Deriving Bisimulations by Simplifying Partitions

Isabella Mastroeni

VMCAI 2008

INTRODUCTION

INGREDIENTS:

BISIMULATION: ... compares systems also on their capability of simulating each others.

INTRODUCTION

INGREDIENTS:

BISIMULATION: ... compares systems also on their capability of simulating each others.

STABILITY: ... requires a *bisimulation* between a system and one of its abstractions (partitions of states).

INTRODUCTION

INGREDIENTS:

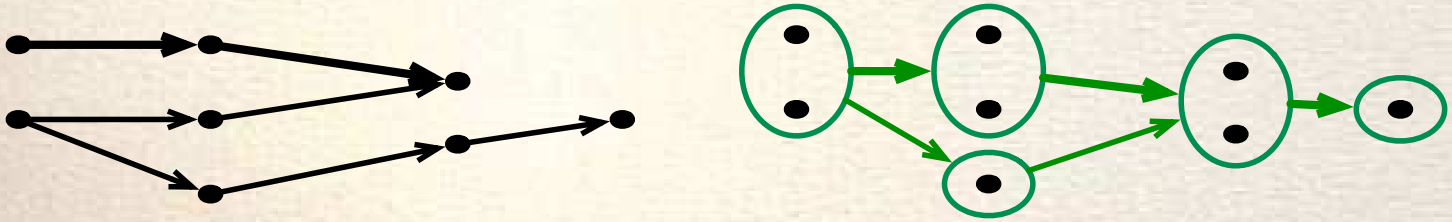
BISIMULATION: ... compares systems also on their capability of simulating each others.

STABILITY: ... requires a *bisimulation* between a system and one of its abstractions (partitions of states).

COMPLETENESS: ... models the precision of an abstract domain wrt an operator.

INTRODUCTION

INGREDIENTS:



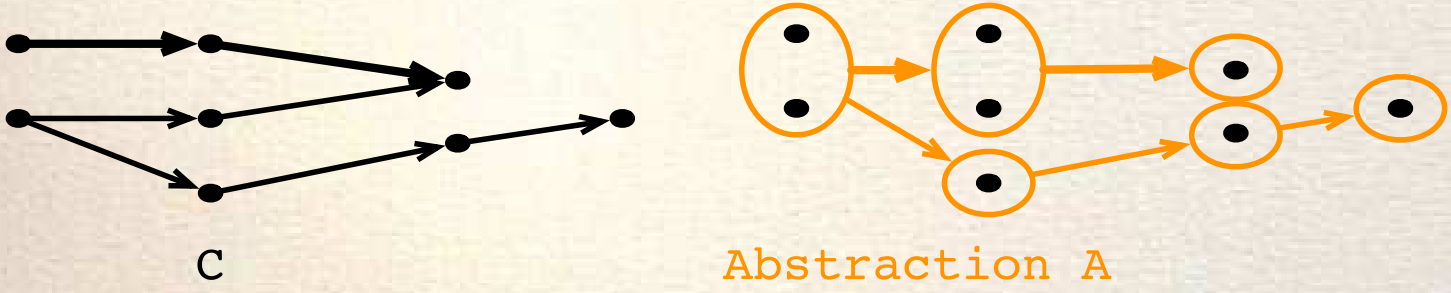
C

Abstraction A

A not bisimilar to C
=
A not stable [PT'87]
=
A not complete [GQ'01,RT'05]

INTRODUCTION

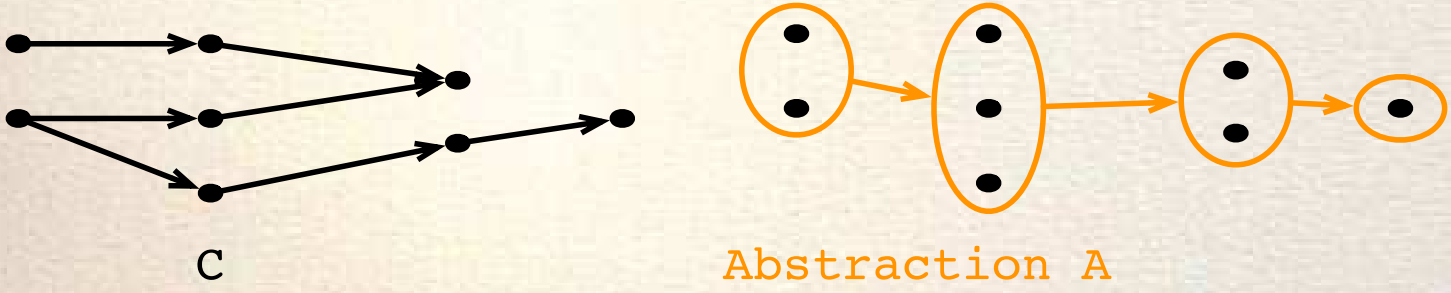
INGREDIENTS:



A bisimilar to C
=
A stable [PT'87]
=
A complete [GQ'01,RT'05]

INTRODUCTION

INGREDIENTS:



A bisimilar to C
=
A stable [PT'87]
=
A complete [GQ'01,RT'05]

ABSTRACT INTERPRETATION

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

Lattice of Abstract Domains $\equiv$ Lattice uco

$\mathsf{A} \equiv \rho(C)$

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

Abstract Interpretation

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

Lattice of Abstract Domains $\equiv$ Lattice uco

$\mathsf{A} \equiv \wp(C)$

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

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

ABSTRACT INTERPRETATION

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

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

A $\equiv \rho(C)$

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

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

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

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*

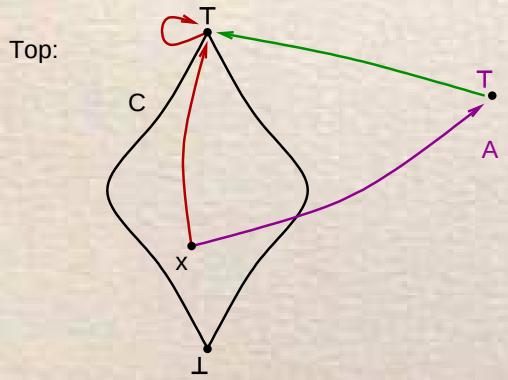
A $\equiv \rho(C)$

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

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

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

$$\sqcup_i A_i = \cap_i A_i$$



Abstract Interpretation

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

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

A $\equiv \rho(C)$

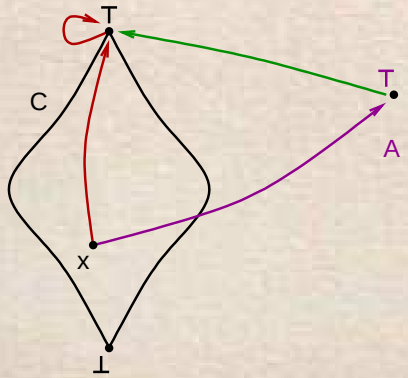
$< uco(C), \sqsubseteq, \sqcap, \sqcup, \lambda x.\ \top, \lambda x.\ x >$

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

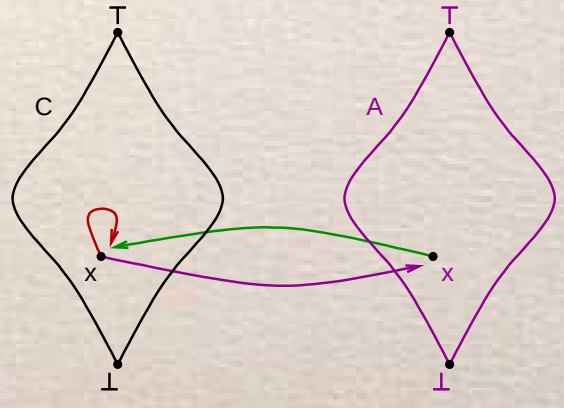
$$\sqcap_i A_i = \mathcal{M}(\cup_i A_i)$$

$$\sqcup_i A_i = \cap_i A_i$$

Top:



Bottom:



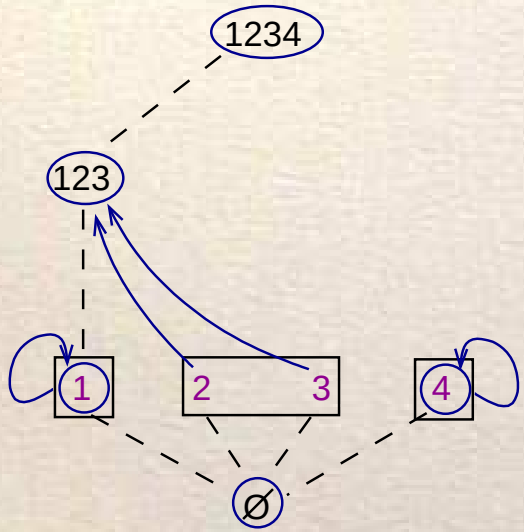
PARTITIONS VS ABSTRACT DOMAINS

Partitions uniquely correspond to particular abstract domains: PARTITIONING [RT'04,HM'05]

PARTITIONS VS ABSTRACT DOMAINS

Partitions uniquely correspond to particular abstract domains: PARTITIONING [RT'04,HM'05]

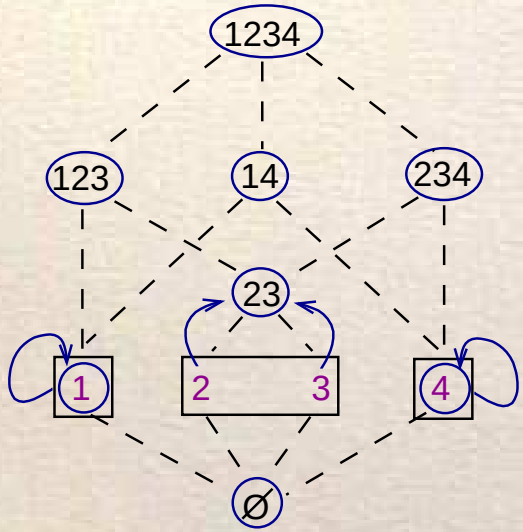
$\eta \in \mathit{uco}(\wp(C)) \;\Rightarrow\; \forall x, y.\, x\ \mathit{Rel}^{\eta}\ y$ iff $\eta(x) = \eta(y)$



PARTITIONS VS ABSTRACT DOMAINS

Partitions uniquely correspond to particular abstract domains: PARTITIONING [RT'04,HM'05]

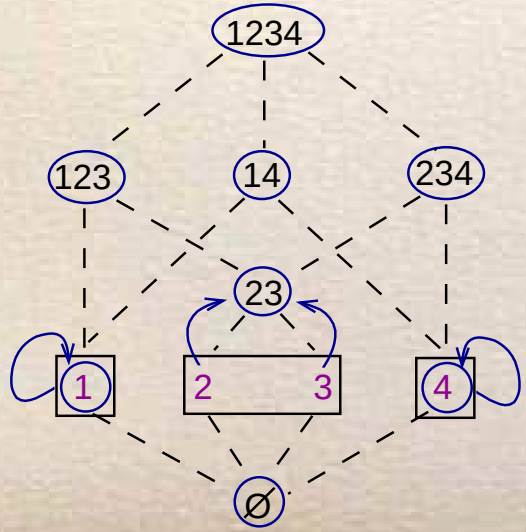
$\eta \in \mathit{uco}(\wp(C)) \;\Rightarrow\; \forall x, y.\, x\ \mathit{Rel}^{\eta}\ y$ iff $\eta(x) = \eta(y)$



PARTITIONS VS ABSTRACT DOMAINS

Partitions uniquely correspond to particular abstract domains: PARTITIONING [RT'04,HM'05]

- $\eta \in \mathit{uco}(\wp(C)) \;\Rightarrow\; \forall x, y.\, x \; \mathit{Rel}^{\eta} \; y$ iff $\eta(x) = \eta(y)$
- $\mathsf{R} \in \mathit{Eq}(C) \;\Rightarrow\; \mathit{Clo}^{\mathsf{R}}(X) \stackrel{\text{def}}{=} \bigcup_{x \in X} [x]_{\mathsf{R}}$

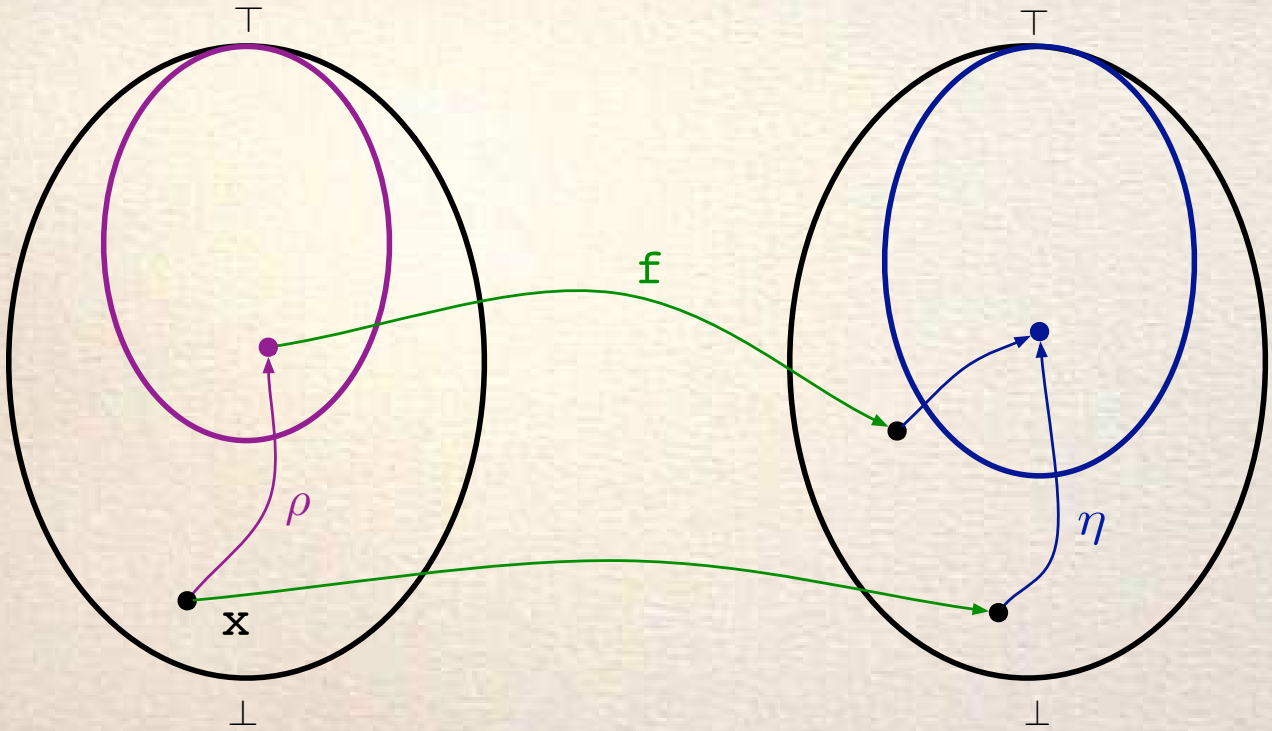


PARTITIONS VS ABSTRACT DOMAINS

Partitions uniquely correspond to particular abstract domains: PARTITIONING [RT'04,HM'05]

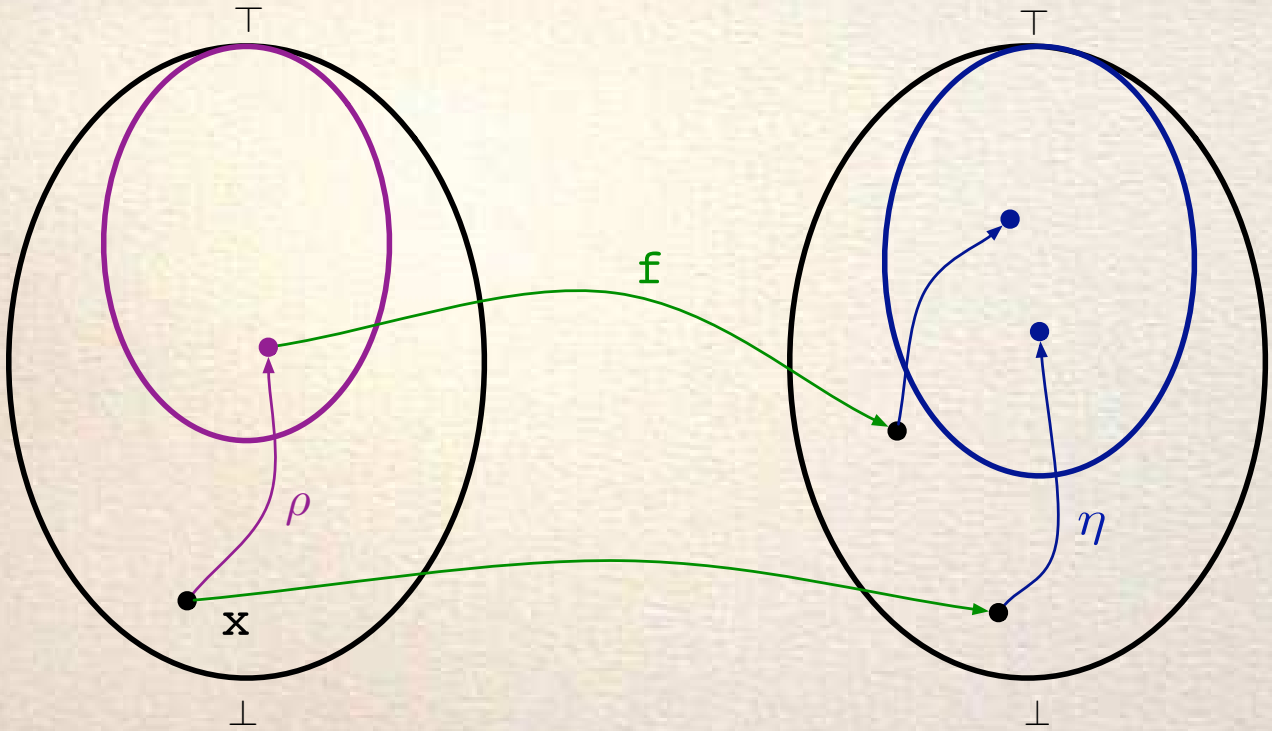
- $\eta \in \mathit{uco}(\wp(C)) \;\Rightarrow\; \forall x, y.\, x\ \mathit{Rel}^{\eta}\ y$ iff $\eta(x) = \eta(y)$
- $\mathsf{R} \in \mathit{Eq}(C) \;\Rightarrow\; \mathit{Clo}^{\mathsf{R}}(X) \stackrel{\mathsf{def}}{=} \bigcup_{x \in X} [x]_{\mathsf{R}}$
- $\Pi(\eta) \stackrel{\mathsf{def}}{=} \mathit{Clo}^{\mathit{Rel}^{\eta}} \sqsubseteq \eta$

COMPLETENESS



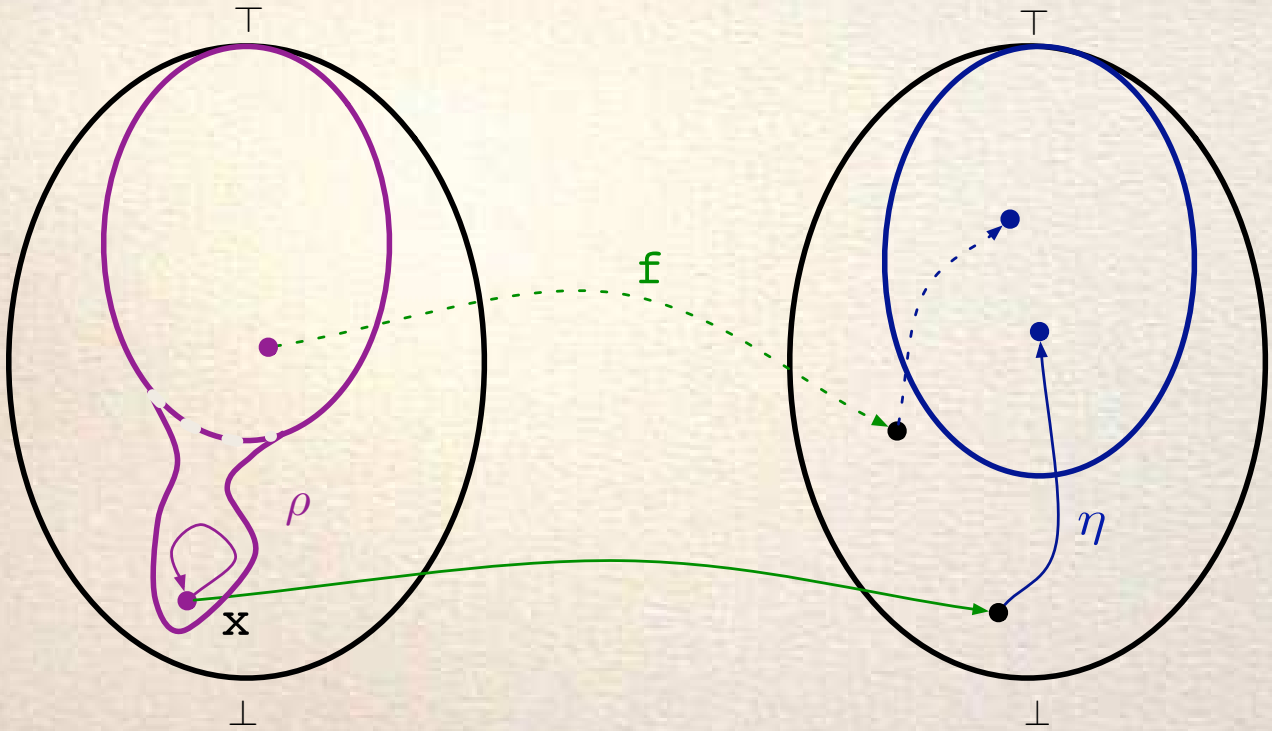
BACKWARD COMPLETENESS: $\eta \circ f \circ \rho = \eta \circ f$

COMPLETENESS



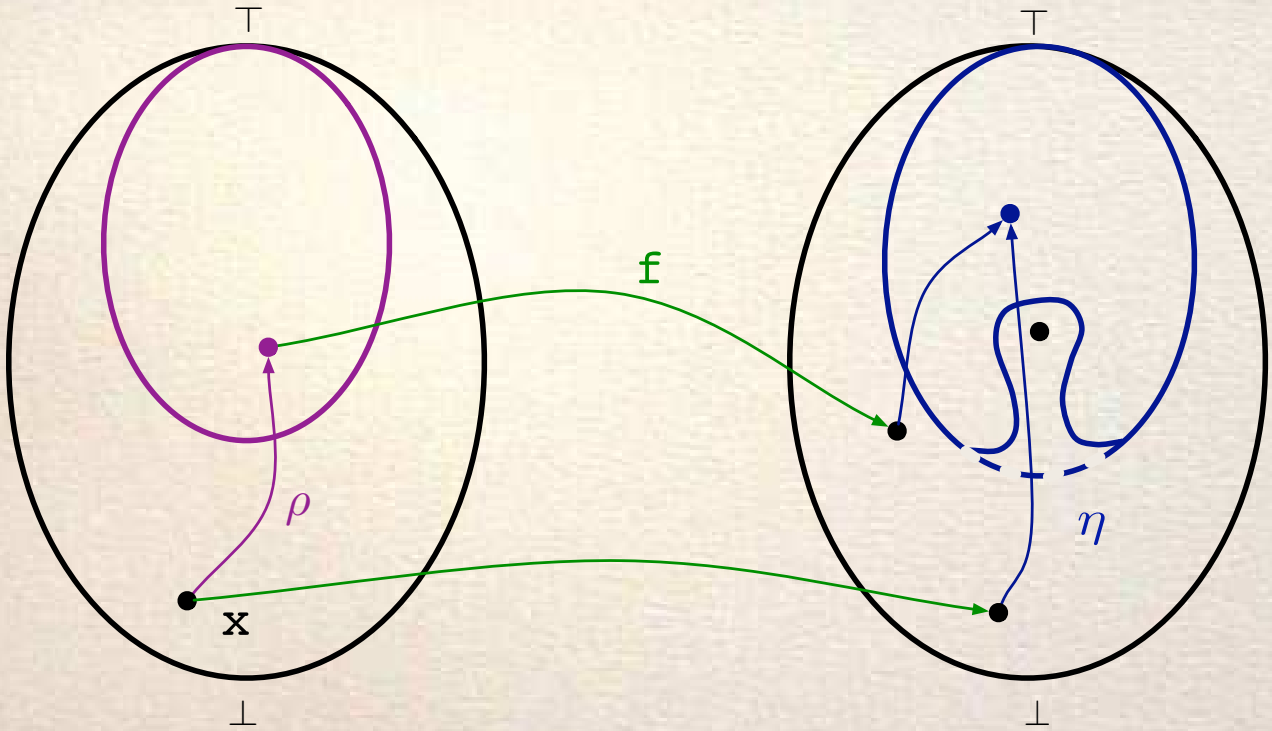
BACKWARD IN-COMPLETENESS: $\eta \circ f \circ \rho \geq \eta \circ f$

COMPLETENESS



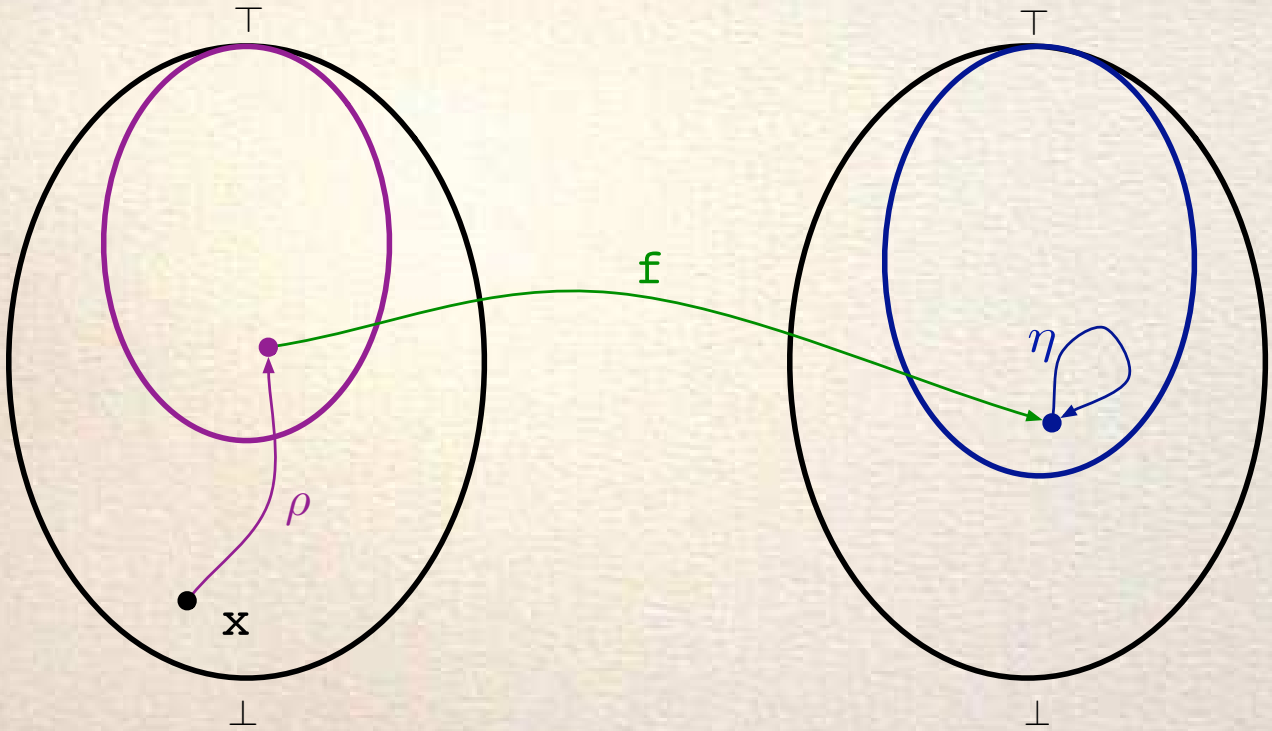
Making BACKWARD COMPLETE: Refining input domains [GRS'00]

COMPLETENESS



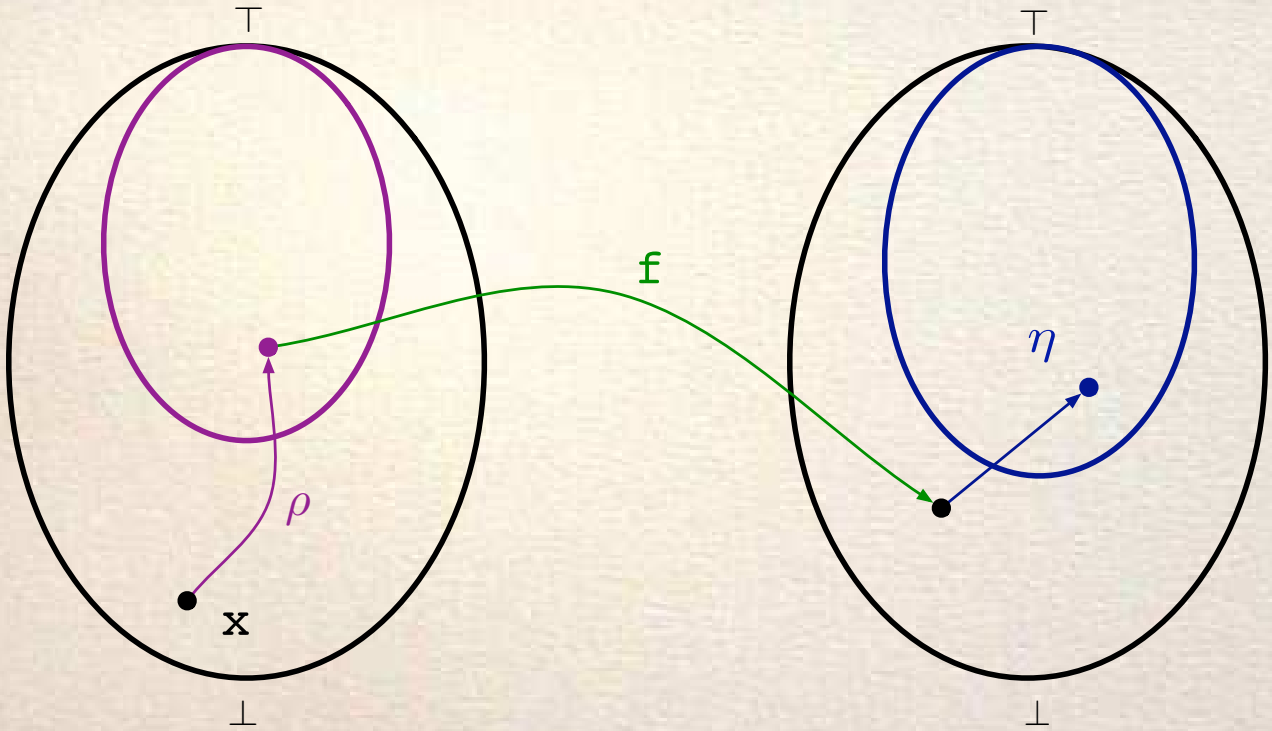
Making BACKWARD COMPLETE: Simplifying output domains [GRS'00]

COMPLETENESS



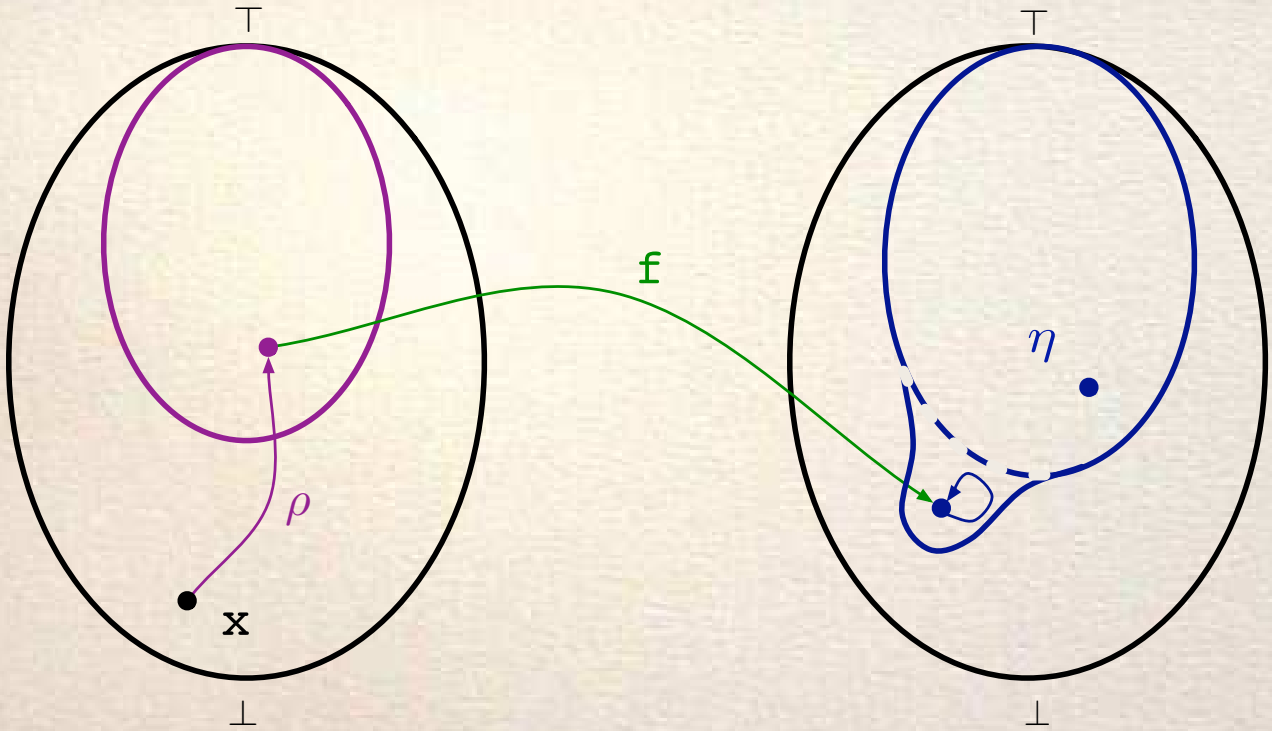
FORWARD COMPLETENESS: $\eta \circ f \circ \rho = f \circ \rho$

COMPLETENESS



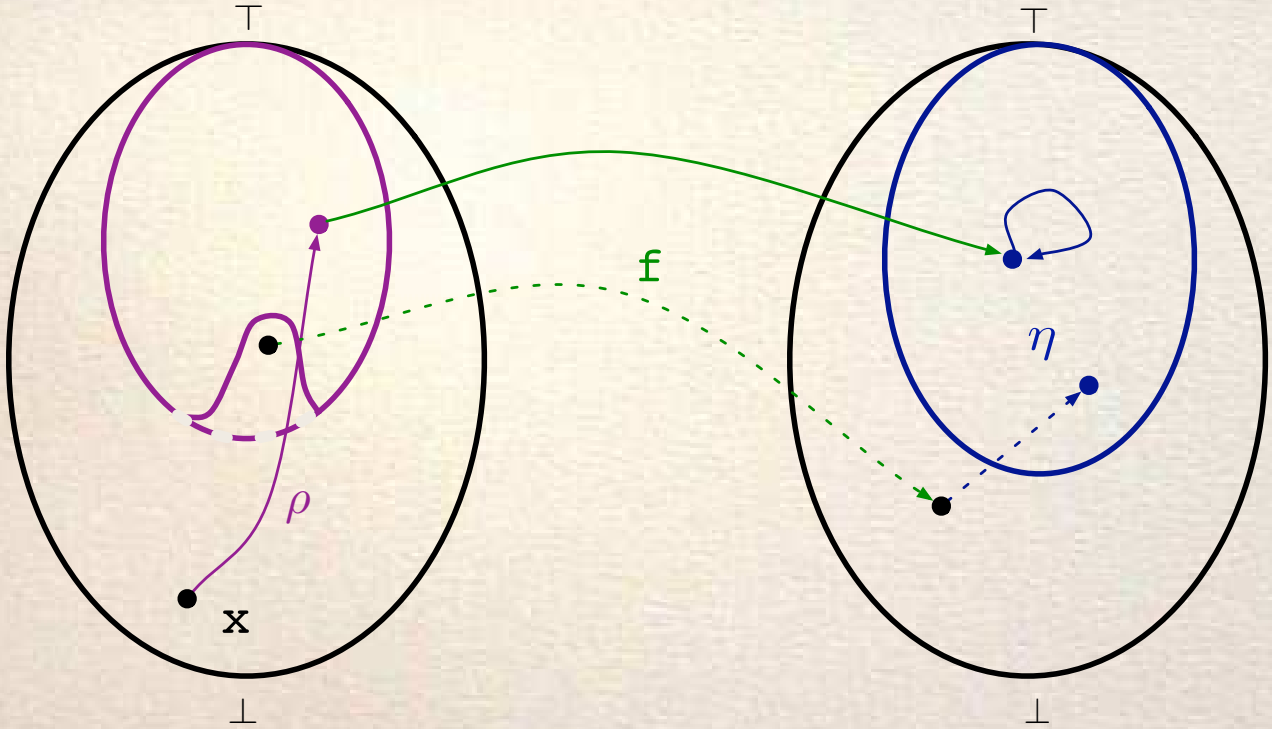
FORWARD IN-COMPLETENESS: $\eta \circ f \circ \rho \geq f \circ \rho$

COMPLETENESS



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

COMPLETENESS



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

STABILITY

Let S and R, resp., an output and an input partition, let *p* be a binary relation:

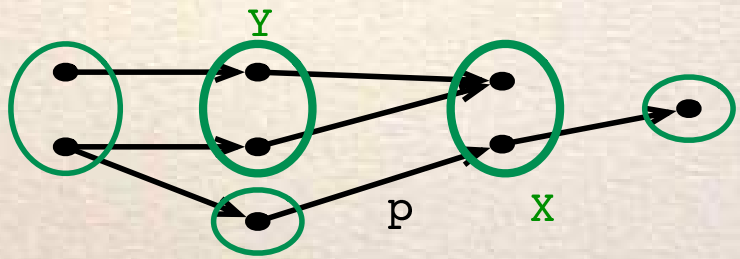
STABILITY: S is *stable* wrt R if $\forall X \in \mathsf{S}$, $Y \in \mathsf{R}$ we have

$$X \cap p(Y) \neq \varnothing \;\Rightarrow\; X \subseteq p(Y)$$

STABILITY

Let S and R, resp., an output and an input partition, let *p* be a binary relation:

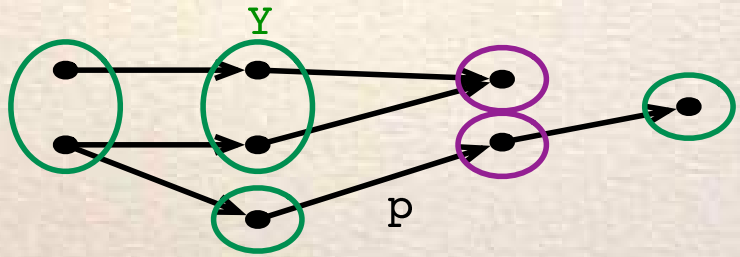
> STABILITY: S is *stable* wrt R if $\forall X \in \mathsf{S}$, $Y \in \mathsf{R}$ we have
> $$X \cap p(Y) \neq \varnothing \;\Rightarrow\; X \subseteq p(Y)$$



STABILITY

Let S and R, resp., an output and an input partition, let *p* be a binary relation:

> STABILITY: S is *stable* wrt R if $\forall X \in \mathsf{S}$, $Y \in \mathsf{R}$ we have
> $$X \cap p(Y) \neq \varnothing \;\Rightarrow\; X \subseteq p(Y)$$



MAIN CONTRIBUTION

WHAT ALREADY EXISTS:

- A correspondence between *stability* and forward completeness [RANZATO & TAPPARO '05];
- A *refinement* algorithm for *partition stability* [PAIGE & TARJAN'87];
- A *refinement* transformer for *abstract domain completeness* [GIACOBAZZI ET AL.'00, GIACOBAZZI & QUINTARELLI'01];
- A *simplification* transformer for *abstract domain completeness* [GIACOBAZZI ET AL.'00, GIACOBAZZI & QUINTARELLI'01];;

MAIN CONTRIBUTION

WHAT DOES NOT EXIST:

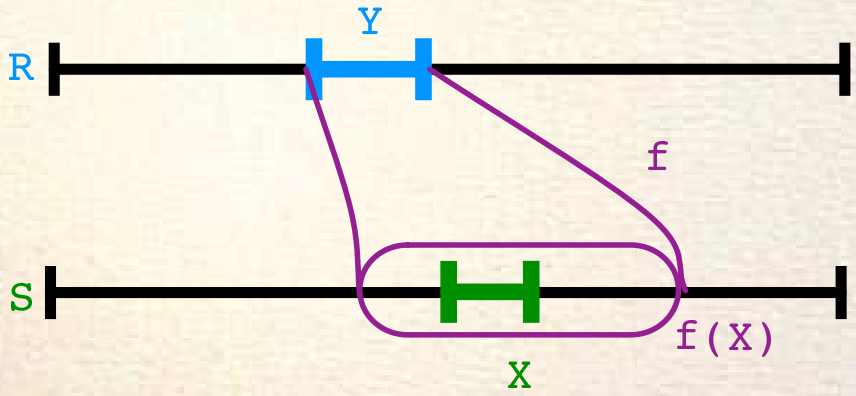
- A characterization of completeness for partitions;
- A notion of partition stability/completeness for the *backward* direction;
- A *simplification* algorithm for *partition stability*

STABILITY VS FORWARD COMPLETENESS

F-STABILITY: $X \cap f(Y) \neq \varnothing \;\Rightarrow\; X \subseteq f(Y)$ [PT'87,RT'05]

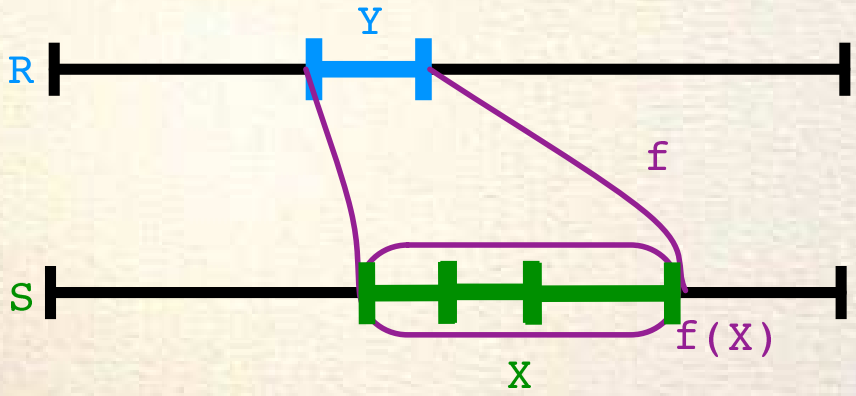
STABILITY VS FORWARD COMPLETENESS

F-STABILITY: $X \cap f(Y) \neq \varnothing \;\Rightarrow\; X \subseteq f(Y)$ [PT'87,RT'05]



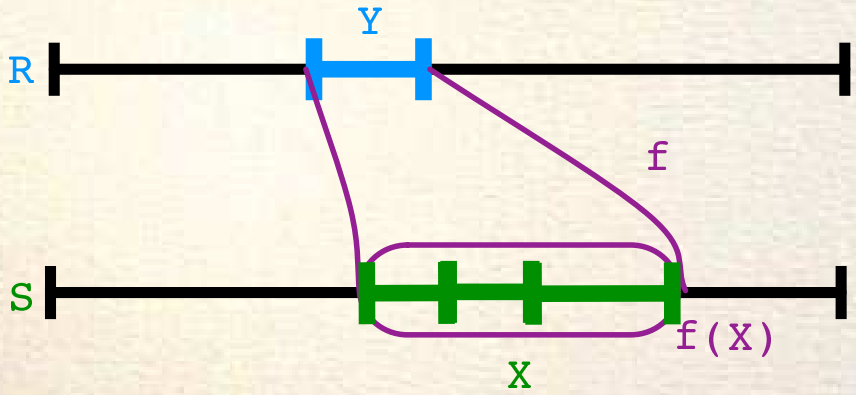
STABILITY VS FORWARD COMPLETENESS

F-STABILITY: $X \cap f(Y) \neq \varnothing \;\Rightarrow\; X \subseteq f(Y)$ [PT'87,RT'05]



STABILITY VS FORWARD COMPLETENESS

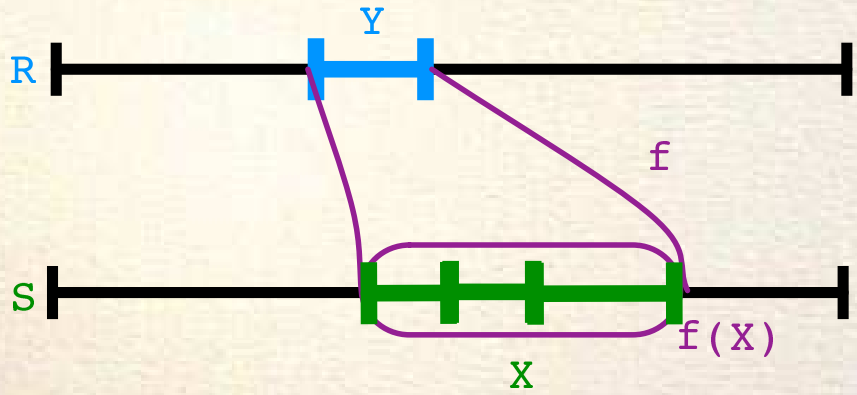
F-STABILITY: $X \cap f(Y) \neq \varnothing \;\Rightarrow\; X \subseteq f(Y)$ [PT'87,RT'05]



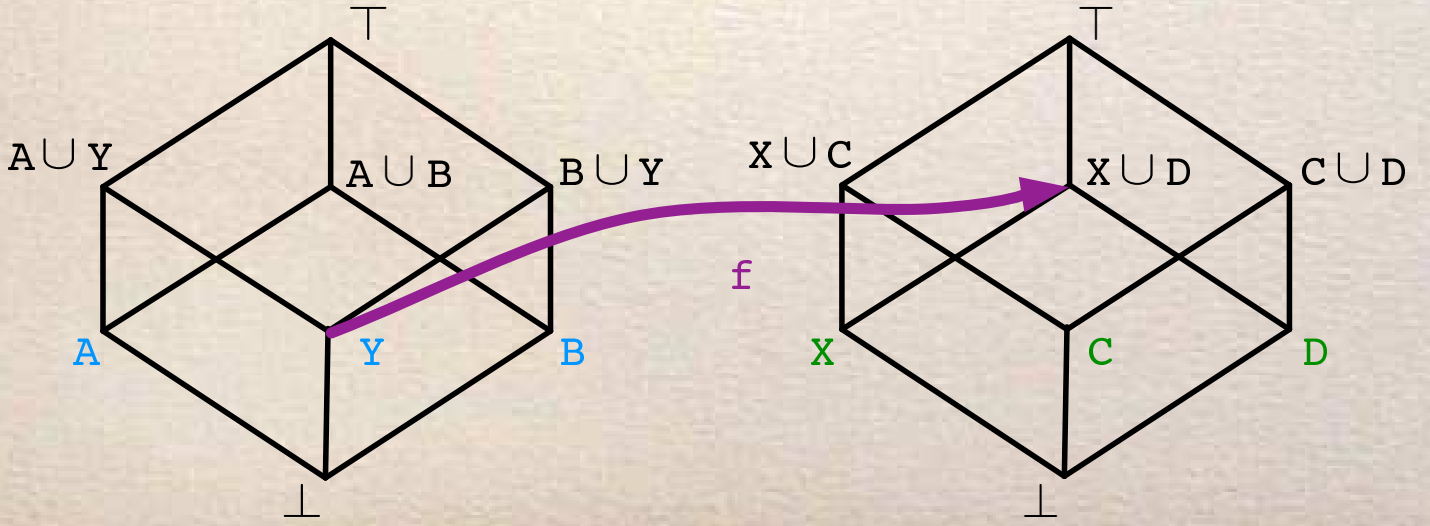
F-COMPLETENESS: $[f([x]_R)]_S = f([x]_R)$

STABILITY VS FORWARD COMPLETENESS

F-STABILITY: $X \cap f(Y) \neq \varnothing \;\Rightarrow\; X \subseteq f(Y)$ [PT'87,RT'05]



F-COMPLETENESS: $[f([x]_R)]_S = f([x]_R) \;\Leftrightarrow\; (\forall X \in Clo_R \;\Rightarrow\; f(X) \in Clo_S)$

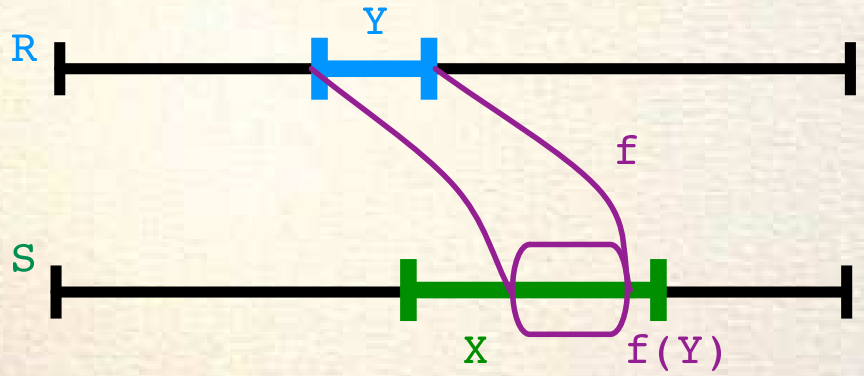


STABILITY VS BACKWARD COMPLETENESS

B-STABILITY: $X \cap f(Y) \neq \varnothing \;\Rightarrow\; f(Y) \subseteq X$

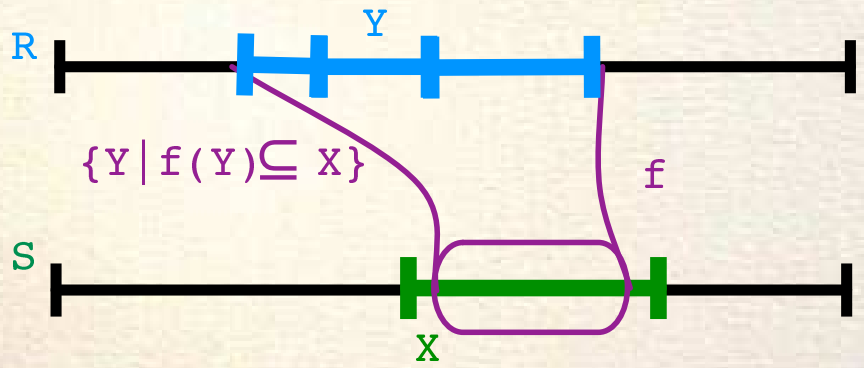
STABILITY VS BACKWARD COMPLETENESS

B-STABILITY: $X \cap f(Y) \neq \varnothing \;\Rightarrow\; f(Y) \subseteq X$



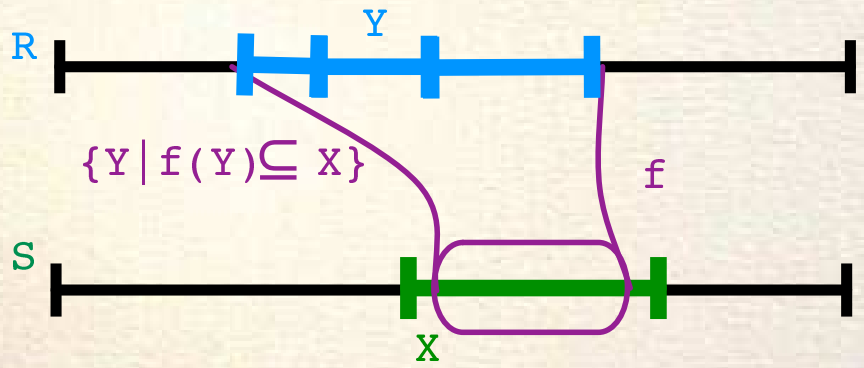
STABILITY VS BACKWARD COMPLETENESS

B-STABILITY: $X \cap f(Y) \neq \varnothing \Rightarrow f(Y) \subseteq X$



STABILITY VS BACKWARD COMPLETENESS

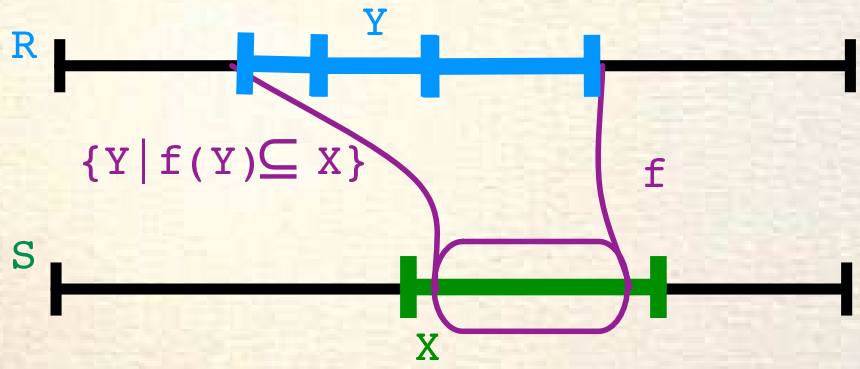
B-STABILITY: $X \cap f(Y) \neq \varnothing \;\Rightarrow\; f(Y) \subseteq X$



B-COMPLETENESS:
$[f([x]_{\mathsf{R}})]_{\mathsf{S}} = [f(x)]_{\mathsf{S}}$

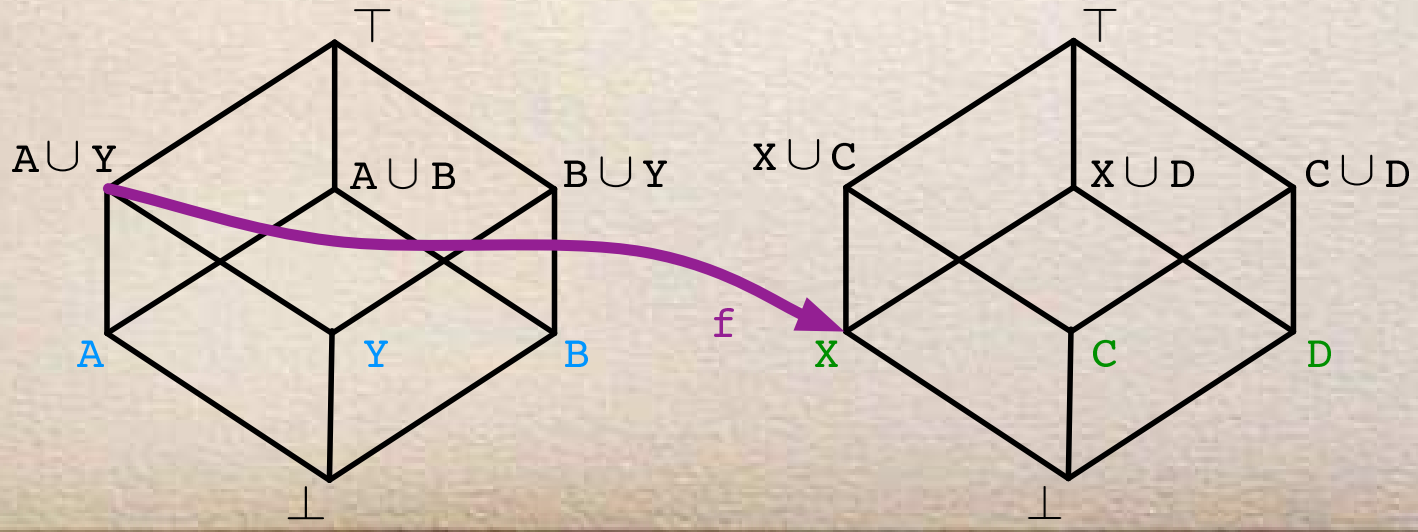
STABILITY VS BACKWARD COMPLETENESS

B-STABILITY: $X \cap f(Y) \neq \varnothing \;\Rightarrow\; f(Y) \subseteq X$



B-COMPLETENESS:

$[f([x]_{\mathsf{R}})]_{\mathsf{S}} = [f(x)]_{\mathsf{S}} \;\Leftrightarrow\; (\forall X \in Clo_{\mathsf{S}} \;\Rightarrow\; \max\left\{\, Y \;\middle|\; f(Y) \subseteq X \,\right\} \in Clo_{\mathsf{R}})$



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};

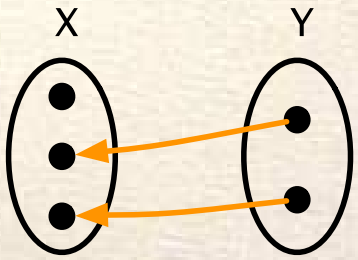
A backward problem can always be transformed in a forward one, but the viceversa is not always possible!

REFINING FOR STABILITY: PT GENERALIZED

The best refinement towards forward stability always exists!
[PT'87]

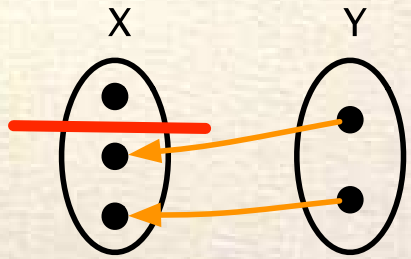
REFINING FOR STABILITY: PT GENERALIZED

The best refinement towards forward stability always exists!
[PT'87]



REFINING FOR STABILITY: PT GENERALIZED

The best refinement towards forward stability always exists!
[PT'87]



REFINING FOR STABILITY: PT GENERALIZED

The best refinement towards forward stability always exists!
[PT'87]

$$\texttt{PTSplit}_f(S,\mathsf{P}) : \begin{cases} \mathsf{P} : \text{Partition} \\ \text{Partition obtained from } \mathsf{P} \text{ by replacing} \\ \text{each block } B \in \mathsf{P} \text{ with } B \cap f(S) \text{ and } B \setminus f(S) \end{cases}$$

$$\texttt{PTRefiners}_f(\mathsf{P}) \stackrel{\text{def}}{=} \left\{ S \;\middle|\; \mathsf{P} \neq \texttt{PTSplit}_f(S,\mathsf{P}) \;\wedge\; \exists \{B_i\}_i \subseteq \mathsf{P}.\, S = \bigcup_i B_i \right\}$$

$$\texttt{PT-Algorithm}_f : \begin{cases} \textbf{while } (\mathsf{P} \text{ is not stable}) \textbf{ do} \\ \quad \textbf{choose } S \in \texttt{PTRefiners}_f(\mathsf{P}); \\ \quad \mathsf{P} := \texttt{PTSplit}_f(S,\mathsf{P}); \\ \textbf{endwhile} \end{cases}$$

[RT'05]

REFINING FOR STABILITY: PT GENERALIZED

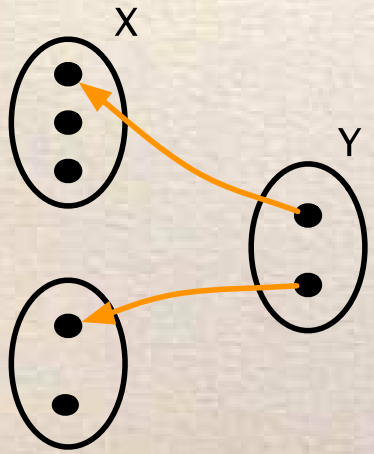
The best refinement towards forward stability always exists!
[PT'87]

The best refinement towards backward stability always exists!

REFINING FOR STABILITY: PT GENERALIZED

The best refinement towards forward stability always exists!
[PT'87]

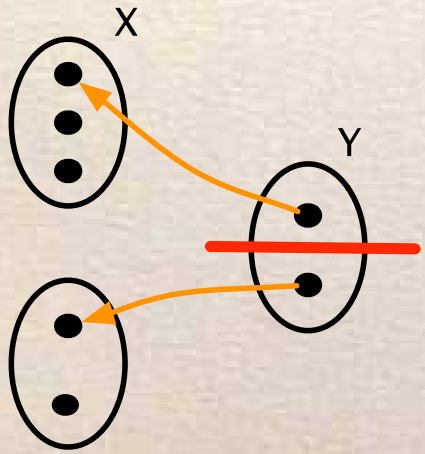
The best refinement towards backward stability always exists!



REFINING FOR STABILITY: PT GENERALIZED

The best refinement towards forward stability always exists!
[PT'87]

The best refinement towards backward stability always exists!



REFINING FOR STABILITY: PT GENERALIZED

The best refinement towards forward stability always exists!
[PT'87]

The best refinement towards backward stability always exists!

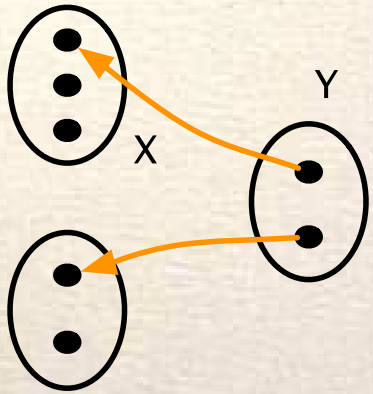
We can use the PT algorithm since a backward problem wrt f corresponds always to a forward problem wrt f^{-1}.

SIMPLIFYING FOR STABILITY

The best simplification towards backward stability always exists!

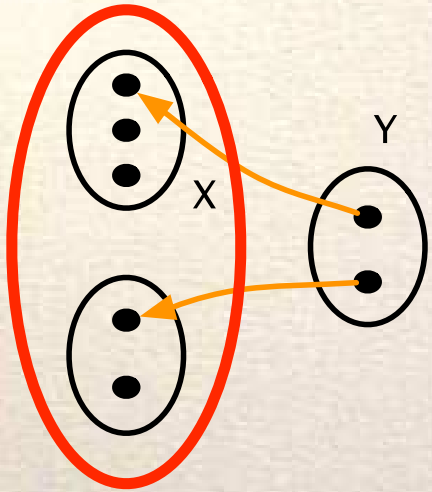
SIMPLIFYING FOR STABILITY

The best simplification towards backward stability always exists!



SIMPLIFYING FOR STABILITY

The best simplification towards backward stability always exists!



SIMPLIFYING FOR STABILITY

The best simplification towards backward stability always exists!

$$\texttt{PTSimplifiers}_f(S) \stackrel{\text{def}}{=} \left\{ X \;\middle|\; X \cap f(S) \neq \varnothing \right\}$$

$$\texttt{PTMerge}_f(S, \texttt{P}) : \begin{cases} \text{Partition obtained from } \texttt{P} \text{ by replacing} \\ \text{all the blocks } X \in \texttt{PTSimplifiers}_f(S) \text{ with} \\ \bigcup \texttt{PTSimplifiers}_f(S) \end{cases}$$

$$\texttt{DPT-Algorithm}_f : \begin{cases} \textbf{while } (\texttt{P} \text{ is not stable}) \textbf{ do} \\ \quad \textbf{choose } S \in \texttt{PTSimplifiers}_f(\texttt{P}); \\ \quad \texttt{P} := \texttt{PTMerge}_f(S, \texttt{P})); \\ \textbf{endwhile} \end{cases}$$

SIMPLIFYING FOR STABILITY

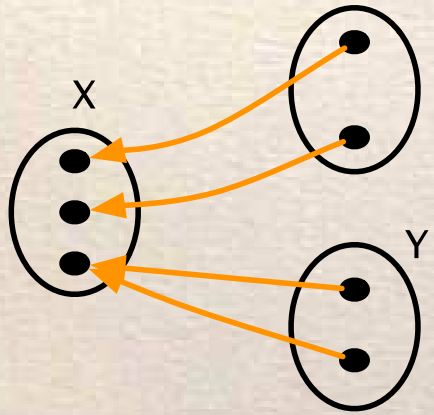
The best simplification towards backward stability always exists!

The best simplification towards forward stability DOES NOT always exist!

SIMPLIFYING FOR STABILITY

The best simplification towards backward stability always exists!

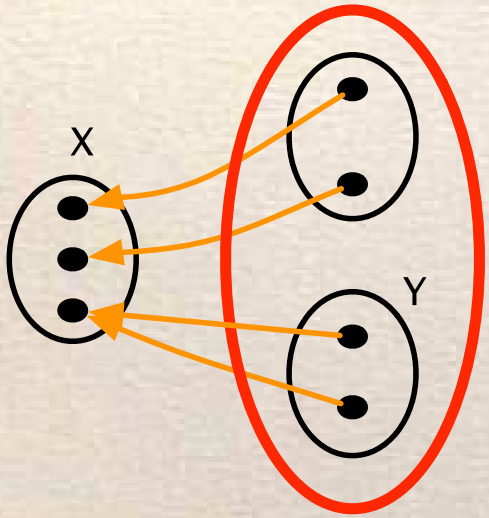
The best simplification towards forward stability DOES NOT always exist!



SIMPLIFYING FOR STABILITY

The best simplification towards backward stability always exists!

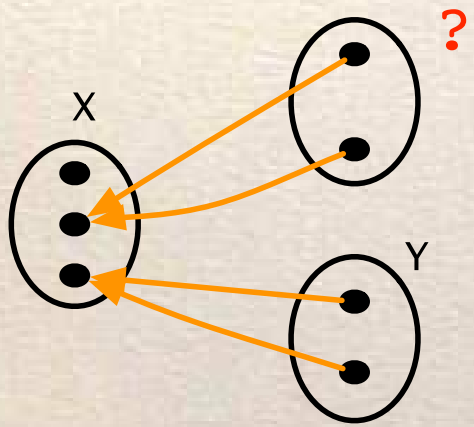
The best simplification towards forward stability DOES NOT always exist!



SIMPLIFYING FOR STABILITY

The best simplification towards backward stability always exists!

The best simplification towards forward stability DOES NOT always exist!



SIMPLIFYING FOR STABILITY

The best simplification towards backward stability always exists!

The best simplification towards forward stability DOES NOT always exist!

EXAMPLE:

Consider $f(x) = 2x$. $\top$ is not stable ($f(\top) = even \subset \top$).

Consider R $= \{even, odd\}$ (Parity partition), then

$even \cap f(odd) \neq \varnothing$ since $6 \in f(3)$ and

$even \not\subseteq f(odd)$ since $4 \notin f(odd)$

$\Rightarrow$ A forward stable simplification does not exist!

STABILITY IN ABSTRACT NON INTERFERENCE

ABSTRACT NON INTERFERENCE: [GM'04]

Fixed an observation of public input, the variation of private input has not to *interfere* with the observation of the public output.

$$\Downarrow$$

$$\forall l_1, l_2 \in \mathbb{V}^{\mathsf{L}}, h_1, h_2 \in \mathbb{V}^{\mathsf{H}}. \eta(l_1) = \eta(l_2) \Rightarrow \rho([\![P]\!](h_1, l_1)^{\mathsf{L}}) = \rho([\![P]\!](h_2, l_2)^{\mathsf{L}})$$

STABILITY IN ABSTRACT NON INTERFERENCE

ABSTRACT NON INTERFERENCE: [GM'04]

Fixed an observation of public input, the variation of private input has not to *interfere* with the observation of the public output.

$$\forall l_1, l_2 \in \mathbb{V}^{\mathsf{L}}, h_1, h_2 \in \mathbb{V}^{\mathsf{H}}. \eta(l_1) = \eta(l_2) \Rightarrow \rho([\![P]\!](h_1, l_1)^{\mathsf{L}}) = \rho([\![P]\!](h_2, l_2)^{\mathsf{L}})$$

Let $\Upsilon(\eta(l))$ denote the sets of value that has to be indistinguishable by a malicious attacker observing η in input ($\Upsilon(\eta(l)) = [\![P]\!](\mathbb{V}^{\mathsf{H}}, \eta(l))^{\mathsf{L}}$).

STABILITY IN ABSTRACT NON INTERFERENCE

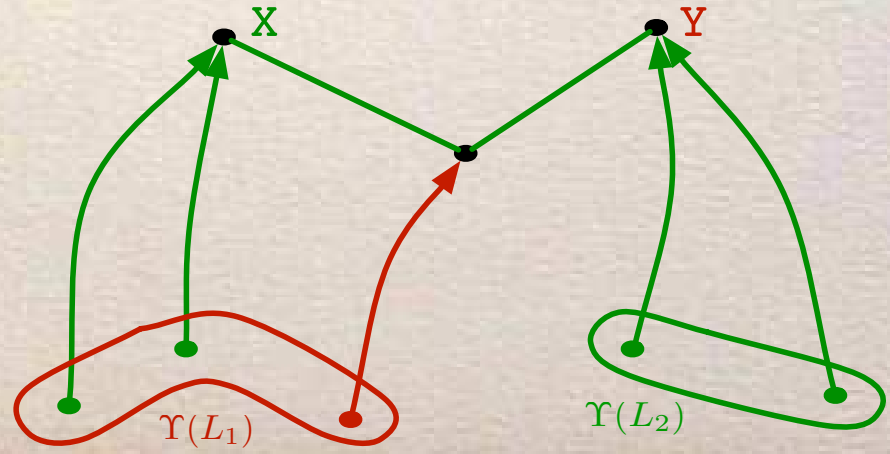
ABSTRACT NON INTERFERENCE: [GM'04]

Fixed an observation of public input, the variation of private input has not to *interfere* with the observation of the public output.

$$\forall l_1, l_2 \in \mathbb{V}^{\mathsf{L}}, h_1, h_2 \in \mathbb{V}^{\mathsf{H}}. \eta(l_1) = \eta(l_2) \Rightarrow \rho([\![P]\!](h_1, l_1)^{\mathsf{L}}) = \rho([\![P]\!](h_2, l_2)^{\mathsf{L}})$$

Let $\Upsilon(\eta(l))$ denote the sets of value that has to be indistinguishable by a malicious attacker observing η in input ($\Upsilon(\eta(l)) = [\![P]\!](\mathbb{V}^{\mathsf{H}}, \eta(l))^{\mathsf{L}}$).

There is interference when:



STABILITY IN ABSTRACT NON INTERFERENCE

ABSTRACT NON INTERFERENCE: [GM'04]

Fixed an observation of public input, the variation of private input has not to *interfere* with the observation of the public output.

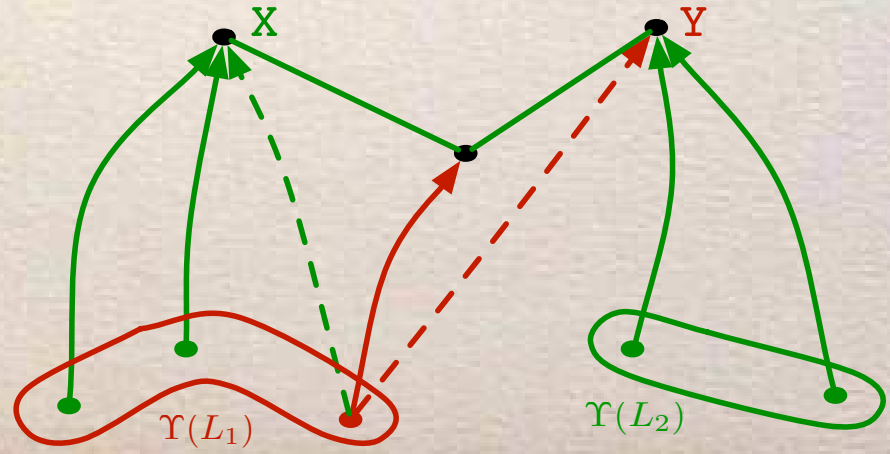
$$\Downarrow$$

$$\forall l_1, l_2 \in \mathbb{V}^{\mathsf{L}}, h_1, h_2 \in \mathbb{V}^{\mathsf{H}}. \eta(l_1) = \eta(l_2) \Rightarrow \rho([\![P]\!](h_1, l_1)^{\mathsf{L}}) = \rho([\![P]\!](h_2, l_2)^{\mathsf{L}})$$

Let $\Upsilon(\eta(l))$ denote the sets of value that has to be indistinguishable by a malicious attacker observing η in input ($\Upsilon(\eta(l)) = [\![P]\!](\mathbb{V}^{\mathsf{H}}, \eta(l))^{\mathsf{L}}$).

There is interference when:

$Y \cap \Upsilon(L_1) \neq \varnothing$ and $\Upsilon(L_1) \not\subseteq Y$



STABILITY IN ABSTRACT NON INTERFERENCE

ABSTRACT NON INTERFERENCE: [GM'04]

Fixed an observation of public input, the variation of private input has not to *interfere* with the observation of the public output.

$$\forall l_1, l_2 \in \mathbb{V}^{\mathsf{L}}, h_1, h_2 \in \mathbb{V}^{\mathsf{H}}. \eta(l_1) = \eta(l_2) \Rightarrow \rho([\![P]\!](h_1, l_1)^{\mathsf{L}}) = \rho([\![P]\!](h_2, l_2)^{\mathsf{L}})$$

Let $\Upsilon(\eta(l))$ denote the sets of value that has to be indistinguishable by a malicious attacker observing η in input ($\Upsilon(\eta(l)) = [\![P]\!](\mathbb{V}^{\mathsf{H}}, \eta(l))^{\mathsf{L}}$).

THEOREM: The domain $\left\{ X \;\middle|\; X \text{ is backward stable wrt } \Upsilon \right\}$ is the strongest harmless attacker for deterministic systems.

STABILITY FOR OPACITY

OPAQUE PREDICATE: A predicate ϕ over the semantics of a system, is opaque wrt the observation function *obs* if, for every execution t_1 satisfying ϕ there is an execution t_2 which does not satisfy ϕ, such that $obs(t_1) = obs(t_2)$. [Bryans et al. '05]

STABILITY FOR OPACITY

OPAQUE PREDICATE: A predicate ϕ over the semantics of a system, is opaque wrt the observation function *obs* if, for every execution t_1 satisfying ϕ there is an execution t_2 which does not satisfy ϕ, such that $obs(t_1) = obs(t_2)$. [Bryans et al. '05]

$\Downarrow$

$\forall t.\ obs(t) \cap \phi \neq \varnothing$ and $obs(t) \not\subseteq \phi$
(ϕ NOT backward stable wrt *obs*)

STABILITY FOR OPACITY

OPAQUE PREDICATE: A predicate ϕ over the semantics of a system, is opaque wrt the observation function *obs* if, for every execution t_1 satisfying ϕ there is an execution t_2 which does not satisfy ϕ, such that $obs(t_1) = obs(t_2)$. [Bryans et al. '05]

$\Downarrow$

$\forall t.\ obs(t) \cap \phi \neq \varnothing$ and $obs(t) \not\subseteq \phi$
(ϕ NOT backward stable wrt *obs*)

EXAMPLE:
$\phi = 3|(x^3 - x)$, attacker capability $\alpha = \{\mathbb{Z}, 3\mathbb{Z}, \mathbb{Z} \setminus 3\mathbb{Z}, \varnothing\}$

$\Rightarrow$ If the attacker can observe the predicate as the bca of all the function composing ϕ then $obs(2) \cap \phi = \mathbb{Z} \cap \phi \neq \varnothing$, while $obs(2) \not\subseteq \phi$.

STABILITY FOR OPACITY

OPAQUE PREDICATE: A predicate ϕ over the semantics of a system, is opaque wrt the observation function *obs* if, for every execution t_1 satisfying ϕ there is an execution t_2 which does not satisfy ϕ, such that $obs(t_1) = obs(t_2)$. [Bryans et al. '05]

$\Downarrow$

$\forall t.\ obs(t) \cap \phi \neq \varnothing$ and $obs(t) \not\subseteq \phi$
(ϕ NOT backward stable wrt *obs*)

$\Rightarrow$ *Completeness* can be exploited for certifying the resilience of opaque predicates to reverse engineering;

$\Rightarrow$ *Opacity* provides new expectations in seeking domain transformers increasing incompleteness,

DISCUSSION

- We extend the existing notion of stability (corresponding to forward completeness for partitions) also to the backward direction;
- We dualize the existing refinement algorithm for stability in order to simplify partitions;
- The simplification algorithm can be considered for simplifying abstract models in abstract model checking;
- We show fields of computer science where the new stability notion models existing concepts:
  - The strongest harmless attacker in abstract non-interference [Giacobazzi & Mastroeni ’04, Hunt & Mastroeni ’05]
  - Opacity for abstract observations of programs