HIDING INFORMATION IN COMPLETENESS HOLES

NEW PERSPECTIVES IN CODE OBFUSCATION AND WATERMARKING

Roberto Giacobazzi

Dipartimento di Informatica
Università di Verona
Italy

SEFM'08, Cape Town November 2008

THE PROBLEM: PROTECTION

- In SW much of the know-how is located in the product itself!
- According to Business Software Alliance (BSA):
  - the worldwide weighted average piracy rate is 35%, the median piracy rate is 62%, meaning half of the countries have a piracy rate of 62% or higher of the market, which grows to 75% in one-third of the countries
  - In 2007, every 2.00USD worth of software purchased legitimately, 1.00USD worth was obtained illegally!!
- knowledge extraction by static and dynamic analysis
- program decomposition for code reuse
- source code disassembly and decompilation for reverse engineering
- integrity corruption for code hacking

THE PROBLEM: PROTECTION

We need adequate strategies for

Intellectual Property Protection (IPP)

and

Digital Right Management (DRM)

- Make difficult source code analysis
- Make difficult program decomposition, disassembly and decompiation
- Steganography (watermarking and fingerprinting) against theft
- Tamper proofing against integrity corruption

THE PROBLEM: ATTACK

Malware represents malicious software.

Malware detector is a program $\mathcal{D}$ that determines whether another program P is infected with a malware M.

$$\mathcal{D}(P, M) = \begin{cases} \text{True} & \text{if } P \text{ is infected with } M \\ \text{False} & \text{otherwise} \end{cases}$$

The problem: Attack

Malware represents malicious software.

Malware detector is a program $\mathcal{D}$ that determines whether another program P is infected with a malware M.

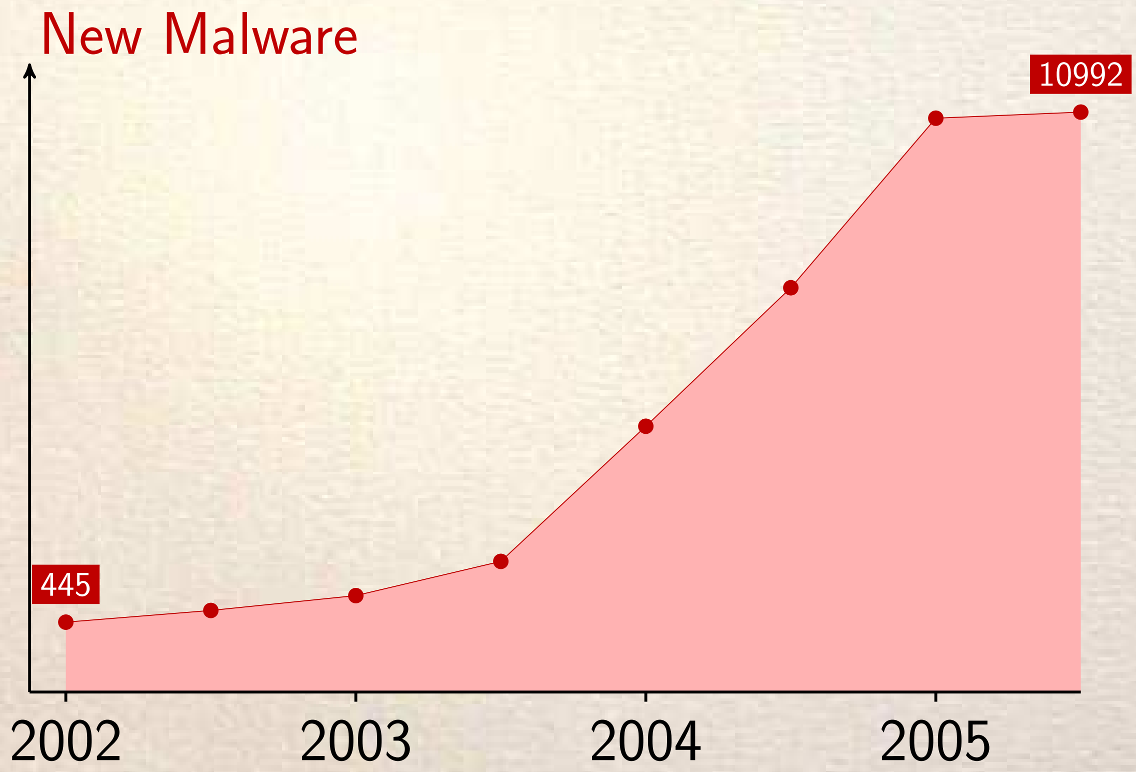
$$\mathcal{D}(P, M) = \begin{cases} \text{True} & \text{if } P \text{ is infected with } M \\ \text{False} & \text{otherwise} \end{cases}$$

An ideal malware detector detects all and only the programs infected with M, i.e., it is sound and complete.

- Sound = no false positives (no false alarms)
- Complete = no false negatives (no missed alarms)

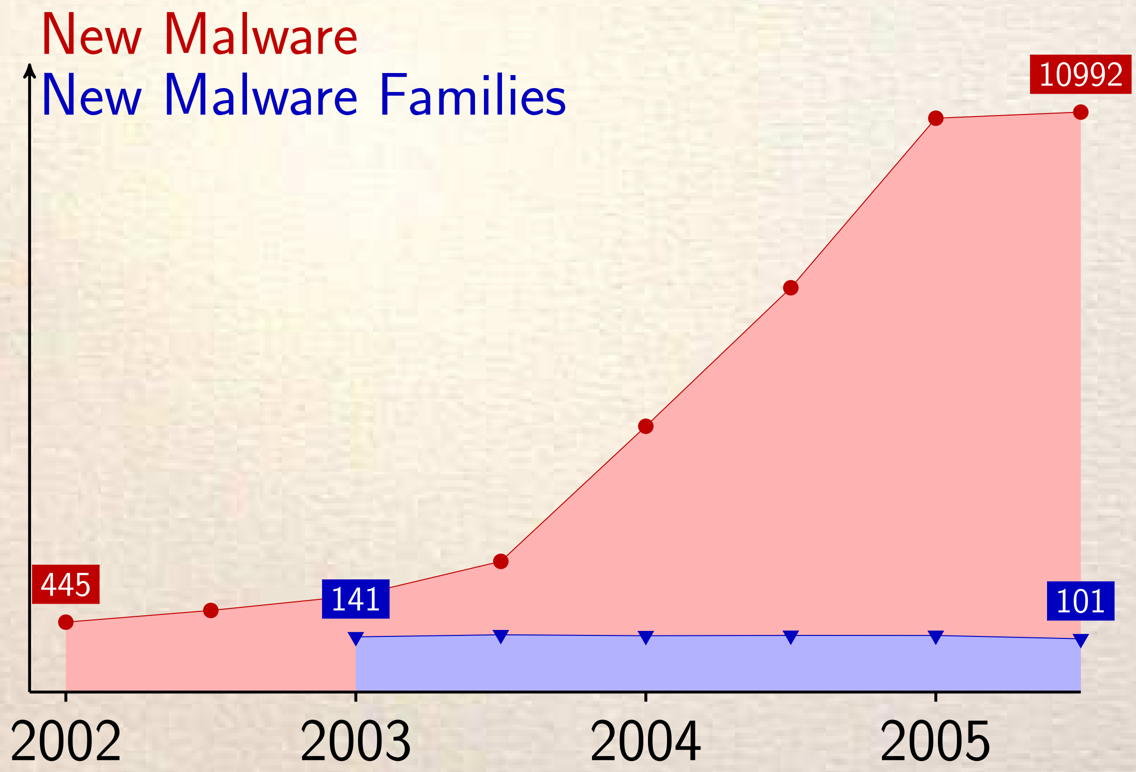
Malware Trends

There is more malware every year.



MALWARE TRENDS

There is more malware every year.



But the number of malware families has almost no variation.

Beagle family has 197 variants (as on Jan. 2007).
Warezov family has 218 variants (as on Jan. 2007).

SW PROTECTION VS. SW ATTACKS

malicious SW

malicious host

SW PROTECTION VS. SW ATTACKS

malicious SW

viruses

worms

malicious host

SW PROTECTION VS. SW ATTACKS

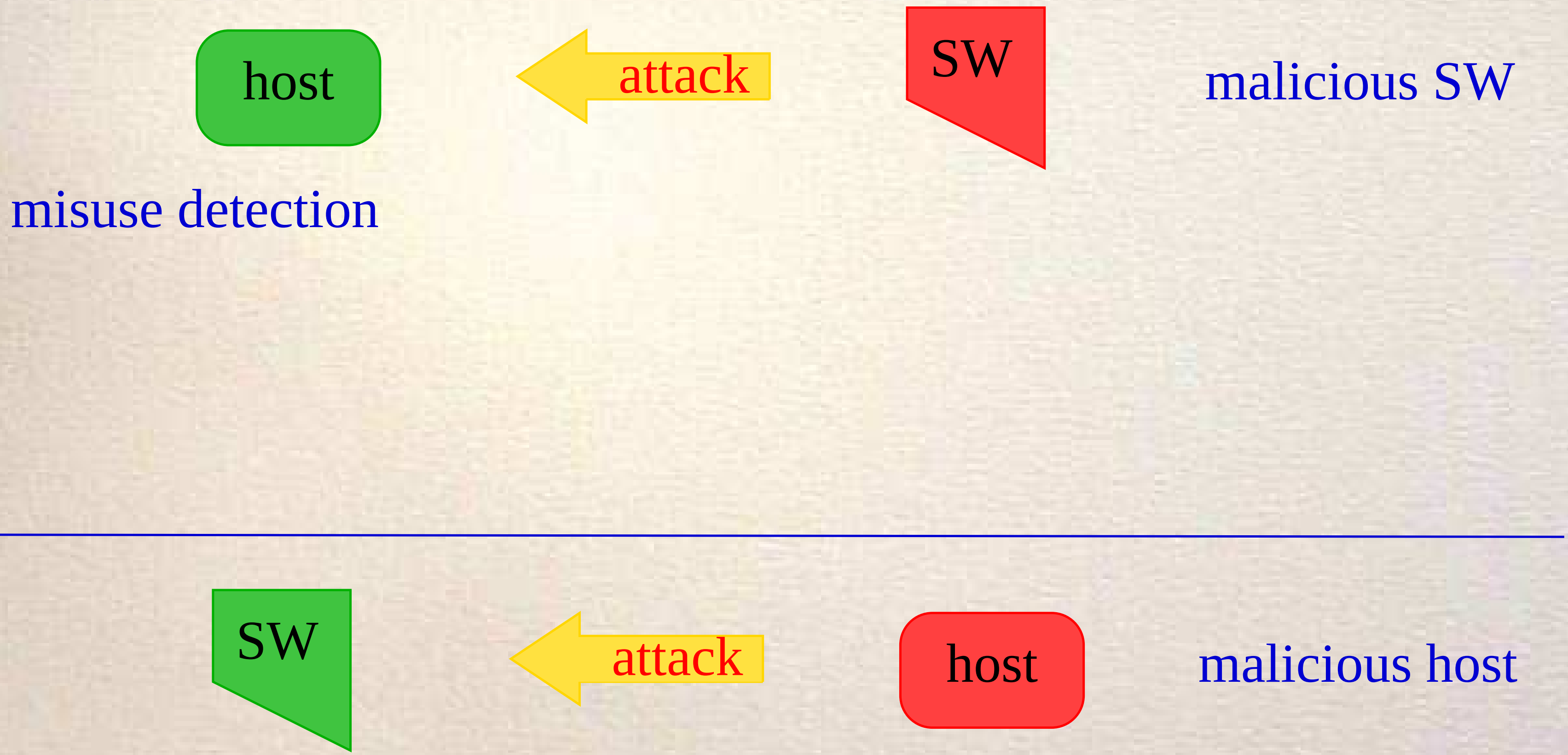
malicious SW

viruses
worms

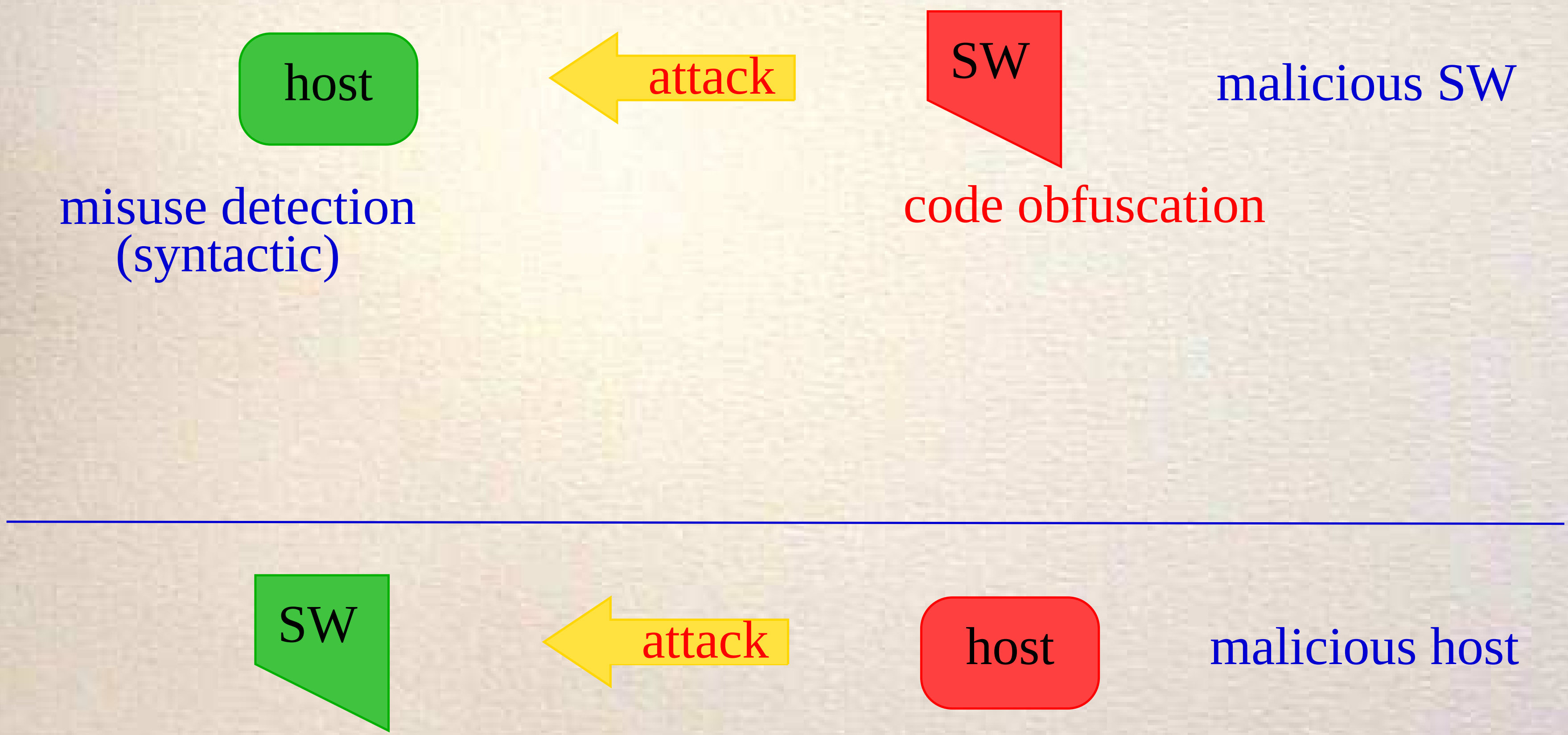
malicious host

IP
integrity

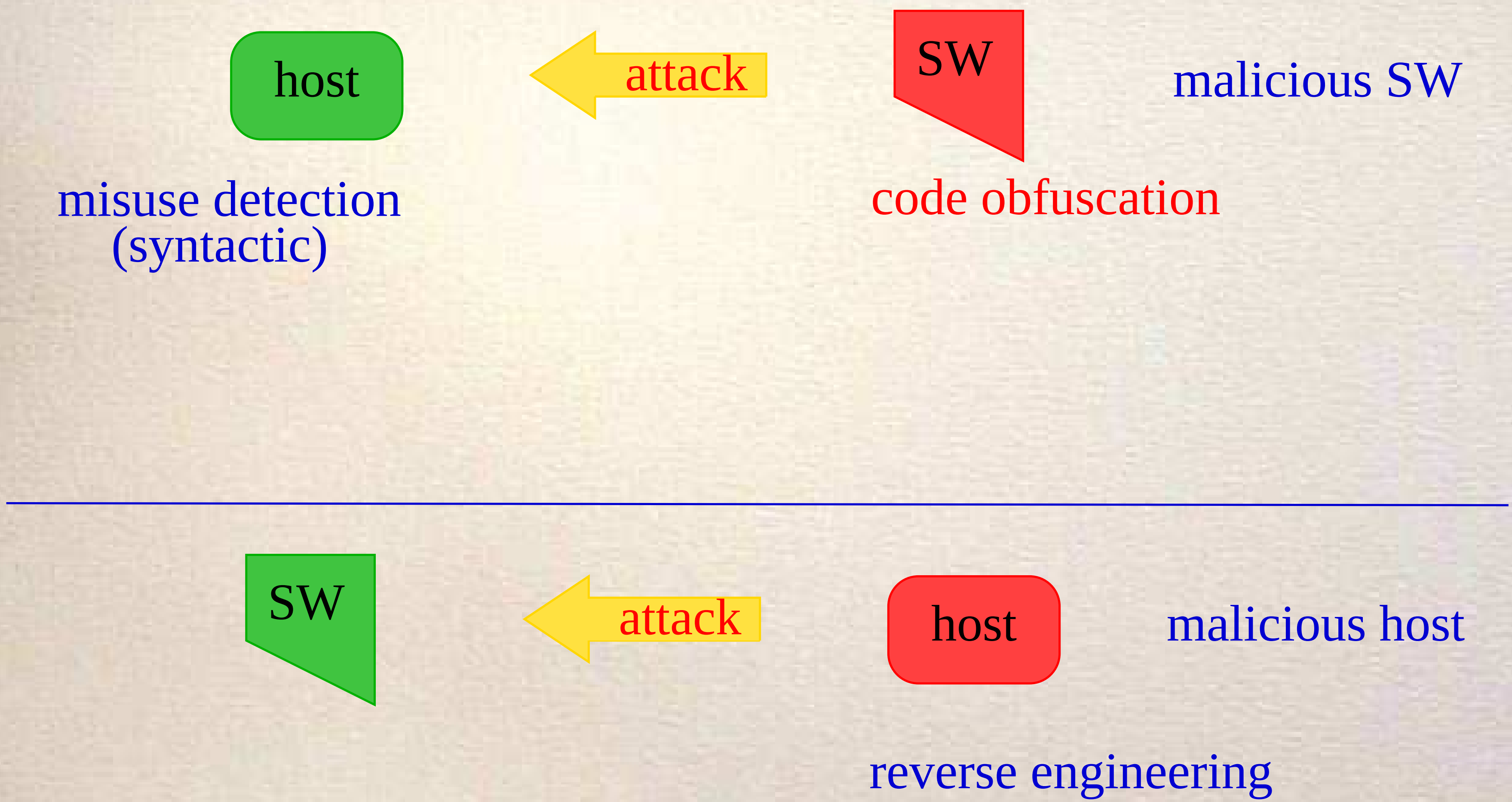
SW PROTECTION VS. SW ATTACKS



SW PROTECTION VS. SW ATTACKS



SW PROTECTION VS. SW ATTACKS



SW PROTECTION VS. SW ATTACKS

malicious SW

misuse detection
(syntactic)

code obfuscation

malicious host

code obfuscation

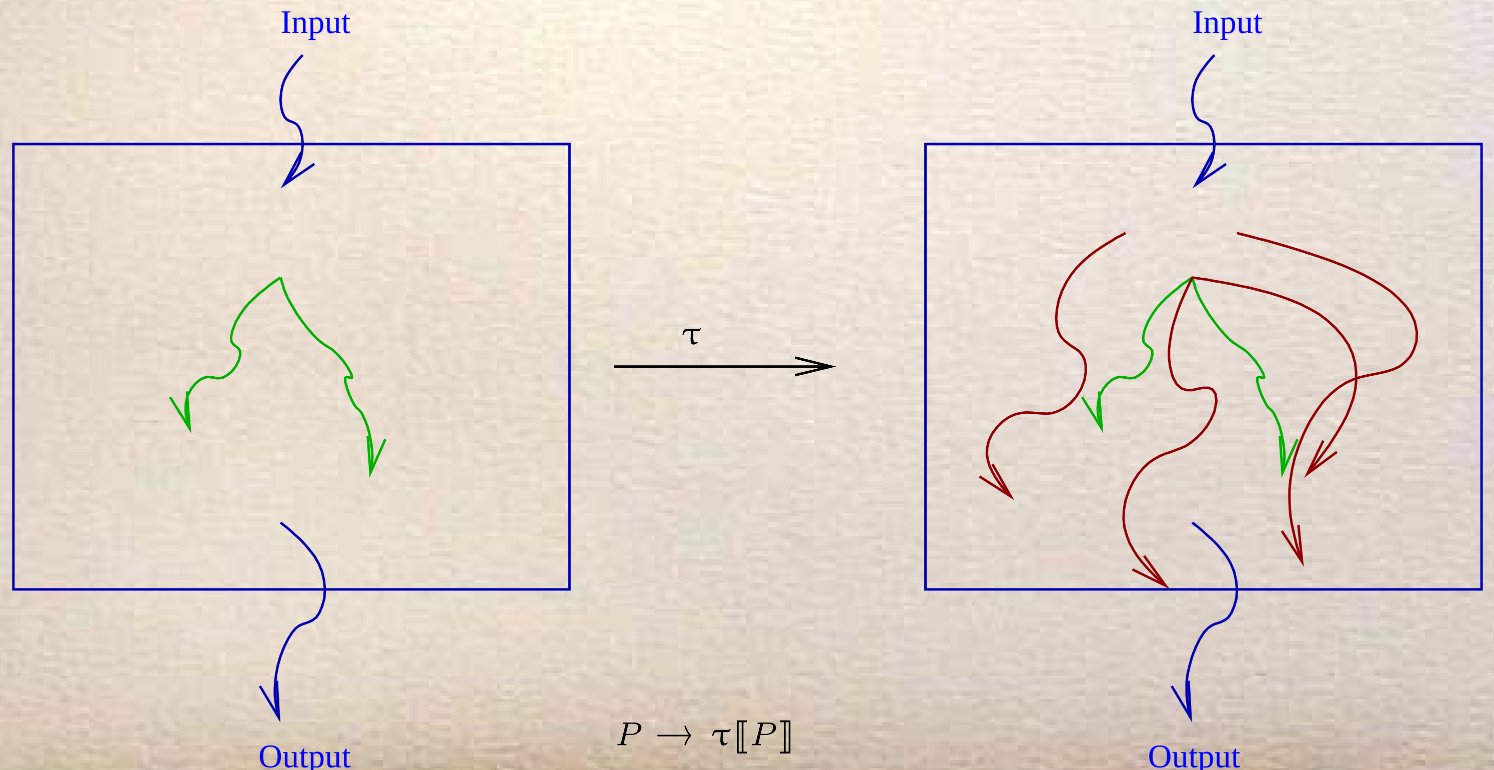
reverse engineering
(behaviour)

SW PROTECTION VS. SW ATTACKS

Protection by obscurity: Code Obfuscation

$\tau : \mathbb{P} \to \mathbb{P}$ is a code obfuscation if it is an obfuscating compiler:

- it is potent: $\tau(P)$ is *more complex* (ideally unintelligible) than P;
- it preserves the observational behaviour of programs $[\![\tau(P)]\!] = [\![P]\!]$ [C. Collberg et al. '97, '98].



$P \to \tau[\![P]\!]$

PROTECTION BY OBSCURITY: CODE OBFUSCATION

$\tau : \mathbb{P} \to \mathbb{P}$ is a code obfuscation if it is an obfuscating compiler:

- it is potent: $\tau(P)$ is *more complex* (ideally unintelligible) than P;
- it preserves the observational behaviour of programs $[\![\tau(P)]\!] = [\![P]\!]$ [C. Collberg et al. '97, '98].

The limit. Obfuscating programs is (im)possible:

Even under restrictive hypothesis a general purpose obfuscator generating perfectly unintelligible code (virtual black-box) does not exist! [Barak et al. '01].

The challenge. Design obfuscators that work against specific attacks

Extensional properties of programs are undecidable [Rice '53].
....so formal methods and static analysis are born!

An Example

(Pseudo-)Code:

```
mov eax, [edx+0Ch]
push ebx
push [eax]
call ReleaseLock
```

AN EXAMPLE

(Pseudo-)Code:

```
mov eax, [edx+0Ch]
push ebx
push [eax]
call ReleaseLock
```

Obfuscated code (junk):

```
mov eax, [edx+0Ch]
inc eax
push ebx
dec eax
push [eax]
call ReleaseLock
```

AN EXAMPLE

(Pseudo-)Code:

```
mov eax, [edx+0Ch]
push ebx
push [eax]
call ReleaseLock
```

Obfuscated code (junk + reordering):

```
mov eax, [edx+0Ch]
jmp +3
push ebx
dec eax
jmp +4
inc eax
jmp -3
call ReleaseLock
jmp +2
push [eax]
jmp -2
```

State of the Art

[Collberg et al. ’97, ’98]

- opaque predicate insertion
- code flattening,
- variable splitting,
- bogus code insertion,
- spurious aliases

Potency measure by standard metrics:

code size, number of predicates, number of methods in OO code, height of inheritance, and variable dependence length

State of the Art

[Wang et al. '00]

- spurious aliases

Potency measure by complexity of static analysis

- 1-level aliasing is easy **P** [Banning '79]
- ≥ 2-level aliasing is hard **NP** [Horowitz '97]
- with dynamic memory allocation is undecidable!!

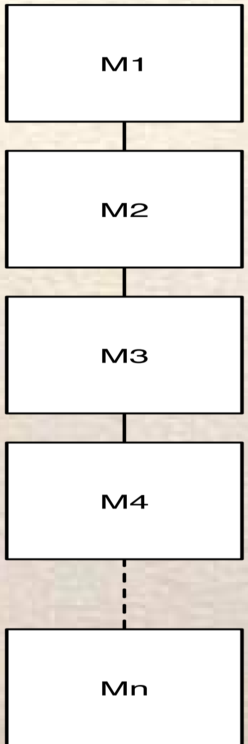
understanding control-flow = solve a ≥ 2-level aliasing problem

State of the Art

[Cloackware '00]

- code flattening

Potency is related with the PSPACE complexity of reachability in dispatchers

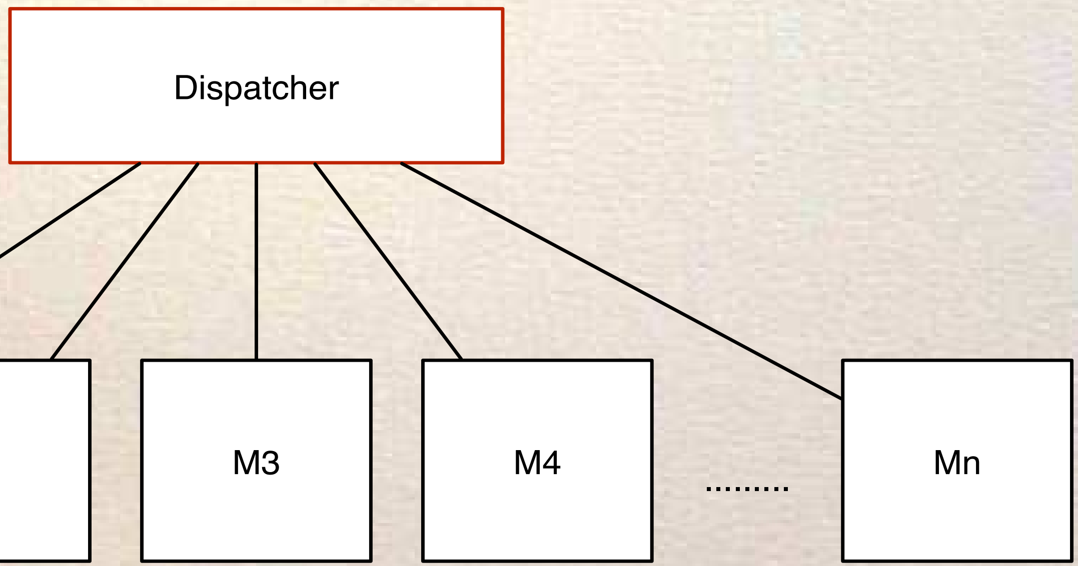


State of the Art

[Cloackware '00]

- code flattening

Potency is related with the PSPACE complexity of reachability in dispatchers



State of the Art

[Drape et al '05 and '07]

- data obfuscation
- slicing obfuscation: enlarging slices by adding dependencies

Potency is related with data-refinement

- If D is a data-type, $\mathfrak{D}$ is a refinement of D if $\langle \mathfrak{D}, \alpha, \gamma, D \rangle$ is a GI
- Correctness: $[\![P]\!] = \alpha \circ [\![\tau(P)]\!] \circ \gamma$
- ...i.e.: P and $\gamma; \tau(P); \alpha$ are observationally equivalent!

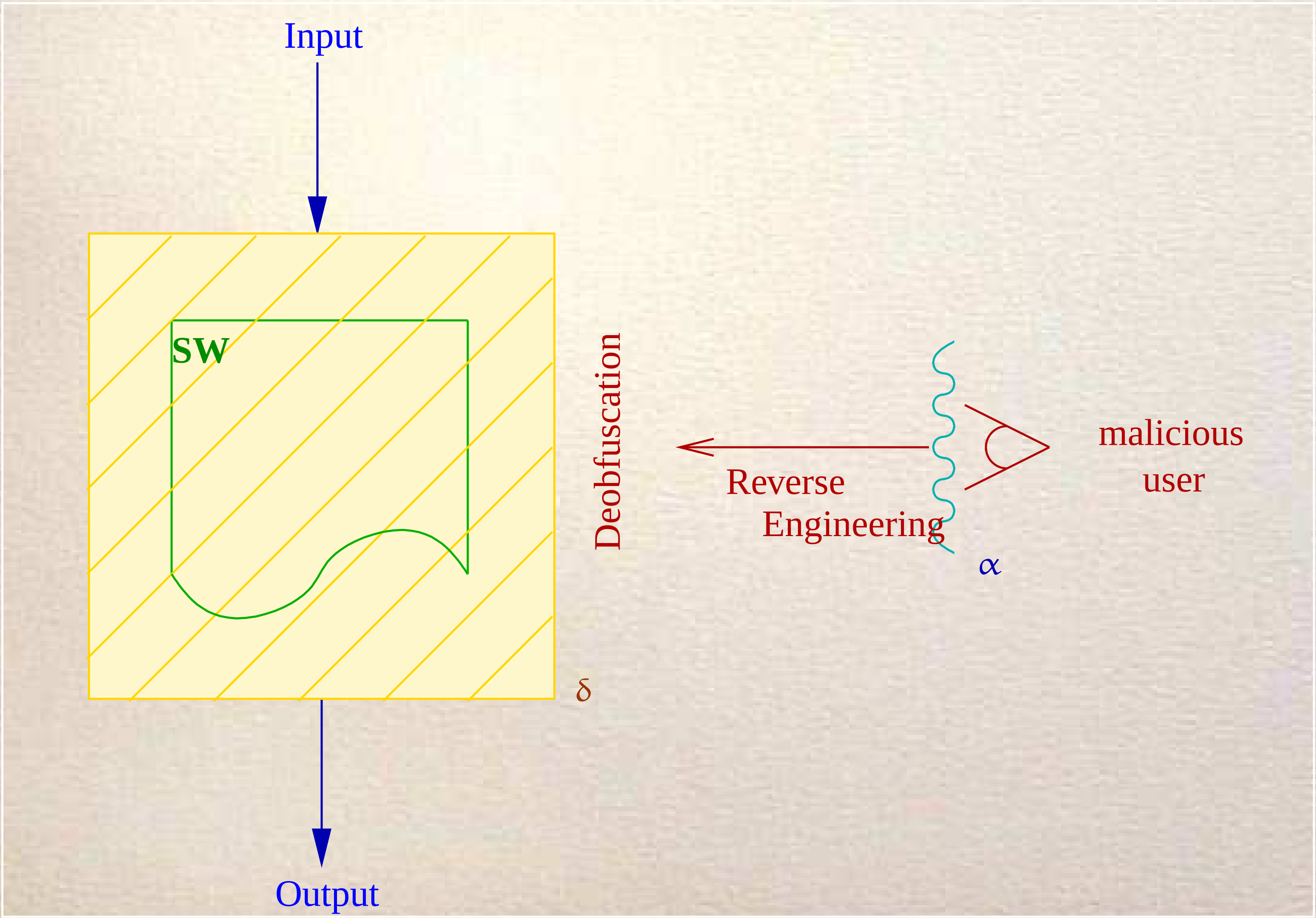
Obfuscation corresponds precisely to concretise (in the sense of abstract interpretation) a data-type

The problem: hiding and unveiling in SW

- Understanding programs corresponds to understand their semantics
  - ✔ The attacker is an interpreter (static or dynamic)
- Potency is related with the degree of precision of the interpreter
  - ✔ $\tau(P)$ is an obfuscation of P if the interpretation of $\tau(P)$ fails (is less precise) than the same interpretation of P: $[\![P]\!] \leq [\![\tau(P)]\!]$
  - ✔ In this case τ defeats $[\![\cdot]\!]$!!
- We need a theory of interpreters at different levels of abstraction

We need Abstract Interpretation

THE PROBLEM: HIDING AND UNVEILING IN SW



WHY ABSTRACT INTERPRETATION?

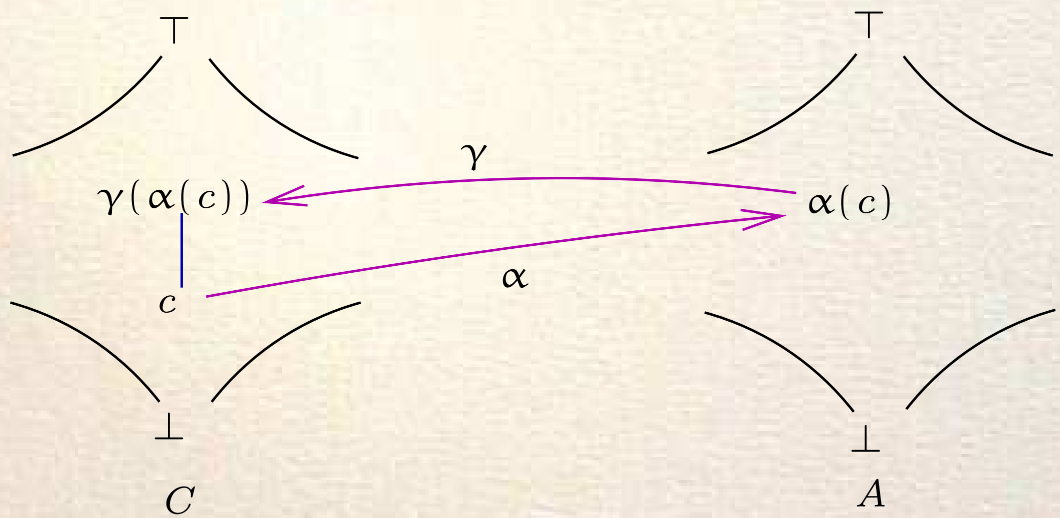
- The attacker
  - Reverse engineering needs (static or dynamic) analysis
  - Watermark extraction or violation need (static or dynamic) analysis
- The defender
  - Can exploit attack flaws to embed information
  - Can exploit attack limitations (complexity, accuracy, time, space etc) for obscuring information

Abstract Interpretation (1977) is the most general model for the (static or dynamic) approximation of semantics of discrete dynamic systems

- Including: Static program analysis, type checking and type inference, model checking and predicate abstraction, trajectory evaluation, testing, proof systems, etc.

ABSTRACT INTERPRETATION

Design approximate semantics of programs [Cousot & Cousot ’77, ’79].



Galois Connection: $\langle C, \alpha, \gamma, A \rangle$, A and C are complete lattices.

$\langle uco(C), \sqsubseteq \rangle$ set of all possible abstract domains,

$A_1 \sqsubseteq A_2$ if A_1 is more concrete than A_2

Abstract Interpretation

[Cousot & Cousot '79]

- A program P
- A domain of computation for P: C typically a complete lattice
- Semantic specification (interpreter): $[\![P]\!] : C \longrightarrow C$
- (Approximate) observable properties: $\rho \in uco(C)$
- DERIVE A SOUND APPROXIMATE SPECIFICATION $[\![P]\!]^\sharp$

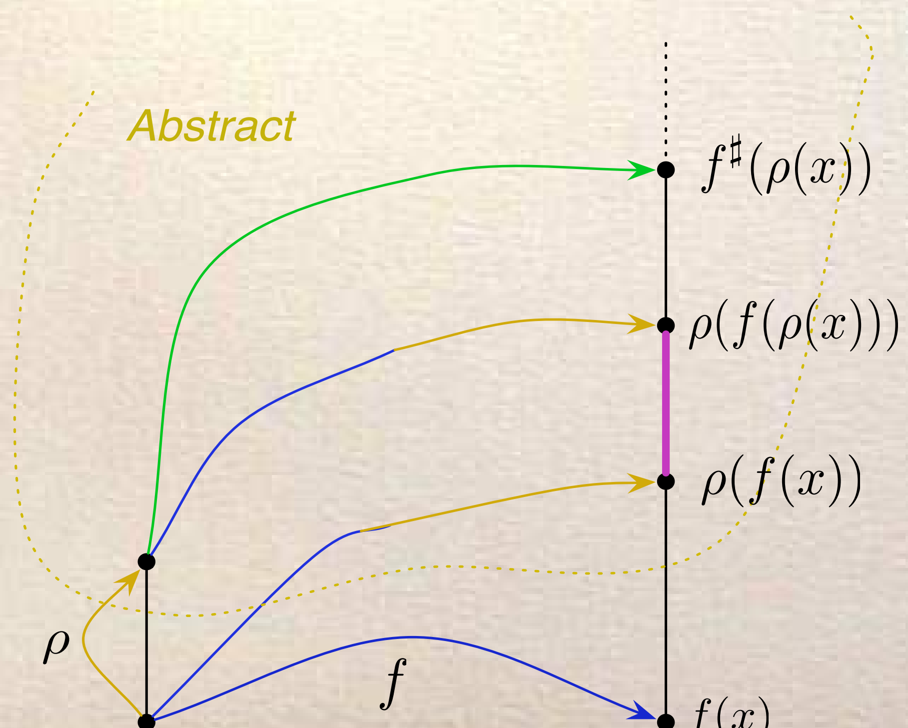
  $$\rho([\![P]\!](x)) \leq [\![P]\!]^\sharp(x)$$

- THE LIMIT CASE: COMPLETENESS

  $$\rho([\![P]\!](x)) = [\![P]\!]^\sharp(x) \text{ iff } \rho([\![P]\!](x)) = \rho([\![P]\!](\rho(x)))$$

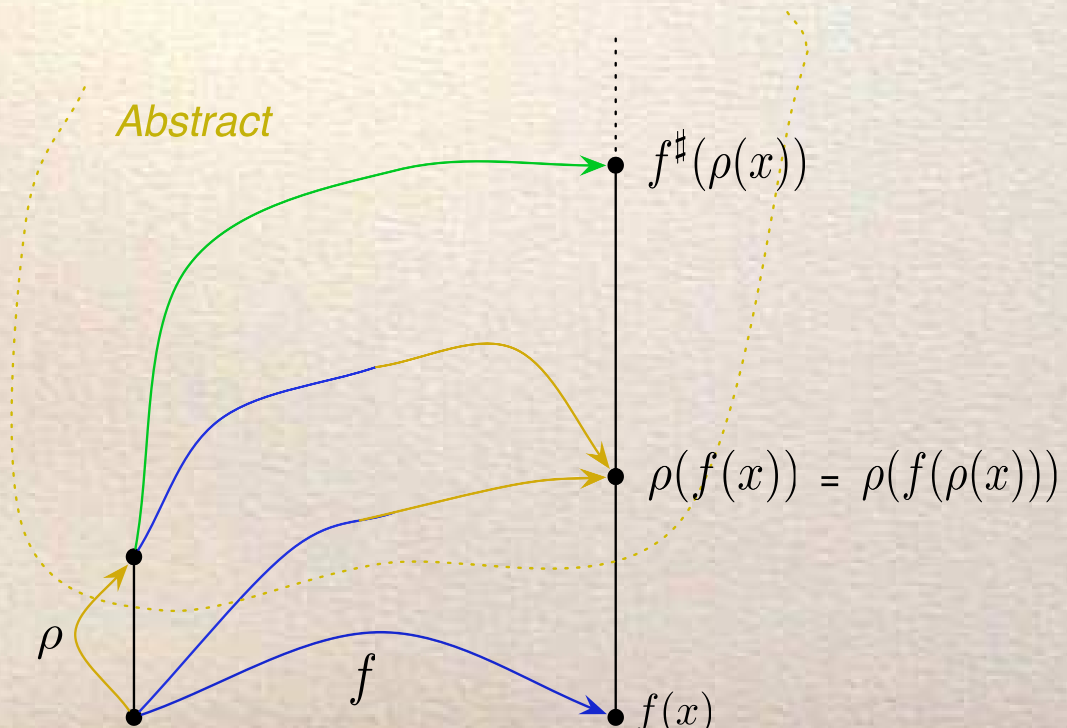
COMPLETENESS IN ABSTRACT INTERPRETATION

- BACKWARD SOUNDNESS: NO INFORMATION IS LOST BY APPROXIMATING THE INPUT/OUTPUT
- $\rho \circ f \leq \rho \circ f \circ \rho$



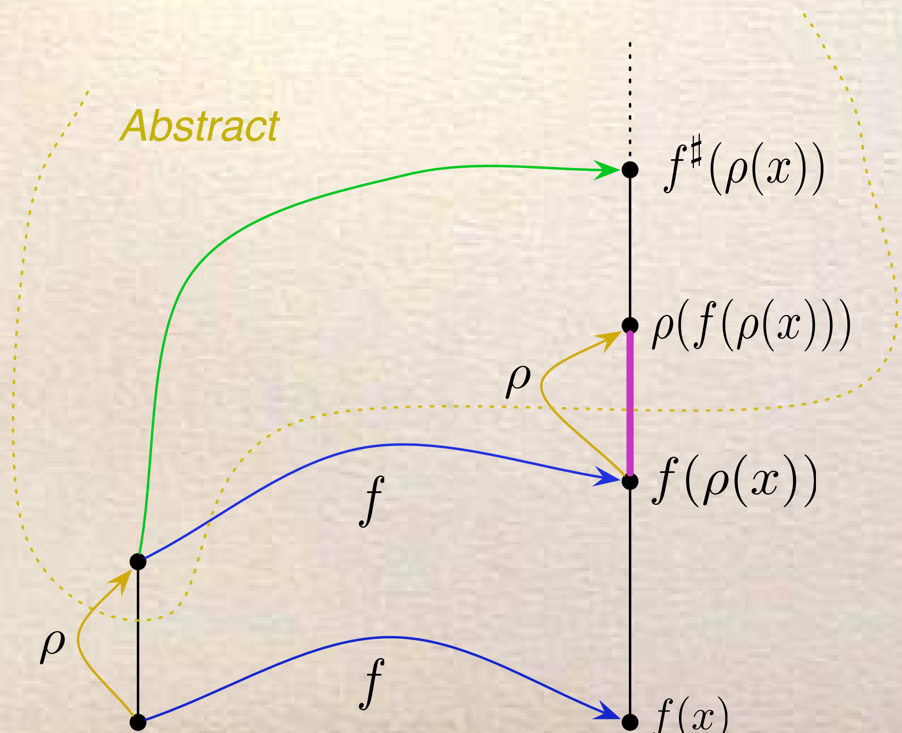
COMPLETENESS IN ABSTRACT INTERPRETATION

- BACKWARD COMPLETENESS: NO LOSS OF PRECISION IS ACCUMULATED BY APPROXIMATING THE INPUT
- $\rho \circ f = \rho \circ f \circ \rho$



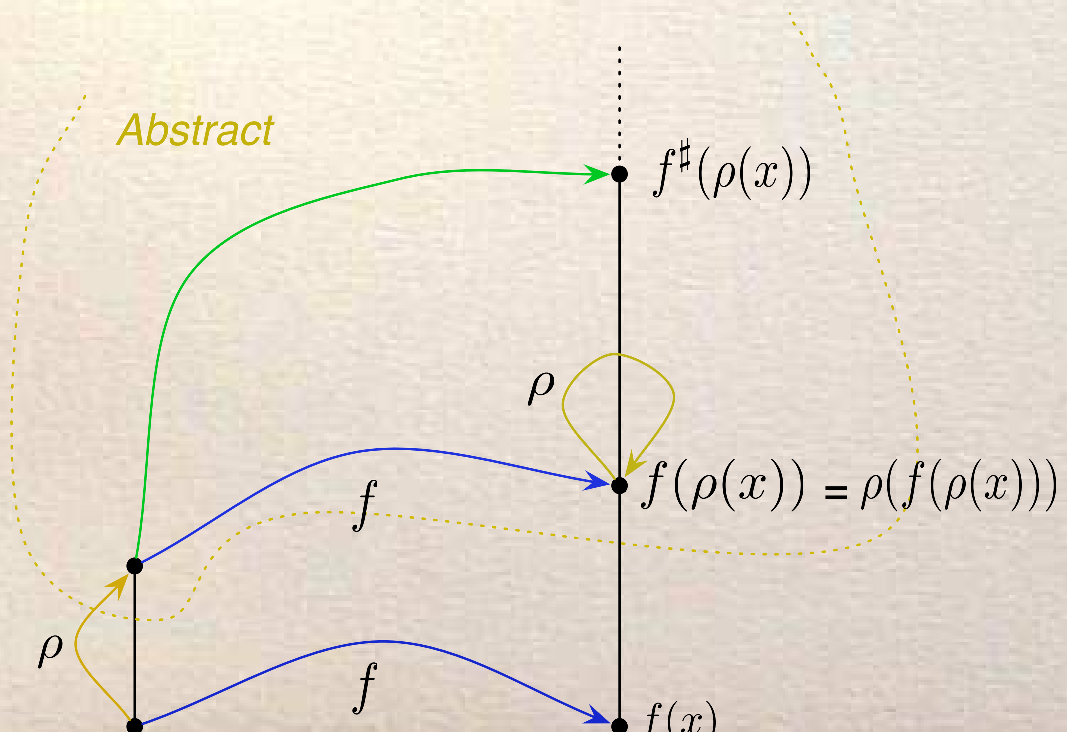
Completeness in Abstract Interpretation

- Forward Completeness: no information is lost by approximating the output
- $f \circ \rho \leq \rho \circ f \circ \rho$



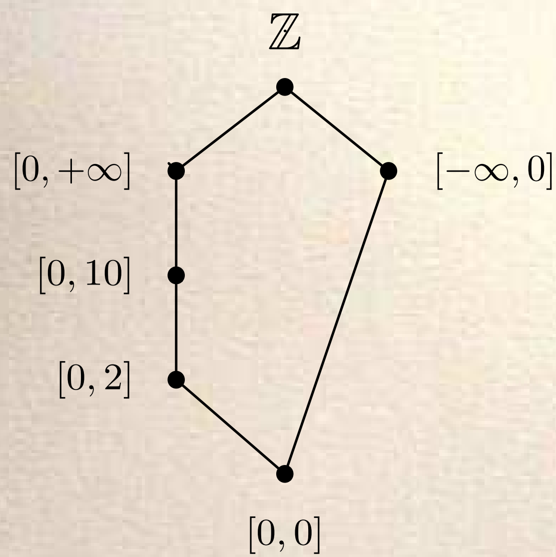
COMPLETENESS IN ABSTRACT INTERPRETATION

- FORWARD COMPLETENESS: NO INFORMATION IS LOST BY APPROXIMATING THE OUTPUT
- $f \circ \rho = \rho \circ f \circ \rho$



AN EXAMPLE

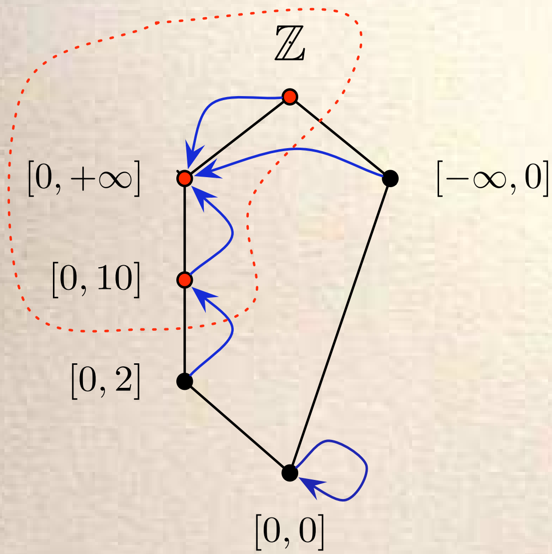
A SIMPLE EXAMPLE IN INTERVAL ANALYSIS



- A simple domain of intervals

An example

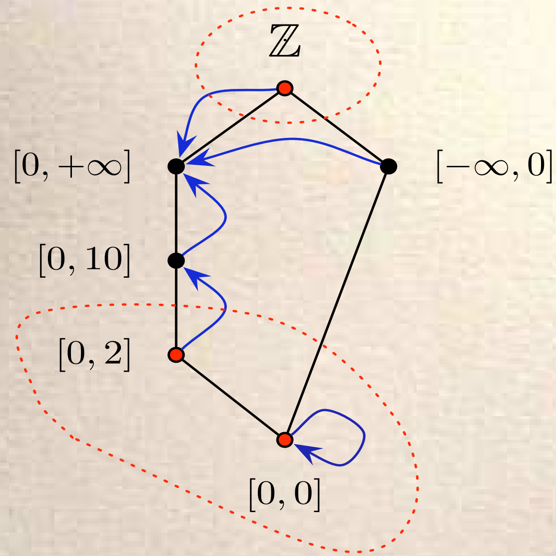
A simple example in Interval analysis



- A simple domain of intervals
- $sq(X) = \left\{ x^2 \;\middle|\; x \in X \right\}$
- $\{\mathbb{Z}, [0,+\infty], [0,10]\}$ is Forward but not Backward complete

AN EXAMPLE

A SIMPLE EXAMPLE IN INTERVAL ANALYSIS



- A simple domain of intervals
- $sq(X) = \left\{\, x^2 \;\middle|\; x \in X \,\right\}$
- $\{\mathbb{Z}, [0,+\infty], [0,10]\}$ is Forward but not Backward complete
- $\{\mathbb{Z}, [0,2], [0,0]\}$ is Backward but not Forward complete

OBSCURITY BY INCOMPLETENESS

Failing precision means failing completeness!

Obfuscating programs is making abstract interpreters incomplete

- Let $\rho \in uco(\Sigma)$ with Σ semantic objects (data, traces etc)
- A program transformation $\tau : \mathbb{P} \to \mathbb{P}$: $[\![P]\!] = [\![\tau(P)]\!]$.
- ρ $\mathcal{B}$-complete for $[\![\cdot]\!]$: $\rho([\![P]\!]) = [\![P]\!]^{\rho}$

τ obfuscates P if

$$[\![P]\!]^{\rho} \sqsubset [\![\tau(P)]\!]^{\rho} \iff \rho([\![\tau(P)]\!]) \sqsubset [\![\tau(P)]\!]^{\rho}$$

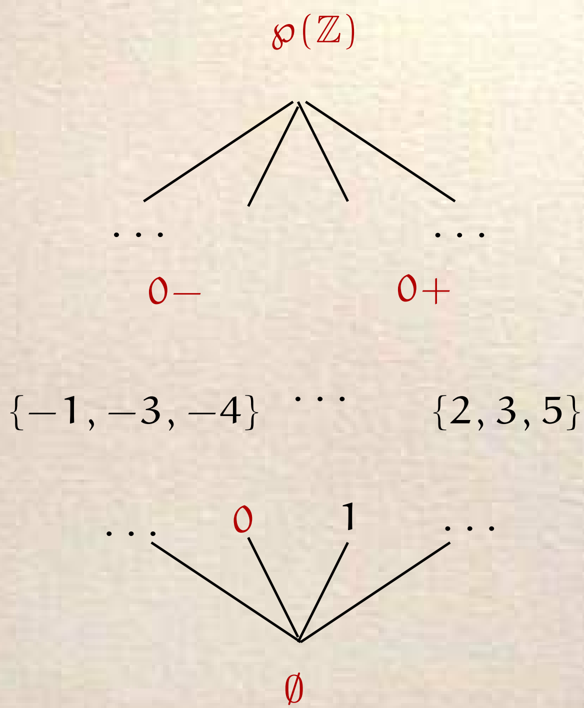
OBSCURITY BY INCOMPLETENESS

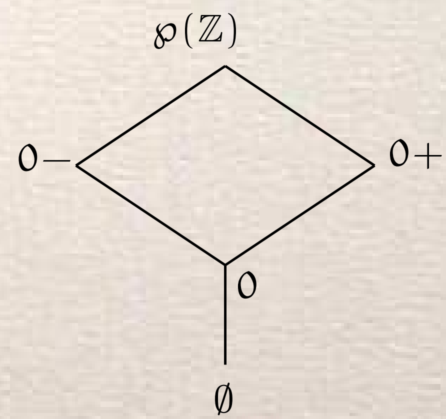
Failing precision means failing completeness!

Obfuscating programs is making abstract interpreters incomplete

C : $\mathsf{x} = \mathsf{a} * \mathsf{b}$

$Sign$ is an abstraction of $\wp(\mathbb{Z})$:





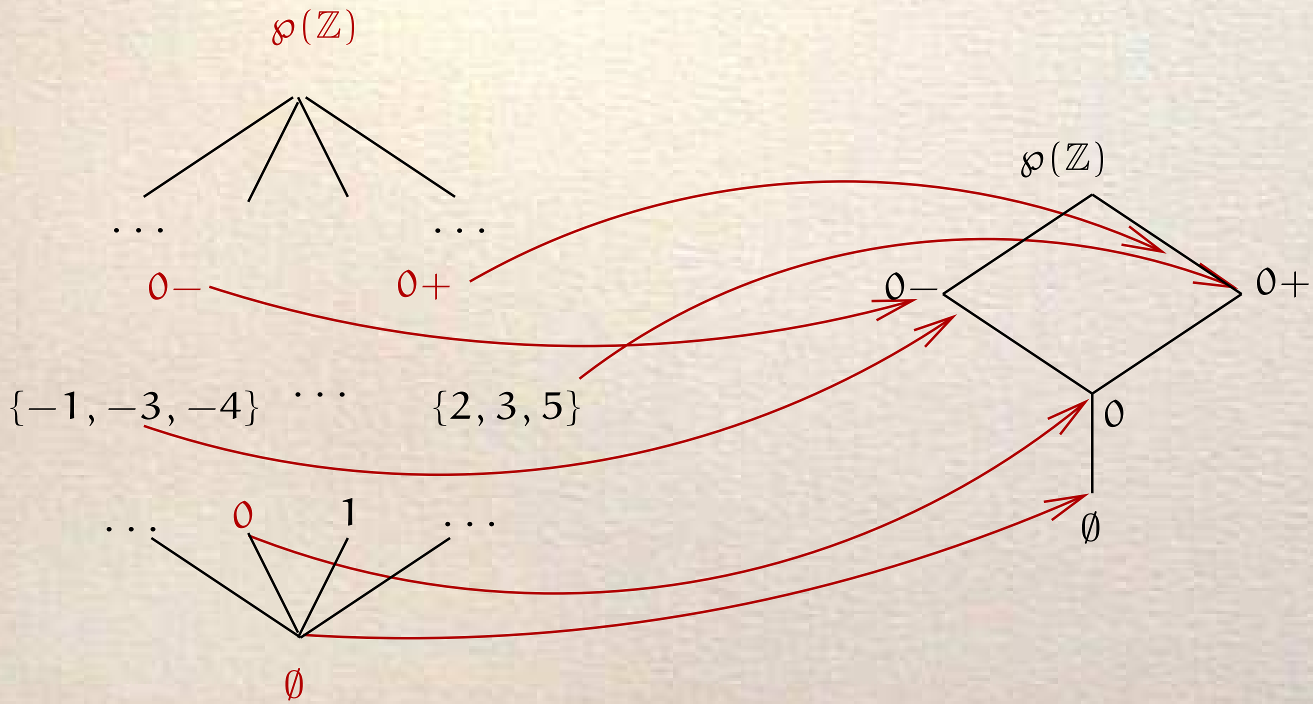
Obscurity by incompleteness

Failing precision means failing completeness!

Obfuscating programs is making abstract interpreters incomplete

C : x $=$ a $*$ b

$Sign$ is an abstraction of $\wp(\mathbb{Z})$:



OBSCURITY BY INCOMPLETENESS

Failing precision means failing completeness!

Obfuscating programs is making abstract interpreters incomplete

$$C: \quad \mathtt{x} = \mathtt{a} * \mathtt{b} \quad \longrightarrow \quad \tau(C): \begin{array}{l} \mathtt{x} = 0; \\ \mathtt{if}\ \mathtt{b} \leq 0\ \mathtt{then}\ \{\mathtt{a} = -\mathtt{a}; \mathtt{b} = -\mathtt{b}\}; \\ \mathtt{while}\ \mathtt{b} \neq 0\ \{\mathtt{x} = \mathtt{a} + \mathtt{x}; \mathtt{b} = \mathtt{b} - 1\} \end{array}$$

- Sign is complete for C
  - ✔ $[\![C]\!]^{\mathit{Sign}} = \lambda \mathsf{a}, \mathsf{b}.\ \mathit{Sign}(\mathsf{a} * \mathsf{b})$

- Sign is incomplete for $\tau(C)$
  - ✔ $[\![\tau(C)]\!]^{\mathit{Sign}} = \lambda \mathsf{a}, \mathsf{b}. \begin{cases} 0 & \text{if } \mathsf{a} = 0 \vee \mathsf{b} = 0 \\ \wp(\mathbb{Z}) & \text{otherwise} \end{cases}$

GENERALISING DATA-REFINEMENT I

We consider variable splitting

$v \in \mathit{Var}(P)$ is split into $\langle v_1, v_2 \rangle$ such that
$v_1 = f_1(v)$, $v_2 = f_2(v)$ and $v = g(v_1, v_2)$

$$f_1(v) = v \div 10$$
$$f_2(v) = v \mod 10$$
$$g(v_1, v_2) = 10 \cdot v_1 + v_2$$

And the interval analysis: $\iota(x) = [\min(x), \max(x)]$

$$P : \left[\begin{array}{l} v = 0; \\ \textbf{while } v < N \ \{v++\} \end{array} \right. \qquad [\![P]\!]^{\iota} = \lambda v.\ [0, N]$$

GENERALISING DATA-REFINEMENT I

We consider variable splitting

$v \in \mathit{Var}(P)$ is split into $\langle v_1, v_2\rangle$ such that
$v_1 = f_1(v)$, $v_2 = f_2(v)$ and $v = g(v_1, v_2)$

$$f_1(v) = v \div 10$$
$$f_2(v) = v \mod 10$$
$$g(v_1, v_2) = 10 \cdot v_1 + v_2$$

And the interval analysis: $\iota(x) = [\min(x), \max(x)]$

$$\tau(P): \left[\begin{array}{l} v_1 = 0; \\ v_2 = 0; \\ \textbf{while} \quad 10 \cdot v_1 + v_2 < N \; \{ \\ \qquad v_1 = v_1 + (v_2 + 1) \div 10 \\ \qquad v_2 = (v_2 + 1) \mod 10 \\ \quad \}; \end{array}\right.$$
$$c: \quad v = 10 \cdot v_1 + v_2$$

$$\begin{array}{ll} [\![\tau(P); c]\!]^{\iota} & = \\ \lambda v.\ 10 \odot [0, \frac{N \ominus [0,9]}{10}] \oplus [0, 9] & = \\ \lambda v.\ [0, N] \oplus [0, 9] & = \\ \lambda v.\ [0, N+9] & \end{array}$$

GENERALISING DATA-REFINEMENT II

We consider array splitting for weakening the invariant of Fibonacci's

$$\mathbf{Inv} = 2 \le i \le N \wedge \forall j \in [2, i].\ a[j] = a[j-1] + a[j-2]$$

The invariant **Inv** can be generated by relational interval-**Fib** analysis

- $\eta = \alpha^+ \circ \alpha$ where
- $$\alpha(X) = \begin{cases} \mathbf{Fib} & \text{if } \forall \langle S, x \rangle \in X.\ S \subseteq D_x \wedge \quad (S = \{0\} \wedge x[0] = 0) \vee \\ & \quad (S = \{0, 1\} \wedge x[0] = 0 \wedge x[1] = 1) \vee \\ & \quad (\forall j \in S.\ x[j] = x[j-1] + x[j-2]) \\ \mathbf{Any} & \text{otherwise} \end{cases}$$
- $I \longrightarrow \mathbf{Fib}$ represents Fibonacci's sequences until $\max(I)$
- $I \longrightarrow \mathbf{Any}$ represents any array with domain including I (no overlow)
- $[n, m] \longrightarrow \mathbf{Fib} = [n, m-1] \longrightarrow \mathbf{Fib} \oplus [n, m-2] \longrightarrow \mathbf{Fib}$

GENERALISING DATA-REFINEMENT II

We consider array splitting for weakening the invariant of Fibonacci's

$$\mathbf{Inv} = 2 \leq i \leq N \wedge \forall j \in [2, i].\ a[j] = a[j-1] + a[j-2]$$

$$P : \left[\begin{array}{l} a[0] = 0; \\ a[1] = 1; \\ i = 2; \\ \textbf{while} \quad i \leq N \ \{ \\ \qquad\qquad a[i] = a[i-1] + a[i-2]; \\ \qquad\qquad i++ \\ \qquad\quad \} \end{array}\right.$$

$$[\![P]\!]^{\mathfrak{u} \longrightarrow \eta} = a \in [0, N] \longrightarrow \mathbf{Fib} \wedge i \in [2, N+1]$$

GENERALISING DATA-REFINEMENT II

We consider array splitting for weakening the invariant of Fibonacci's

$$\textbf{Inv} = 2 \le i \le N \wedge \forall j \in [2, i].\ a[j] = a[j-1] + a[j-2]$$

$$\tau(P) : \left[\begin{array}{l} b[0] = 0; \\ c[0] = 1; \\ i = 2; \\ \textbf{while} \quad i \le N \,\{ \\ \qquad \textbf{if} \quad i \mod 2 == 0 \\ \qquad\qquad \{b[i \div 2] = c[(i-1) \div 2] + b[(i-2) \div 2]\} \\ \qquad\qquad \{c[i \div 2] = b[(i) \div 2] + c[(i-2) \div 2]\}; \\ \qquad\qquad\ i++ \\ \qquad\qquad \} \end{array} \right.$$

$$[\![\tau(P)]\!]^{\mathfrak{u} \longrightarrow \eta} = b, c \in [0, N \div 2] \longrightarrow \textbf{Any} \wedge i \in [2, N+1]$$

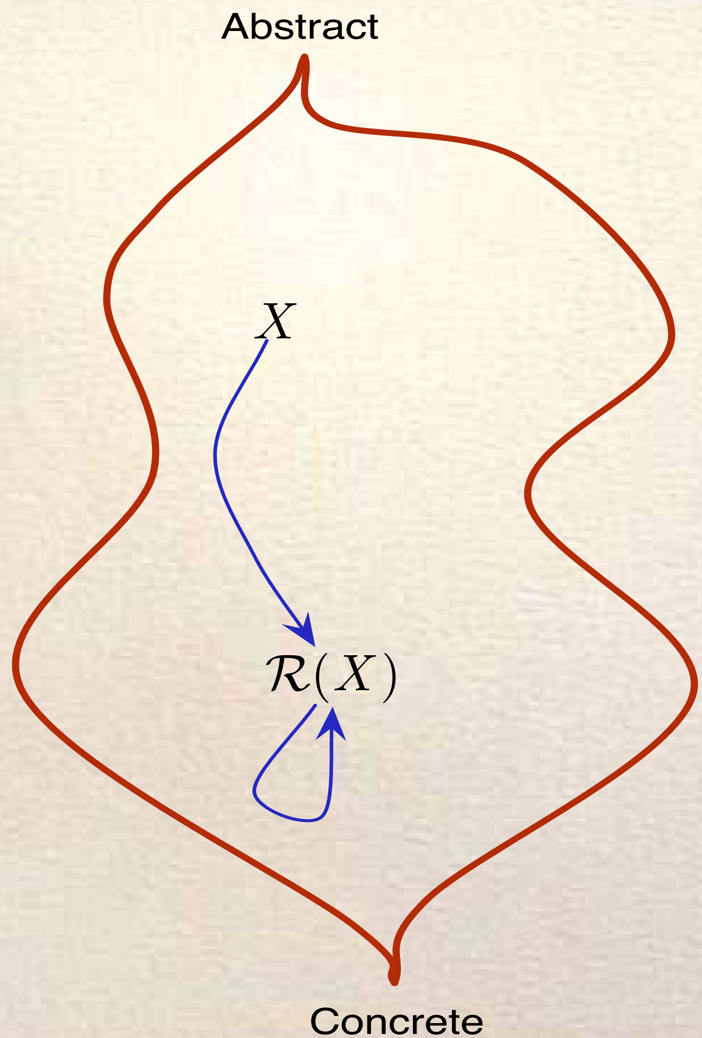
Generalising data-refinement II

We consider array splitting for weakening the invariant of Fibonacci's

$$\mathbf{Inv} = 2 \leq i \leq N \wedge \forall j \in [2, i].\ a[j] = a[j-1] + a[j-2]$$

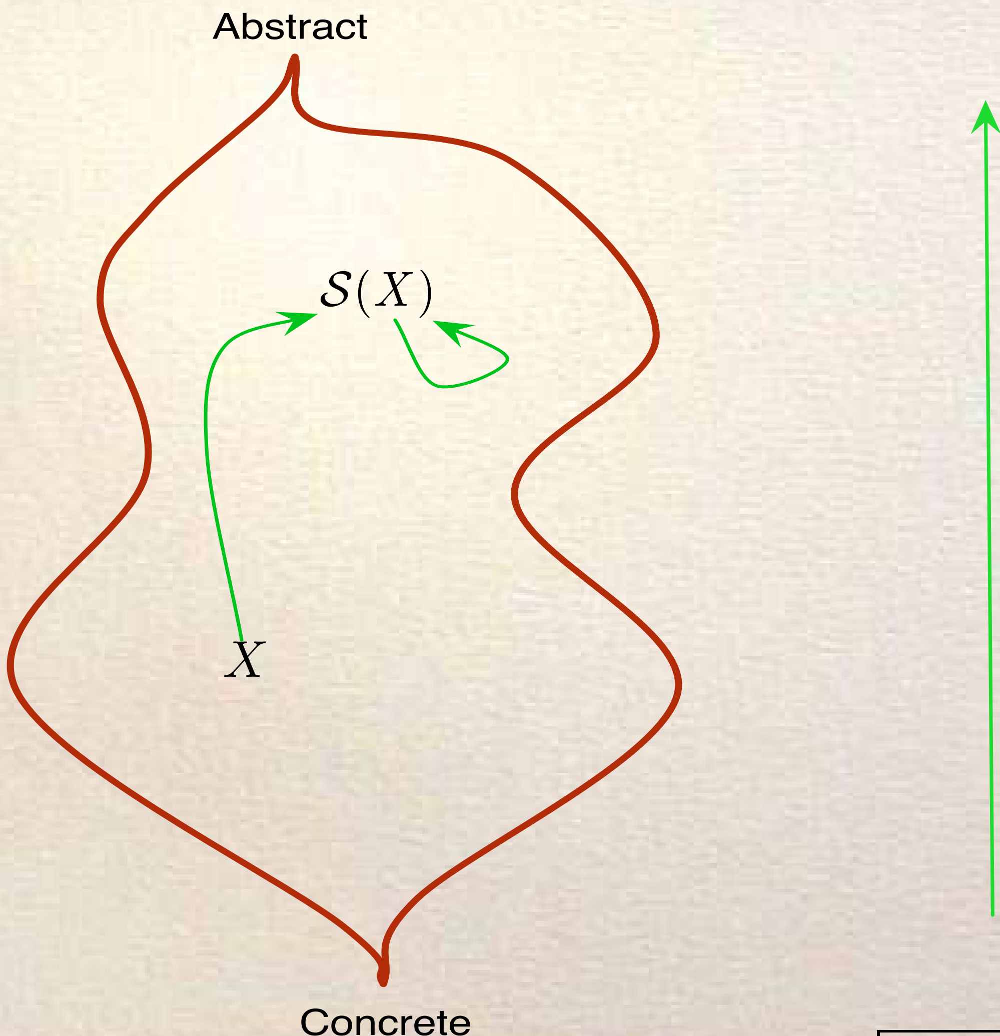
How can we attack $\tau(P)$ and get $\mathbf{Inv}$ back?

THE GEOMETRY OF ATTACKERS



lco – REFINEMENT

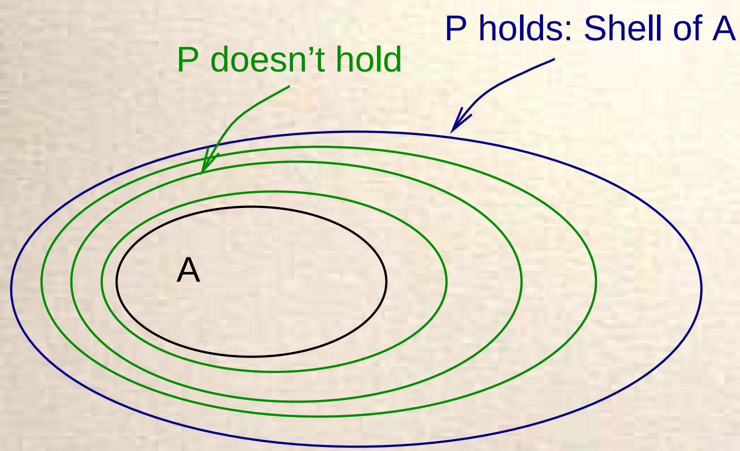
THE GEOMETRY OF ATTACKERS



$\boxed{uco - \text{SIMPLIFICATION}}$

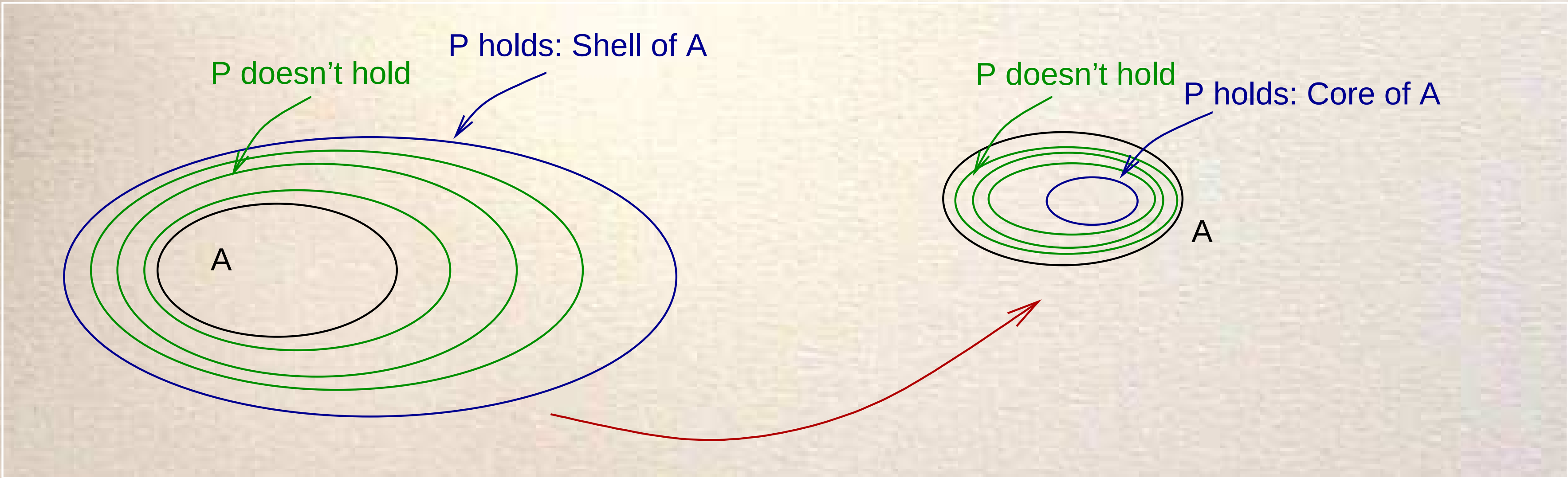
SHELL/CORE

Let P be completeness

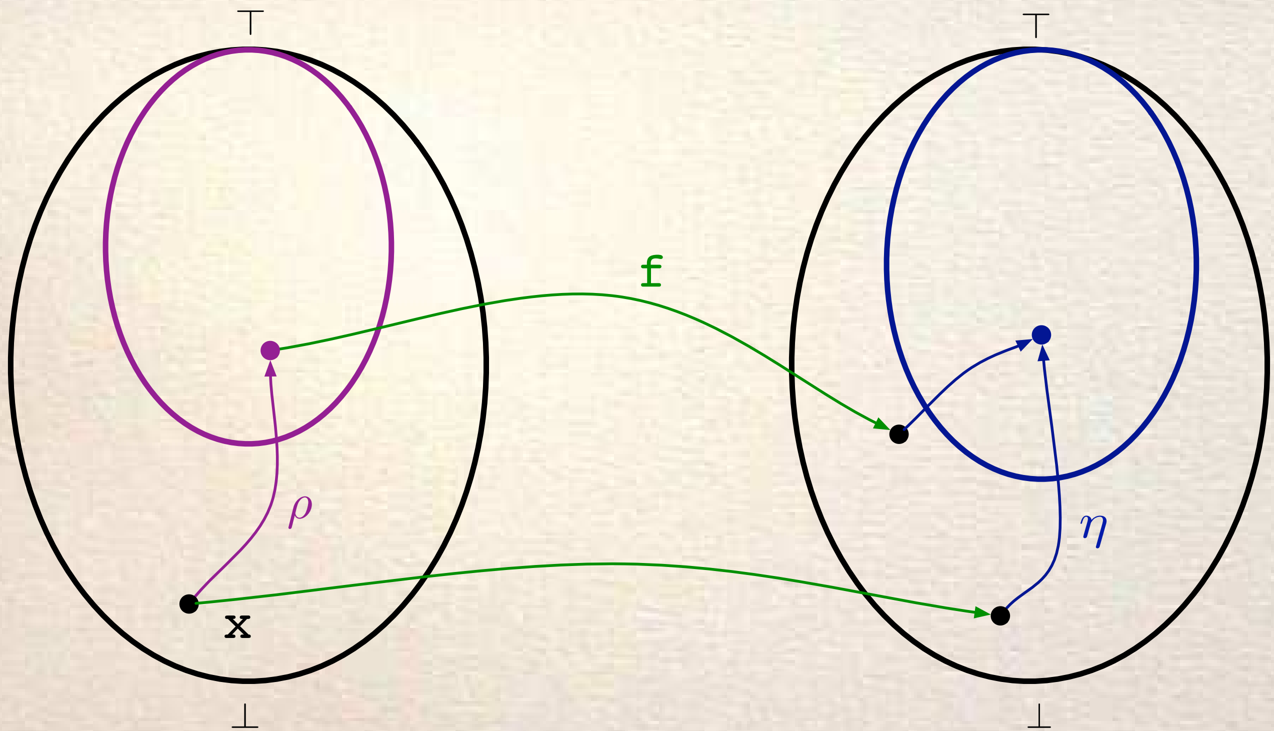


Shell/Core

Let P be completeness

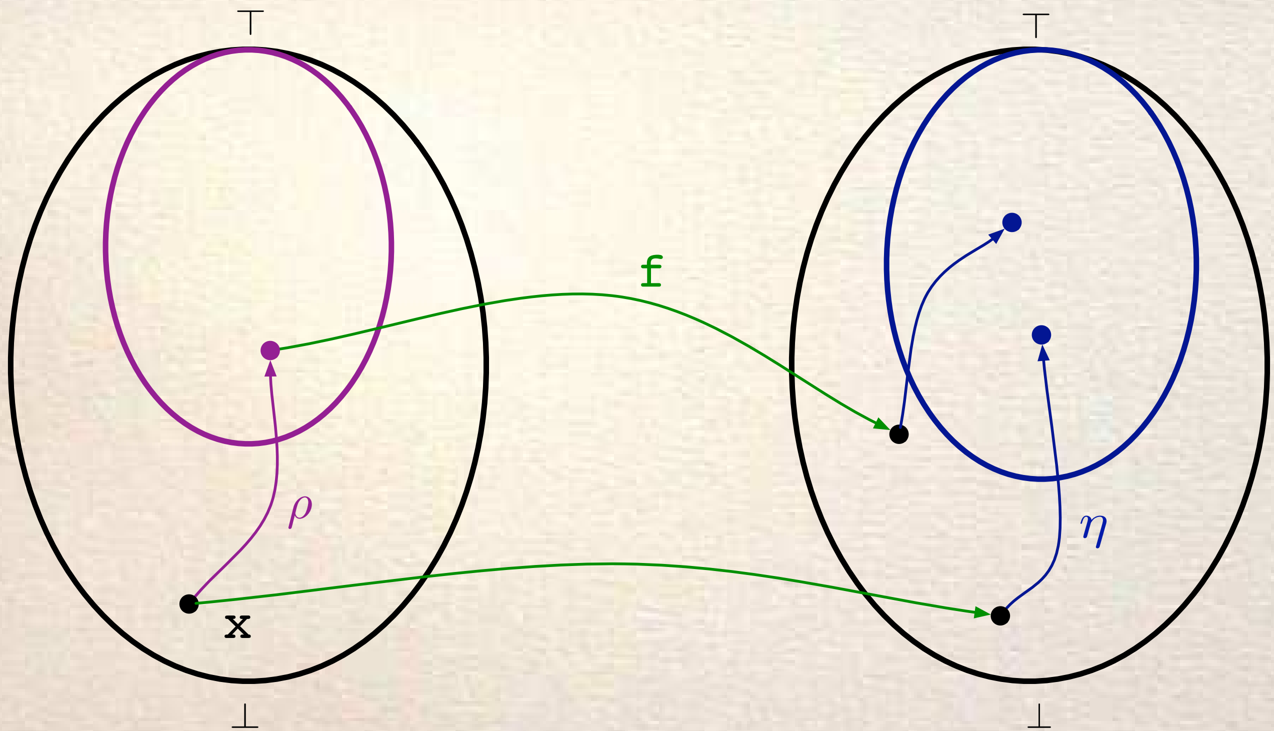


DOMAIN COMPLETENESS: SHELL/CORE



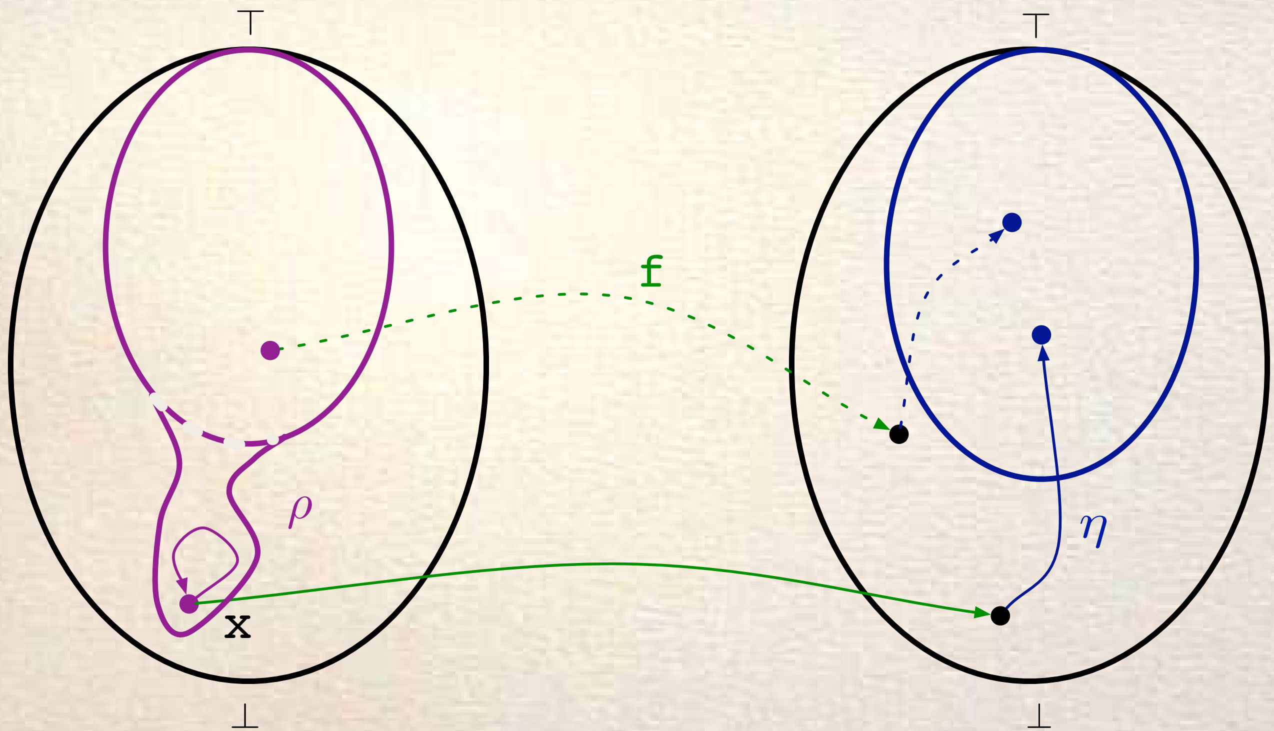
BACKWARD COMPLETENESS: $\eta \circ f \circ \rho = \eta \circ f$

DOMAIN COMPLETENESS: SHELL/CORE



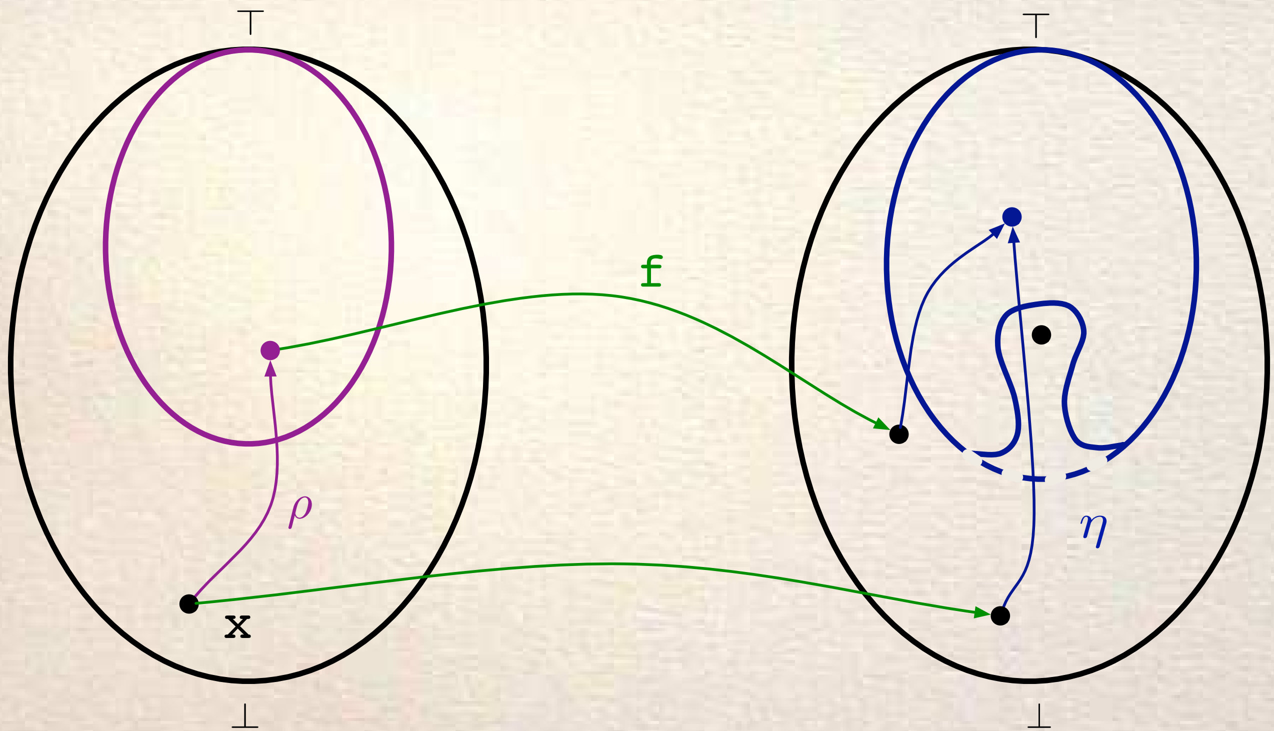
BACKWARD IN-COMPLETENESS: $\eta \circ f \circ \rho \geq \eta \circ f$

DOMAIN COMPLETENESS: SHELL/CORE



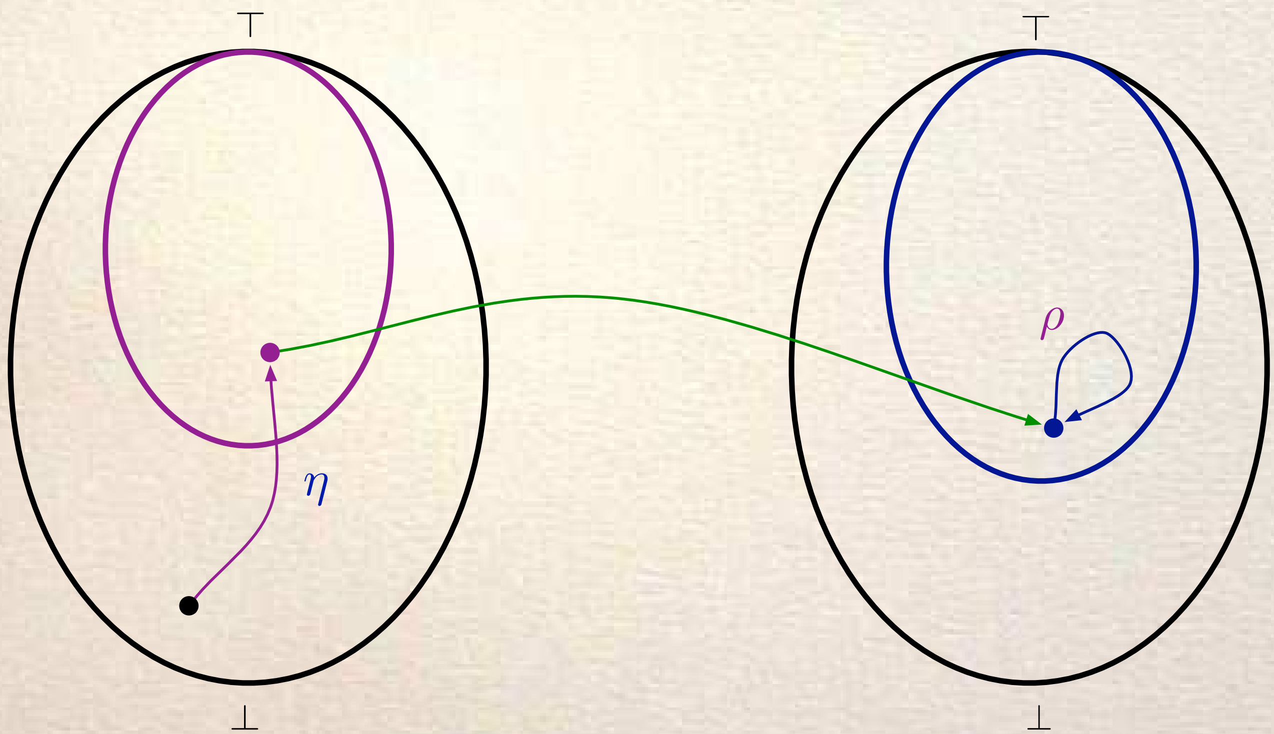
Making BACKWARD COMPLETE: Refining input domains [GRS'00]

DOMAIN COMPLETENESS: SHELL/CORE



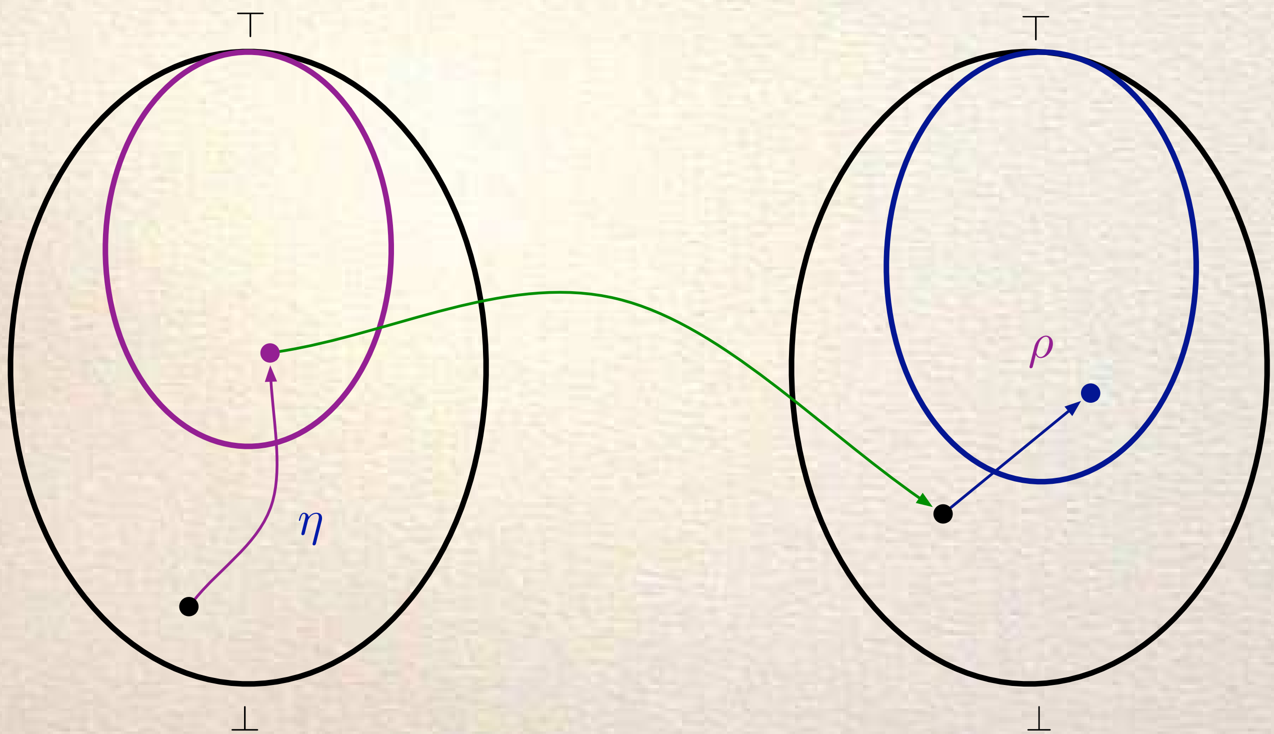
Making BACKWARD COMPLETE: Simplifying output domains [GRS'00]

DOMAIN COMPLETENESS: SHELL/CORE



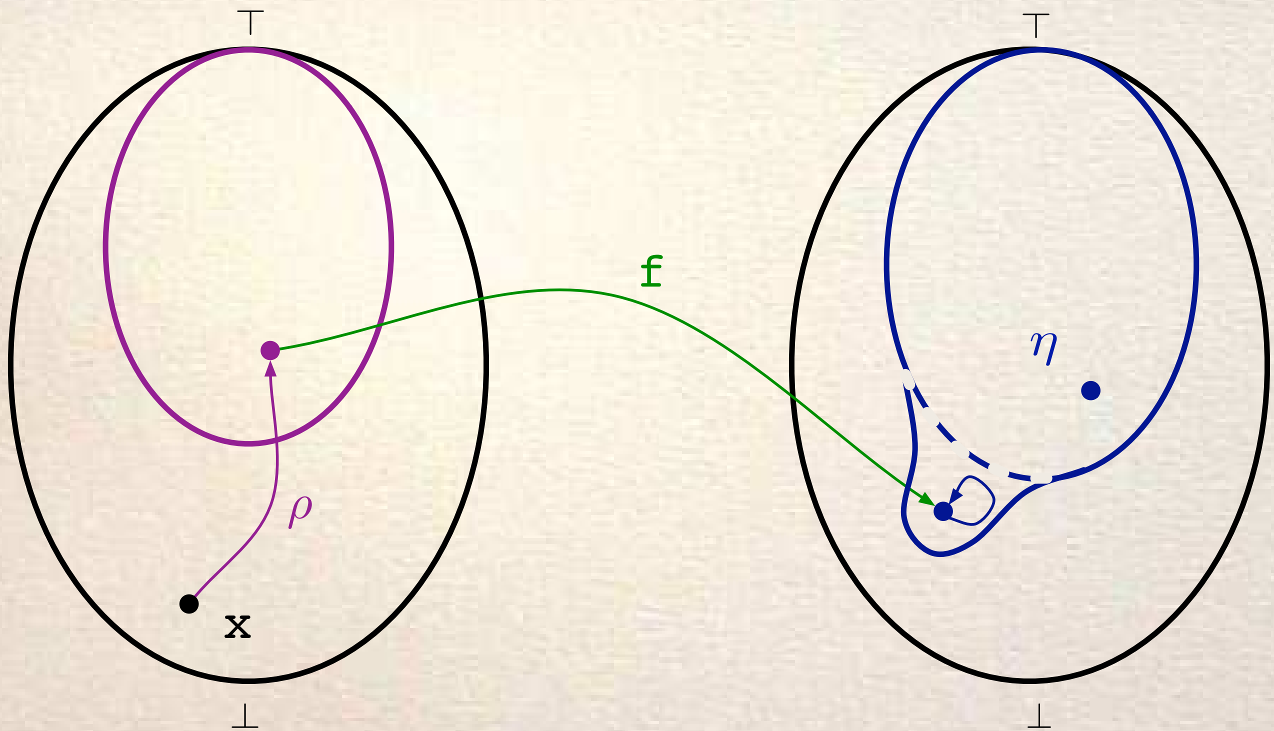
FORWARD COMPLETENESS: $\eta \circ f \circ \rho = f \circ \rho$

DOMAIN COMPLETENESS: SHELL/CORE



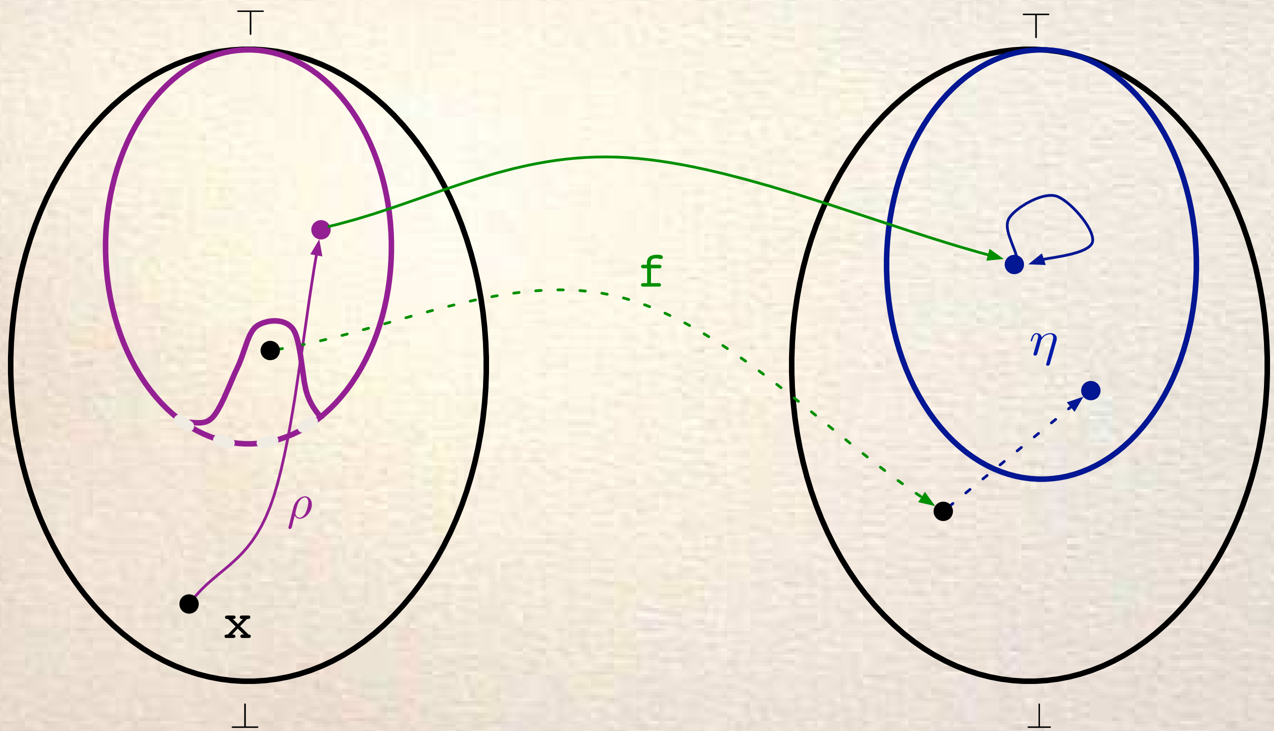
FORWARD IN-COMPLETENESS: $\eta \circ f \circ \rho \geq f \circ \rho$

DOMAIN COMPLETENESS: SHELL/CORE



Making FORWARD COMPLETE: Refining output domains [GQ'01]

DOMAIN COMPLETENESS: SHELL/CORE



Making FORWARD COMPLETE: Simplifying input domains [GQ'01]

Backward vs Forward

- A domain is *backward complete* wrt f iff it is *forward complete* wrt $f^{+} = \lambda X.\, \bigcup \left\{ Y \,\middle|\, f(Y) \subseteq X \right\}$;
- A (not trivial) partition is *backward stable* wrt f iff it is *forward stable* wrt $f^{-1} = \lambda X.\, \left\{ y \,\middle|\, f(y) \in X \right\}$;
- If f is injective, a (not trivial) partition is *forward stable* wrt f iff it is *backward stable* wrt f^{-1};

BACKWARD VS FORWARD

- A domain is *backward complete* wrt f iff it is *forward complete* wrt $f^+ = \lambda X.\ \bigcup \left\{ Y \,\middle|\, f(Y) \subseteq X \right\}$;

- A (not trivial) partition is *backward stable* wrt f iff it is *forward stable* wrt $f^{-1} = \lambda X.\ \left\{ y \,\middle|\, f(y) \in X \right\}$;

- If f is injective, a (not trivial) partition is *forward stable* wrt f iff it is *backward stable* wrt f^{-1};

A backward problem can always be transformed in a forward one, but the viceversa is not always possible!

GENERALISING DATA-REFINEMENT III

$$
\tau(P):\left[\begin{array}{l}
b[0]=0;\\
c[0]=1;\\
i=2;\\
\textbf{while}\quad i\leq N\,\{\\
\qquad\textbf{if}\quad i \mod 2 == 0\\
\qquad\qquad\{b[i\div 2]=c[(i-1)\div 2]+b[(i-2)\div 2]\}\\
\qquad\qquad\{c[i\div 2]=b[(i)\div 2]+c[(i-2)\div 2]\};\\
\qquad\qquad\ i++\\
\qquad\qquad\}
\end{array}\right.
$$

The complete shell $\mathcal{S}=\mathcal{R}^{\mathcal{B}}_{[\![\tau(P)]\!]}(\mathfrak{u}\longrightarrow\eta)$ includes odd and even Fibonacci's sequences:

- $[\![\tau(P)]\!]^{\mathcal{S}}= b\in[0,N\div 2]\longrightarrow\textbf{eFib}\wedge c\in[0,N\div 2]\longrightarrow\textbf{oFib}\wedge i\in[2,N+1]$
- $\textbf{Inv}=2\leq i\leq N\wedge\forall j\in[2,i].\ a[j]=a[j-1]+a[j-2]$

CAN WE MAKE SW OBSCURE BY TRANSFORMING SEMANTICS?

PROGRAM TRANSFORMATION

[Cousot & Cousot POPL'02]

$$
\begin{tikzcd}[column sep=huge, row sep=huge]
\text{Subject program } P \arrow[rr, "\text{Syntactic}", "\text{transformation } \tau"'] \arrow[d, shift left, "S"] & & \text{Transformed program } \tau[\![P]\!] \arrow[d, shift left, "S"] \\
\text{Subject program semantics } S[\![P]\!] \arrow[u, shift left, "p"] \arrow[rr, "\text{Semantic}", "\text{transformation } t"'] & & \text{Transformed program semantics } t[S[\![P]\!]] \sqsubseteq S[\![\tau[\![P]\!]]\!] \arrow[u, shift left, "p"]
\end{tikzcd}
$$

Syntactic transformation: $\tau = p \circ t \circ S$

THE GEOMETRY OF SEMANTICS TRANSFORMERS

MAKING SEMANTICS COMPLETE (FROM ABOVE AND BELOW):

$$\mathbb{F}^{\uparrow}_{\eta,\rho}(f) = \bigsqcap\{h : C \longrightarrow C \mid f \sqsubseteq h,\ \rho \circ h \circ \eta = h \circ \eta\}$$
$$\mathbb{F}^{\downarrow}_{\eta,\rho}(f) = \bigsqcup\{h : C \longrightarrow C \mid f \sqsupseteq h,\ \rho \circ h \circ \eta = h \circ \eta\}$$

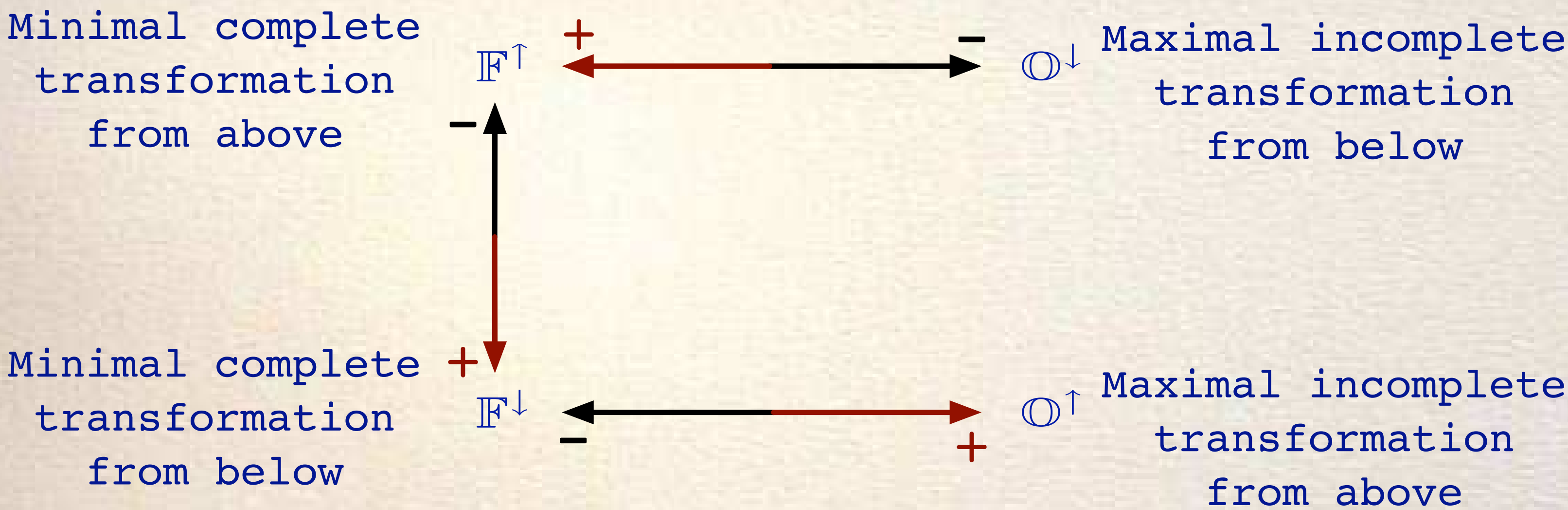
$\mathbb{F}^{\uparrow}_{\eta,\rho}(f)$ and $\mathbb{F}^{\downarrow}_{\eta,\rho}(f)$ are (Forward) complete

MAKING SEMANTICS MAXIMALLY IN-COMPLETE (FROM ABOVE AND BELOW):

$$\mathbb{O}^{\uparrow}_{\eta,\rho}(f) = \bigsqcup\{g : C \longrightarrow C \mid \mathbb{F}^{\downarrow}_{\eta,\rho}(g) = \mathbb{F}^{\downarrow}_{\eta,\rho}(f)\}$$
$$\mathbb{O}^{\downarrow}_{\eta,\rho}(f) = \bigsqcap\{g : C \longrightarrow C \mid \mathbb{F}^{\uparrow}_{\eta,\rho}(g) = \mathbb{F}^{\uparrow}_{\eta,\rho}(f)\}$$

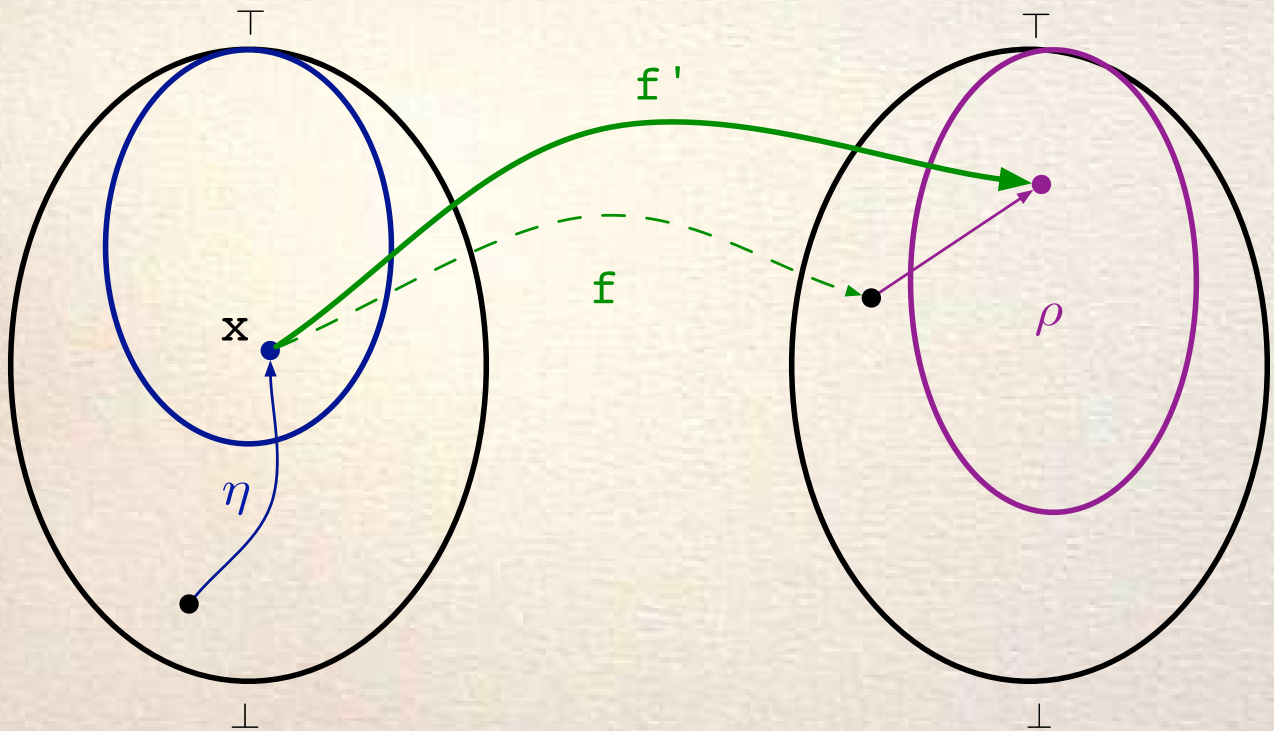
$\mathbb{O}^{\uparrow}_{\eta,\rho}(f)$ and $\mathbb{O}^{\downarrow}_{\eta,\rho}(f)$ are generally in-complete

THE GEOMETRY OF SEMANTICS TRANSFORMERS



$$(\mathbb{F}^{\uparrow})^{+} = \mathbb{F}^{\downarrow} \qquad \text{and} \qquad (\mathbb{F}^{\uparrow})^{-} = \mathbb{O}^{\downarrow}$$

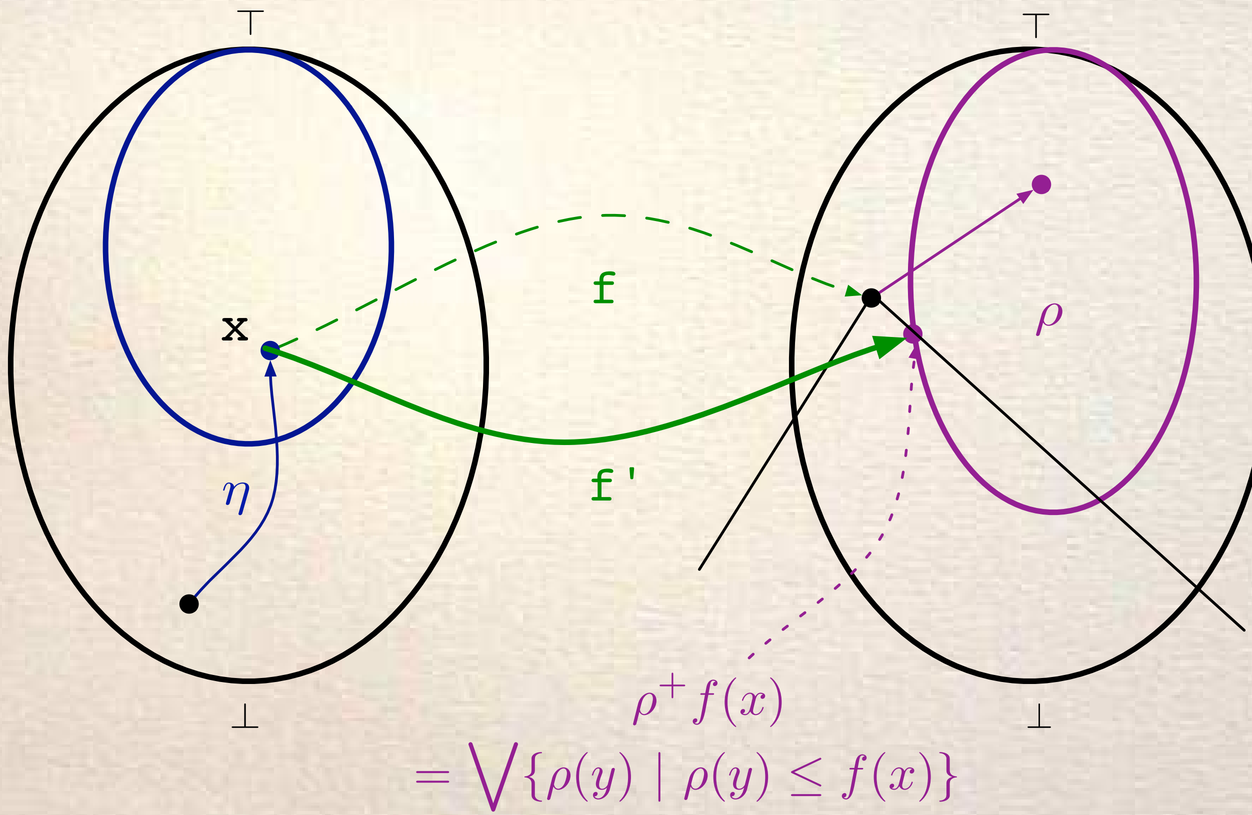
The Geometry of Semantics Transformers



Making FORWARD COMPLETENESS: Transforming the semantics upwards

$$\mathbb{F}^{\uparrow}_{\eta,\rho} = \lambda f.\lambda x. \begin{cases} \rho \circ f(x) & \text{if } x \in \eta(C) \\ f(x) & \text{otherwise} \end{cases}$$

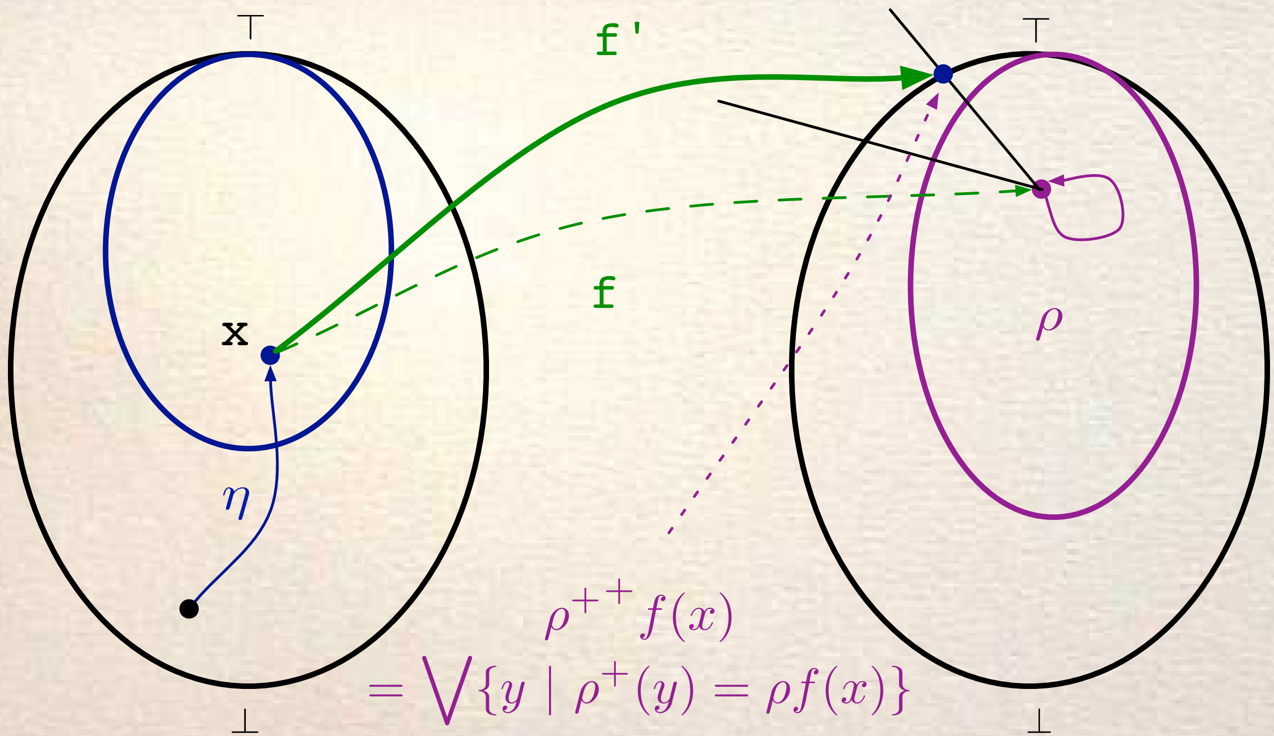
THE GEOMETRY OF SEMANTICS TRANSFORMERS



Making FORWARD COMPLETENESS: Transforming the semantics downwards

$$\mathbb{F}^{\downarrow}_{\eta,\rho} = \lambda f.\lambda x. \begin{cases} \rho^{+} \circ f(x) & \text{if } x \in \eta(C) \\ f(x) & \text{otherwise} \end{cases}$$

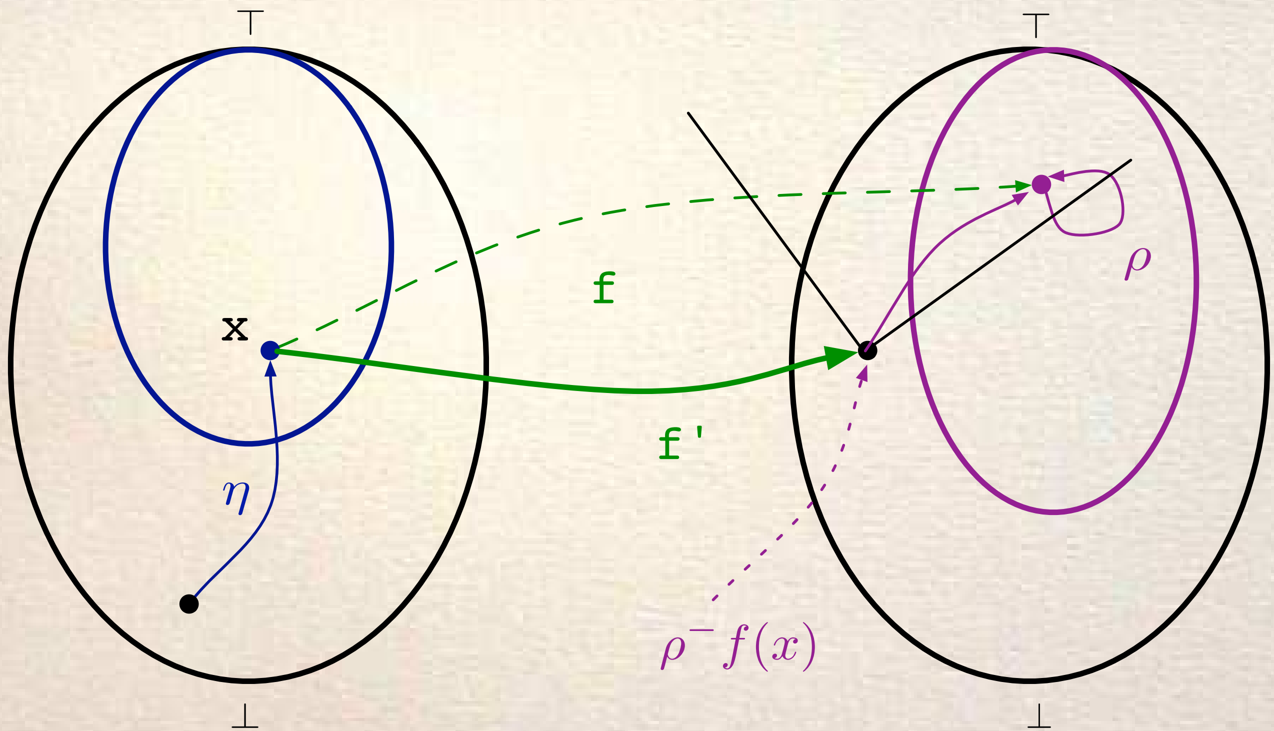
THE GEOMETRY OF SEMANTICS TRANSFORMERS



Making FORWARD IN-COMPLETENESS: Transforming the semantics upwards

$$\mathbb{O}^{\uparrow}_{\eta,\rho}(f)(x) = \begin{cases} (\rho^{+})^{+}(f(x)) = \bigvee\left\{ y \,\middle|\, \rho^{+}(y) = \rho^{+}(f(x)) \right\} & \text{if } x \in \eta \\ f(x) & \text{otherwise} \end{cases}$$

THE GEOMETRY OF SEMANTICS TRANSFORMERS



Making FORWARD IN-COMPLETENESS: Transforming the semantics downwards

$$\mathbb{O}^{\downarrow}_{\eta,\rho}(f)(x) = \begin{cases} \rho^-(f(x)) = \bigwedge \left\{ y \;\middle|\; \rho(y) = \rho(f(x)) \right\} & \text{if } x \in \eta \\ f(x) & \text{otherwise} \end{cases}$$

Obfuscation as incompleteness

We transform semantics in order to induce maximal incompleteness

$$P : \left[\begin{array}{ll} & x = x * x; \\ c: & \textbf{if } 10 \leq x \leq 100\ \{y = 5\}\ \{y = 5000\}; \\ & \textbf{return}(y) \end{array}\right.$$

- $[\![P]\!]^{\iota}(x \in [5, 8]) = x \in [25, 64] \wedge y \in [5]$

- $\mathbf{wlp}[\![c]\!]^{\iota}(y \leq 100) = x \in [10, 100]$ and
  $\mathbf{wlp}[\![x = x * x]\!]^{\iota}(x \in [10, 100]) = x \in [4, 10]$.

- Find c' such that

$$\begin{aligned} &\mathbf{wlp}[\![c']\!]^{\iota}(x \in [10, 100]) = \\ &\quad \mathbb{O}^{\downarrow}_{\iota,\iota}(\lambda X.\, \mathbf{wlp}[\![x = x * x]\!]^{\iota}(X))(x \in [10, 100]) = \\ &\quad \iota^{-}(\mathbf{wlp}[\![x = x * x]\!]^{\iota}(x \in [10, 100])) = \{4, 10\} \end{aligned}$$

OBFUSCATION AS INCOMPLETENESS

We transform semantics in order to induce maximal incompleteness

$$P: \left[\begin{array}{ll} & x = x * x; \\ c: & \textbf{if } 10 \leq x \leq 100 \, \{y = 5\} \, \{y = 5000\}; \\ & \textbf{return}(y) \end{array} \right.$$

- $c' : \textbf{if } x == 4 \vee x == 10 \, \{x = 16\} \, \{x = x * 200\}$

- In order to ensure behaviour equivalence we derive

$$\begin{array}{ll} \textbf{if} & 4 \leq x \leq 10 \\ & \{x = x - (x - 4) \,\Box\, x = x - (x - 10)\} \\ & \{\textbf{nil}\} \end{array}$$

Obfuscation as incompleteness

We transform semantics in order to induce maximal incompleteness

$$P : \left[\begin{array}{ll} & x = x * x; \\ c: & \textbf{if } 10 \leq x \leq 100 \, \{y = 5\} \, \{y = 5000\}; \\ & \textbf{return}(y) \end{array} \right.$$

- The resulting obfuscated code is:

$$\tau(P) : \left[\begin{array}{l} \textbf{if} \quad 4 \leq x \leq 10 \\ \qquad \{x = x - (x - 4) \, \Box \, x = x - (x - 10)\} \\ \qquad \{\textbf{nil}\}; \\ \textbf{if } x == 4 \vee x == 10 \, \{x = 16\} \, \{x = x * 200\}; \\ \textbf{if } 10 \leq x \leq 100 \, \{y = 5\} \, \{y = 5000\}; \\ \textbf{return}(y) \end{array} \right.$$

For $x = 7$ we have

$$[\![\tau(P)]\!]^{\iota}(x \in [5, 8]) = x \in [16, 1400] \wedge y \in [5, 5000]$$

Obfuscation as incompleteness

We transform semantics in order to induce maximal incompleteness

$$P: \left[\begin{array}{ll} & x = x * x; \\ c: & \textbf{if}\ 10 \le x \le 100\ \{y = 5\}\ \{y = 5000\}; \\ & \textbf{return}(y) \end{array}\right.$$

- The resulting obfuscated code is:

$$\tau(P): \left[\begin{array}{l} \textbf{if}\quad 4 \le [5,8] \le 10 \\ \qquad \{x = [5,8] - ([5,8] - 4)\ \square\ x = x - (x - 10)\} \\ \qquad \{\textbf{nil}\}; \\ \{x \in [1,7]\} \\ \textbf{if}\ x == 4 \vee x == 10\ \{x = 16\}\ \{x = x * 200\}; \\ \textbf{if}\ 10 \le x \le 100\ \{y = 5\}\ \{y = 5000\}; \\ \textbf{return}(y) \end{array}\right.$$

For $x = 7$ we have

$$[\![\tau(P)]\!]^{\iota}(x \in [5,8]) = x \in [16, 1400] \wedge y \in [5, 5000]$$

Obfuscation as Incompleteness

We can derive a method for systematically making code obscure:

- $P = M1; \ldots; M_j; \Phi_j\, M_{j+1}; \ldots; M_n$
- Assume the invariant Φ_j can be generated with abstract interpretation α
- Find C such that:

$$\mathbf{wlp}[\![C]\!]^{\alpha}(\Phi_j) = \mathbb{O}^{\downarrow,\uparrow}_{\alpha,\alpha}(\lambda X.\, \mathbf{wlp}[\![M_j]\!]^{\iota}(X))(\Phi_j)$$

- Adjust C in order to keep concrete observational (I/O) behaviour ($C \models \Phi_j$)
- $\tau(P) = M1; \ldots; C; \Phi_j\, M_{j+1}; \ldots; M_n$

HIDING IN OBSCURITY

We generalize Cousot' Abstract Watermarking [Cousot & Cousot '04]

- **Stegomarker**: $\mathfrak{M} : \mathcal{S} \longrightarrow \mathbb{P}$ encodes the signature $s \in \mathcal{S}$ into a program $\mathfrak{M}(s) \in \mathbb{P}$ (the stegomark)
- **Stegolayer**: $\mathfrak{L} : \mathbb{P} \times \mathbb{P} \longrightarrow \mathbb{P}$ is used to compose the stegomark with the source (cover) program.
- **Stegoprogram**: $\mathfrak{S} : \mathbb{P} \times \mathcal{S} \longrightarrow \mathbb{P}$ such that $\mathfrak{S}(P, s) = \mathfrak{L}(P, \mathfrak{M}(s))$

STATIC WATERMARKING

Watermarks are encoded as syntactic (static) properies of $\mathfrak{S}(P, s)$

HIDING IN OBSCURITY

We generalize Cousot' Abstract Watermarking [Cousot & Cousot '04]

- Stegomarker: $\mathfrak{M} : \mathcal{S} \longrightarrow \mathbb{P}$ encodes the signature $s \in \mathcal{S}$ into a program $\mathfrak{M}(s) \in \mathbb{P}$ (the stegomark)
- Stegolayer: $\mathfrak{L} : \mathbb{P} \times \mathbb{P} \longrightarrow \mathbb{P}$ is used to compose the stegomark with the source (cover) program.
- Stegoprogram: $\mathfrak{S} : \mathbb{P} \times \mathcal{S} \longrightarrow \mathbb{P}$ such that $\mathfrak{S}(P, s) = \mathfrak{L}(P, \mathfrak{M}(s))$

DYNAMIC WATERMARKING

Watermarks are encoded as semantic (dynamic) properies of $\mathfrak{S}(P, s)$

Hiding in Obscurity

We generalize Cousot' Abstract Watermarking [Cousot & Cousot '04]

- Stegomarker: $\mathfrak{M} : \mathcal{S} \longrightarrow \mathbb{P}$ encodes the signature $s \in \mathcal{S}$ into a program $\mathfrak{M}(s) \in \mathbb{P}$ (the stegomark)
- Stegolayer: $\mathfrak{L} : \mathbb{P} \times \mathbb{P} \longrightarrow \mathbb{P}$ is used to compose the stegomark with the source (cover) program.
- Stegoprogram: $\mathfrak{S} : \mathbb{P} \times \mathcal{S} \longrightarrow \mathbb{P}$ such that $\mathfrak{S}(P, s) = \mathfrak{L}(P, \mathfrak{M}(s))$

Abstract Watermarking

Watermarks are encoded as abstract properies of $\mathfrak{S}(P, s)$

HIDING IN OBSCURITY

Static and dynamic are instances of Abstract Watermarking!

- $P \in \mathbb{P}$ (source), $\alpha, \varpi, \eta \in uco(\Sigma)$ be program properties such that $\alpha \sqsubseteq \varpi$
- If $\{\!|\mathfrak{M}(s)|\!\}^{\alpha} \in \varpi$ then $\mathfrak{L}$ is a stegolayer for P and $\mathfrak{M}(s)$ if

$$\{\!|\mathfrak{L}(P, \mathfrak{M}(s))|\!\}^{\alpha} = \lambda x. \begin{cases} \{\!|\mathfrak{M}(s)|\!\}^{\alpha}(x) & \text{if } x \in \eta \\ \{\!|P|\!\}^{\alpha}(x) & \text{otherwise} \end{cases}$$

STATIC WATERMARKING

α decidable (static) and $\eta = id$

$\Downarrow$

$\mathfrak{S}(P, s)$ always reveals the watermark

Hiding in Obscurity

Static and dynamic are instances of Abstract Watermarking!

- $P \in \mathbb{P}$ (source), $\alpha, \omega, \eta \in uco(\Sigma)$ be program properties such that $\alpha \sqsubseteq \omega$
- If $\{\!|\mathfrak{M}(s)|\!\}^{\alpha} \in \omega$ then $\mathfrak{L}$ is a stegolayer for P and $\mathfrak{M}(s)$ if

$$\{\!|\mathfrak{L}(P, \mathfrak{M}(s))|\!\}^{\alpha} = \lambda x. \begin{cases} \{\!|\mathfrak{M}(s)|\!\}^{\alpha}(x) & \text{if } x \in \eta \\ \{\!|P|\!\}^{\alpha}(x) & \text{otherwise} \end{cases}$$

Dynamic Watermarking

α generic interpreter (dynamic) and $\eta \neq id$

$\Downarrow$

$\mathfrak{S}(P, s)$ reveals the watermark only on input η

Hiding and completeness

A stegoprogram reveals the watermark ϖ under input η if its abstract semantics is $\mathcal{F}$-complete for ϖ and η

$\mathfrak{S}(s, P)$ is a stegoprogram if:

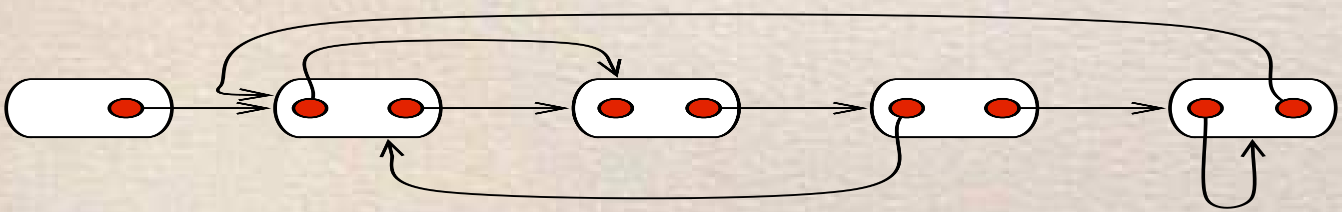
$$\{\!|\mathfrak{S}(s, P)|\!\}^{\alpha} = \mathbb{F}^{\uparrow\downarrow}_{\eta,\mathfrak{M}\{\!|s|\!\}}(\{\!|P|\!\}^{\alpha})$$

- $\{\!|\cdot|\!\}^{\alpha}$ performs watermark extraction (an abstract interpretation)
- Credibility: $\{\!|P|\!\}^{\alpha} \notin \varpi$ (i.e., $\varpi(\{\!|P|\!\}^{\alpha}) \approx \top$)
- Resilience: α is preserved by most program transformations
- Stealthy: α hard to guess + good stegolayer

Block Reordering

Static watermarking ($\eta = id$) with traces in Σ^+ as semantic objects

- $\mathcal{E} : \mathbb{N} \longrightarrow \mathbb{G}$ encoding of numbers in graphs
- $\mathfrak{M}\{\!|s|\!\}$ is the atomic closure $\{\mathcal{G}_s, \Sigma^+\} \in uco(\Sigma^+)$ where
  $\mathcal{G}_s = \left\{ \sigma \in \Sigma^+ \;\middle|\; \mathcal{E}(s) = \mathsf{CFG}(\sigma) \right\}$
- $\{\!|P|\!\}^\alpha$ extracts the CFG of P, which is an (incomplete) abstract interpretation of the trace semantics $\{\!|P|\!\}$



$$2 \cdot 5^3 + 0 \cdot 5^2 + 1 \cdot 5^1 + 4 \cdot 5^0 = 259$$

GRAPH-BASED WATERMARKING

Dynamic watermarking ($\eta \neq id$) states $\langle c, \mathfrak{R}, \mathcal{H}, i \rangle$, where $\mathcal{H} \in \mathbb{H}$ is a heap, c is the current instruction, i is an input sequence, and $\mathfrak{R} : Var(P) \longrightarrow \mathscr{R}$ is register allocation.

- $\mathcal{E} : \mathbb{N} \longrightarrow \mathbb{G}$ encoding of numbers in graphs
- $\mathfrak{M}\{\!|s|\!\}$ is the atomic closure $\{\mathcal{E}(s), \Sigma^+\} \in uco(\Sigma^+)$
- $\mathfrak{H} : \mathbb{H} \longrightarrow \mathbb{G}$ extracts the set of all graphs allocated in memory with **root** allocated as last,
- $\alpha = \delta^+ \circ \delta$ where $\delta : \wp(\Sigma^+) \longrightarrow \mathbb{G}$ is such that:

$$\delta(X) = \left\{ \mathcal{G} \;\middle|\; \begin{array}{l} \sigma \in X,\ |\sigma| = n+1,\ \sigma_n = \langle c, \mathfrak{R}, \mathcal{H}_n, \varepsilon \rangle \\ \mathcal{G} \in \mathfrak{H}(\mathcal{H}_n),\ \mathbf{root}(\mathcal{G}) \in \mathcal{H}_n \\ \forall j \in [0, n-1].\ \mathbf{root}(\mathcal{G}) \notin \mathcal{H}_j \end{array} \right\}$$

DISCUSSION: THE FUCSIA IDEA

Obfuscation and Steganography by Abstract Interpretation

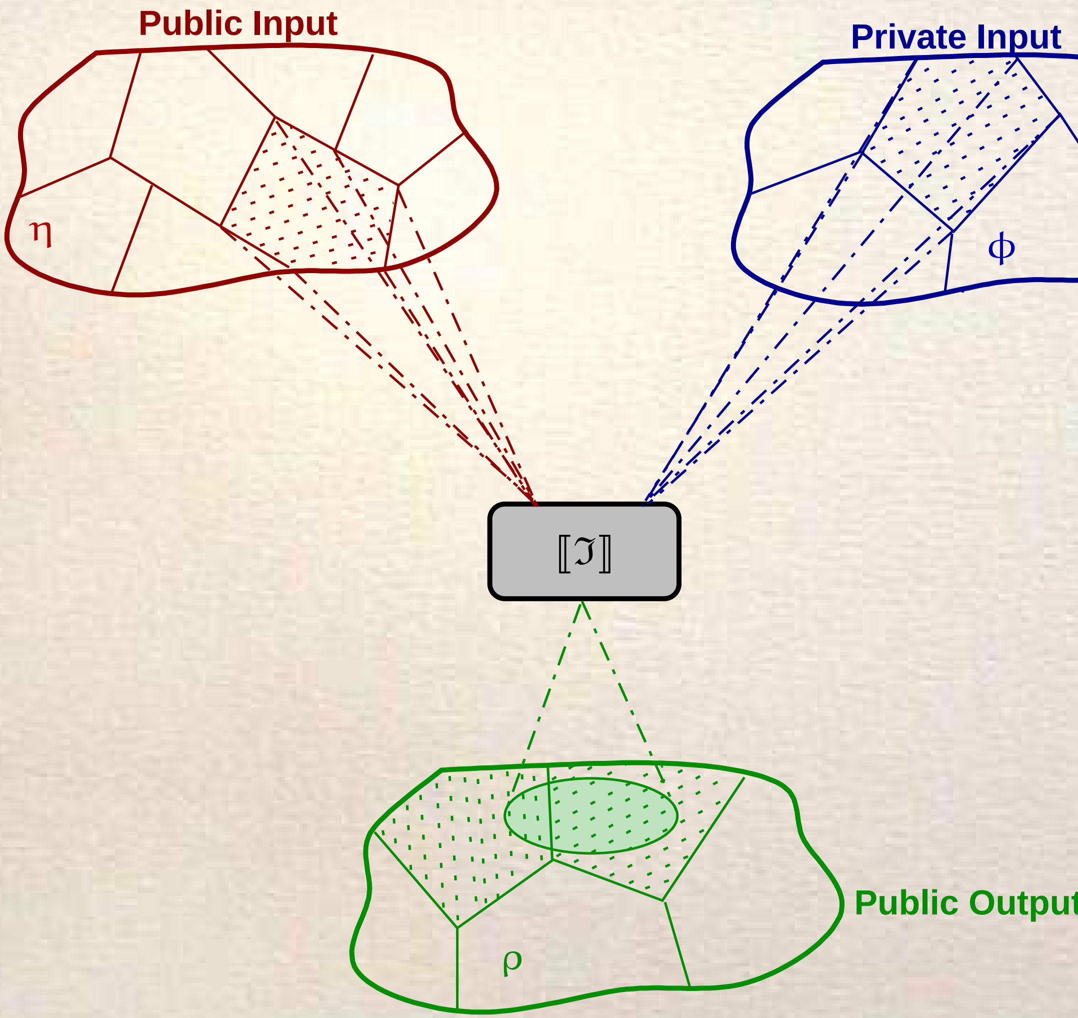
- Define a uniform framework for information concealment in programming languages
  - General enough to include most known methods
  - Formal enough to provide a (possibly) provable secure environment for obfuscation and steganography
  - Rich enough to provide advanced design and evaluation tools
  - Practical enough to become a standard in the obfuscation and steganographic design and evaluation

- The goal: develop a theory and practice for code obfuscation and steganography in order to make these technologies as practical as analogous ones in other media (e.g., in DRM of audio and video)
  - The code is a new media
  - Known concepts in digital media (compression, noise etc.) have to be studied on software

FUTURE DIRECTIONS

- Move from syntactic to semantic-based metrics
  - measuring incompleteness
  - measuring complexity of complete refinements
- Obscuring and watermarking require program integration $\mathfrak{I} : \mathbb{P} \times \mathbb{P} \longrightarrow \mathbb{P}$
- Explore (HO)ANI for isolating completeness holes?
  - The obfuscated parts and the stegomarks have to preserve the semantics of the cover program when integrated
  - $\mathbb{P}$ is partitioned in
    - *cover programs* $\mathcal{P} \subseteq \mathbb{P}$
    - *secret programs* $\mathcal{Q} \subseteq \mathbb{P}$

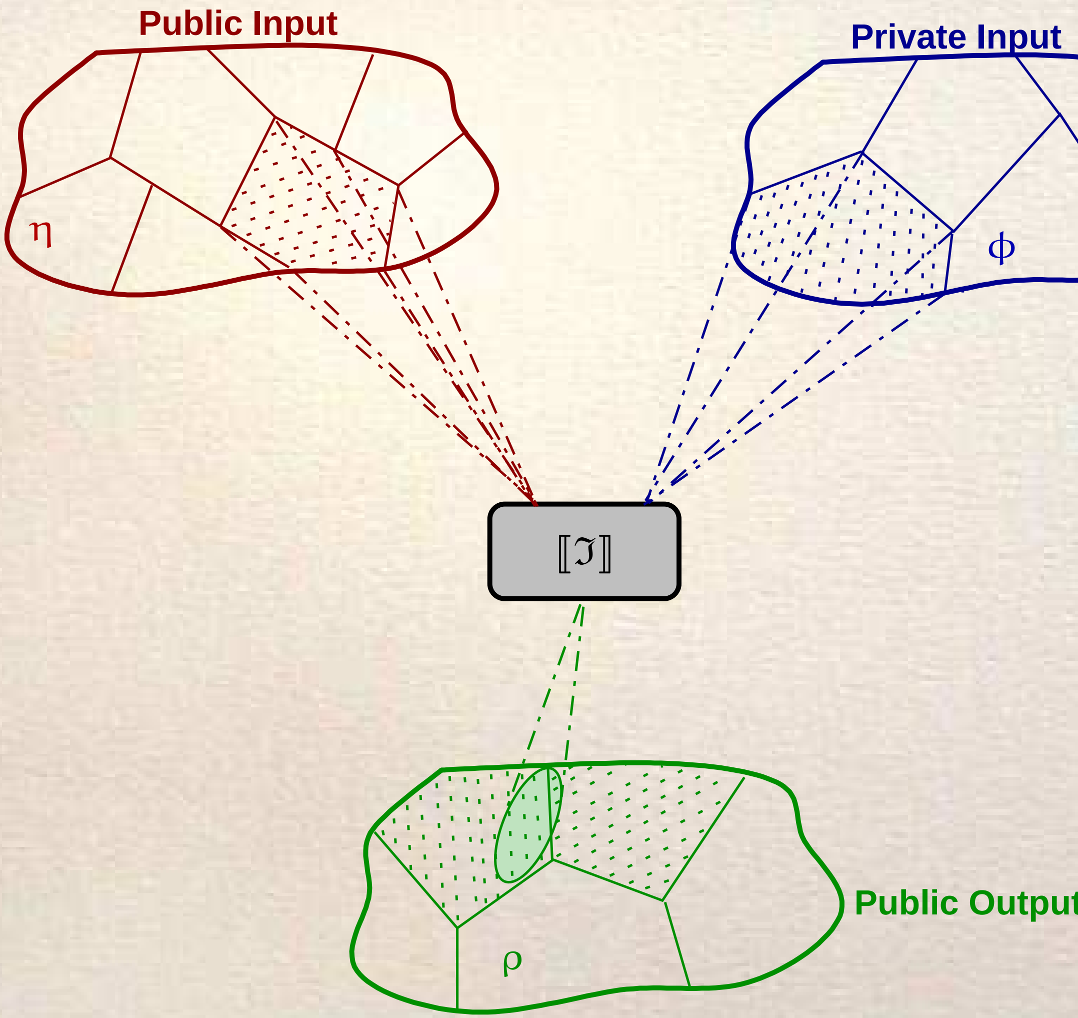
HOANI FOR SW WATERMARKING?



$$(\eta)\mathfrak{I}(\phi \rightsquigarrow\!\!| \rho) : [\![P_1]\!]^{\eta} = [\![P_2]\!]^{\eta} \implies$$

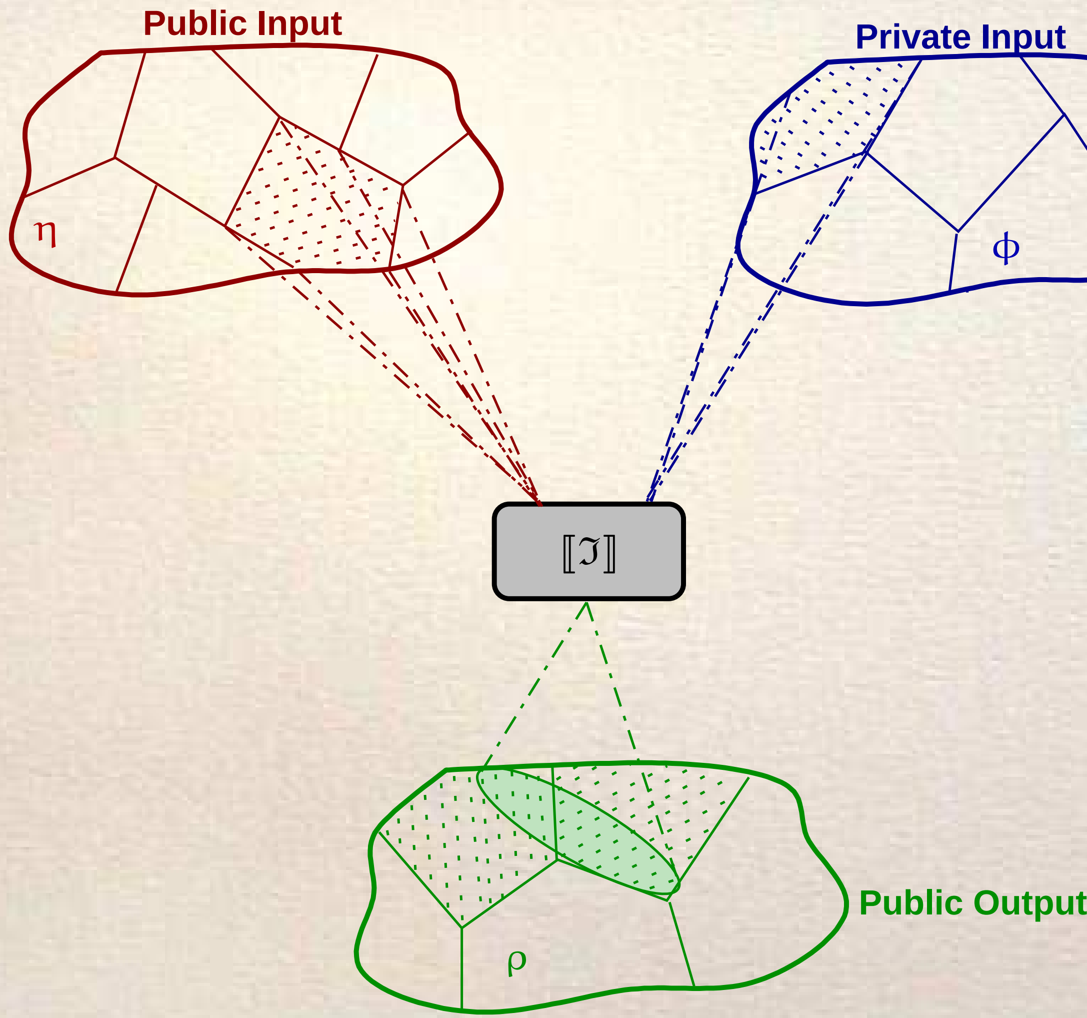
$$\rho^{\rho}([\![\mathfrak{I}]\!]^{\phi,\eta}([\![Q_1]\!]^{\phi}, [\![P_1]\!]^{\eta}) = \rho^{\rho}([\![\mathfrak{I}]\!]^{\phi,\eta}([\![Q_2]\!]^{\phi}, [\![P_2]\!]^{\eta})$$

HOANI FOR SW WATERMARKING?



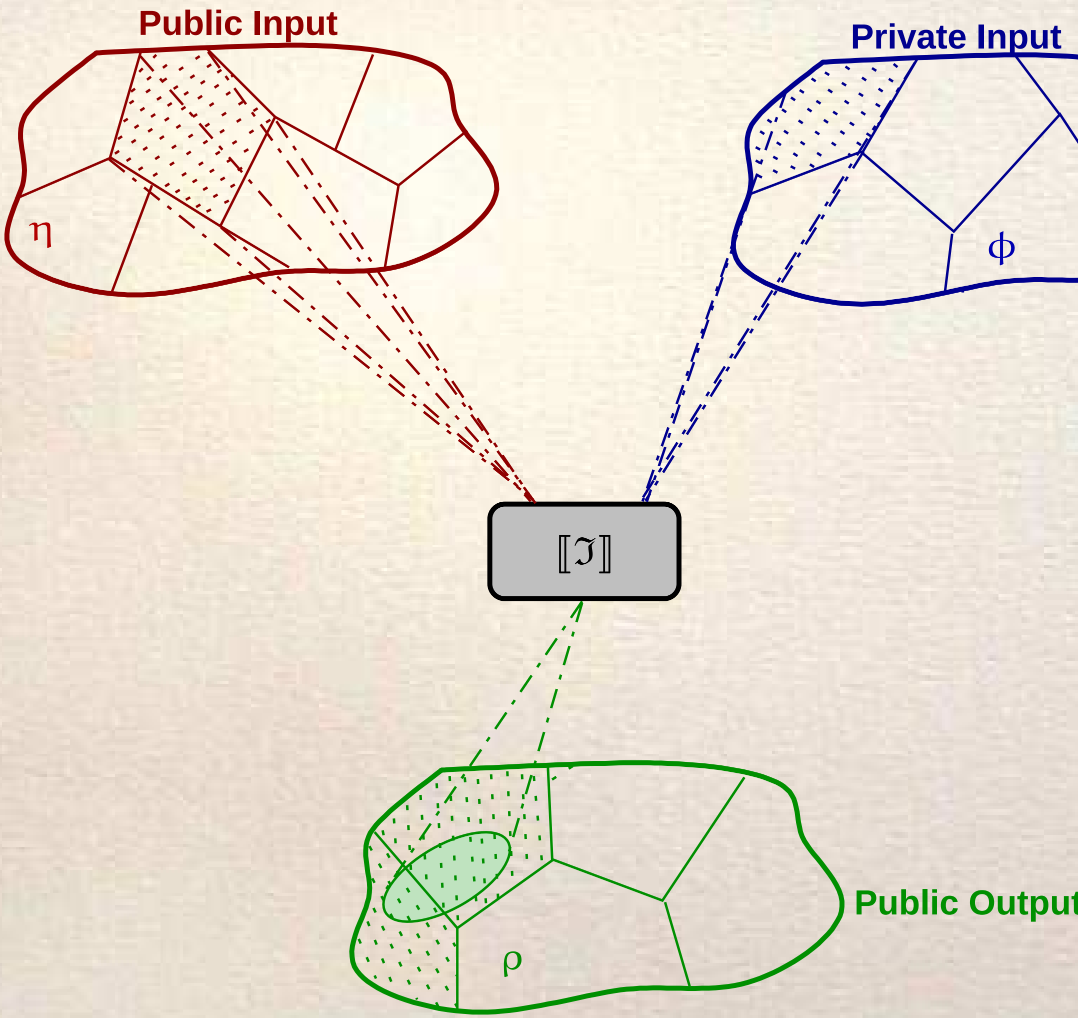
$$(\eta)\mathfrak{I}(\phi \rightsquigarrow\!\!\parallel \rho) : [\![P_1]\!]^{\eta} = [\![P_2]\!]^{\eta} \implies$$
$$\rho^{\rho}([\![\mathfrak{I}]\!]^{\phi,\eta}([\![Q_1]\!]^{\phi}, [\![P_1]\!]^{\eta}) = \rho^{\rho}([\![\mathfrak{I}]\!]^{\phi,\eta}([\![Q_2]\!]^{\phi}, [\![P_2]\!]^{\eta})$$

HOANI FOR SW WATERMARKING?



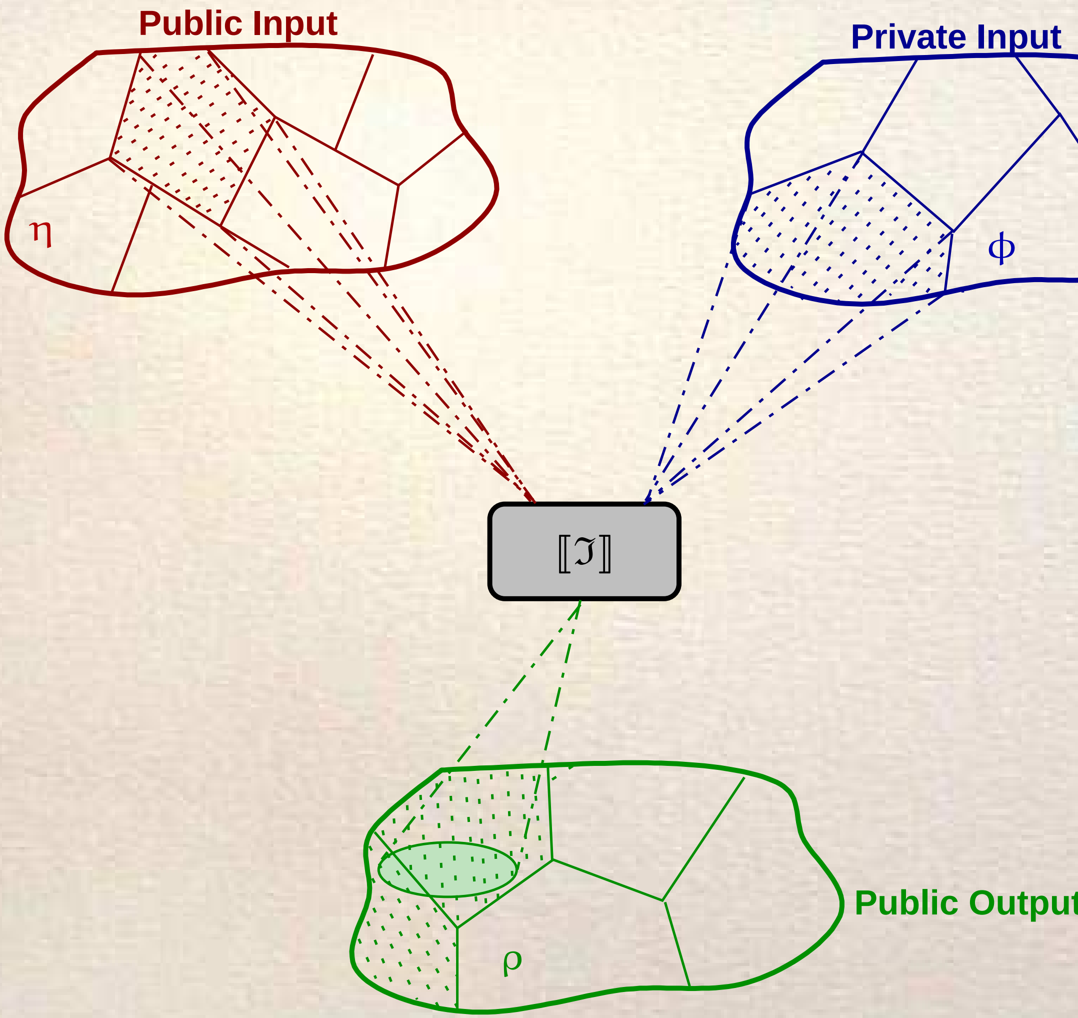
$$(\eta)\mathfrak{I}(\phi \rightsquigarrow\!\!\Vert \rho) : [\![P_1]\!]^{\eta} = [\![P_2]\!]^{\eta} \implies$$
$$\rho^{\rho}([\![\mathfrak{I}]\!]^{\phi,\eta}([\![Q_1]\!]^{\phi}, [\![P_1]\!]^{\eta}) = \rho^{\rho}([\![\mathfrak{I}]\!]^{\phi,\eta}([\![Q_2]\!]^{\phi}, [\![P_2]\!]^{\eta})$$

HOANI FOR SW WATERMARKING?



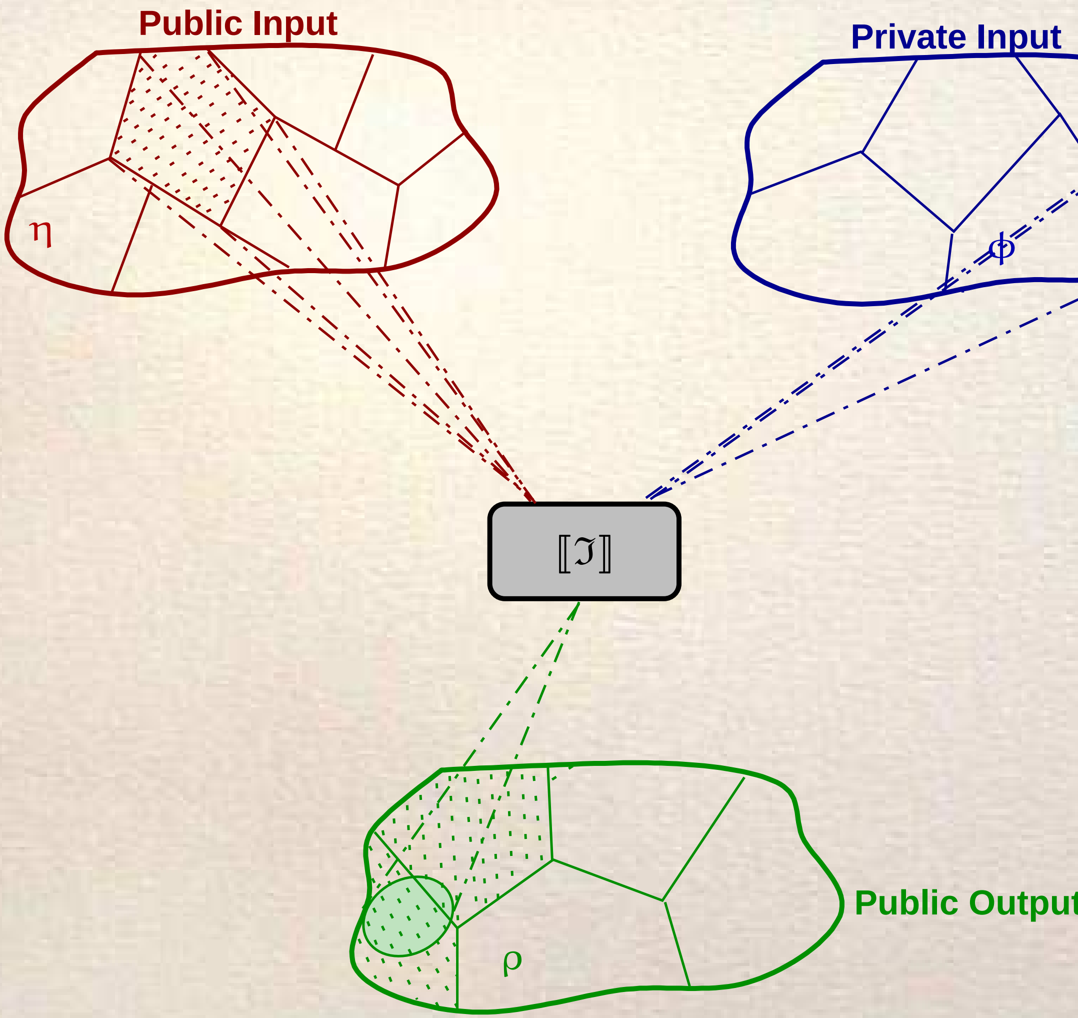
$$(\eta)\mathfrak{I}(\phi \rightsquigarrow\!\!\mid \rho) : [\![P_1]\!]^\eta = [\![P_2]\!]^\eta \implies$$
$$\rho^\rho([\![\mathfrak{I}]\!]^{\phi,\eta}([\![Q_1]\!]^\phi, [\![P_1]\!]^\eta) = \rho^\rho([\![\mathfrak{I}]\!]^{\phi,\eta}([\![Q_2]\!]^\phi, [\![P_2]\!]^\eta)$$

HOANI FOR SW WATERMARKING?



$$(\eta)\mathfrak{I}(\phi \rightsquigarrow\!\!\!\Vdash \rho) : [\![P_1]\!]^{\eta} = [\![P_2]\!]^{\eta} \implies$$
$$\rho^{\rho}([\![\mathfrak{I}]\!]^{\phi,\eta}([\![Q_1]\!]^{\phi}, [\![P_1]\!]^{\eta}) = \rho^{\rho}([\![\mathfrak{I}]\!]^{\phi,\eta}([\![Q_2]\!]^{\phi}, [\![P_2]\!]^{\eta})$$

HOANI FOR SW WATERMARKING?



$$(\eta)\mathfrak{I}(\phi \rightsquigarrow\!\!\Vdash \rho) : [\![P_1]\!]^{\eta} = [\![P_2]\!]^{\eta} \implies$$
$$\rho^{\rho}([\![\mathfrak{I}]\!]^{\phi,\eta}([\![Q_1]\!]^{\phi}, [\![P_1]\!]^{\eta}) = \rho^{\rho}([\![\mathfrak{I}]\!]^{\phi,\eta}([\![Q_2]\!]^{\phi}, [\![P_2]\!]^{\eta})$$

MANY THANKS!!