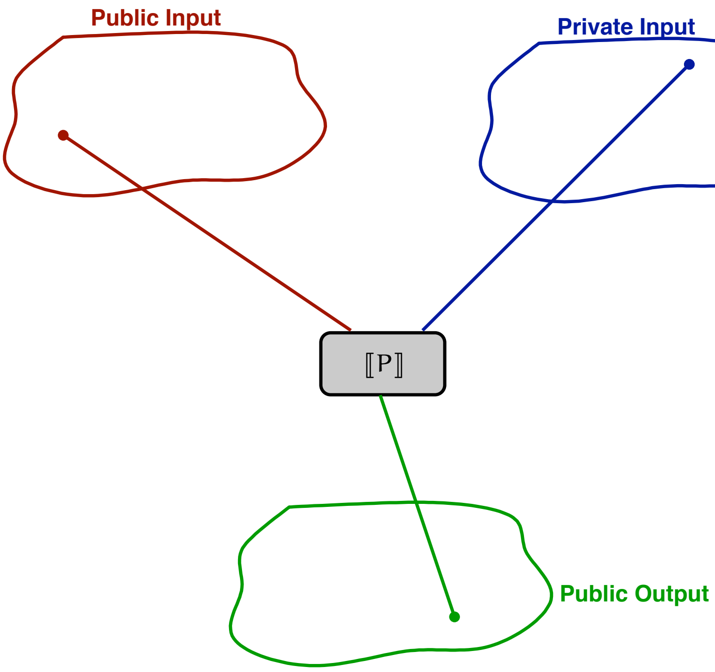
$$\forall l : \mathtt{L}, \forall h_1, h_2 : \mathtt{H}.\ [\![P]\!](h_1, l)^{\mathtt{L}} = [\![P]\!](h_2, l)^{\mathtt{L}}$$

Standard Non-Interference



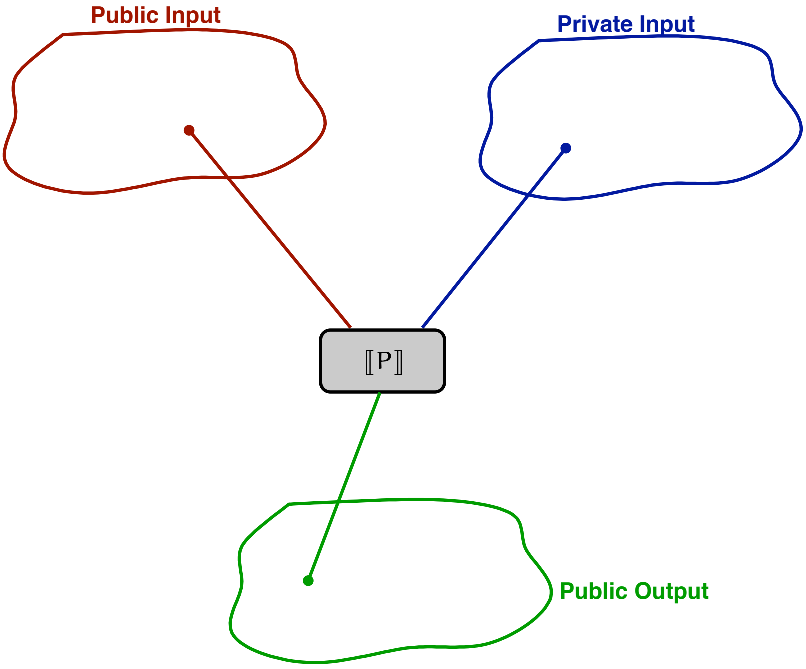
$$\forall l : \mathtt{L}, \forall h_1, h_2 : \mathtt{H}.\ [\![P]\!](h_1, l)^{\mathtt{L}} = [\![P]\!](h_2, l)^{\mathtt{L}}$$

Standard Non-Interference



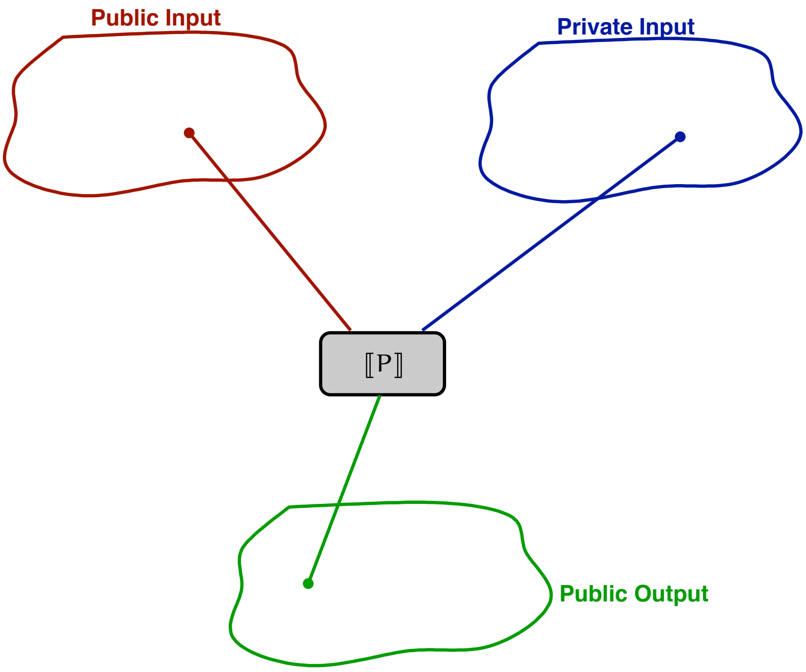
$$\forall l : \mathtt{L}, \forall h_1, h_2 : \mathtt{H}.\ [\![P]\!](h_1, l)^{\mathtt{L}} = [\![P]\!](h_2, l)^{\mathtt{L}}$$

Standard Non-Interference



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

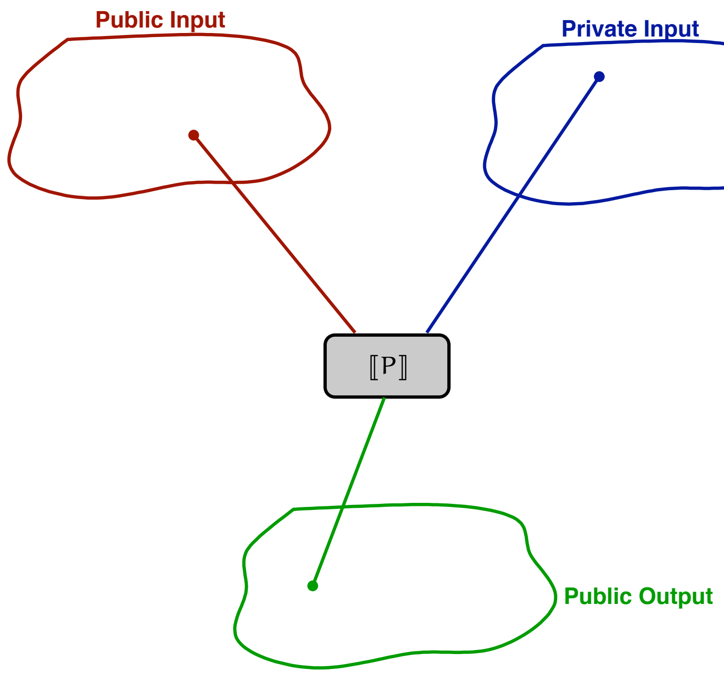
Standard Non-Interference



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

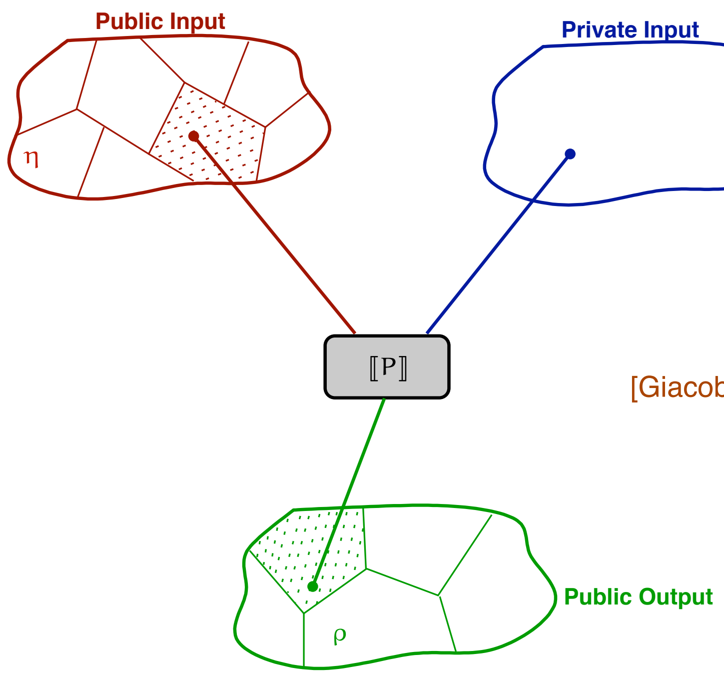
Language-based Security: Abstract Non-Interferece – p.7/32

Standard Non-Interference



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

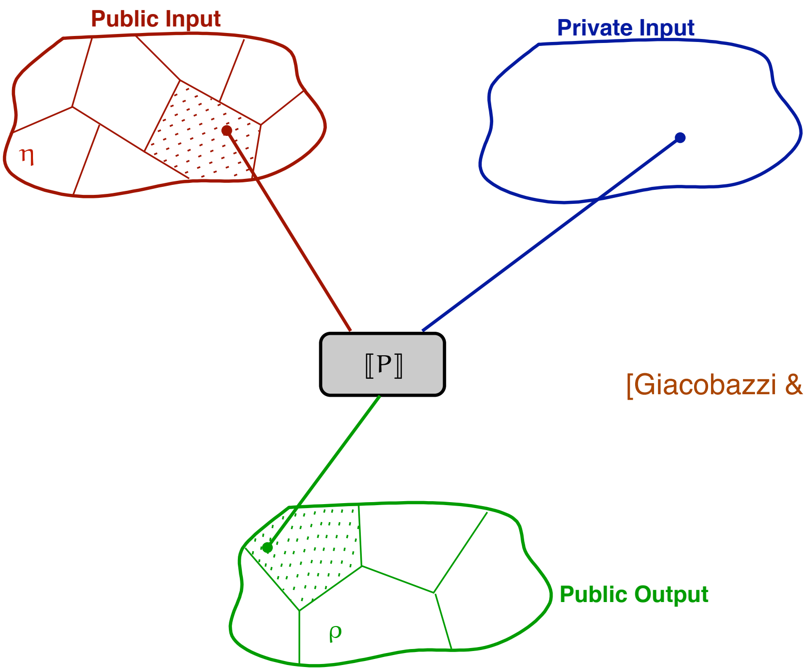
Abstract Non-Interference (Narrow)



[Giacobazzi & Mastroeni ’04]

$$\rho, \eta \in Abs(\wp(\mathbb{V}^{\mathtt{L}})) \colon [\eta]P(\rho) \colon \eta(l_1) = \eta(l_2) \Rightarrow \rho([\![P]\!](h_1, l_1)^{\mathtt{L}}) = \rho([\![P]\!](h_2, l_2)^{\mathtt{L}})$$

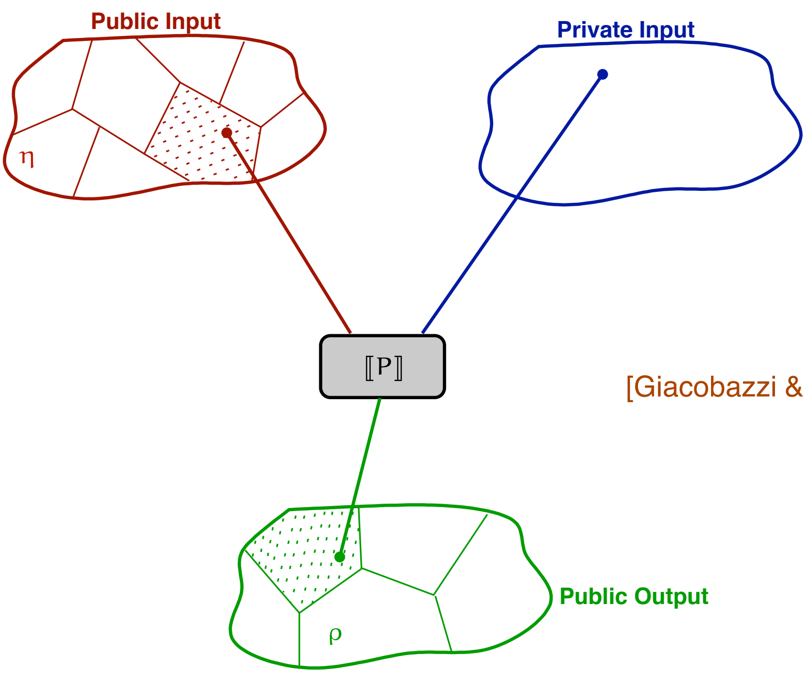
Abstract Non-Interference (Narrow)



[Giacobazzi & Mastroeni '04]

$\rho, \eta \in Abs(\wp(\mathbb{V}^{L}))$: $[\eta]P(\rho)$: $\eta(l_1) = \eta(l_2) \Rightarrow \rho(\llbracket P \rrbracket(h_1, l_1)^{L}) = \rho(\llbracket P \rrbracket(h_2, l_2)^{L})$

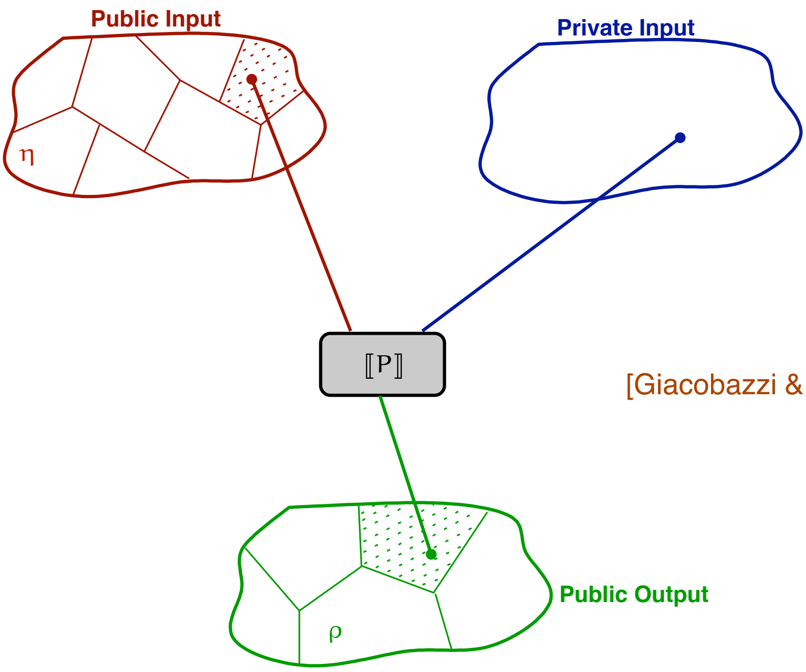
Abstract Non-Interference (Narrow)



[Giacobazzi & Mastroeni '04]

$\rho, \eta \in Abs(\wp(\mathbb{V}^{L}))$: $[\eta]P(\rho)$: $\eta(l_1) = \eta(l_2) \Rightarrow \rho(\llbracket P \rrbracket(h_1, l_1)^{L}) = \rho(\llbracket P \rrbracket(h_2, l_2)^{L})$

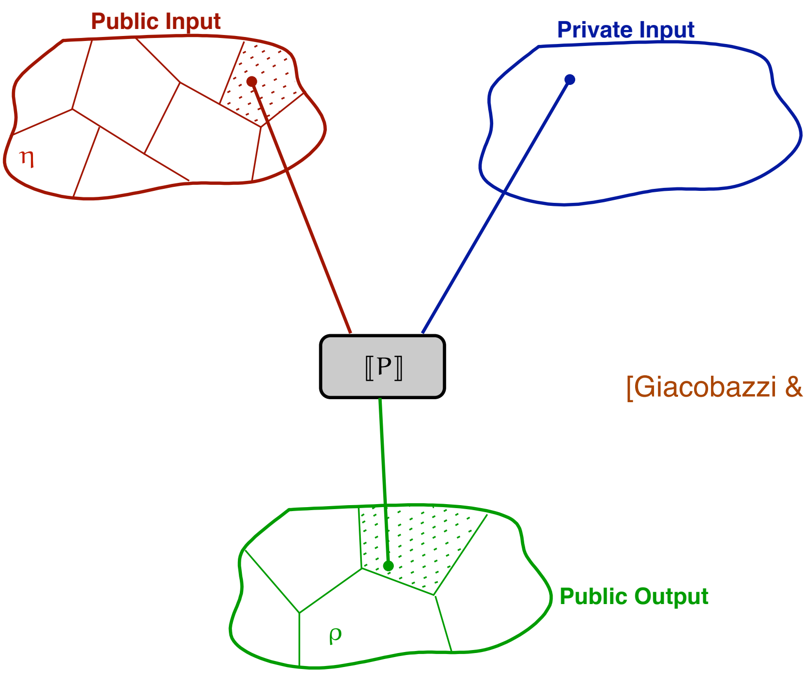
Abstract Non-Interference (Narrow)



[Giacobazzi & Mastroeni '04]

$\rho, \eta \in \mathit{Abs}(\wp(\mathbb{V}^{L}))$: $[\eta]P(\rho)$: $\eta(l_1) = \eta(l_2) \Rightarrow \rho([\![P]\!](h_1, l_1)^{L}) = \rho([\![P]\!](h_2, l_2)^{L})$

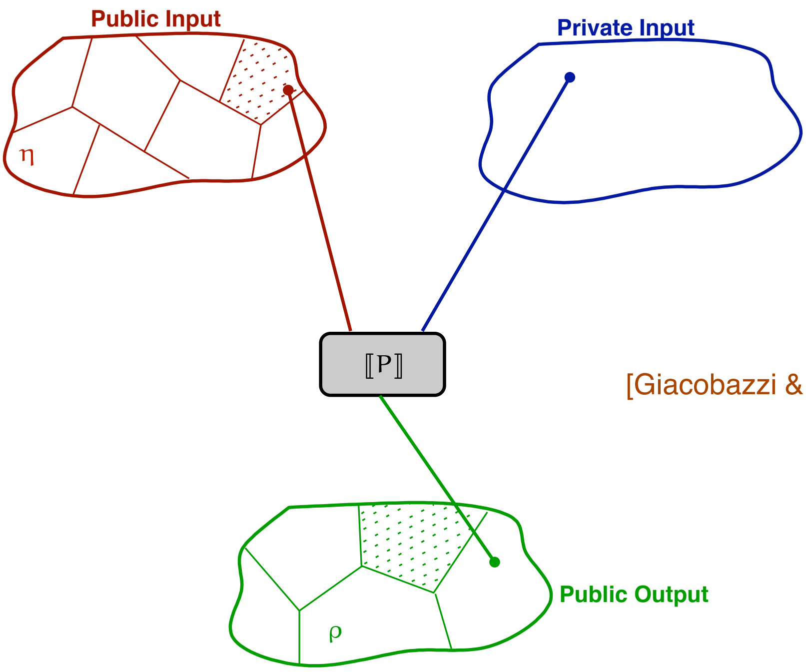
Abstract Non-Interference (Narrow)



[Giacobazzi & Mastroeni '04]

$\rho, \eta \in \mathit{Abs}(\wp(\mathbb{V}^{L}))$: $[\eta]P(\rho)$: $\eta(l_1) = \eta(l_2) \Rightarrow \rho([\![P]\!](h_1, l_1)^{L}) = \rho([\![P]\!](h_2, l_2)^{L})$

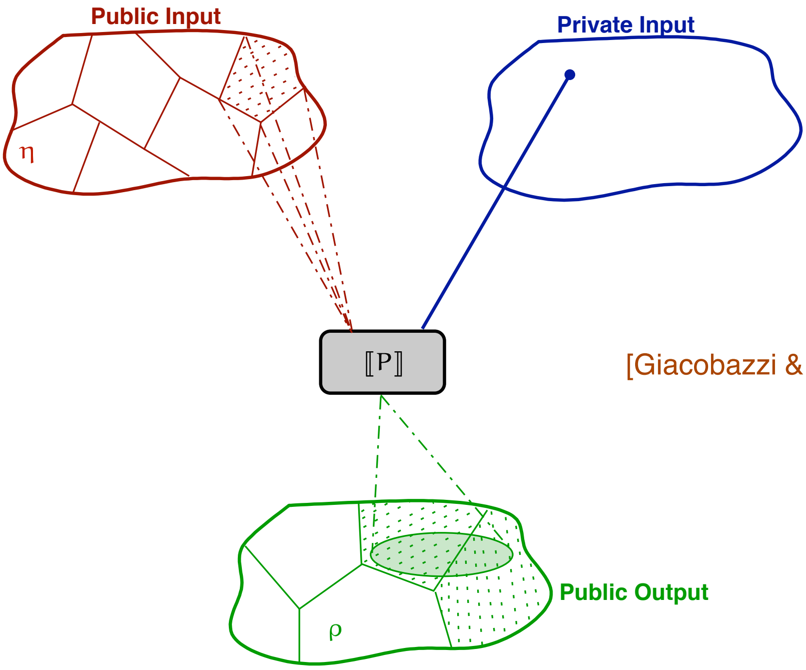
Abstract Non-Interference (Narrow)



[Giacobazzi & Mastroeni '04]

$$\rho, \eta \in \mathit{Abs}(\wp(\mathbb{V}^{\mathrm{L}})) \colon [\eta]P(\rho) \colon \eta(l_1) = \eta(l_2) \Rightarrow \rho(⟦P⟧(h_1, l_1)^{\mathrm{L}}) = \rho(⟦P⟧(h_2, l_2)^{\mathrm{L}})$$

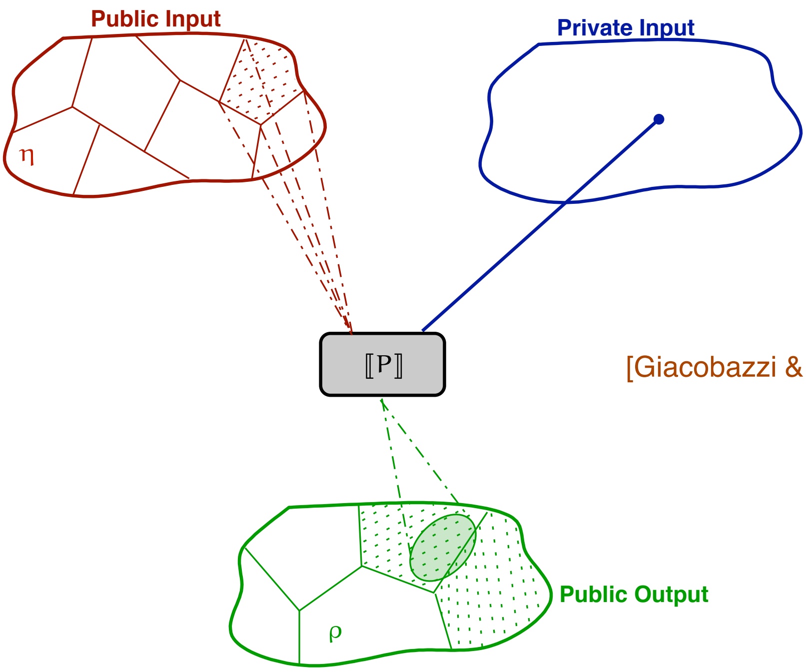
Abstract Non-Interference (ANI)



[Giacobazzi & Mastroeni '04]

$$\rho, \eta \in \mathit{Abs}(\wp(\mathbb{V}^{\mathrm{L}})) \colon (\eta)P(\rho) \colon$$
$$\eta(l_1) = \eta(l_2) \Rightarrow \rho(⟦P⟧(h_1, \eta(l_1))^{\mathrm{L}}) = \rho(⟦P⟧(h_2, \eta(l_2))^{\mathrm{L}})$$

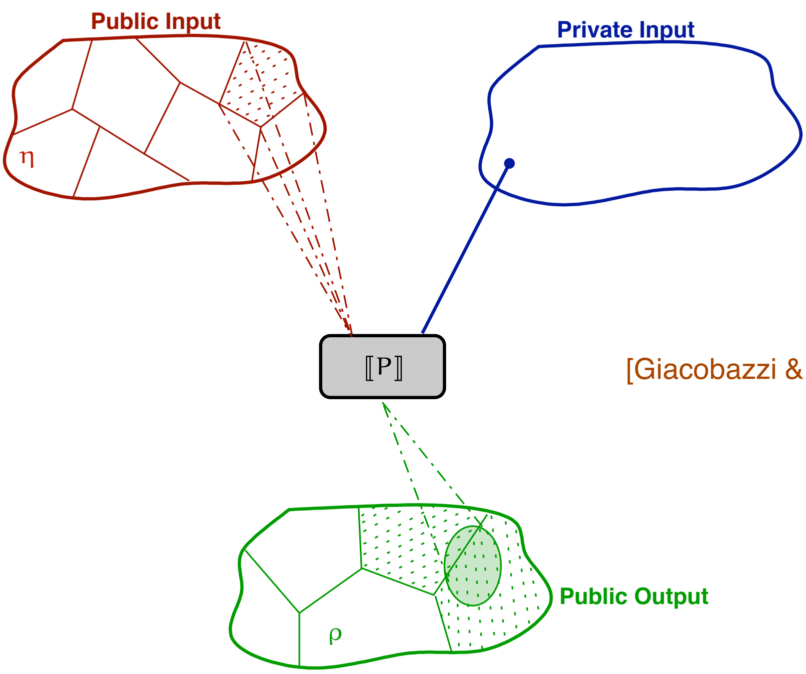
Abstract Non-Interference (ANI)



[Giacobazzi & Mastroeni '04]

$$\rho, \eta \in \mathit{Abs}(\wp(\mathbb{V}^{L})) : (\eta)P(\rho) :$$
$$\eta(l_1) = \eta(l_2) \Rightarrow \rho([\![P]\!](h_1, \eta(l_1))^{L}) = \rho([\![P]\!](h_2, \eta(l_2))^{L})$$

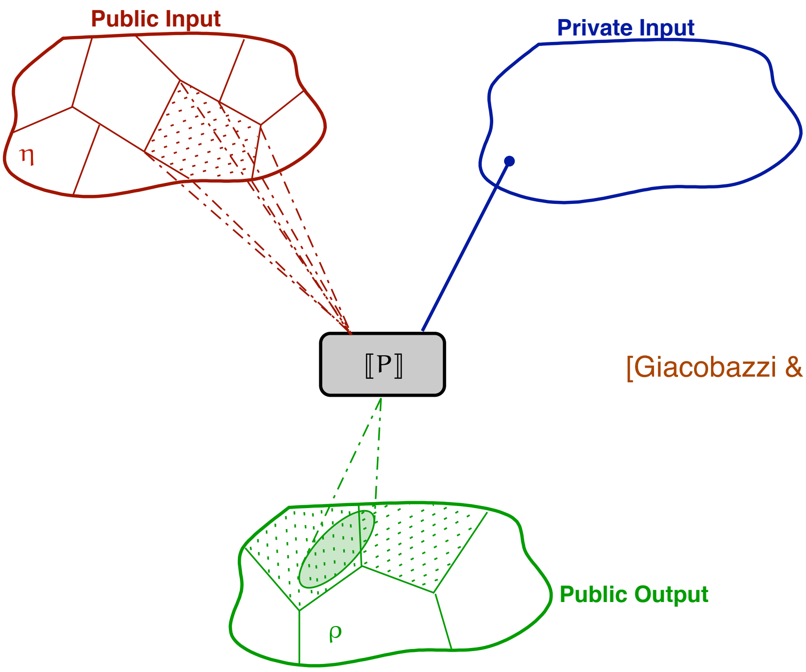
Abstract Non-Interference (ANI)



[Giacobazzi & Mastroeni '04]

$$\rho, \eta \in \mathit{Abs}(\wp(\mathbb{V}^{L})) : (\eta)P(\rho) :$$
$$\eta(l_1) = \eta(l_2) \Rightarrow \rho([\![P]\!](h_1, \eta(l_1))^{L}) = \rho([\![P]\!](h_2, \eta(l_2))^{L})$$

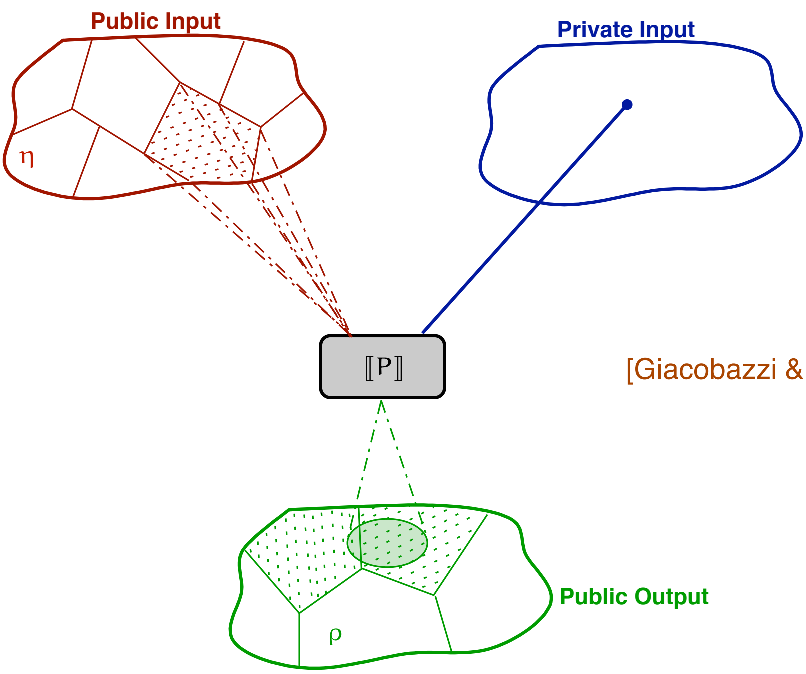
Abstract Non-Interference (ANI)



[Giacobazzi & Mastroeni '04]

$$\rho, \eta \in \mathit{Abs}(\wp(\mathbb{V}^{L})) \colon (\eta)P(\rho) \colon$$
$$\eta(l_1) = \eta(l_2) \Rightarrow \rho(\llbracket P \rrbracket(h_1, \eta(l_1))^{L}) = \rho(\llbracket P \rrbracket(h_2, \eta(l_2))^{L})$$

Abstract Non-Interference (ANI)



[Giacobazzi & Mastroeni '04]

$$\rho, \eta \in \mathit{Abs}(\wp(\mathbb{V}^{L})) \colon (\eta)P(\rho) \colon$$
$$\eta(l_1) = \eta(l_2) \Rightarrow \rho(\llbracket P \rrbracket(h_1, \eta(l_1))^{L}) = \rho(\llbracket P \rrbracket(h_2, \eta(l_2))^{L})$$

Examples

EXAMPLE I:

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

$$\boxed{\text{Standard Non-Interference} \equiv [id]P(id)}$$

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

Examples

EXAMPLE I:

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

$$\boxed{\text{Standard Non-Interference} \equiv [id]P(id)}$$

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

$$\Downarrow$$

$$\boxed{[id]P(Par)}$$

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

Examples

EXAMPLE II:

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

$$\boxed{[Par]P(Sign)}$$

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

Examples

EXAMPLE II:

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

$$\boxed{[Par]P(Sign)}$$

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

$$\Downarrow$$

$$\boxed{(Par)P(Sign)}$$

$$h = -3,\ Par(l) = \text{even} \rightsquigarrow Sign(l) = \text{I don't know}$$
$$h = 1,\ Par(l) = \text{even} \rightsquigarrow Sign(l) = \text{I don't know}$$

Examples

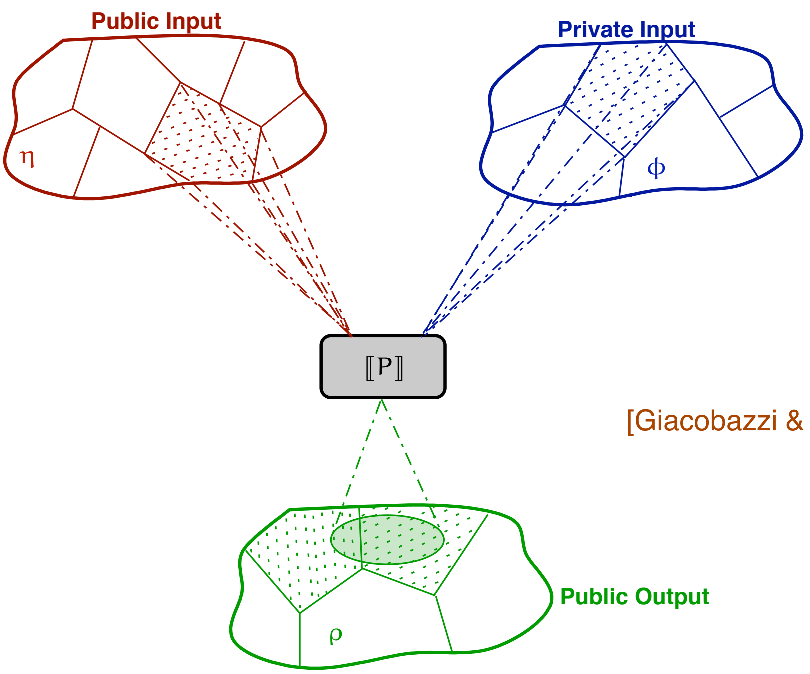
EXAMPLE III:

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

$$\boxed{(id)P(Par)}$$

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

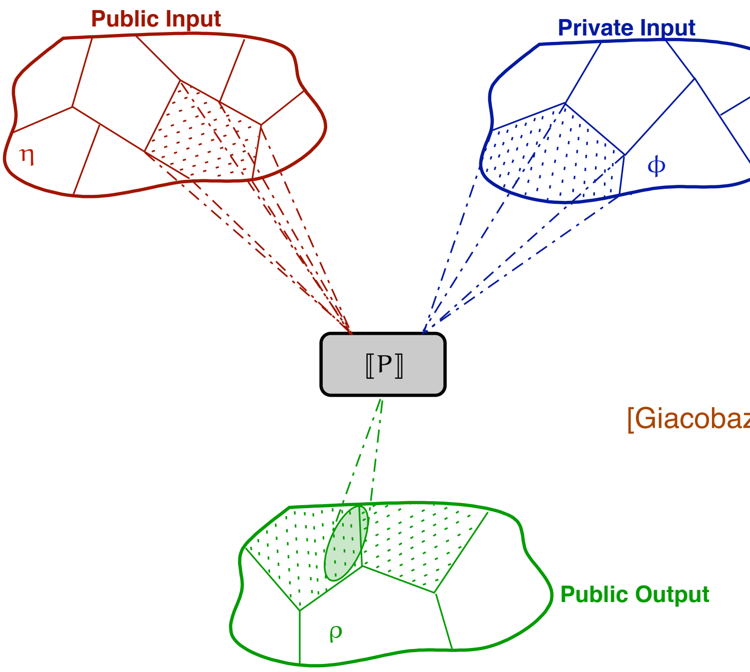
Declassified ANI via blocking



[Giacobazzi & Mastroeni '04]

$$\rho, \eta \in Abs(\wp(\mathbb{V}^L)), \phi \in Abs(\wp(\mathbb{V}^H)) \colon (\eta)P(\phi \rightsquigarrow\!\!\| \rho)\colon$$
$$\eta(l_1) = \eta(l_2) \Rightarrow \rho(\llbracket P \rrbracket(\phi(h_1), \eta(l_1))^L) = \rho(\llbracket P \rrbracket(\phi(h_2), \eta(l_2))^L)$$

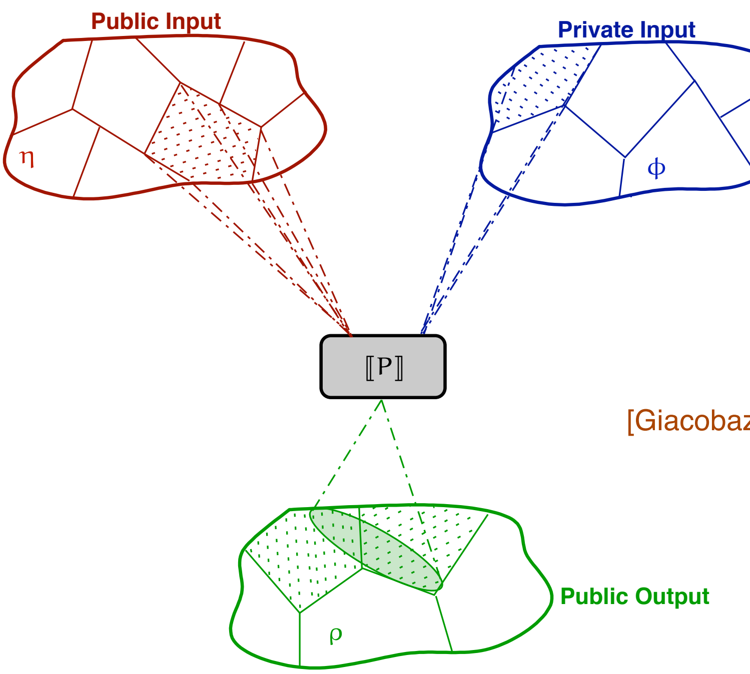
Declassified ANI via blocking



[Giacobazzi & Mastroeni '04]

$$\rho,\eta \in Abs(\wp(\mathbb{V}^{L})),\phi \in Abs(\wp(\mathbb{V}^{H})) : (\eta)P(\phi \rightsquigarrow\!\!\parallel \rho):$$
$$\eta(l_1)=\eta(l_2) \Rightarrow \rho(\llbracket P\rrbracket(\phi(h_1),\eta(l_1))^{L})=\rho(\llbracket P\rrbracket(\phi(h_2),\eta(l_2))^{L})$$

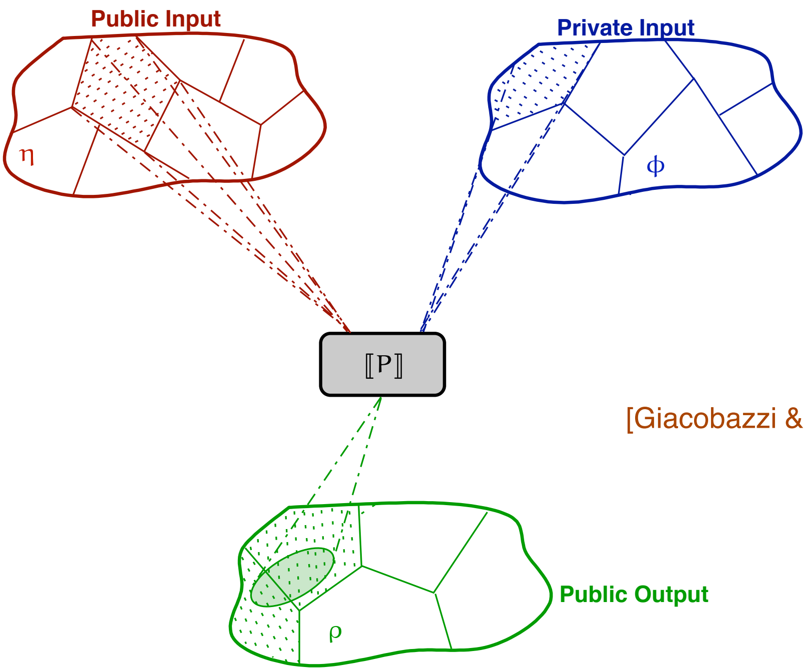
Declassified ANI via blocking



[Giacobazzi & Mastroeni '04]

$$\rho,\eta \in Abs(\wp(\mathbb{V}^{L})),\phi \in Abs(\wp(\mathbb{V}^{H})) : (\eta)P(\phi \rightsquigarrow\!\!\parallel \rho):$$
$$\eta(l_1)=\eta(l_2) \Rightarrow \rho(\llbracket P\rrbracket(\phi(h_1),\eta(l_1))^{L})=\rho(\llbracket P\rrbracket(\phi(h_2),\eta(l_2))^{L})$$

Declassified ANI via blocking

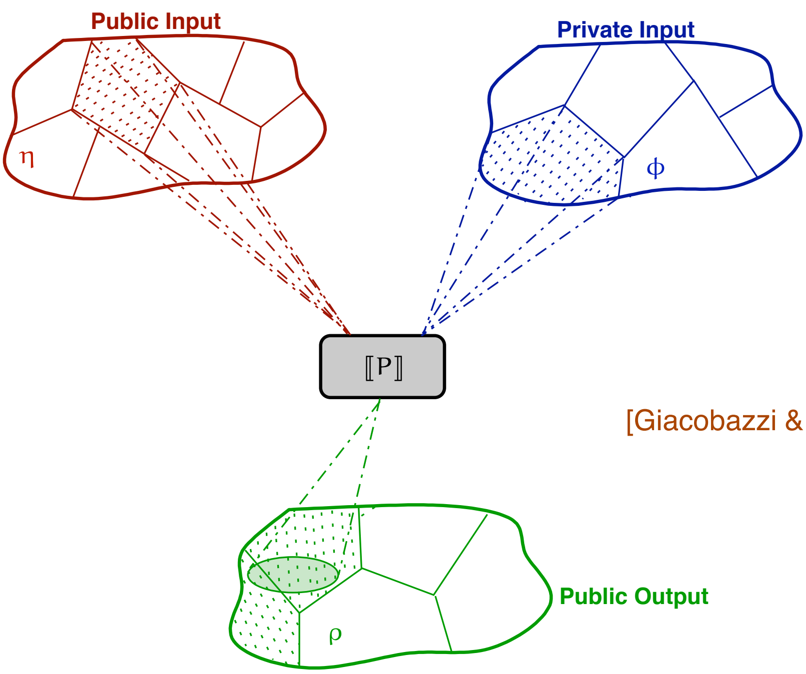


[Giacobazzi & Mastroeni '04]

$$\rho,\eta \in \mathit{Abs}(\wp(\mathbb{V}^{L})), \phi \in \mathit{Abs}(\wp(\mathbb{V}^{H})): (\eta)P(\phi \rightsquigarrow\!\!\| \rho):$$

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

Declassified ANI via blocking

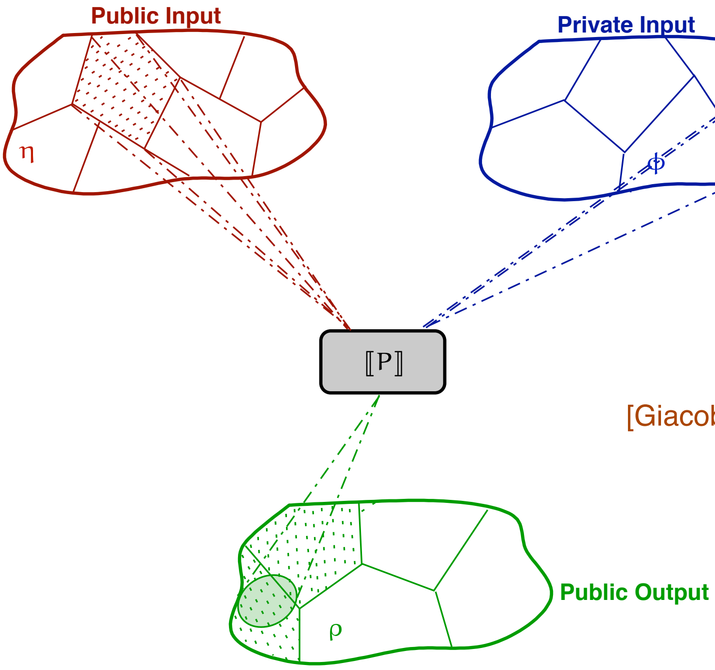


[Giacobazzi & Mastroeni '04]

$$\rho,\eta \in \mathit{Abs}(\wp(\mathbb{V}^{L})), \phi \in \mathit{Abs}(\wp(\mathbb{V}^{H})): (\eta)P(\phi \rightsquigarrow\!\!\| \rho):$$

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

Declassified ANI via blocking



[Giacobazzi & Mastroeni '04]

$$\rho, \eta \in \mathit{Abs}(\wp(\mathbb{V}^{L})), \phi \in \mathit{Abs}(\wp(\mathbb{V}^{H})) \colon (\eta)P(\phi \rightsquigarrow\!\!\| \rho) \colon$$
$$\eta(l_1) = \eta(l_2) \Rightarrow \rho(⟦P⟧(\phi(h_1), \eta(l_1))^{L}) = \rho(⟦P⟧(\phi(h_2), \eta(l_2))^{L})$$

Example

EXAMPLE:

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

$$\boxed{(\mathit{id})P(\mathit{Par})}$$

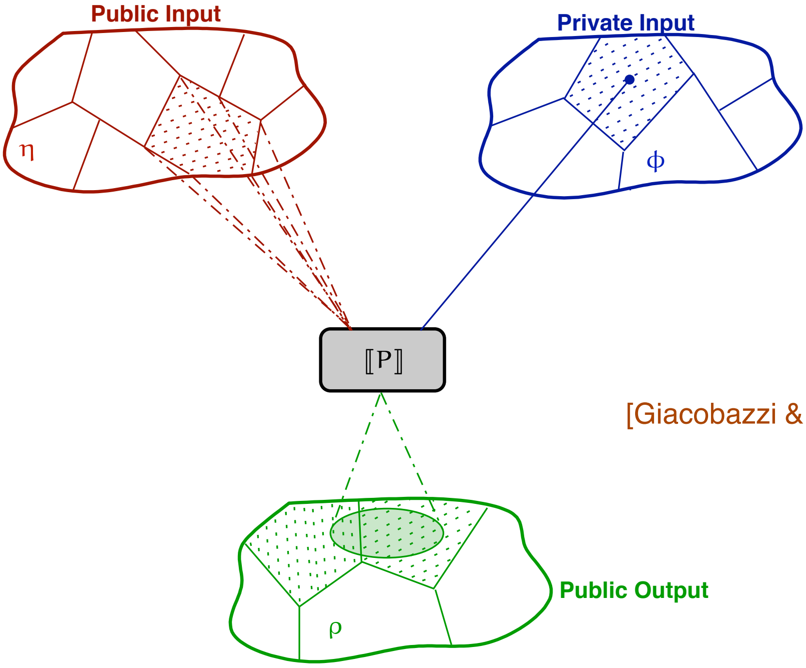
$$h = 2,\ l = 1 \ \rightsquigarrow\ \mathit{Par}(l) = \text{even}$$
$$h = 3,\ l = 1 \ \rightsquigarrow\ \mathit{Par}(l) = \text{odd}$$
$$h = n,\ l = 1 \ \rightsquigarrow\ \mathit{Par}(l) = \mathit{Par}(n)$$

$$\Downarrow$$

$$\boxed{(\mathit{id})P(\mathit{Sign} \rightsquigarrow\!\!\| \mathit{Par})}$$

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

Declassified ANI (via allowing)

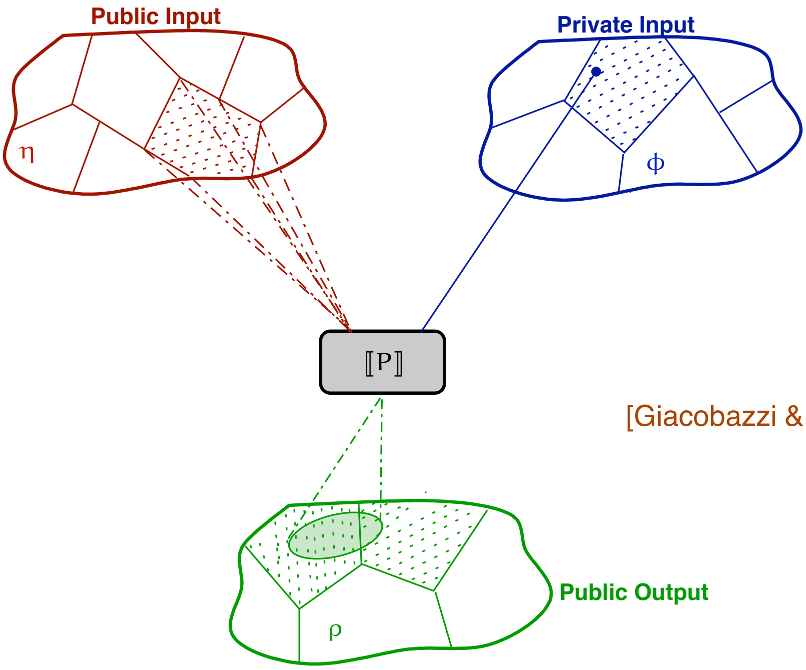


[Giacobazzi & Mastroeni '04]

$\rho, \eta \in Abs(\wp(\mathbb{V}^{L})), \phi \in Abs(\wp(\mathbb{V}^{H}))$: $(\eta)P(\phi \Rightarrow \rho)$:

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

Declassified ANI (via allowing)

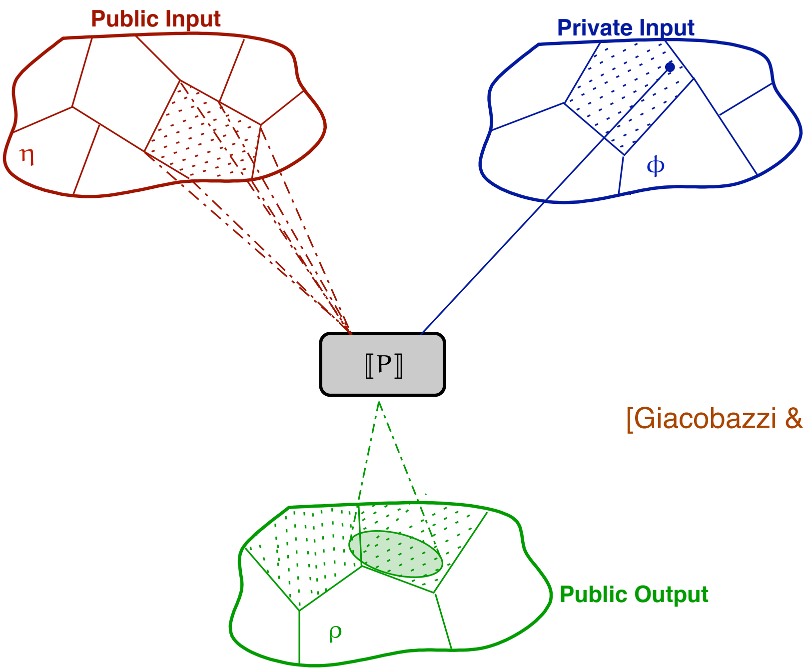


[Giacobazzi & Mastroeni '04]

$\rho, \eta \in Abs(\wp(\mathbb{V}^{L})), \phi \in Abs(\wp(\mathbb{V}^{H}))$: $(\eta)P(\phi \Rightarrow \rho)$:

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

Declassified ANI (via allowing)

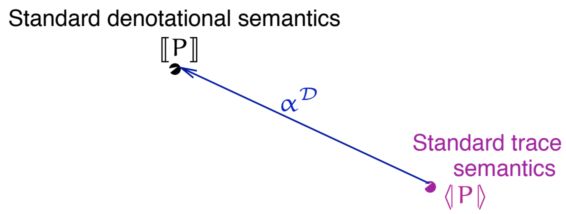


[Giacobazzi & Mastroeni '04]

$$\rho,\eta \in Abs(\wp(\mathbb{V}^L)), \phi \in Abs(\wp(\mathbb{V}^H)) : (\eta)P(\phi \Rightarrow \rho):$$

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

Timed abstract non-interference

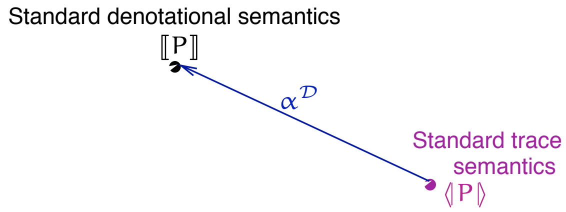


Trace semantics

$\Downarrow$

Traces' length = Time elapsed

Timed abstract non-interference

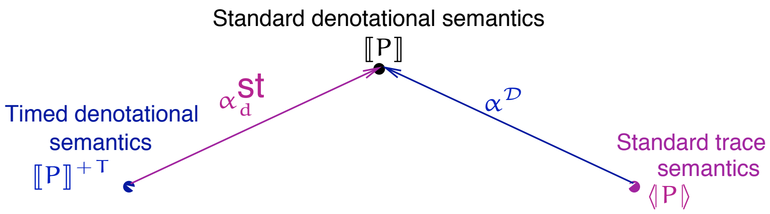


$\boxed{\textit{Traces' length} = \textsc{Time elapsed}}$

Stuttering removes time from traces!

TRACE SEMANTICS	TRACE WITHOUT STUTTERING
$2\mathbb{Z} \longrightarrow 2\mathbb{Z} \longrightarrow 2\mathbb{Z}+1 \longrightarrow 2\mathbb{Z}$	$2\mathbb{Z} \longrightarrow 2\mathbb{Z}+1 \longrightarrow 2\mathbb{Z}$
$\neq$	$=$
$2\mathbb{Z} \longrightarrow 2\mathbb{Z}+1 \longrightarrow 2\mathbb{Z}+1 \longrightarrow 2\mathbb{Z}$	$2\mathbb{Z} \longrightarrow 2\mathbb{Z}+1 \longrightarrow 2\mathbb{Z}$

Timed abstract non-interference

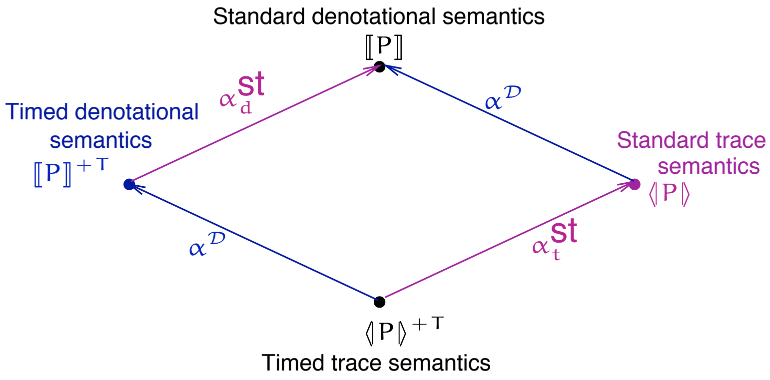


Timed denotational semantics

$\Downarrow$

$\boxed{\textit{Time counter} = \textsc{Time elapsed}}$

Timed abstract non-interference

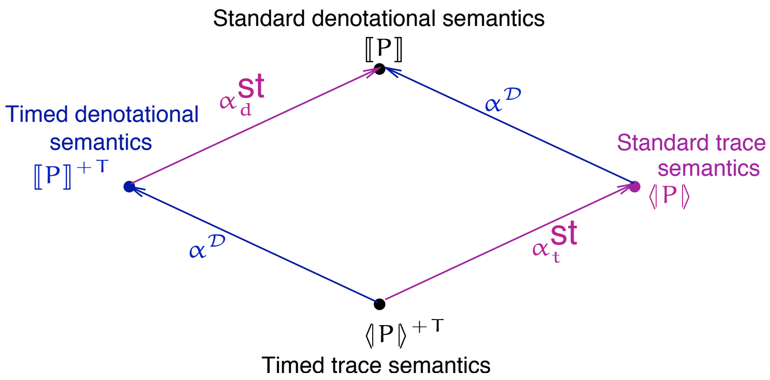


Timed denotational semantics

$$\Downarrow$$

Time counter = TIME ELAPSED

Timed abstract non-interference



Time counter = TIME ELAPSED

Abstraction removes time from timed denotational semantics!

TIMED SEMANTICS	UNTIMED SEMANTICS
$Par([\![P]\!](2,4,0))^{TL} = Par(6,3) = \langle 2\mathbb{Z}, 2\mathbb{Z}+1\rangle$	$\Pi^T(\langle 2\mathbb{Z}, 2\mathbb{Z}+1\rangle) = \langle 2\mathbb{Z}, \mathbb{Z}\rangle$
$\neq$	$=$
$Par([\![P]\!](4,4,0))^{TL} = Par(8,6) = \langle 2\mathbb{Z}, 2\mathbb{Z}\rangle$	$\Pi^T(\langle 2\mathbb{Z}, 2\mathbb{Z}\rangle) = \langle 2\mathbb{Z}, \mathbb{Z}\rangle$

CHARACTERIZING AND ENFORCING ABSTRACT NON-INTERFERENCE

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 ρ such that $(\eta)P(\phi \rightsquigarrow\!\!\!] \rho)$

  [The most powerful *public observer*]

Deriving output attackers

The following theorems hold:

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

Deriving output attackers

The following theorems hold:

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

$\Rightarrow$ 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 $(\delta)P(\phi \rightsquigarrow\!\!\!\!\!\!\!\!\;\;\|\delta)$

  [The most powerful *canonical* public observer]

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

Deriving canonical attackers

EXAMPLE:

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

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

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

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

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

Proving Abstract Non-Interference

Abstract Non-Interference is not defined *inductively* on the syntax

Proving Abstract Non-Interference

Abstract Non-Interference is not defined *inductively* on the syntax

$\Downarrow$

Use of Abstract Non-Interference hard in automatic program verification mechanisms.

Proving Abstract Non-Interference

Abstract Non-Interference is not defined *inductively* on the syntax

$\Downarrow$

Use of Abstract Non-Interference hard in automatic program verification mechanisms.

$\Downarrow$

Definition of *sound* proof systems for Abstract Non-Interference.

$$\frac{[\eta]c_1(\rho),\ [\rho]c_2(\beta)}{[\eta]c_1;c_2(\beta)} \qquad \frac{(\eta)c_1(\Upsilon(\rho)),\ [\rho]c_2(\Upsilon(\beta))}{(\eta)c_1;c_2(\Upsilon(\beta))}$$

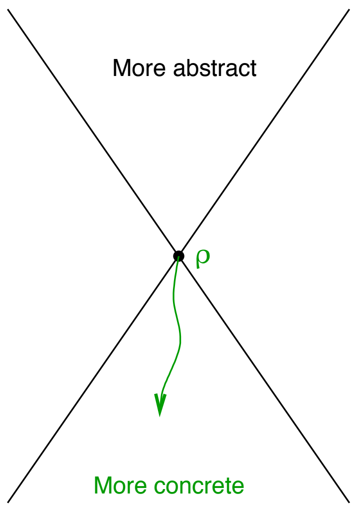
Language-based Security: Abstract Non-Interferece – p.18/32

Observer vs Observable

Consider $\models (\eta)P(\phi \rightsquigarrow \rho)$: *In order to preserve non-interference...*

Observer vs Observable

Consider $\models (\eta)P(\phi \rightsquigarrow \rho)$: *In order to preserve non-interference...*

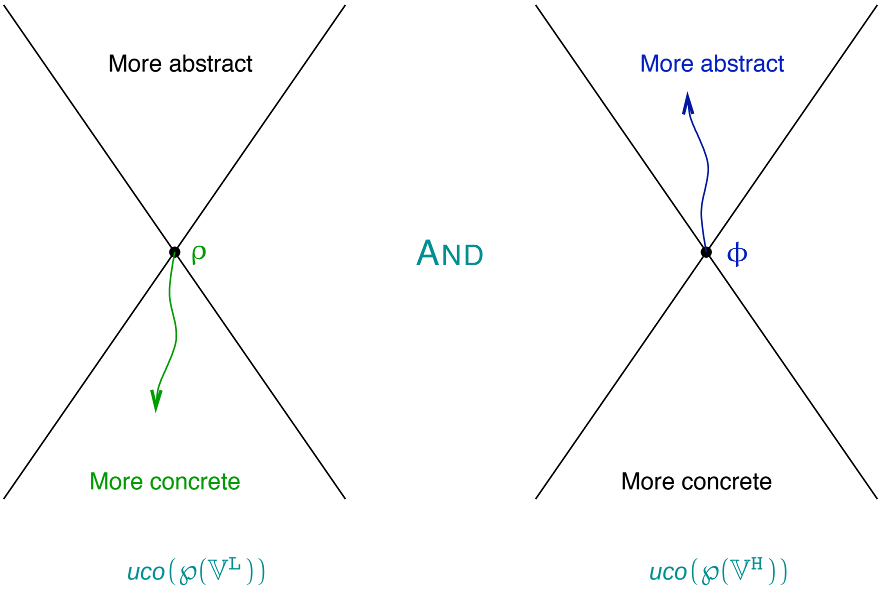


$uco(\wp(\mathbb{V}^{L}))$

Language-based Security: Abstract Non-Interferece – p.19/32

Observer vs Observable

Consider $\models (\eta)P(\phi \rightsquigarrow\!\!\![\![\rho)$: *In order to preserve non-interference...*



ANI: A completeness problem

Recall that [Joshi & Leino'00]

$$P \text{ is } \textit{secure} \quad \text{iff} \quad \mathtt{HH}\,;\, P;\, \mathtt{HH} \doteq P\,;\, \mathtt{HH}$$

ANI: A completeness problem

Recall that [Joshi & Leino'00]

$$\mathtt{P} \text{ is } \textit{secure} \quad \text{iff} \quad \mathtt{HH} \,;\, \mathtt{P};\, \mathtt{HH} \doteq \mathtt{P} \,;\, \mathtt{HH}$$

$$\text{Let } \mathtt{X} = \langle \mathtt{X}^{\mathtt{H}}, \mathtt{X}^{\mathtt{L}} \rangle \Rightarrow \mathcal{H}(\mathtt{X}) \stackrel{\text{def}}{=} \langle \top^{\mathtt{H}}, \mathtt{X}^{\mathtt{L}} \rangle \in \textit{uco}(\wp(\mathbb{V}))$$

$$\mathtt{HH} \,;\, \mathtt{P};\, \mathtt{HH} \doteq \mathtt{P} \,;\, \mathtt{HH}$$

$$\Downarrow$$

$$\mathcal{H} \circ [\![\mathtt{P}]\!] \circ \mathcal{H} = \mathcal{H} \circ [\![\mathtt{P}]\!]$$

ANI: A completeness problem

Recall that [Joshi & Leino'00]

$$\mathtt{P} \text{ is } \textit{secure} \quad \text{iff} \quad \mathtt{HH} \,;\, \mathtt{P};\, \mathtt{HH} \doteq \mathtt{P} \,;\, \mathtt{HH}$$

$$\text{Let } \mathtt{X} = \langle \mathtt{X}^{\mathtt{H}}, \mathtt{X}^{\mathtt{L}} \rangle \Rightarrow \mathcal{H}(\mathtt{X}) \stackrel{\text{def}}{=} \langle \top^{\mathtt{H}}, \mathtt{X}^{\mathtt{L}} \rangle \in \textit{uco}(\wp(\mathbb{V}))$$

$$\mathtt{HH} \,;\, \mathtt{P};\, \mathtt{HH} \doteq \mathtt{P} \,;\, \mathtt{HH}$$

$$\Downarrow$$

$$\mathcal{H} \circ [\![\mathtt{P}]\!] \circ \mathcal{H} = \mathcal{H} \circ [\![\mathtt{P}]\!]$$

$\Rightarrow$ A COMPLETENESS PROBLEM

ANI: A completeness problem

Let $X = \langle X^{H}, X^{L} \rangle \Rightarrow \mathcal{H}(X) \stackrel{\text{def}}{=} \langle \top^{H}, X^{L} \rangle \in uco(\wp(\mathbb{V}))$

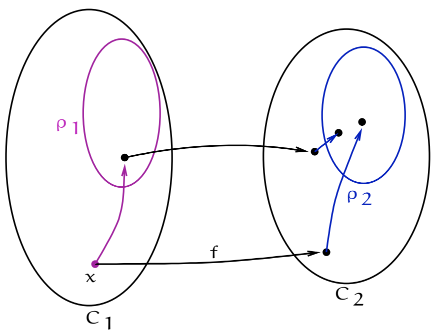
$$\mathcal{H} \circ [\![P]\!] \circ \mathcal{H} \quad = \quad \mathcal{H} \circ [\![P]\!]$$

COMPLETENESS = NON-INTERFERENCE

$$\Downarrow$$

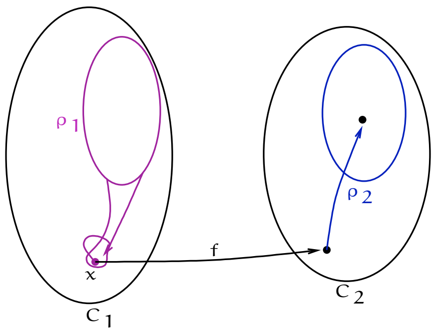
- Transform $\mathcal{H}$ vs *Core*;
- Transform $\mathcal{H}$ vs *Shell*. [Giacobazzi et al.'00]

Making Backward complete



$$\rho_2 f \rho_1 = \rho_2 f$$

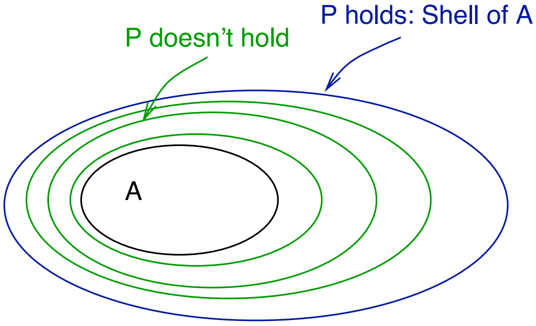
Making Backward complete



$$\rho_2 f \rho_1 = \rho_2 f$$

Completeness shells and cores

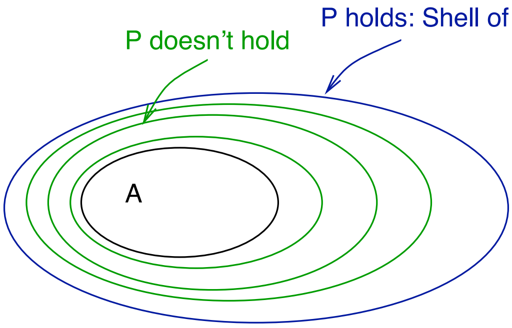
[Giacobazzi et al.'00]



$$R_f \stackrel{\text{def}}{=} \lambda\rho.\mathcal{M}(\bigcup_{y\in\rho} max(f^{-1}(\downarrow y)))$$

Completeness shells and cores

[Giacobazzi et al.'00]

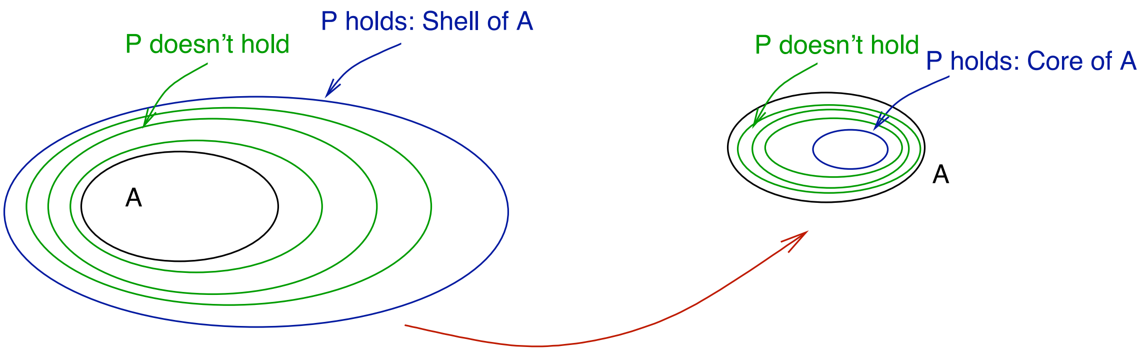


$$R_f \stackrel{def}{=} \lambda\rho.\mathcal{M}\Big(\bigcup_{y\in\rho} max(f^{-1}(\downarrow y))\Big)$$

- *Absolute shell* of ρ: $\mathcal{R}_f(\rho) = gfp^{\sqsubseteq}_{\rho}\lambda\varphi.\rho \sqcap R^{\mathcal{B}}_f(\varphi)$;
- *Relative shell of* η *relative to* ρ: $\mathcal{R}^{\rho}_f(\eta) = \eta \sqcap R_f(\rho)$.

Completeness shells and cores

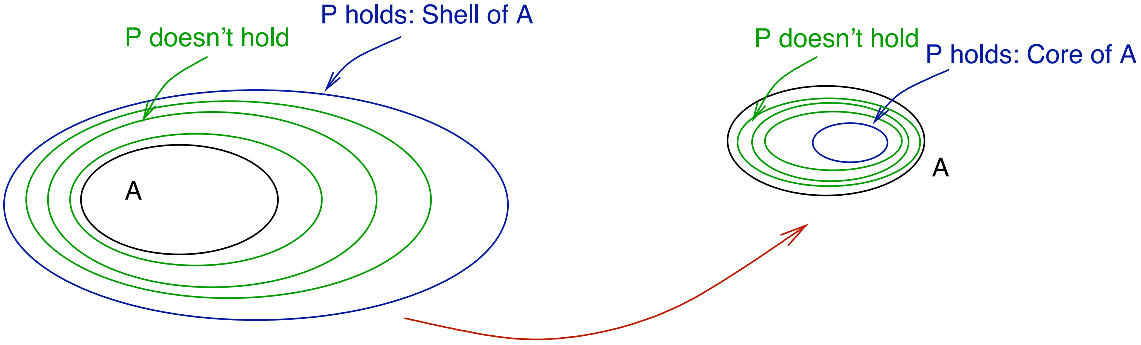
[Giacobazzi et al.'00]



$$C_f \stackrel{def}{=} \lambda\rho.\left\{\, y \in C \;\middle|\; max(f^{-1}(\downarrow y)) \subseteq \rho \,\right\}$$

Completeness shells and cores

[Giacobazzi et al.'00]



$$C_f \stackrel{\text{def}}{=} \lambda\rho.\left\{\, y \in C \;\middle|\; \mathit{max}(f^{-1}(\downarrow y)) \subseteq \rho \,\right\}$$

- *Absolute core of* ρ: $\mathcal{C}_f(\rho) = \mathit{lfp}_{\rho}^{\sqsubseteq}\lambda\varphi.\rho \sqcup C_f^{\mathcal{B}}(\varphi)$;
- *Relative core of* ρ *relative to* η: $\mathcal{C}_f^{\eta}(\rho) = \rho \sqcup C_f(\eta)$.

ANI as completeness

Let $\rho \in \mathit{uco}(\wp(\mathbb{V}^{\mathtt{L}})) \Rightarrow \mathcal{H}_\rho(X) \stackrel{\text{def}}{=} \langle \top^{\mathtt{H}}, \rho(X^{\mathtt{L}})\rangle \in \mathit{uco}(\wp(\mathbb{V}))$

- *Narrow abstract non-interference:* $\mathcal{H}_\rho \circ [\![P]\!] \circ \mathcal{H}_\eta = \mathcal{H}_\rho \circ [\![P]\!]$;
- *Abstract non-interference:* $\mathcal{H}_\rho \circ [\![P]\!]^{\eta,\phi} \circ \mathcal{H}_\eta = \mathcal{H}_\rho \circ [\![P]\!]^{\eta,\phi}$

ANI as completeness

Let $\rho \in uco(\wp(\mathbb{V}^{\mathtt{L}})) \Rightarrow \mathcal{H}_\rho(X) \stackrel{\text{def}}{=} \langle \top^{\mathtt{H}}, \rho(X^{\mathtt{L}}) \rangle \in uco(\wp(\mathbb{V}))$

- *Narrow abstract non-interference:* $\mathcal{H}_\rho \circ [\![P]\!] \circ \mathcal{H}_\eta = \mathcal{H}_\rho \circ [\![P]\!]$;
- *Abstract non-interference:* $\mathcal{H}_\rho \circ [\![P]\!]^{\eta,\phi} \circ \mathcal{H}_\eta = \mathcal{H}_\rho \circ [\![P]\!]^{\eta,\phi}$

$$\Downarrow$$

- PUBLIC OBSERVER AS COMPLETENESS CORE: $\mathcal{C}^{\mathcal{H}_\eta}_{[\![P]\!]^{\eta,\phi}}(\mathcal{H}) = (\eta)[\![P]\!](\phi \rightsquigarrow\!\!] \mathit{id})$

ANI as completeness

Let $\rho \in uco(\wp(\mathbb{V}^{\mathtt{L}})) \Rightarrow \mathcal{H}_\rho(X) \stackrel{\text{def}}{=} \langle \top^{\mathtt{H}}, \rho(X^{\mathtt{L}}) \rangle \in uco(\wp(\mathbb{V}))$

- *Narrow abstract non-interference:* $\mathcal{H}_\rho \circ [\![P]\!] \circ \mathcal{H}_\eta = \mathcal{H}_\rho \circ [\![P]\!]$;
- *Abstract non-interference:* $\mathcal{H}_\rho \circ [\![P]\!]^{\eta,\phi} \circ \mathcal{H}_\eta = \mathcal{H}_\rho \circ [\![P]\!]^{\eta,\phi}$

$$\Downarrow$$

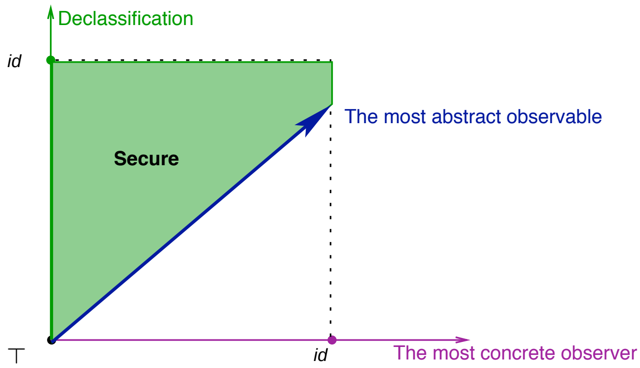
- PUBLIC OBSERVER AS COMPLETENESS CORE: $\mathcal{C}^{\mathcal{H}_\eta}_{[\![P]\!]^{\eta,\phi}}(\mathcal{H}) = (\eta)[\![P]\!](\phi \rightsquigarrow\!\!] \mathit{id})$
- PRIVATE OBSERVABLE AS COMPLETENESS SHELL: $(\eta)P(\mathcal{R}^{\mathcal{H}_\rho}_{[\![P]\!]^{\eta,\mathit{id}}}(\mathcal{H}_\eta) \Rightarrow \rho)$

Language-based Security: Abstract Non-Interferece – p.23/32

ANI as completeness

- PUBLIC OBSERVER AS COMPLETENESS CORE: $\mathcal{C}^{\mathcal{H}_\eta}_{[\![P]\!]^{\eta,\phi}}(\mathcal{H}) = (\eta)[\![P]\!](\phi \rightsquigarrow\!\!\!\| id)$
- PRIVATE OBSERVABLE AS COMPLETENESS SHELL: $(\eta)P(\mathcal{R}^{\mathcal{H}_\rho}_{[\![P]\!]^{\eta,id}}(\mathcal{H}_\eta) \Rightarrow \rho)$
- ADJOINING ATTACKERS AND DECLASSIFICATION

$$id \sqsubset (\eta)[\![P]\!](id \rightsquigarrow\!\!\!\| id) \Leftrightarrow \mathcal{P}(\sqcap_{L\in\eta}\mathcal{M}(\Pi_P(\eta, id)_{|L})) \sqsubset \top$$

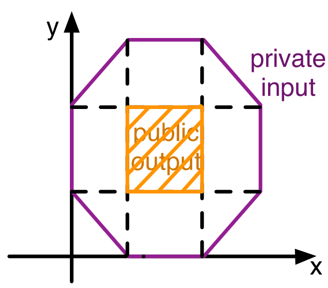


Declassification

[Banerjee, Giacobazzi and Mastroeni '07]

- By exploiting the strong relation between completeness and non-iterference we can obtain the following results:
  - Model declassification as a forward completeness problem for the weakest precondition semantics;
  - Derive counterexamples to a given declassification policy;
  - Refine a given declassification policy;

$$P \stackrel{\text{def}}{=} \left[\begin{array}{l} \textbf{if}(d \le x+y \le d+d_x+d_y \;\wedge\; -d_y \le x-y \le d_x)\;\textbf{then} \\ \quad \textbf{if}(x \ge 0 \wedge x \le d)\;\textbf{then}\; x_L := d; \\ \quad \textbf{if}(x > d \;\wedge\; x \le d_x)\;\textbf{then}\; x_L := x; \\ \quad \textbf{if}(x > d_x \;\wedge\; x \le d_x + d)\;\textbf{then}\; x_L := d_x; \\ \quad \textbf{if}(y \ge 0 \wedge y \le d)\;\textbf{then}\; y_L := d; \\ \quad \textbf{if}(y > d \;\wedge\; y \le d_y)\;\textbf{then}\; y_L := y; \\ \quad \textbf{if}(y > d_y \;\wedge\; y \le d_y + d)\;\textbf{then}\; y_L := d_y; \end{array}\right.$$



Declassification

[Banerjee, Giacobazzi and Mastroeni '07]

- By exploiting the strong relation between completeness and non-iterference we can obtain the following results:
  - Model declassification as a forward completeness problem for the weakest precondition semantics;
  - Derive counterexamples to a given declassification policy;
  - Refine a given declassification policy;
- We can model declassification as a model checking problem (see the relation with robust declassification)

ABSTRACT NON-INTERFERENCE

VS

OTHER APPROACHES

Generalized abstract non-interference

NON-INTERFERENCE

Corresponds to asking that the behavior of the chosen relevant aspects of the computation be invariant with respect to what an attacker may observe.

Generalized abstract non-interference

NON-INTERFERENCE

Corresponds to asking that the behavior of the chosen relevant aspects of the computation be invariant with respect to what an attacker may observe.

- α_{OBS}: Specifies the semantics of the computations relevant for interference (*observation abstraction*);

Generalized abstract non-interference

NON-INTERFERENCE

Corresponds to asking that the behavior of the chosen relevant aspects of the computation be invariant with respect to what an attacker may observe.

- α_{OBS}: Specifies the semantics of the computations relevant for interference (*observation abstraction*);
- α_{INT}: Specifies the maximum amount of information that an attacker may observe concerning a computation (*interference abstraction*);

Generalized abstract non-interference

NON-INTERFERENCE

Corresponds to asking that the behavior of the chosen relevant aspects of the computation be invariant with respect to what an attacker may observe.

- α_{OBS}: Specifies the semantics of the computations relevant for interference (*observation abstraction*);
- α_{INT}: Specifies the maximum amount of information that an attacker may observe concerning a computation (*interference abstraction*);
- α_{ATT}: Characterizes what the model of the attacker can observe about the system behavior (*attacker abstraction*).

Generalized abstract non-interference

NON-INTERFERENCE

Corresponds to asking that the behavior of the chosen relevant aspects of the computation be invariant with respect to what an attacker may observe.

- α_{OBS}: Specifies the semantics of the computations relevant for interference (*observation abstraction*);
- α_{INT}: Specifies the maximum amount of information that an attacker may observe concerning a computation (*interference abstraction*);
- α_{ATT}: Characterizes what the model of the attacker can observe about the system behavior (*attacker abstraction*).

$$\boxed{\alpha_{ATT} \circ \alpha_{OBS}(\llbracket P \rrbracket) = \alpha_{ATT} \circ \alpha_{INT} \circ \alpha_{OBS}(\llbracket P \rrbracket).}$$

Generalized abstract non-interference

NON-INTERFERENCE

Corresponds to asking that the behavior of the chosen relevant aspects of the computation be invariant with respect to what an attacker may observe.

- α_{OBS}: Specifies the semantics of the computations relevant for interference (*observation abstraction*);
- α_{INT}: Specifies the maximum amount of information that an attacker may observe concerning a computation (*interference abstraction*);
- α_{ATT}: Characterizes what the model of the attacker can observe about the system behavior (*attacker abstraction*).

$\Rightarrow$ We characterize the minimal abstraction of α_{ATT} that guarantees GANI.

Generalized abstract non-interference

NON-INTERFERENCE

Corresponds to asking that the behavior of the chosen relevant aspects of the computation be invariant with respect to what an attacker may observe.

- α_{OBS}: Specifies the semantics of the computations relevant for interference (*observation abstraction*);
- α_{INT}: Specifies the maximum amount of information that an attacker may observe concerning a computation (*interference abstraction*);
- α_{ATT}: Characterizes what the model of the attacker can observe about the system behavior (*attacker abstraction*).

SNNI = $\alpha_{\mathrm{T}} \circ \alpha_{low} \circ id(\llbracket \mathrm{P} \rrbracket) = \alpha_{\mathrm{T}} \circ \alpha_{low} \circ \alpha_{\mathrm{L}} \circ id(\llbracket \mathrm{P} \rrbracket)$.

Generalized abstract non-interference

NON-INTERFERENCE

Corresponds to asking that the behavior of the chosen relevant aspects of the computation be invariant with respect to what an attacker may observe.

- α_{OBS}: Specifies the semantics of the computations relevant for interference (*observation abstraction*);
- α_{INT}: Specifies the maximum amount of information that an attacker may observe concerning a computation (*interference abstraction*);
- α_{ATT}: Characterizes what the model of the attacker can observe about the system behavior (*attacker abstraction*).

BNDC = $\alpha_{\mathrm{B}} \circ \alpha_{\mathrm{L}} \circ id(\llbracket \mathrm{P} || \Pi \rrbracket) = \alpha_{\mathrm{B}} \circ \alpha_{\mathrm{L}} \circ \alpha_{sec} \circ id(\llbracket \mathrm{P} || \Pi \rrbracket)$.

Generalized abstract non-interference

NON-INTERFERENCE

Corresponds to asking that the behavior of the chosen relevant aspects of the computation be invariant with respect to what an attacker may observe.

- α_{OBS}: Specifies the semantics of the computations relevant for interference (*observation abstraction*);
- α_{INT}: Specifies the maximum amount of information that an attacker may observe concerning a computation (*interference abstraction*);
- α_{ATT}: Characterizes what the model of the attacker can observe about the system behavior (*attacker abstraction*).

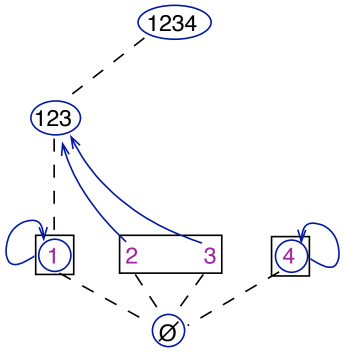
$\mathfrak{n}$-interference for Timed Automata = $\alpha_{low} \circ \alpha_{\mathfrak{n}}(\llbracket P \rrbracket) = \alpha_{low} \circ \alpha_{L} \circ \alpha_{\mathfrak{n}}(\llbracket P \rrbracket)$.

PER model of ANI

[Hunt & Mastroeni '05]

Partitioning Closure

[Ranzato and Tapparo '04]

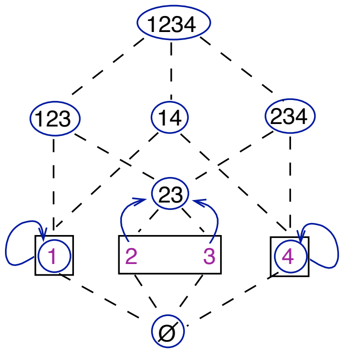


PER model of ANI

[Hunt & Mastroeni '05]

Partitioning Closure

[Ranzato and Tapparo '04]



$\Pi(\eta)$ is the most concrete partitioning closure containing η!

PER model of ANI

[Hunt & Mastroeni '05]

$\Pi(\eta)$ is the most concrete partitioning closure containing η!

$$
\begin{array}{lll}
[\eta]P(\rho) & \text{iff} & [\![P]\!] : \mathit{All} \times \mathit{Rel}^{\eta} \twoheadrightarrow \mathit{All} \times \mathit{Rel}^{\rho} \\
 & \text{iff} & [\Pi(\eta)]P(\Pi(\rho))
\end{array}
$$

PER model of ANI

[Hunt & Mastroeni '05]
$\Pi(\eta)$ is the most concrete partitioning closure containing η!

$$[\eta]P(\rho) \quad \text{iff} \quad [\![P]\!] : \mathit{All} \times \mathit{Rel}^{\eta} \rightharpoonup \mathit{All} \times \mathit{Rel}^{\rho}$$
$$\quad\quad\quad \text{iff} \quad [\Pi(\eta)]P(\Pi(\rho))$$

$$\Downarrow$$

$$[R]P(S) \text{ sse } [\![P]\!] : \mathit{All} \times R \rightharpoonup \mathit{All} \times S$$

ANI vs Robust Declassification

[Zdancewic & Myers '01]

- The PER model of Abstract Non-interference on the *maximal trace semantics* IS EQUIVALENT TO the security properrty introduced for Robust Declassification!

ANI vs Robust Declassification

[Zdancewic & Myers '01]

- Let us recall that Declassificazione robusta transform the attacker observational capability in order to derive what the program releases:

$$\forall \sigma, \sigma' \in \Sigma \,.\, \langle \sigma, \sigma' \rangle \in S[\approx] \;\Leftrightarrow\; \mathrm{Obs}_{\sigma}(S, \approx) \equiv \mathrm{Obs}_{\sigma'}(S, \approx)$$

- [Banerjee, Giacobazzi and Mastroeni '07] This is a backward completeness problem!

ANI vs Robust Declassification

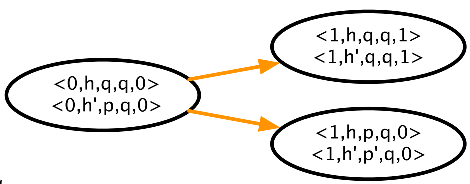
[Zdancewic & Myers '01]

- Let us recall that Declassificazione robusta transform the attacker observational capability in order to derive what the program releases:

$$\forall \sigma, \sigma' \in \Sigma \,.\, \langle \sigma, \sigma' \rangle \in S[\approx] \;\Leftrightarrow\; \mathrm{Obs}_{\sigma}(S, \approx) \equiv \mathrm{Obs}_{\sigma'}(S, \approx)$$

- Example:

$$\begin{array}{lcll} \langle t, h, p, q, r \rangle & \mapsto & \langle t, h, p, q, r \rangle & \\ \langle 0, h, q, q, 0 \rangle & \mapsto & \langle 1, h, q, q, 1 \rangle & \\ \langle 0, h, q, q, 1 \rangle & \mapsto & \langle 1, h, q, q, 0 \rangle & \\ \langle 0, h, p, q, 0 \rangle & \mapsto & \langle 1, h, p, q, 0 \rangle & p \neq q \\ \langle 0, h, p, q, 1 \rangle & \mapsto & \langle 1, h, p, q, 1 \rangle & p \neq q \end{array}$$



The public variables are t, q, r, hence the partition induced by $\mathcal{H}$ is:

$$\langle t, h, p, q, r \rangle \;\equiv\; \langle t', h', p', q', r' \rangle \text{ iff } t = t' \;\wedge\; q = q' \;\wedge\; r = r'$$

ANI vs Robust Declassification

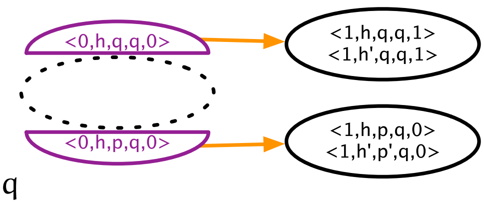
[Zdancewic & Myers '01]

- Let us recall that Declassificazione robusta transform the attacker observational capability in order to derive what the program releases:

$$\forall \sigma, \sigma' \in \Sigma \,.\, \langle \sigma, \sigma' \rangle \in S[\approx] \;\Leftrightarrow\; \mathrm{Obs}_{\sigma}(S, \approx) \equiv \mathrm{Obs}_{\sigma'}(S, \approx)$$

- Example:

$$\begin{array}{lcll} \langle t, h, p, q, r\rangle & \mapsto & \langle t, h, p, q, r\rangle & \\ \langle 0, h, q, q, 0\rangle & \mapsto & \langle 1, h, q, q, 1\rangle & \\ \langle 0, h, q, q, 1\rangle & \mapsto & \langle 1, h, q, q, 0\rangle & \\ \langle 0, h, p, q, 0\rangle & \mapsto & \langle 1, h, p, q, 0\rangle & p \neq q \\ \langle 0, h, p, q, 1\rangle & \mapsto & \langle 1, h, p, q, 1\rangle & p \neq q \end{array}$$



$$\langle 0, h, p, q, 0\rangle \mapsto \begin{cases} \langle 1, h, q, q, 1\rangle \\ \langle 1, h, p, q, 0\rangle \end{cases} \quad \widetilde{pre}_P : \begin{cases} \langle 1, h, q, q, 1\rangle \mapsto \langle 0, h, q, q, 0\rangle & \\ \langle 1, h, p, q, 0\rangle \mapsto \langle 0, h, p, q, 0\rangle & p \neq q \end{cases}$$

ANI vs Enforcing Robust Declassification

[Myers et al. '04]

Language=IMP+*declassify*(e)+$[\bullet]$
$P[a]$ is the program P under the attack a!

$\Downarrow$

$P[\bullet]$ è ROBUSTO se

$$\forall s_1, s_2 \in \Sigma . \forall a, a' :\; [\![P[a]]\!](s_1)^{L} = [\![P[a]]\!](s_2)^{L} \;\Rightarrow\; [\![P[a']]\!](s_1)^{L} = [\![P[a']]\!](s_2)^{L}$$

If a *controls only public inputs* then it is ANI!

EXAMPLE:

$$P \stackrel{\text{def}}{=} [\bullet]; l := l + \textit{declassify}(h \;\textit{mod}\; 3)$$

$\Downarrow$

P satisfies *robust declassification*!

ANI vs Enforcing Robust Declassification

[Myers et al. '04]

Language=IMP+*declassify*(e)+$[\bullet]$
$P[a]$ is the program P under the attack a!

$\Downarrow$

$P[\bullet]$ è ROBUSTO se

$$\forall s_1, s_2 \in \Sigma. \forall a, a' : \; [\![P[a]]\!](s_1)^L = [\![P[a]]\!](s_2)^L \;\Rightarrow\; [\![P[a']]\!](s_1)^L = [\![P[a']]\!](s_2)^L$$

If a *controls only public inputs* then it is ANI!

EXAMPLE:

$$P \stackrel{\text{def}}{=} [\bullet]; l := l + \textit{declassify}(h \ \textit{mod}\ 3)$$

P satisfies *robust declassification*!

$\Downarrow$

Declassified ANI shows that P satisfies *robust declassification* declassifying the MINIMAL amount of information!

ANI vs Enforcing Robust Declassification

[Myers et al. '04]

Language=IMP+*declassify*(e)+$[\bullet]$
$P[a]$ is the program P under the attack a!

$\Downarrow$

$P[\bullet]$ è ROBUSTO se

$$\forall s_1, s_2 \in \Sigma. \forall a, a' : \; [\![P[a]]\!](s_1)^L = [\![P[a]]\!](s_2)^L \;\Rightarrow\; [\![P[a']]\!](s_1)^L = [\![P[a']]\!](s_2)^L$$

If a *controls only public inputs* then it is ANI!

EXAMPLE: Consider

$$P = \; l := l + (h \ \textit{mod}\ 3)$$

$\Downarrow$

The MAXIMAL amount of information released is
$\phi = \{\top, 3\mathbb{Z}, 3\mathbb{Z}+1, 3\mathbb{Z}+2, \varnothing\}$.

ANI vs Delimited release

[Sabelfeld & Myers '04]

Language=IMP+*declassify*(e)

$s_1 \approx_E s_2$ iff $\forall e \in E.\ [\![e]\!](s_1) = [\![e]\!](s_2)$

$$\Downarrow$$

P satisfies DELIMITED RELEASE, $E = \left\{ e \,\middle|\, \textit{declassify}(e) \text{ in } P \right\}$

$$\boxed{\forall s_1, s_2 \in \Sigma.\ s_1^L = s_2^L \wedge s_1 \approx_E s_2 \ \Rightarrow\ [\![P]\!](s_1)^L = [\![P]\!](s_2)^L}$$

EXAMPLE:

The program P satisfies delimited release while P' doesn't:

$$P \stackrel{\text{def}}{=} \textbf{if } \textit{declassify}(h \geq k) \textbf{ then } (h := h - k; l := l + k) \textbf{ else nil}$$

$$\Downarrow$$

$$\phi_k = \{\mathbb{V}^H, \left\{ h \,\middle|\, h \geq k \right\}, \left\{ h \,\middle|\, h < k \right\}, \varnothing\}$$

$$\phi = \textstyle\bigsqcap_k \phi_k = \textit{id}$$

ANI vs Delimited release

[Sabelfeld & Myers '04]

Language=IMP+*declassify*(e)

$s_1 \approx_E s_2$ iff $\forall e \in E.\ [\![e]\!](s_1) = [\![e]\!](s_2)$

$$\Downarrow$$

P satisfies DELIMITED RELEASE, $E = \left\{ e \,\middle|\, \textit{declassify}(e) \text{ in } P \right\}$

$$\boxed{\forall s_1, s_2 \in \Sigma.\ s_1^L = s_2^L \wedge s_1 \approx_E s_2 \ \Rightarrow\ [\![P]\!](s_1)^L = [\![P]\!](s_2)^L}$$

EXAMPLE:

The program P satisfies delimited release while P' doesn't:

$$P' \stackrel{\text{def}}{=} \left[\begin{array}{ll} l := 0; & \\ \textbf{while } n \geq 0 \textbf{ do} & k := 2^{n+1} \\ & \textbf{if } \textit{declassify}(h \geq k) \textbf{ then } (h := h - k; l := l + k) \textbf{ else nil} \\ & n := n - 1 \end{array} \right.$$

$$\Downarrow$$

$$\phi = \textit{id}$$

ANI vs Relaxed non-interference

[Li & Zdancewic '05]

Language=λ-calculus (no explicit declassification)

$\Downarrow$

P satisfies RELAXED NON-INTERFERENCE, if

$$P \equiv f(n_1\sigma_1)(n_2\sigma_2)\ldots(n_k\sigma_k)$$

EXAMPLE:

$$P \stackrel{\text{def}}{=} \left[\begin{array}{l} x := \textit{hash}(sec); y := x \textit{ mod } 2^{64}; \\ \textbf{if } y = in \textbf{ then } out = 1 \textbf{ else } out := 0; \end{array} \right.$$

$\Downarrow$

$$\phi_{in} = \left\{ \begin{array}{l} \mathbb{V}^H, \left\{ sec \,\middle|\, \textit{hash}(sec) \textit{ mod } 2^{64} = in \right\}, \\ \left\{ sec \,\middle|\, \textit{hash}(sec) \textit{ mod } 2^{64} \neq in \right\}, \varnothing \end{array} \right\}$$

$$\phi = \bigsqcap_{in} \phi_{in} = \{\mathbb{V}^H, \varnothing\} \cup \left\{ 2^{64}\mathbb{Z} + n \,\middle|\, 0 \leq n < 2^{64} \right\}$$

ANI vs Relaxed non-interference

[Li & Zdancewic '05]

Language=λ-calculus (no explicit declassification)

$\Downarrow$

P satisfies RELAXED NON-INTERFERENCE, if

$$P \equiv f(n_1\sigma_1)(n_2\sigma_2)\ldots(n_k\sigma_k)$$

EXAMPLE:
Password check:

$$P = \lambda in.\textbf{if } in = \sigma_{pw} \textbf{ then } out := 1 \textbf{ else } out := 0$$

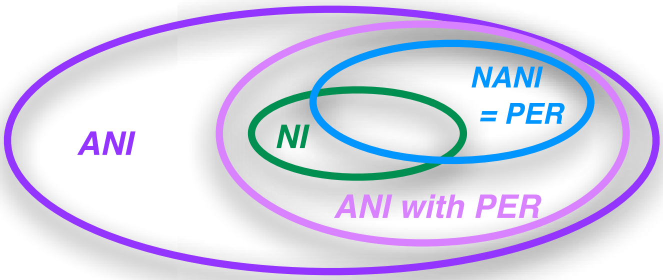
satisfies *relaxed non-interference* being equivalent to:

$$\lambda x.\lambda g : \mathbb{Z} \longrightarrow \mathbb{Z}.(\textbf{if } g(x) \textbf{ then } out := 1 \textbf{ else } out := 0) in((\lambda x.\lambda y.x = y)\sigma_{pw})$$

$\Rightarrow$ As far as *Declassified ANI* is concerned the program is not secure : $\phi = \textit{id}$.

Conclusions

WHAT HAVE WE DONE AND WHAT HAVE WE STILL TO DO?



Conclusions

WHAT HAVE WE DONE AND WHAT HAVE WE STILL TO DO?