Domain Precision in Galois Connection-less Abstract Interpretation

Isabella Mastroeni[0000−0003−1213−536X] and Michele Pasqua[0000−0002−9475−4836]

University of Verona - Computer Science Department, Verona, Italy
isabella.mastroeni@univr.it and michele.pasqua@univr.it

Abstract. The ever growing pervasiveness of software systems in modern days technology results in an increasing need of *software/program correctness proofs*. The latter, allow developers to spot software failures before production, hence preventing potentially catastrophic repercussions on our society, as in the case of safety-critical infrastructures.
Unfortunately, correctness proofs may fail (even when software is actually correct) due to program analysis imprecision: program analysis sacrifices precision in order to gain decidability. In standard abstract interpretation-based static analyses, such imprecision is "measured" in terms of *completeness* of the chosen observation (i.e., of the chosen abstract domain) w.r.t. the programming language semantics. In this setting, fixed the language language, it is crucial to have decidable techniques to determine whether the chosen abstraction is sufficiently precise to analyze the program under consideration.
In this paper, we characterize abstract domain precision from a novel point of view, providing a formal framework for *characterizing* and (statically) *verifying* abstract domain precision, that can be adopted also in the case of "weakened", i.e., Galois Connection-less, static analysis frameworks. Distinctive examples adopting such frameworks are the Convex Polyhedra and Automata domains, for which standard approaches to reason about analysis precision (i.e., completeness) cannot be applied.

Keywords: Abstract interpretation, Abstract Non-Interference, Completeness, Hyperproperties, (Hyper) Static analysis.

1 Introduction

Software-driven technology is becoming more and more pervasive in our everyday life. For this reason, software failures deeply impact our society (think of safety-critical infrastructures, in which software failures may have serious consequences on public safety). This implies that it is extremely important to be confident in what programs do. Dijkstra [15] remarked that *"the only effective way to raise the confidence level of a program significantly is to give a convincing proof of its correctness"*. Program analysis is a prominent method supporting programmers and software engineers in producing reliable software, and abstract interpretation [10, 11] is a general framework enabling to design correct-by-construction program analysis tools. An abstract interpreter provides an approximate sound semantics of a programming language, by computing programs on an (approximate) *abstract domain*. In this context, soundness here means that all true alarms are captured and reported by the analysis. Of course, false alarms may be reported as well, and their presence is in general due to the need of program analysis to make *decidable* the computation of a semantic program property of interest (e.g., variable overflows, formal specifications verification, etc.). When false alarms are not generated, we say the analysis is precise, or *complete*. There is a wide literature about completeness, that spans from the systematic refinement of abstract domains for achieving completeness [25] to the definition of weakened forms of completeness (for instance, verifying completeness locally, on single input properties [5], or by accepting a bounded error in the abstract computation [6]). However, all these works build up on the archetypal abstract interpretation framework adopting Galois connections between concrete and abstract semantics [10, 11]. Nevertheless, there are several static analyses developed in weaker frameworks of abstract interpretation [12], where some hypotheses are relaxed. This led, for instance, to the development of powerful numerical static analyzers implementing Convex Polyhedra [13], or abstract domains exploited in machine learning for neural networks verification [1], such as Zonotope [26]. In these relaxed abstract interpretation frameworks, completeness has not been formally studied, even if the absence of false alarms remains surely a crucial point to investigate.

The whole literature concerning abstract domain completeness characterization and enforcing, also its weakened forms, is strongly based on the standard, previous called archetypal, framework of abstract interpretation, that is based on Galois connections (or, equivalently, on upper closure operators). In particular, this framework assumes the existence of the best correct approximation (bca for short) of each concrete element. In the present paper, we reason about completeness in a *weakened framework*, where the bca existence assumption (and the need of Galois connections) is relaxed [12]. In this new, more general, framework we provide a characterization of completeness, together with effective verification techniques. This allows to characterize and verify completeness even in the case of Galois connection-less abstract interpretation.

Paper contribution. We first propose a possible *formalization* for a weakened abstract interpretation framework (Sect. 3), generalizing the standard one to abstract domains not modeled by means of Galois connections. Then, we exploit its strong correlation with *Abstract Non-Interference* in order to characterize *abstract domain completeness*. Abstract Non-Interference (ANI for short) has been introduced in the context of language-based security as a non-interference policy between observable properties [18, 31], but it is a much more general property of computations [23]. In literature, it has been observed that ANI (in all its different variants) can be modeled as a completeness problem [21, 20], allowing to exploit completeness transformers, defined in the standard abstract interpretation framework, for deeper understanding ANI in the context of language-based security. However, in [21, 20], the authors consider a security-driven notion of ANI, let us call it *secANI*, where data is partitioned into private and public, and variations of the private input must not affect a property of the public output. Then, they prove that any secANI policy π for a program P can be formulated as a completeness problem for P w.r.t. specific abstractions derived from those defining π. In this paper, we follow the opposite direction: we prove that any completeness problem $\mathcal{C}$ for a program P can be formulated as an ANI (not just secANI) property for P, defined in terms of the abstractions considered in $\mathcal{C}$ (Sect. 4). This becomes particularly useful when we move towards the weaker abstract interpretation framework [12]. Indeed, the completeness results of [25] are built on top of the standard abstract interpretation framework, based on Galois connections, while here we relax such assumption, making our contribution a strict generalization of [25]. Moreover, this relation allows us to adapt the deductive approach defined for ANI [19, 22] in order to cope with completeness, providing a logic system for deducing completeness properties (Subsect. 5). Here, the contribution consists in generalizing the proof system provided in [19, 22] to deal with generic non-interference properties, not necessarily splitting data in public and private.

Finally, we provide an effective method to verify completeness, by exploiting a hyper static analysis [29], or *hyperanalysis* (Sect. 6). The latter, has been developed to verify *hyperproperties* and, in particular, to verify ANI (that is an hyperproperty). In this way, we can prove that the completeness property of the abstract domains is a hyperproperty of program semantics, and therefore it can be analyzed and verified (even if in restricted versions) by means of an hyperanalysis. Indeed, [32] provides a general methodology for verifying hyperproperties by means of Abstract Interpretation. Here, we show how to define an effective (hyper) static analysis, based on abstract interpretation, making completeness verification decidable.

2 Background

If S is a set, $\wp(S)$ denotes the powerset of S. If $f : S \to T$ is a function, we often abuse notation by calling f also its additive lifting $f : \wp(S) \to \wp(T)$ to sets of values: $f(X) \triangleq \{\, f(x) \,|\, x \in X \subseteq S \,\}$. If $f : S \to T$ and $g : T \to U$, we denote by $g \circ f$ (or simply gf) their composition. If $f : S \to S$, and $n \in \mathbb{N}$ we define $f^n : S \to S$ inductively as: $f^0 \triangleq id_S$ (the identity on S); and $f^{n+1} \triangleq f \circ f^n$.

In ordered structures (e.g., lattices) L we use $\leq_L$ to denote its partial order relation, $\vee_L$ to denote its least upper bound (lub), $\wedge_L$ to denote its greatest lower bound (glb), $\top_L$ to denote its top element and $\bot_L$ to denote its bottom element[1]. A function f on complete lattices is additive if it preserves arbitrary lubs (co-additivity is dually defined). The least fixpoint of $f : C \to C$ on a poset C, when it exists, is denoted $\mathit{lfp}\, f$. If f is (Scott)continuous on a complete lattice, then $\mathit{lfp}\, f = \bigvee_{n\in\mathbb{N}} f^n(\bot)$.

If $S \subseteq P$ then $\downarrow S \triangleq \{x \in P \mid \exists y \in S \,.\, x \leq y\}$. We will often denote $f(\{x\})$ as $f(x)$ and $\downarrow\{x\}$ as $\downarrow x$.

[1] We avoid the pedex when the structure is clear form the context or it is not relevant.

2.1 Abstract Interpretation

Abstract interpretation [10, 11] is a formal framework for approximating programs semantics, defined in terms of a concrete domain C and an abstract domain A of C. Given complete lattices C and A, a pair of functions $\alpha : C \to A$ and $\gamma : A \to C$ forms a Galois connection (GC for short) if for any $x \in C$ and $y \in A$ we have $\alpha(x) \leq_A y \Leftrightarrow x \leq_C \gamma(y)$. In this case, α (resp. γ) is the abstraction/left adjoint (resp. concretization/right adjoint), and it is additive (resp. co-additive). Co-additive functions f admits left adjoint $f^- \triangleq \lambda x.\ \bigwedge \{\ y \,|\, x \leq f(y)\ \}$. An *upper closure operator* (uco for short) $\rho : P \to P$ on a poset P is monotone, idempotent, and extensive (i.e., $\forall x \in P.\ x \leq_P \rho(x)$). If $\alpha \circ \gamma = id_A$ then the GC forms a Galois insertion (GI for short) and $\gamma \circ \alpha$ is an uco. Let us denote by $Abs(C)$ the class of abstract domains (GI or uco) of C. In particular, we denote it by $A_{\alpha,\gamma}$ or by A_ρ depending on what we want to make explicit. The *disjunctive completion* of a domain is defined as: $\curlyvee(\rho) \triangleq \bigsqcup\{\eta \in Abs(C) \mid \eta \sqsubseteq \rho \wedge \eta \text{ is additive}\}$. A closure η is called *partitioning* [35] if $\eta = \mathcal{P}(\eta)$, where $\mathcal{P}(\eta) \triangleq \curlyvee(\{[x]_\eta \mid x \in S\})$, with $[x]_\eta \triangleq \{y \mid \eta(x) = \eta(y)\}$, is the most concrete closure inducing the same partition of η.

Soundness and Completeness. Given an abstract domain $A_{\alpha,\gamma} \in Abs(C)$ and a concrete function $f : C \to C$, an abstract function $f^A : A \to A$ is a *sound* approximation of f when $\alpha \circ f \leq_A f^A \circ \alpha$. The best correct approximation (bca for short) of f in A is the function $\overline{f}^A \triangleq \alpha \circ f \circ \gamma$. Any possible approximation of f is less precise or as precise as the bca of f.

The abstract function f^A is a *complete* [11, 25] approximation of f on A if $\alpha \circ f = f^A \circ \alpha$. An abstract domain A is called complete for f if there exists a complete approximation f^A of f. Completeness of f^A intuitively means that f^A is the most precise approximation of f. Completeness can be characterized also in terms of uco. Let $A_\rho \in Abs(C)$, then A is a complete abstraction for f if $\rho \circ f \circ \rho = \rho \circ f$. This implies that completeness is a property of the abstract domain, since if there exists a complete approximation of f, then the bca is itself complete. Abstract domains can be made complete [25]. In a more general setting, let $f : D \to C$ be a function on complete lattices D and C (potentially different), and $A_\rho \in Abs(C)$, $A_\eta \in Abs(D)$ be abstractions, respectively, of input and output domains. In this case, we say that $\langle \rho, \eta \rangle$ is a pair of complete abstract domains for f if $\rho \circ f \circ \eta = \rho \circ f$. A pair of domain transformers can be associated with such completeness problem. We follow [16, 24] by defining *domain refinement* τ_r and *simplification* τ_s as any monotone function $\tau_r, \tau_s : Abs(C) \to Abs(C)$ such that $X \subseteq \tau_r(X)$ and $\tau_s(X) \subseteq X$, respectively. In [25], a constructive characterization of the most abstract refinement, called *complete shell*, and of the most concrete simplification, called *complete core*, of any domain, making it complete, for a given continuous function f, is given as a solution of a simple domain equation. Consider the following operators on closures:

$$R_f \triangleq \lambda X.\, \mathcal{M}(\textstyle\bigcup_{y \in X} \mathsf{max}(f^{-1}(\downarrow y))) \qquad C_f \triangleq \lambda X.\, \{y \in L \mid \mathsf{max}(f^{-1}(\downarrow y)) \subseteq X\}$$

where $\mathcal{M}$ denotes the *Moore closure* (i.e., a function that closes all sets by glb) and max retrieves the maximal elements from a set (for the formal details of these constructions see [25]). In [25], the authors proved that the only interesting cases, as far as the refinement and simplification towards completeness are concerned, are respectively the most concrete $\beta \sqsupseteq \rho$ such that $\langle \beta, \eta \rangle$ is complete and the most abstract $\beta \sqsubseteq \eta$ such that $\langle \rho, \beta \rangle$ is complete. In particular, given $A_\rho \in Abs(C)$, the complete shell of $A_\eta \in Abs(D)$ is $\mathcal{R}^\rho_f(\eta) \triangleq \eta \sqcap R_f(\rho)$; and, given $\eta \in uco(D)$, the complete core of $\rho \in uco(C)$ is $\mathcal{C}^\eta_f(\rho) \triangleq \rho \sqcup C_f(\eta)$. When we consider $f : C \to C$ and the constraint $\eta = \rho$, the above construction requires a fixpoint iteration on abstract domains [25].

2.2 The Reference Language

Following the approach of [5] (see also [37, 34]) we consider the language $\mathfrak{L}$ of regular commands[2] (RComm in Fig. 1, where $\oplus$ denotes non-deterministic choice and $*$ is the Kleene closure), which is general enough to cover deterministic imperative languages as well as other programming paradigms [5]. The language is parametric on the syntax of basic transfer functions $\mathsf{c} \in \mathsf{btFun}$, that can be instantiated with different kinds of instructions such as, for instance, assignments and boolean guards.

The concrete semantics. We consider as concrete semantics a standard denotational semantics. We assume that btFun is provided with a semantics function $(\!|\cdot|\!) : \mathsf{btFun} \to (C \to C)$ on a complete lattice C such that $(\!|\mathsf{c}|\!)$ is additive. This assumption is not restrictive since basic transfer functions are always defined by additive lifting [5]. The concrete semantics $[\![\cdot]\!] : \mathsf{RComm} \to (C \to C)$ of regular commands is then inductively defined as follows:

$$\begin{array}{ll} [\![\mathsf{c}]\!]c \triangleq (\!|\mathsf{c}|\!)c & [\![\mathsf{C}_1 \oplus \mathsf{C}_2]\!]c \triangleq [\![\mathsf{C}_1]\!]c \vee [\![\mathsf{C}_2]\!]c \\ [\![\mathsf{C}^*]\!]c \triangleq \bigvee\{[\![\mathsf{C}]\!]^n c \mid n \in \mathbb{N}\} & [\![\mathsf{C}_1; \mathsf{C}_2]\!]c \triangleq [\![\mathsf{C}_2]\!]([\![\mathsf{C}_1]\!]c) \end{array}$$

The semantics of regular commands corresponds to the denotational semantics defined in [9] starting from the operational semantics for non deterministic choice and iteration [37].

To complete the language, we consider standard basic transfer functions used in deterministic while languages: skip instruction, assignments and boolean guards, as defined in Fig. 1. In $\mathfrak{L}$ we consider just integers values in $\mathbb{Z}$ and integer variables, where *Var* is an enumerable set of variable names. Standard imperative language commands can be easily defined by using guarded branching and loop commands as syntactic sugar [5]:

$$\textbf{if } \mathsf{b} \textbf{ then } \mathsf{c}_1 \textbf{ else } \mathsf{c}_2 \textbf{ fi} \triangleq (\mathsf{b}?; \mathsf{c}_1) \oplus (\neg\mathsf{b}?; \mathsf{c}_2) \quad \textbf{while } \mathsf{b} \textbf{ do } \mathsf{c} \textbf{ ew} \triangleq (\mathsf{b}?; \mathsf{c})^*; \neg\mathsf{b}?$$

[2] We choose to keep the language as simple as possible, avoiding non necessary language features, in order to keep the focus on the analysis from a purely semantic point of view.

$$\begin{aligned}
\text{Exp} \ni \mathsf{e} &::= \mathsf{a} \mid \mathsf{b} \\
\text{AExp} \ni \mathsf{a} &::= x \mid n \mid \mathsf{a} + \mathsf{a} \mid \mathsf{a} - \mathsf{a} \mid \mathsf{a} * \mathsf{a} \\
\text{BExp} \ni \mathsf{b} &::= \mathit{true} \mid \mathit{false} \mid \mathsf{e} = \mathsf{e} \mid \mathsf{e} < \mathsf{e} \mid \mathsf{b} \wedge \mathsf{b} \mid \neg \mathsf{b} \\
\mathsf{btFun} \ni \mathsf{c} &::= \textbf{skip} \mid x := \mathsf{a} \mid \mathsf{b}? \\
\mathsf{RComm} \ni \mathtt{C} &::= \mathsf{c} \mid \mathtt{C}; \mathtt{C} \mid \mathtt{C} \oplus \mathtt{C} \mid (\mathtt{C})^* \qquad \mathit{Var} \ni x \text{ (variables)} \qquad \mathbb{Z} \ni n \text{ (values)}
\end{aligned}$$

Fig. 1. Syntax of the language $\mathfrak{L}$ of regular commands.

A program *memory* $\mathbbm{m} : V \to \mathbb{Z}$ is a total function from a finite set of variables $V \subseteq \mathit{Var}$ to values. Memory update $[x \mapsto v]$ is defined as usual: $\mathbbm{m}[x \mapsto v](x) \triangleq v$, when $x = y$; and $\mathbbm{m}[x \mapsto v](y) = \mathbbm{m}(y)$ otherwise. Let us call $\mathbb{M} \triangleq V \to \mathbb{Z}$ the set of program memories, the concrete domain of $\mathfrak{L}$ semantics is $C \triangleq \wp(\mathbb{M})$, denoting sets of memories on variables in V. The basic transfer function semantics $(\!|\mathsf{c}|\!) : \wp(\mathbb{M}) \to \wp(\mathbb{M})$ is defined, for a set of memories $\mathsf{M} \subseteq \mathbb{M}$, as:

$$(\!|\textbf{skip}|\!)\mathsf{M} \triangleq \mathsf{M} \quad (\!|x := \mathsf{a}|\!)\mathsf{M} \triangleq \{\mathbbm{m}[x \mapsto \{\!|\mathsf{a}|\!\}\mathbbm{m}] \mid \mathbbm{m} \in \mathsf{M}\}$$
$$(\!|\mathsf{b}?|\!)\mathsf{M} \triangleq \{\mathbbm{m} \in \mathsf{M} \mid \{\!|\mathsf{b}|\!\}\mathbbm{m} = \mathit{true}\}$$

where $\{\!|\mathsf{a}|\!\} : \mathbb{M} \to \mathbb{Z}$ and $\{\!|\mathsf{b}|\!\} : \mathbb{M} \to \{\mathit{true}, \mathit{false}\}$ are the standard evaluation semantics for arithmetic and boolean expressions, respectively. For arithmetical expressions, we abuse notation by denoting with $\{\!|\mathsf{a}|\!\}$ also their additive lift to sets of memories.

3 Weak Abstract Interpretation Framework

In the standard abstract interpretation framework [10, 11], based on Galois connections, even when only the concretization function is given, a monotone abstraction function always exists and it can be mathematically derived from the concretization (this is guaranteed by the properties of a GC). Whenever such abstraction cannot be defined, as it happens in the case of Convex polyedra [13] or Automata [3] domains, then it means that the concretization does not yields a GC, i.e., it is not co-additive. Indeed, the requirement of having a GC between a concrete domain C and an abstract one A may be in practice too restrictive (e.g., when developing a static analyzer). In [12] the authors provide a more general framework for abstract interpretation, describing *necessary* and *releasable* (i.e., that can be relaxed) assumptions. In particular, it is strictly required that *any concrete element must have an approximation* (to garantee soundness). For instance, this is violated by concretization functions γ mapping the abstract top to a concrete element strictly smaller than the concrete top, i.e., when $\gamma(\top_A) <_C \top_C$. In this case, some concrete elements (those greater that $\gamma(\top_A)$) do not have a sound approximation. Indeed, even if we map all these elements (including $\top_C$) to $\top_A$, then soundness would be violated since the approximation of $\top_C$ would be an object with a strictly smaller concretization[3]. Instead, the existence of the *best approximation* for each concrete element is releasable assumption. The best abstraction does not exist, for instance, when the set of sound approximations is an infinite strictly descending chain or a set of finite/infinite non-comparable strictly decreasing chains [12].

In this section, we provide a *closure*-based characterization of such abstract domains, by formalizing a weakened abstract interpretation framework where the *abstraction* function may be missing and, therefore, where the adoption of the best abstraction assumption is not viable.

Definition 1 (Weak Adjoint). *Let $\gamma : A \to C$ be a monotone and one-to-one[4] function such that $\gamma(\top_A) = \top_C$, a* weak adjoint *$\gamma^{\sim} : C \to A$ of γ is any function in $\widetilde{\Gamma}(\gamma)$, where:*

$$\widetilde{\Gamma}(\gamma) \triangleq \{\dot{\alpha} : C \to A \mid \forall c \in C, a \in A.\ c \leq_C \gamma \circ \dot{\alpha}(c) \wedge \dot{\alpha} \circ \gamma(a) = a\}$$

The pair $(\gamma, \gamma^{\sim})$, with $\gamma^{\sim}$ weak adjoint of γ, form a *weak Galois connection*. Hence, a weak adjoint simply fixes the approximations for the meanings of abstract elements given by γ (unique by the on-to-one property of γ), and allows a range of possibilities for the other concrete elements, only forcing soundness, as required in [12]. Hence, Def. 1 gives rise to a *family* of abstraction functions, each depending on the strategy adopted to select the elements in $\{a \in A \mid c \leq_C \gamma(a)\}$ when it does not exist an abstract element a such that $c = \gamma(a)$. For instance, when possible, we could take the minimal elements of such set. Furthermore, when considering a not co-additive γ, then $\gamma^{\sim}$ is in general not monotone. However, when γ is co-additive, taking $\gamma^{\sim}(c) = \mathsf{min}\{a \in A \mid c \leq_C \gamma(a)\}$, we have that $\gamma^{\sim}$ is precisely the left adjoint γ^{-} of γ, since a minimal element always exists and it is unique (the glb).

Theorem 1. *Let $\gamma : A \to C$ be a monotone and one-to-one function such that $\gamma(\top_A) = \top_C$, and $\gamma^{\sim} \in \widetilde{\Gamma}(\gamma)$, then $\gamma^{\sim} \circ \gamma = id$, and $\rho \triangleq \gamma \circ \gamma^{\sim}$ is idempotent, i.e., $\forall c \in C . \rho \circ \rho(c) = \rho(c)$, and extensive, i.e., $\forall c \in C . c \leq_C \rho(c)$.*

Proof. For the sake of readability, we denote $\gamma^{\sim}$ more concisely with $\dot{\alpha}$, and we omit the function composition operator.

- $\dot{\alpha}\gamma = id$ holds by construction.
- $\gamma\dot{\alpha}$ is idempotent, indeed: $\gamma\dot{\alpha}\gamma\dot{\alpha} = \gamma(\dot{\alpha}\gamma)\dot{\alpha} = \gamma\dot{\alpha}$.
- $\gamma\dot{\alpha}$ is extensive, indeed: if $c = \gamma(a)$, for some $a \in A$, then $\gamma\dot{\alpha}(c) = \gamma(a) = c$, hence trivially we have $c \leq \gamma\dot{\alpha}(c)$; otherwise, $\gamma\dot{\alpha}(c) \in \gamma(\{a' \in A \mid c \leq_C \gamma(a')\})$, if $\{a' \in A \mid c \leq_C \gamma(a')\} \neq \varnothing$, then its γ image is greater than c, otherwise it is $\top$, again greater than c.

In other words, by losing the abstraction we lose closure monotonicity, while keeping the other properties. We call such operators *weak closures*.

[3] In GC-based abstract interpretation, this condition is implied by γ being co-additive.

[4] The one-to-one hypothesis is not restrictive, being implicit in the GI-based framework. Indeed, γ can always be made one-to-one by collapsing the elements of A with the same concrete meaning w.r.t. γ.

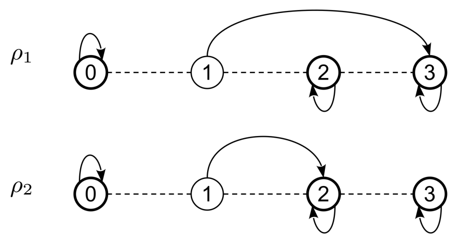

Fig. 2. Different weak closures with the same set of fixpoints.

Definition 2 (Weak closure). *Let $\langle C, \leq_C\rangle$ be a poset. Then, $\rho : C \to C$ is a* weak upper closure operator*, or* weak closure*, if it is idempotent and extensive.*

We denote with $wAbs(C)$ the set of all weak abstract interpretations of C, characterized by a weak GC or, equivalently, by a weak closure.

The set $wAbs(C)$, equipped with the standard point-wise ordering between functions $\sqsubseteq$, forms a partial order. Note that, differently from upper closure operators, weak closures cannot be uniquely identified by the set of their fixpoints. Indeed, it is easy to find different weak closures having the same set of fix points, as depicted in Fig. 2. Here, we have two weak closures ρ_1 and ρ_2[5] on the poset $\langle\{0, 1, 2, 3\}, \leq\rangle$, that have the same set of fixpoints $\{0, 2, 3\}$, even if $\rho_2 \sqsubset \rho_1$.

Example 1. Consider the abstract domain REG [3, 2] of automata/regular languages on a finite alphabet Σ. It is well known that automata/regular languages are not closed by infinite intersection, and therefore they do not form an abstract domain in the standard framework. In particular, if we consider as concrete domain the powerset of all strings on the alphabet Σ, i.e., $C = \wp(\Sigma^*)$, we define the abstract domain $\textsc{Reg} \triangleq \{L \subseteq \Sigma^* \mid \exists D \in \text{DFA}.\ L = \mathcal{L}(D)\}$, where DFA is the set of deterministic finite state automata, and $\mathcal{L}(D)$ is the language of strings on Σ accepted by D. Also non-deterministic finite state automata NFA precisely characterize regular languages. The fact that REG is not closed by infinite intersection means that it does not exists an abstraction function associating with any $L \in \wp(\Sigma^*)$ the least regular language containing L. However, a monotone and one-to-one concretization function is the identity (any regular language is a language), hence we can build a weak Galois connection by fixing the regular overapproximation to associate with any L. In order to be a bit more precise, we could provide a way to associate a regular language overapproximating a context-free language. A context-free language L is such that there exists a pushdown automata P, i.e., $P \in \text{PDA}$, such that $L = \mathcal{L}(P)$. Then, let $\gamma_{\textsc{Reg}} \triangleq id$ on $\wp(\Sigma^*)$, we define $\gamma^{\sim}_{\textsc{Reg}}$ for each $L \in \wp(\Sigma^*)$ as:

$$\gamma^{\sim}_{\textsc{Reg}}(L) \triangleq \begin{cases} L & \text{if } \exists D \in \text{DFA}.\ L = \mathcal{L}(D) \\ \mathcal{L}(N) & \text{if } \exists P \in \text{PDA}.\ L = \mathcal{L}(P) \ \wedge\ N \triangleq \mathsf{reg}(P) \\ \top & \text{otherwise} \end{cases}$$

[5] The function ρ_2 is also an upper closure operator.

Without entering in too much details, reg takes the transition relation δ_P of P (depending also on the symbol on the top of the stack) and defines the transition relation δ_N of a NFA N, ignoring the stack, as:

$$\delta_N(q, a) \triangleq \{q' \mid \exists A \text{ stack symbol in } P.\, \delta(q, a, A) = q'\}$$

Note that, we do not care about decidability issues concerning the existence of DFA and/or PDA, since we do not necessarily need a minimality condition, namely we could map to $\top$ regular or CF languages for which we are not able to provide an automaton.

Completeness (Subsect. 2.1) can be defined in this weakened framework, simply by considering $A_\eta \in wAbs(D)$ and $A_\rho \in wAbs(C)$. In the following, we will call such extended notion *weak completeness*. As we will see in the next sections, some known results holding in the standard abstract interpretation framework hold also in the weak framework, making them applicable even to Galois connection-less abstract domains (like Convex Polyhedra and Automata).

4 Characterizing Weak Completeness

In this section, we characterize domain completeness transformers in the proposed *weakened* framework of abstract interpretation. The existing completeness transformers [25] are strongly based on the monotonicity assumption, relaxed in weak abstract interpretation, hence, they cannot be naively generalized. The idea we propose here consists in exploring the connection between domain completeness and Abstract Non-Interference [18, 23], where similar domain transformers have been characterized without using monotonicity.

It is worth noting that, this is not the first attempt to explore such kind of connection. In [20, 21], the authors provide a completeness model for ANI in the context of language-based security, where input and output values are only partially observable (i.e., split in public and secret data). In particular, in such previous works we can observe that:

1. abstractions are modeled in the *standard* GC-based abstract interpretation framework (requiring even more restrictive conditions on closures);
2. domain completeness is instantiated (by using *specific* abstractions, e.g., abstracting secret data to the top) to model ANI in language-based security.

In this paper, we will investigate such connection by taking the opposite direction, applicable to a wider context: we will provide an ANI model of domain completeness in the weak framework of abstract interpretation. In particular:

1. abstractions are modeled as *weak* upper closure operators (Def. 2), making the results strictly more general;
2. ANI is instantiated to model abstract domain completeness in the (weak) abstract interpretation framework.

In this way, we can exploit the ANI domain transformers [18, 23] for *making abstract domains (weak) complete*, namely complete also in the weak framework of abstract interpretation (i.e., when monotonicity constraint on the closures is relaxed), and to extend the static verification approaches existing for ANI [30, 32, 19] to cope with (weak) completeness.

Abstract Non-Interference. Non-interference has been introduced in [8, 27, 36] as a confidentiality policy determining whether the variation of sensible input information has effect on the observable part of the computation. This notion has been weakened by considering the variation of properties affecting a, potentially, abstract computation of functions [23]. Moreover, also the split of data in secret/relevant and public/observable can be seen as an abstraction of the data. Formally, Non-Interference has been generalized and weakened by means of abstract interpretation, as follows.

Definition 3 (Abstract Non-Interference [23]). *Let $f : D \to C$, $A_\eta \in Abs(D)$ and $A_\rho \in Abs(C)$. We say that f satisfies* Abstract Non-Interference[6] *w.r.t. $\langle \eta, \rho \rangle$, written $[\eta]f(\rho)$, if:*

$$\forall x_1, x_2 \in D.\, \eta(x_1) = \eta(x_2) \;\Rightarrow\; \rho \circ f(x_1) = \rho \circ f(x_2) \tag{1}$$

We recall that, for ANI we have two domain transformers, parametric on the set of observable and relevant input and output data, allowing to characterize: the most concrete harmless (i.e., unable to observe variations) output abstraction ρ (fixed the other observations) [23]; and the most abstract observable input property η, called *revealed information*, (fixed the other observations) [23].

Also Eq. 1 can be defined in the weak abstract interpretation framework, by considering $A_\eta \in wAbs(D)$ and $A_\rho \in wAbs(C)$. In this case we say that f satisfies *weak ANI.*

4.1 Completeness is Abstract Non-Interference

We now show how to require completeness of abstract domains w.r.t. a function means to prove that the function inputs sharing the same property are mapped into outputs that also share the same property (that is precisely ANI). This highlights the strong connection between Completeness and ANI, allowing us to export the domain transformers defined for ANI in the context of static analysis, and precisely in the weak abstract interpretation framework. In addition, there is another important side effect. Completeness has been always characterized as a domain property, even if local completeness [5] showed how much completeness depends on the way the analyzed program is written. Indeed, rewriting completeness as ANI, allows us to see completeness as a program *hyperproperty* [4, 29] instead of as a domain property [11, 25]. The deep gain in this change of

[6] Note that what we call here Abstract Non-Interference is a specific version of the notion in [23], there called narrow.

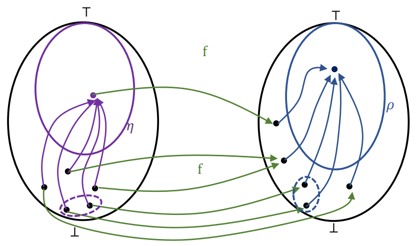

Fig. 3. Completeness is Abstract Non-Interference.

perspective consists in making it possible to provide a framework for verifying completeness on programs by static analysis. Let us start by proving the connection between completeness and ANI.

Theorem 2. *Let $f : D \to C$, $A_\eta \in wAbs(D)$ and $A_\rho \in wAbs(C)$. We have that $\langle \rho, \eta \rangle$ is weak complete w.r.t f iff f satisfies weak ANI w.r.t. $\langle \eta, \rho \rangle$, namely: $\rho \circ f \circ \eta = \rho \circ f \;\Leftrightarrow\; [\eta]f(\rho)$.*

Proof. ($\Rightarrow$) Suppose $\langle \rho, \eta \rangle$ completeness holds, namely $\forall x \in D \,.\, \rho \circ f \circ \eta(x) = \rho \circ f(x)$. We have to prove that ANI holds, namely $\forall x_1, x_2 \in D \,.\, \eta(x_1) = \eta(x_2) \Rightarrow \rho \circ f(x_1) = \rho \circ f(x_2)$. Suppose $\eta(x_1) = \eta(x_2)$, otherwise the implication vacuously holds. Then we have $\rho \circ f(x_1) = \rho \circ f \circ \eta(x_1) = \rho \circ f \circ \eta(x_2) = \rho \circ f(x_2)$. Hence, ANI $[\eta]f(\rho)$ holds. ($\Leftarrow$) Suppose ANI holds, namely $\forall x_1, x_2 \in D \,.\, \eta(x_1) = \eta(x_2) \Rightarrow \rho \circ f(x_1) = \rho \circ f(x_2)$. We have to prove that $\langle \rho, \eta \rangle$ completeness holds, namely that $\forall x \in D \,.\, \rho \circ f \circ \eta(x) = \rho \circ f(x)$. Let assume that $\eta(x) \neq x$, otherwise completeness trivially holds. Let $x_1 = \eta(x) \in D$ and $x_2 = x \in D$. Then we have $\eta(x_1) = \eta \circ \eta(x) = \eta \circ \eta(x_2) = \eta(x_2)$. Since we suppose that ANI holds, we have that $\eta(x_1) = \eta(x_2)$ implies $\rho \circ f(x_1) = \rho \circ f(x_2)$. By definition, we have that $x_1 = \eta(x)$ and $x_2 = x$, resulting in $\rho \circ f \circ \eta(x) = \rho \circ f(x)$. Since x has been chosen arbitrarily, we can conclude that $\langle \rho, \eta \rangle$ completeness holds.

It is trivial to observe that, since closures are particular (i.e., monotone) weak-closures, by Thm. 2, we have that the equivalence between completeness and ANI holds also in the standard, GC-based, abstract interpretation framework.

Corollary 1. *Let $f : D \to C$, $A_\eta \in Abs(D)$ and $A_\rho \in Abs(C)$. We have that $\langle \rho, \eta \rangle$ is complete w.r.t f iff f satisfies ANI w.r.t. $\langle \eta, \rho \rangle$.*

In other words, completeness requires that all the inputs sharing the same property η are led by f to outputs sharing the same property ρ, while ANI checks the same relation on pairs of computations (as depicted in Fig. 3).

4.2 Making Abstract Domains Weak Complete

Most Concrete Weak Complete Output Observation. Here, we aim at characterizing the (weak) completeness core [25], by exploiting the new formalization of (weak) completeness in terms of (weak) Abstract Non-Interference.

First, let us define the set of all the images by $f : D \to C$ resulting from elements having the same abstraction $A_\eta \in wAbs(D)$, that is, $\forall x \in D$ we define the set $\kappa^\eta_f(x) \triangleq \{f(y) \mid \eta(y) = \eta(x)\}$.

In particular, this means that if $\eta(x_1) = \eta(x_2)$, then also $\kappa^\eta_f(x_1) = \kappa^\eta_f(x_2)$. At this point, we aim at abstracting in the same way all the concrete elements in $\{f(y) \mid \eta(y) = \eta(x)\}$. Following the construction proposed for ANI [18], let us define the following predicate determining which concrete elements do not cause different abstractions for elements in $\{f(y) \mid \eta(y) = \eta(x)\}$. Let $y \in C$:

$$Nint^\eta_f(y) \;\Leftrightarrow\; (\exists x \in D\, \exists z \in \kappa^\eta_f(x) \,.\, (y \geq z \;\Rightarrow\; y \geq \bigvee \kappa^\eta_f(x)))$$

This predicate holds for y if whenever y is above the image (by f) of an element x, then it is above the images of all the elements sharing the same η property with x.

Finally, let us define $\eta^\wedge_f \triangleq \{y \in C \mid Nint^\eta_f(y)\}$, which is the abstract domain for the output elements of f unable to distinguish the execution of f starting from inputs with the same property η. It is worth noting that, since we characterize $\eta^\wedge_f$ by identifying its fix points, we necessarily obtain an uco. In other words, this simplification provides the most concrete *uco* satisfying completeness, not excluding the potential existence of more concrete weak complete weak closures.

Lemma 1. *Let $f : D \to C$, $A_\eta \in wAbs(D)$ and $A_\rho \in wAbs(C)$, then $\eta^\wedge_f$ is an upper closure operator of C.*

Theorem 3. *Let $f : D \to C$ and $A_\eta \in wAbs(D)$. $\eta^\wedge_f \in uco(C)$ is the* most concrete (not weak) output observation *such that f satisfies ANI w.r.t. $\langle \eta^\wedge_f, \eta \rangle$.*

The previous result provides a characterization of the completeness domain core that is applicable even in the weak abstract interpretation framework. However, as already observed, the construction will always result in a standard upper closure operator, which guarantees the uniqueness of the construction but not necessarily its optimality. Therefore, the development of techniques for constructing optimal solutions in the weak abstract interpretation framework remains an interesting avenue for further study. As expected, whenever we project the construction in the standard framework, this transformer collapses to the well known one presented in [25].

Corollary 2. *If $\rho \in Abs(C)$ and $\eta \in Abs(D)$, then $\rho \sqcup \eta^\wedge_f$ is the most concrete abstraction more abstract than ρ making f to satisfy ANI w.r.t. $\langle \rho \sqcup \eta^\wedge_f, \eta \rangle$ and, hence, it is exactly the* completeness core *[25] of ρ w.r.t. f.*

This corresponds to take only the elements of ρ satisfying the predicate $Nint^\eta_f$. If we aim at computing the most concrete output observation with a fixed input observation we have just to compute precisely $\eta^\wedge_f$.

Example 2. Let us consider a very simple example, the semantics of $0 < x?$, which, for $\mathsf{M} \in \wp(\mathbb{M})$ is $(\!|0 < x?|\!)\mathsf{M} = \mathsf{M} \cap \{\, \mathbb{m} \in \mathbb{M} \,|\, \mathbb{m}(x) > 0 \,\}$. Let us consider the input observation $\widetilde{Sign} \in wAbs(\wp(\mathbb{M}))$ defined on memories as $\widetilde{Sign}(\mathsf{M}) \triangleq \lambda x \in Var.\, \widetilde{Sign}(\{\mathbb{m}(x) \mid \mathbb{m} \in \mathsf{M}\})$[7] and where, abusing notation, we define

$$\widetilde{Sign} \triangleq \lambda X \in \wp(\mathbb{Z}).\begin{cases} \mathbb{Z}_{\leq 0} \text{ if } X \subseteq \left\{\, n \in \mathbb{Z} \,\middle|\, n \leq 0 \,\right\} \triangleq \mathbb{Z}_{\leq 0} \\ \mathbb{Z}_{\geq 0} \text{ if } X \subseteq \left\{\, n \in \mathbb{Z} \,\middle|\, n \geq 0 \,\right\} \triangleq \mathbb{Z}_{\geq 0},\ X \neq \{0\} \\ \varnothing \quad \text{if } X = \varnothing \\ \mathbb{Z} \quad \text{otherwise} \end{cases}$$

This is a weak closure since, for instance, $\{0\} \subseteq \{0,1\}$ but $\widetilde{Sign}(\{0\}) = \mathbb{Z}_{\leq 0} \not\leq_{\widetilde{Sign}} \widetilde{Sign}(\{0,1\}) = \mathbb{Z}_{\geq 0}$.
Now, for each M, and therefore for any $\widetilde{Sign}(\mathsf{M})$, we have to build $\kappa^{\widetilde{Sign}}_{(\!|0\,<\,x?|\!)}(\mathsf{M})$, which is the set of all the elements that should have the same output abstraction. Since $\widetilde{Sign}$ has only four fix points, we have only for of such sets:

$$\left\{\, (\!|0 < x?|\!)\mathsf{M}' \,\middle|\, \widetilde{Sign}(\mathsf{M}') = [x \mapsto \mathbb{Z}_{\leq 0}] \,\right\} = \varnothing = \left\{\, (\!|0 < x?|\!)\mathsf{M}' \,\middle|\, \widetilde{Sign}(\mathsf{M}') = [x \mapsto \varnothing] \,\right\}$$
$$\left\{\, (\!|0 < x?|\!)\mathsf{M}' \,\middle|\, \widetilde{Sign}(\mathsf{M}') = [x \mapsto \mathbb{Z}_{\geq 0}] \,\right\} = \mathbb{Z}_{>0} = \left\{\, (\!|0 < x?|\!)\mathsf{M}' \,\middle|\, \widetilde{Sign}(\mathsf{M}') = [x \mapsto \mathbb{Z}] \,\right\}$$

Hence, $\widetilde{Sign}^{\wedge}_{(\!|0\,<\,x?|\!)} = \{\mathbb{Z}, \varnothing, \mathbb{Z}_{>0}\} \cup \{X \mid X \subseteq \mathbb{Z}_{\leq 0}\}$, meaning that any set of positive numbers should be abstracted in the same way in output, while we can observe precisely any set of negative numbers.

Most Abstract Weak Complete Input Perturbation. Now we aim at characterizing the (weak) completeness shell [25], by exploiting the new formalization of (weak) completeness in terms of (weak) Abstract Non-Interference. Note that, in this case we may not have a most abstract complete input perturbation, but a family of optimal input perturbations.

Hence, we are going to identify the set of elements that should share the same input property. Let us characterize the sets of elements that an optimal input perturbation can associate with $x \in D$, depending on $f : D \to C$, $A_\rho \in wAbs(C)$ abstraction on C, where $\mathsf{max}(X)$ extracts the maximal elements from X. Let

$$\rho_f^{\vee} \in \left\{\, \mu : D \to D \,\middle|\, \forall x.\, \mu(x) \in \mathsf{max}\{y \in D \mid \rho f(y) = \rho f(x)\} \,\right\}$$

This function associates with each element the most abstract one (in the worst case it is the identity on the element) sharing the same output observation ρ of its image by f.

Lemma 2. *Let $f : D \to C$, $A_\rho \in wAbs(C)$, then we have $A_{\rho_f^{\vee}} \in wAbs(D)$.*

Theorem 4. *Let $f : D \to C$ and $A_\rho \in wAbs(C)$. We have that $\rho_f^{\vee}$ is maximal, w.r.t. $\sqsubseteq$, among the input observations such that f satisfies ANI w.r.t. $\langle \rho, \rho_f^{\vee} \rangle$.*

The above result characterizes the completeness domain shell even in the weak abstract interpretation framework, and unlike the core, the construction of the shell can result in an optimal solution within the entire weak framework. As expected, whenever we make the construction in the standard framework, this transformer collapses to the well known one designed in [25].

[7] In the following, $\lambda x \in Var.X$ (or $[x \mapsto X]$) denotes the set $\{\mathbb{m} \in \mathbb{M} \mid \mathbb{m}(x) \in X\}$.

Corollary 3. *If $A_\eta \in Abs(D)$ and $A_\rho \in Abs(C)$, then $\eta \sqcap \rho_f^\vee$ is the ANI shell of η w.r.t. f, and therefore it is the* completeness shell *[25] of η w.r.t. f.*

Example 3. Let us consider the simple semantics $(\!|2 * x|\!)\mathsf{M}$ for all $\mathsf{M} \in \wp(\mathbb{M})$. Let Par be the standard parity domain $\{\mathbb{Z}, \mathbb{Z}^{even}, \mathbb{Z}^{odd}, \varnothing\}$ returning the parity of sets of integer numbers, let us define

$$\widetilde{ParSign} \triangleq \lambda X \in \wp(\mathbb{Z}).\ \begin{cases} \mathbb{Z}^{Par(X)}_{\leq\, 0} & \text{if } X \subseteq \left\{\, n \in \mathbb{Z} \,\middle|\, n \leq 0 \,\right\}, Par(X) \neq \mathbb{Z} \\ \mathbb{Z}^{Par(X)}_{\geq\, 0} & \text{if } X \subseteq \left\{\, n \in \mathbb{Z} \,\middle|\, n \geq 0 \,\right\},\ X \neq \{0\}, Par(X) \neq \mathbb{Z} \\ Par(X) & \text{otherwise} \end{cases}$$

with fix points denoted $\{\mathbb{Z}, \mathbb{Z}^{even}, \mathbb{Z}^{odd}, \mathbb{Z}^{even}_{\leq\, 0}, \mathbb{Z}^{odd}_{\leq\, 0}, \mathbb{Z}^{even}_{\geq\, 0}, \mathbb{Z}^{odd}_{\geq\, 0}, \varnothing\}$, where, for instance, $\mathbb{Z}^{even}_{\leq\, 0} \triangleq \left\{\, n \,\middle|\, n \leq 0, n \text{ even} \,\right\}$ (analogous for the other cases). As $\widetilde{Sign}$ of the previous example, also $\widetilde{ParSign}$ is a weak closure. In this case, we have to characterize the sets of maximal elements with the same output abstraction, hence we have to find out how many of such sets we have w.r.t. $\widetilde{ParSign}$: First of all we observe that the results may only be of even numbers, hence the only output fix points to consider are those involving even numbers. In the following, we abuse notation by using $\widetilde{ParSign}$ on integers and on memories

$$\begin{aligned}
&\mathsf{max}\{\mathsf{M}' \mid \widetilde{ParSign}((\!|2 * x|\!)\mathsf{M}') = [x \mapsto \mathbb{Z}^{even}_{\leq\, 0}]\} = [x \mapsto \mathbb{Z}_{\leq\, 0}] \\
&\mathsf{max}\{\mathsf{M}' \mid \widetilde{ParSign}((\!|2 * x|\!)\mathsf{M}') = [x \mapsto \mathbb{Z}^{even}_{\geq\, 0}]\} = [x \mapsto \mathbb{Z}_{\geq\, 0}] \\
&\mathsf{max}\{\mathsf{M}' \mid \widetilde{ParSign}((\!|2 * x|\!)\mathsf{M}') = [x \mapsto \mathbb{Z}^{even}]\} = [x \mapsto \mathbb{Z}] \\
&\mathsf{max}\{\mathsf{M}' \mid \widetilde{ParSign}((\!|2 * x|\!)\mathsf{M}') = [x \mapsto \varnothing]\} = [x \mapsto \varnothing]
\end{aligned}$$

Hence, $\widetilde{ParSign}^{\vee}_{(\!|2\, *\, x|\!)} = \widetilde{Sign}$, meaning that any more abstract closure would abstract in the same way sets with different output observations, violating ANI.

5 A Deductive System for Completeness

The proposed domain transformers are surely important for deeper understanding how completeness can be characterized, but they do not provide effective methods for retrieving complete abstract domains given a program. In order to make such task effective, we may derive complete domains inductively on the language structure, similarly to what has been done for ANI [19, 22]. The idea is to exploit the given transformers on the language expressions, and then to propagate abstractions inductively on regular commands structure.

Indeed, we show how the deductive system for ANI [19, 22] can be rewritten for the language $\mathfrak{L}$, in the weak abstract interpretation framework. The inference rules of the deductive system for completeness are given in Fig. 4, where for the sake of readability we write $[\eta]\mathsf{C}(\rho)$ instead of $[\eta][\![\mathsf{C}]\!](\rho)$.

The two axioms of rule **R0** just assert trivial facts, namely that it always holds completeness when the input distinguishes everything or when the output does not distinguish anything. Rule **R1** provides the two axioms deriving from the theorems determining complete shell and core of the abstractions involved

$$\textbf{R0: } [\eta]\mathsf{C}(\mathbb{T}) \quad [id]\mathsf{C}(\rho) \qquad \textbf{R1: } [\eta]\mathsf{e}(\eta^{\wedge}_{\{\!|\mathsf{e}|\!\}}) \quad [\rho^{\vee}_{\{\!|\mathsf{e}|\!\}}]\mathsf{e}(\rho) \qquad \textbf{R2: } \frac{\eta \sqsubseteq \rho}{[\eta]\mathbf{skip}(\rho)}$$

$$\textbf{R3: } \frac{[\eta]\mathsf{a}(\rho) \quad \eta \sqsubseteq \rho \text{ if } \mathit{Var} \supsetneq \{x\}}{[\eta]\, x := \mathsf{a}(\rho)} \qquad \textbf{R4: } \frac{[\eta]\mathsf{b}(\rho)}{[\eta]\, \mathsf{b}?(\rho)} \qquad \textbf{R5: } \frac{[\eta]\mathsf{C}(\rho) \;\; \rho \sqsubseteq \eta}{[\eta]\mathsf{C}^*(\eta)}$$

$$\textbf{R6: } \frac{[\eta]\mathsf{C}_1(\rho) \;\; [\eta_1]\mathsf{C}_2(\rho_1) \;\; \rho \sqsubseteq \eta_1}{[\eta]\mathsf{C}_1;\ \mathsf{C}_2(\rho_1)} \qquad \textbf{R7: } \frac{[\eta_1]\mathsf{C}_1(\rho_1) \;\; [\eta_2]\mathsf{C}_2(\rho_2)}{[\eta_1 \sqcap \eta_2]\mathsf{C}_1 \oplus \mathsf{C}_2(\rho_1 \sqcup \rho_2)}$$

$$\textbf{R8: } \frac{[\eta_1]\mathsf{C}(\rho_1) \;\; \eta \sqsubseteq \eta_1 \;\; \rho_1 \sqsubseteq \rho}{[\eta]\mathsf{C}(\rho)} \qquad \textbf{R9: } \frac{\forall i \in I \,.\, [\eta_i]\mathsf{C}(\rho) \;\; \eta_i \text{ partitioning}}{[\bigsqcup_{i\in I} \eta_i]\mathsf{C}(\rho)}$$

$$\textbf{R10: } \frac{\forall i \in I \,.\, [\eta]\mathsf{C}(\rho_i)}{[\eta]\mathsf{C}(\bigsqcup_{i\in I} \rho_i)} \qquad \textbf{R11: } \frac{\forall i \in I \,.\, [\eta]\mathsf{C}(\rho_i)}{[\eta]\mathsf{C}(\bigsqcap_{i\in I} \rho_i)}$$

Fig. 4. Deriving complete abstractions for the language $\mathfrak{L}$ of regular commands.

(Subsect. 4.2), w.r.t. the semantics of the expression. As far as **skip** is concerned **R2**, then still we need the input abstraction η to be more concrete than the output one. **R3**, when dealing with more than one variable, has to ensure (for all the unchanged variables) that η implies ρ. **R4** is quite immediate. Rule **R5** inductively handle iterations. **R6** and **R7** are as expected. Rule **R8** tells us that we can always concretize the input observation and abstract the output observation. Finally, the last rules change input and output abstractions exploiting operations on abstract domains. Note that rule **R9** is applicable only if η_i are all partitioning, namely they necessarily need to be closures, and not just pseudo-closures. This is not a limit of the deductive systems, since all these rule are useful for improving the precision of the deduction but are not necessary for its soundness.

When $[\eta]\mathsf{C}(\rho)$ can be proved in the deductive system in Fig. 4 we write $\vdash [\eta]\mathsf{C}(\rho)$. The pseudo-closures deduced by the derivation $\vdash$ are the most abstract domains for which completeness holds, w.r.t a given program in $\mathfrak{L}$.

Theorem 5 (Soundness). *Let* $\mathsf{C} \in \mathfrak{L}$ *and* $A_\eta, A_\rho \in wAbs(\wp(\mathbb{M}))$*. If* $\vdash [\eta]\mathsf{C}(\rho)$ *then* $\langle \rho, \eta \rangle$ *is (weak) complete for* C*.*

Example 4. Let $\mathsf{C} \triangleq (0 < x?; x := 2 * x)^*; \neg(0 < x)?$. In this case the concrete domain is $\wp(\mathbb{M})$. Let us consider $\widetilde{Sign}$ defined in Ex. 2 and $Par\widetilde{Sign}$ in Ex. 3.
In Ex. 2 we proved that $[\widetilde{Sign}]0 < x(\widetilde{Sign}^{\wedge}_{(\!|0\,<\,x?|\!)})$. By rule **R4** we have that $[\widetilde{Sign}]0 < x?(\widetilde{Sign}^{\wedge}_{(\!|0\,<\,x?|\!)})$ hence, by rule **R8**, we have $[\widetilde{Sign}]0 < x?(\widetilde{Sign})$, being $\widetilde{Sign}^{\wedge}_{(\!|0\,<\,x?|\!)} \sqsubseteq \widetilde{Sign}$.
Moreover, in Ex. 3 we proved that $[\widetilde{Sign}]2 * x(Par\widetilde{Sign})$. Then, by rule **R3**, having

$\mathtt{C}$ only one variable, we have that $[\widetilde{Sign}]x := 2 * x(Par\widetilde{Sign})$. Hence, we can apply **R6** obtaining $[\widetilde{Sign}]0 < x?; x := 2 * x(Par\widetilde{Sign})$, and then, being $Par\widetilde{Sign} \sqsubseteq \widetilde{Sign}$, by **R5**, we derive $[\widetilde{Sign}](0 < x?; x := 2 * x)^*(\widetilde{Sign})$.
Finally, we can prove that $[\widetilde{Sign}]\neg(0 < x)(Par\widetilde{Sign})$ with a reasoning similar to $0 < x$. Hence, we can apply again **R4** to derive $[\widetilde{Sign}]\neg(0 < x)?(Par\widetilde{Sign})$ and **R6** to obtain $[\widetilde{Sign}](0 < x?; x := 2 * x)^*; \neg(0 < x)?(Par\widetilde{Sign})$.

6 Statically Verifying (Weak) Completeness

Program verification aims at checking weather a program complies with a specification, i.e. a formal description of what programs are allowed and are not allowed to do. Recently, *hyperproperties* [7] have been introduced in order to formalize those specifications that cannot be checked observing *single* program executions. The behavior (semantics) $\mathcal{S}[\mathtt{C}]$ of a program $\mathtt{C}$ is usually modeled as *a set* of *denotations* (e.g., execution traces), one for every possible input. In this setting, properties are sets of execution denotations and hyperproperties are collections of sets of executions. This tantamount to say that a program $\mathtt{C}$ satisfies a property P iff $\mathcal{S}[\mathtt{C}] \subseteq P$, while it satisfies a hyperproperty HP iff $\{\mathcal{S}[\mathtt{C}]\} \subseteq HP$. It turns out that a lot of interesting specifications, like (Abstract) Non-Interference, are hyperproperties [4, 29] and, hence, they require specific verification mechanisms that go beyond the classic one adopted for program properties. An interesting side effect of our work goes towards the direction advocated in [14] exploring the idea of analyzing static analyses. Indeed, by proving that abstract domain completeness is a program hyperproperty we show how a property of the abstract domain (used for analyzing a program), can be seen as a hyperproperty of the program to analyze, and, hence, that can be statically verified on the program.

In this section, we firstly investigate the correlation between completeness and hyperproperties. In particular, we will shows that abstract domain completeness w.r.t. a program can be restated as a hyperproperty verification problem for such program. Then, by exploiting previous results on hyperproperties verification, we will show how completeness can be statically verified.

6.1 Hypercompleteness: Completeness as a Hyperproperty

Hyperproperties allow to specify complex program specifications (e.g., in the context of concurrent systems or security [7]) and, of course, they are crucial in making precise practical verification mechanisms. But, we believe that hyperproperties play also a more fundamental role in program analysis. Indeed, thanks to the equivalence between completeness and ANI (a hyperproperty), in this section we can prove that *checking completeness of an analysis w.r.t. a program boils down to check a hyperproperty of that program.* Of course such result is just a first step in the field of analyzing program analyses: whether this correlation is limited to completeness or can be generalized to any analysis property deserves further investigation.

A Program Semantics Suited for Hyperproperty Verification. As outlined in [4, 30], in order to verify hyperproperties, it is necessary to lift the concrete semantics to sets of sets, namely we need a *hypersemantics*. In [30], the authors show how to define a *correct* hypersemantics, starting from the concrete language semantics. Correct here means that the hypersemantics contains the concrete one. For instance, in the case of $\mathfrak{L}$, a hypersemantics[8] $[\![\mathsf{C}]\!]_{\mathcal{H}} : \wp(\wp(\mathbb{M})) \to \wp(\wp(\mathbb{M}))$ of the program C is correct when $[\![\mathsf{C}]\!]\mathsf{M} \in [\![\mathsf{C}]\!]_{\mathcal{H}}\{\mathsf{M}\}$, for any $\{\mathsf{M}\} \subseteq \wp(\mathbb{M})$. With an over-approximation of a (correct) hypersemantics we can soundly verify hyperproperties. The over-approximation is given by an *abstract hypersemantics*, computing on a suitable abstract (hyper)domain.

We can define the hypersemantics of programs in $\mathfrak{L}$ (denoted by the subscript $\mathcal{H}$) following the construction presented in [30, 32]. The (hyper) transfer function for basic commands in btFun is $(\!|\cdot|\!)_{\mathcal{H}} : \mathsf{btFun} \to (\wp(\wp(\mathbb{M})) \to \wp(\wp(\mathbb{M})))$, where $\wp(\wp(\mathbb{M}))$ is a complete lattice by definition and $(\!|\mathsf{c}|\!)_{\mathcal{H}}$ is the additive lift to sets of the concrete semantics $[\![\mathsf{c}]\!] : \wp(\mathbb{M}) \to \wp(\mathbb{M})$ of the language. In particular, for any set of sets of memories $\mathfrak{M} \subseteq \wp(\mathbb{M})$ we define:

$$(\!|\mathbf{skip}|\!)_{\mathcal{H}}\mathfrak{M} \triangleq \mathfrak{M} \quad (\!|x := \mathsf{a}|\!)_{\mathcal{H}}\mathfrak{M} \triangleq \{[\![x := \mathsf{a}]\!]\mathsf{M} \mid \mathsf{M} \in \mathfrak{M}\}$$
$$(\!|\mathsf{b}?|\!)_{\mathcal{H}}\mathfrak{M} \triangleq \{(\!|\mathsf{b}|\!)\mathsf{M} \mid \mathsf{M} \in \mathfrak{M}\}$$

The hypersemantics $[\![\cdot]\!]_{\mathcal{H}} : \mathsf{RComm} \to (\wp(\wp(\mathbb{M})) \to \wp(\wp(\mathbb{M})))$ of regular commands is inductively defined as follows, for any set of sets of memories $\mathfrak{M} \subseteq \wp(\mathbb{M})$:

$$[\![\mathsf{c}]\!]_{\mathcal{H}}\mathfrak{M} \triangleq (\!|\mathsf{c}|\!)_{\mathcal{H}}\mathfrak{M} \qquad [\![\mathsf{C}_1 \oplus \mathsf{C}_2]\!]_{\mathcal{H}}\mathfrak{M} \triangleq \{[\![\mathsf{C}_1]\!]\mathsf{M} \cup [\![\mathsf{C}_2]\!]\mathsf{M} \mid \mathsf{M} \in \mathfrak{M}\}$$
$$[\![\mathsf{C}^*]\!]_{\mathcal{H}}\mathfrak{M} \triangleq \left\{\textstyle\bigcup_{n\in\mathbb{N}}\{[\![\mathsf{C}]\!]^n\mathsf{M}\} \;\middle|\; \mathsf{M} \in \mathfrak{M}\right\} \quad [\![\mathsf{C}_1;\mathsf{C}_2]\!]_{\mathcal{H}}\mathfrak{M} \triangleq [\![\mathsf{C}_2]\!]_{\mathcal{H}}([\![\mathsf{C}_1]\!]_{\mathcal{H}}\mathfrak{M})$$

Note that, $[\![\mathsf{C}]\!]_{\mathcal{H}}$ is not exactly the lift to sets of $[\![\mathsf{C}]\!]$, but it is a correct approximation [30, 32], namely $\{[\![\mathsf{C}]\!]\mathsf{M}\} \subseteq [\![\mathsf{C}]\!]_{\mathcal{H}}\{\mathsf{M}\}$ for any set of memories $\mathsf{M} \subseteq \mathbb{M}$.

Verifying Completeness by Using Program Hypersemantics. Now that we have defined the hypersemantics, we will show how it can be used to verify completeness. To do so, we exploit the equivalence between completeness and ANI and the fact that the latter is a hyperproperty.

The definition of ANI presented in Sect. 4 is given in terms of a generic function f. Hence, ANI for programs in $\mathfrak{L}$ is defined as follows, where we basically instantiate Def. 3 with f being the concrete semantics $[\![\mathsf{C}]\!] : \wp(\mathbb{M}) \to \wp(\mathbb{M})$ of a program $\mathsf{C} \in \mathfrak{L}$. Let $\eta, \rho \in wAbs(\wp(\mathbb{M}))$, we say that the program C in $\mathfrak{L}$ satisfies (weak) ANI w.r.t. $\langle \eta, \rho \rangle$, written $[\eta]\mathsf{C}(\rho)$, when:

$$\forall \mathsf{M}_1, \mathsf{M}_2 \in \wp(\mathbb{M}) \,.\, \eta(\mathsf{M}_1) = \eta(\mathsf{M}_2) \;\Rightarrow\; \rho([\![\mathsf{C}]\!]\mathsf{M}_1) = \rho([\![\mathsf{C}]\!]\mathsf{M}_2)$$

Now consider the set $\mathfrak{M}_2^\eta \triangleq \{\{\mathsf{M}_1, \mathsf{M}_2\} \mid \eta(\mathsf{M}_1) = \eta(\mathsf{M}_2)\}$, consisting in the set of memories pairs indistinguishable by the input observation η, and the set $\mathfrak{M}_2^\rho \triangleq \{\{\mathsf{M}_1, \mathsf{M}_2\} \mid \rho(\mathsf{M}_1) = \rho(\mathsf{M}_2)\}$, consisting in the set of memories pairs

[8] Note that, an hypersemantics can be given in an abstract way on $\wp(C)$, in the same way we defined the concrete semantics on C in Section 2.

indistinguishable by the output observation ρ. We can use the hypersemantics $[\![\cdot]\!]_{\mathcal{H}}$ of $\mathfrak{L}$ to perform ANI verification:

$$[\![\mathtt{C}]\!]_{\mathcal{H}}\mathfrak{M}_2^{\eta} \subseteq \mathfrak{M}_2^{\rho} \;\Rightarrow\; [\eta]\mathtt{C}(\rho) \tag{2}$$

Indeed, Eq. 2 says that the program C executed from η-equivalent set of memories yields always ρ-equivalent set of memories, that is exactly the definition of ANI w.r.t. $\langle\eta,\rho\rangle$. Note that, this works since ANI is 2-bounded, hence *pairs* of executions are sufficient for verification [30, 32]. This, in turn, results into a verification method also for completeness of $[\![\mathtt{C}]\!]$ w.r.t. $\langle\eta,\rho\rangle$, due to its equivalence with ANI.

Theorem 6. *The abstract interpretation $\langle\rho,\eta\rangle$ is (weak) complete for* C *if:*

$$[\![\mathtt{C}]\!]_{\mathcal{H}}\mathfrak{M}_2^{\eta} \subseteq \mathfrak{M}_2^{\rho}$$

Proof. The proof is straightforward. Indeed, $[\![\mathtt{C}]\!]_{\mathcal{H}}\mathfrak{M}_2^{\eta} \subseteq \mathfrak{M}_2^{\rho}$ implies C satisfies ANI w.r.t. $\langle\eta,\rho\rangle$ (by Eq. 2) and, by Thm. 2, we have C is complete for $\langle\rho,\eta\rangle$.

Note that this is a simple implication since $[\![\mathtt{C}]\!]_{\mathcal{H}}$ is a sound approximation of $[\![\mathtt{C}]\!]$, but it is not the precise additive lift to sets of the concrete semantics [30]. Furthermore, the theorem is given in terms of pseudo-closures, hence we can trivially extend the result to upper closure operators and completeness, in the standard GC-based abstract interpretation framework.

Example 5. Consider the usual non-relational abstraction for memories, approximating a set of memories $\mathsf{M} \in \wp(\mathbb{M}) = \wp(V \to \mathbb{Z})$ by means of a non-relational memory $\mathtt{m}^{nr} \in V \to \wp(\mathbb{Z})$ and a value abstraction $\mathcal{I} \in wAbs(\wp(\mathbb{Z}))$ approximating set of integers to intervals. Consider the program $\mathtt{C} \triangleq x := |x|; ((x > 0)?; x := x - 2)^*$ where $|\mathtt{e}|$ is the absolute value operator applied to e that we suppose to add to $\mathfrak{L}$. Then C is not complete for the intervals domain $\mathcal{I}$. Intuitively, the approximation of the concrete semantics on input $\{-1, 1\}$ would yield the interval $\{-1\}$, while the (bca) abstract interpretation on the same input would yield the interval $\{-1, 0\}$. In this setting, $\mathfrak{M}_2^{\mathcal{I}}$ contains all pairs of non-relational memories $\{\mathtt{m}_1^{nr}, \mathtt{m}_2^{nr}\}$ such that $\mathcal{I}(\mathtt{m}_1^{nr}(x)) = \mathcal{I}(\mathtt{m}_2^{nr}(x))$ (for instance $\mathtt{m}_1^{nr}(x) = \{-1, 0, 1\}$ and $\mathtt{m}_2^{nr}(x) = \{-1, 1\}$, that are both approximated in the interval $\{-1, 0, 1\}$). If we execute the hypersemantics on $\mathfrak{M}_2^{\mathcal{I}}$ we will obtain, among others, the set $\{\mathtt{m}_1^{nr'}, \mathtt{m}_2^{nr'}\}$ such that $\mathtt{m}_1^{nr'}(x) = \{-1\}$ and $\mathtt{m}_2^{nr'}(x) = \{-1, 0\}$. But for such non-relational memories we have that $\mathcal{I}(\mathtt{m}_1^{nr'}(x)) = \{-1\} \neq \{-1, 0\} = \mathcal{I}(\mathtt{m}_2^{nr'}(x))$. This means that $[\![\mathtt{C}]\!]_{\mathcal{H}}\mathfrak{M}_2^{\mathcal{I}} \not\subseteq \mathfrak{M}_2^{\mathcal{I}}$, thus violating ANI and, in turn, proving that C is not complete for the interval abstraction.

6.2 Static Analysis for Completeness

As happens for the concrete semantics, also the hypersemantics is in general not computable. To effectively perform a static analysis we indeed need approximation: we need to abstract the computation of the concrete hypersemantics.

The abstract hypersemantics is a correct (and usually computable) approximation of the concrete hypersemantics. In particular, the *abstract hypersemantics* of programs in $\mathfrak{L}$ is a function $\llbracket\cdot\rrbracket^{A}_{\mathcal{H}} : \mathsf{RComm} \to (A \to A)$ on a hyper abstract domain $A_{\alpha,\gamma} \in \mathit{wAbs}(\wp(\wp(\mathbb{M})))$, defined inductively on the structure of regular commands as (here $\sqcup$ is the join of A):

$$\begin{array}{ll} \llbracket \mathtt{c} \rrbracket^{A}_{\mathcal{H}} a \triangleq (\alpha \circ \llbracket \mathtt{c} \rrbracket \circ \gamma) a & \llbracket \mathtt{C}_1 \oplus \mathtt{C}_2 \rrbracket^{A}_{\mathcal{H}} a \triangleq \llbracket \mathtt{C}_1 \rrbracket^{A}_{\mathcal{H}} a \sqcup \llbracket \mathtt{C}_2 \rrbracket^{A}_{\mathcal{H}} a \\ \llbracket \mathtt{C}^* \rrbracket^{A}_{\mathcal{H}} a \triangleq \bigsqcup \{ (\llbracket \mathtt{C} \rrbracket^{A}_{\mathcal{H}})^n a \mid n \in \mathbb{N} \} & \llbracket \mathtt{C}_1 ; \mathtt{C}_2 \rrbracket^{A}_{\mathcal{H}} a \triangleq \llbracket \mathtt{C}_2 \rrbracket^{A}_{\mathcal{H}} (\llbracket \mathtt{C}_1 \rrbracket^{A}_{\mathcal{H}} a) \end{array}$$

Note that, even if using the bca for the semantics of basic transfer functions in btFun is quite standard in abstract interpretation [5], effectively computable abstract hypersemantics (i.e. static analyses) may resort to an approximation of the bca.

The Abstract Interpretation for Completeness. To verify ANI and, hence, completeness we can instantiate the *hyperlevel constants* domain of [30] on $\wp(\mathbb{M})$. Note that, nothing changes in the definition of the hyperdomain when we consider a weak closure $\rho \in \mathit{wAbs}(\wp(\mathbb{M}))$, instead of an upper closure operator. In particular, following the construction of [30], we have that $\rho^\bullet = \wp(\mathit{Atom}(\rho)) \cup \{\rho\}$, where $\mathit{Atom}(\rho)$ denotes the sets of atoms of the fixpoints of ρ, namely the elements of ρ just above the bottom. Applying the lifting transformer of [30] we obtain $\mathcal{L}(\rho) \triangleq \lambda\mathfrak{M}\,.\,\{\rho(\mathsf{M}) \mid \mathsf{M} \in \mathfrak{M}\}$, for any set of sets of memories $\mathfrak{M} \subseteq \wp(\mathbb{M})$. Composing the two, we obtain $\alpha_\bullet \triangleq \lambda\mathfrak{M}\,.\,\rho^\bullet(\mathcal{L}(\rho)(\mathfrak{M}))$, that forms, together with its left adjoint $\gamma_\bullet = \alpha_\bullet^-$, the Galois connection [30]:

$$\langle \wp(\wp(\mathbb{M})), \subseteq \rangle \overset{\gamma_\bullet}{\underset{\alpha_\bullet}{\leftrightarrows}} \langle \rho^\bullet, \subseteq \rangle$$

By using such abstraction, we have that $\alpha_\bullet(\mathfrak{M}^{\rho}_{2}) = \mathit{Atom}(\rho)$, since the set contains only fix points of ρ, which can be seen as constants w.r.t. ρ. Hence, we can approximate the verification of ANI by using an abstract interpretation of the hypersemantics in the hyperdomain $\rho^\bullet$.

Lemma 3. *Let $\llbracket \mathtt{C} \rrbracket^{\rho^\bullet}_{\mathcal{H}}$ be the abstract interpretation of $\mathtt{C}$ in $\rho^\bullet$ and consider the case where $\rho \sqsubseteq \eta$. Then, we have that $\llbracket \mathtt{C} \rrbracket^{\rho^\bullet}_{\mathcal{H}}(\alpha_\bullet(\mathfrak{M}^{\eta}_{2})) \subseteq \mathit{Atom}(\rho) \;\Rightarrow\; [\eta]\mathtt{C}(\rho)$.*

Proof. Since $\llbracket \mathtt{C} \rrbracket^{\rho^\bullet}_{\mathcal{H}}$ is, by design, a correct approximation of $\llbracket \mathtt{C} \rrbracket_{\mathcal{H}}$, we have that $\llbracket \mathtt{C} \rrbracket_{\mathcal{H}} \mathfrak{M}^{\eta}_{2} \subseteq \llbracket \mathtt{C} \rrbracket^{\rho^\bullet}_{\mathcal{H}}(\alpha_\bullet(\mathfrak{M}^{\eta}_{2}))$. Then, if $\llbracket \mathtt{C} \rrbracket^{\rho^\bullet}_{\mathcal{H}}(\alpha_\bullet(\mathfrak{M}^{\eta}_{2})) \subseteq \mathit{Atom}(\rho)$ we obtain that $\llbracket \mathtt{C} \rrbracket_{\mathcal{H}} \mathfrak{M}^{\eta}_{2} \subseteq \mathit{Atom}(\rho)$. This implies that for each pair of (output) memory sets $\{\mathsf{M}_1, \mathsf{M}_2\} \in \llbracket \mathtt{C} \rrbracket_{\mathcal{H}} \mathfrak{M}^{\eta}_{2}$, namely sets of memories resulting from the computation of $\mathtt{C}$ on η-equivalent (input) memory sets, we have that $\rho^\bullet(\{\mathsf{M}_1, \mathsf{M}_2\}) = \{\mathsf{M}_1, \mathsf{M}_2\}$, since $\{\mathsf{M}_1, \mathsf{M}_2\} \in \mathit{Atom}(\rho)$. But this implies $\rho(\mathsf{M}_1) = \rho(\mathsf{M}_2)$, that is the requirement stated by Abstract Non-Interference.

Note that the requirement of having $\rho \sqsubseteq \eta$ is not too restrictive. Indeed, in static analysis an effectively implementable analyzer is a fixpoint computation that necessarily apply the same abstraction in input and output. Hence, only if we consider η more abstract than ρ we obtain a meaningful analysis. Again, Lem. 3 results into a static analysis method also for completeness, due to its equivalence with ANI.

Theorem 7. *Let* $[\![\mathtt{C}]\!]_{\mathcal{H}}^{\rho^\bullet}$ *be the abstract interpretation of* $\mathtt{C}$ *in* $\rho^\bullet$ *and* $\rho \sqsubseteq \eta$*. Then,* $\langle \rho, \eta \rangle$ *is (weak) complete for* $\mathtt{C}$ *if* $[\![\mathtt{C}]\!]_{\mathcal{H}}^{\rho^\bullet}(\alpha_\bullet(\mathfrak{M}_2^\eta)) \subseteq \mathit{Atom}(\rho)$*.*

Proof. The proof is straightforward. Indeed, $[\![\mathtt{C}]\!]_{\mathcal{H}}^{\rho^\bullet}(\alpha_\bullet(\mathfrak{M}_2^\eta)) \subseteq \mathit{Atom}(\rho)$ implies that $\mathtt{C}$ satisfies ANI w.r.t. $\langle \eta, \rho \rangle$ (by Lem. 3) and, therefore, by Thm. 2 we have that $\mathtt{C}$ is complete for $\langle \rho, \eta \rangle$.

Note that, the inclusion in Lem. 3 and, hence, in Thm. 7 is decidable when we bound the height of $\rho^\bullet$, as explained in [33].

7 Conclusions

Abstract domain completeness is a central issue in the theory of abstract interpretation, that has important practical implications (e.g., precision of abstract interpretation-based static analyses). In this paper, we investigated the connection between domain completeness and Abstract Non-Interference, tackling completeness from a different perspective. This allowed us to adapt the verification mechanisms developed for ANI in order to verify completeness, and to characterize new completeness domain transformers. All such completeness-related machinery has been developed in a weakened closure-based abstract interpretation framework, where we consider abstract domains that do not provide a best correct abstraction (i.e., that do not yield a Galois connection), such as the Convex Polyhedra and the Automata domains.

Such weak abstract interpretation framework has been already introduced in [12], but no formalization was provided in terms of closure-like functions. Indeed, having a general setting for comparing abstractions independently of their representation is quite useful to reason about domains properties, like completeness. In the standard abstract interpretation framework this role is played by upper closure operators but, to the best of our knowledge, there is not an analogous notion for Galois connection-less abstract interpretations. Hence, in the present paper, we introduced *weak-closures*, namely upper closure operators relaxing monotonicity, in order to precisely fill this gap.

A preliminary study on the link between ANI and completeness has been already provided in [20, 21], where ANI has been restated as a completeness problem for specific upper closure operators, in the context of language-based security. In the present paper, we followed the opposite direction, by restating completeness as an ANI problem, in the more general setting of weak-closures and not restricted to the specific context of language-based security. As a result, we proved an *equivalence* between completeness and ANI, that holds for abstract domains defined in terms of weak-closures (and, as a trivial consequence, for domains defined on upper closure operators).

As a consequence of the proved equivalence between domain completeness and ANI, in the present paper we highlighted how known domain transformers developed for ANI can be extended to cope with completeness. In particular, we provided transformers making weak-closures complete for a given function and we generalized the deductive system of [19, 22], in order to effectively compute the most concrete (weak-)closures that are complete for a given program. Finally, we exploited the static verification mechanisms, based on *hypersemantics* [30], developed for Non-Interference [32] in order to effectively verify ANI and, in turn, completeness (which is, indeed, a hyperproperty of program semantics).

It should be clear that, this paper can be considered only a first step in the direction of abstract interpretation-based static verification of completeness [14], since the connection we considered here is between ANI and the completeness of the best correct approximation (bca) of the semantics. This means that we can only imply completeness for an abstract semantics less precise than the bca. It will indeed deserve further investigation to design static analysis techniques, based on hyper analysis, verifying completeness directly on sound approximations of the bca. It will also worth extending the results to the verification of weaker forms of completeness (e.g., local [5] or partial [6] completeness).

Moreover, the design of static analysis techniques for ANI may become useful also in other fields of application. For instance, *robustness for neural networks* consists in checking whether by performing a perturbation of the input we can observe a variation in the network output classification, where the network can be abstractly interpreted [17]. In this context the idea of statically verifying ANI could be adapted to verify robustness w.r.t. an input perturbation. Clearly, this field of application needs and deserves further work.

A Selected Proofs

Proof (Proof of Lemma 1). We prove that the set $\eta_f^\wedge$ is a Moore family, namely that is closed under greatest lower bound. Let us consider $Y \subseteq \eta_f^\wedge$ and suppose $\exists x \in D\,.\, z \in \kappa_f^\eta(x)$ and such that $\bigwedge Y \geq z$, then $\forall y \in Y$ we have $Y \geq \bigwedge Y \geq z$. But, by definition of Y, this means that $\forall y \in Y$ we have $y \geq \vee\kappa_f^\eta$, hence by definition of glb we have $\bigwedge Y \geq \vee\kappa_f^\eta$, meaning that $\bigwedge Y \in \eta_f^\wedge$.
Suppose now that $\forall x \in D. \forall z \in \kappa_f^\eta(x)$ we have $\bigwedge Y \not\geq z$, hence the implication defining Nint_f^η is trivially true and again $\bigwedge Y \in \eta_f^\wedge$.

Proof (Proof of Theorem 3). First of all we have to show that:

$$\forall x_1, x_2 \in D.\, \eta(x_1) = \eta(x_2) \;\Rightarrow\; \eta_f^\wedge \circ f(x_1) = \eta_f^\wedge \circ f(x_2)$$

By construction, $\eta_f^\wedge \circ f(x_1) = \bigwedge\{y \mid y \geq f(x_1) \wedge \mathit{Nint}_f^\eta(y)\}$ and $\eta_f^\wedge \circ f(x_2) = \bigwedge\{y \mid y \geq f(x_2) \wedge \mathit{Nint}_f^\eta(y)\}$. By Lemma 1, we know that also the glb of Nint_f^η elements satisfies Nint_f^η, namely is in the set. Let us prove that if y is such that $\mathit{Nint}_f^\eta(y)$ then it is greater than any image of f. Suppose $y \in \{y \mid y \geq f(x_1) \wedge \mathit{Nint}_f^\eta(y)\}$, then $y \geq f(x_1)$, but $f(x_1) \in \kappa_f^\eta(x_2) = \{f(y) \mid \eta(y) = \eta(x_2)\}$ by hypothesis, but then by Nint_f^η hypothesis, $y \geq \bigvee \kappa_f^\eta(x_2) \geq f(x_2)$. Namely $y \in \{y \mid y \geq f(x_2) \wedge \mathit{Nint}_f^\eta(y)\}$. Since we do not have hypotheses on x_1 and x_2, this proves that the two sets are the same, and therefore $\eta_f^\wedge \circ f(x_1) = \eta_f^\wedge \circ f(x_2)$.

We now have to prove that it is the most concrete. This come trivially by construction, since $\eta_f^\wedge$ takes all the elements y such that $Nint_f^\eta(y)$, any more concrete domain ρ' must contain w such that $\neg Nint_f^\eta(w)$. But this means that $\exists x \in D\, \exists z \in \kappa_f^\eta(x)$ such that $w \geq z = f(y)$ (for some $y \in D$) but $w \ngeq \bigvee \kappa_f^\eta(x)$, meaning that there must exists $z' \in \kappa_f^\eta(x)$ such that $w \ngeq z' = f(y')$ (for some $y' \in D$). Hence we have $\eta(y) = \eta(y')$ and $\rho' \circ f(y) = z \leq w$ while $\rho' \circ f(y') = z' \nleq w$ meaning that $\rho' \circ f(y) \neq \rho' \circ f(y')$.

Proof (Proof of Lemma 2). Extensivity holds trivially by definition. Let us prove idempotence. Suppose $\rho_f^\vee(x) = y \in \mathsf{max}\{y \in D \mid \rho \circ f(y) = \rho \circ f(x)\}$. Let us compute $\rho_f^\vee(y) = w \in \mathsf{max}\{w \in D \mid \rho \circ f(w) = \rho \circ f(y)\}$. But the we trivially have that $\rho \circ f(x) = \rho \circ f(y) = \rho \circ f(w)$, hence $w \leq y$ being y maximal, and $y \leq w$ by extensivity of $\rho_f^\vee$, hence $\rho_f^\vee(x) = y = w = \rho_f^\vee(y) = \rho_f^\vee \circ \rho_f^\vee(x)$.

Proof (Proof of Theorem 4). We have to prove that:

$$\forall x_1, x_2.\, \rho_f^\vee(x_1) = \rho_f^\vee(x_2) \;\Rightarrow\; \rho \circ f(x_1) = \rho \circ f(x_2)$$

Suppose that $\rho_f^\vee(x_1) = \rho_f^\vee(x_2)$. Then, $\rho_f^\vee(x_1) = y_1 \in \mathsf{max}\{y \in D \mid \rho \circ f(y) = \rho \circ f(x_1)\}$ and $\rho_f^\vee(x_2) = y_2 \in \mathsf{max}\{y \in D \mid \rho \circ f(y) = \rho \circ f(x_2)\}$, with $y_1 = y_2$, hence $\rho \circ f(x_1) = \rho \circ f(y_1)$ and $\rho \circ f(y_2) = \rho \circ f(x_2)$.
We have now to prove that it is maximal w.r.t. the relative precision order, namely any more abstract abstraction does not satisfy the ANI property. Suppose there exists $\eta' \in wAbs(D)$ more abstract than $\rho_f^\vee$, then it means that there exists $x \in D$ such that $y \triangleq \rho_f^\vee(x) \lneq \eta'(x)$, namely $\eta'(x) \gneq y \in \mathsf{max}\{z \in D \mid \rho \circ f(z) = \rho \circ f(x)\}$. Hence $\eta' \circ \eta'(x) = \eta'(x)$, by idempotence, but $\rho \circ f(\eta'(x)) \neq \rho \circ f(x)$ being y maximal.

Proof (Proof of Theorem 5). Exploiting the correspondence between completeness and Abstract Non-Interference (Theorem 2), we just have to prove that ANI holds. Indeed, we have to prove that if $\vdash [\eta]\mathsf{C}(\rho)$ then $[\eta]\mathsf{C}(\rho)$ holds. Let us prove that all rules in Fig. 4 are sound, namely that the deduced abstraction ensure ANI for P.
Rule **R0**: $\forall x_1, x_2$, independently from the input observation η, we trivially have $\mathbb{T}[\![\mathsf{C}]\!](x_1) = \mathbb{T}[\![\mathsf{C}]\!](x_2)$. On the other hand, $id(x_1) = id(x_2)$ means that $x_1 = x_2$, and therefore trivially, for any ρ, $\rho[\![\mathsf{C}]\!](x_1) = \rho[\![\mathsf{C}]\!](x_2)$.
Rule **R1**: We consider here expressions as base case of the induction. By Corollary 2 we have that $\eta^\wedge_{\{\!|\mathsf{e}|\!\}}$ is such that $\forall x_1, x_2.\, \eta(x_1) = \eta(x_2) \;\Rightarrow\; \eta^\wedge_{\{\!|\mathsf{e}|\!\}}\{\!|\mathsf{e}|\!\}(x_1) = \eta^\wedge_{\{\!|\mathsf{e}|\!\}}\{\!|\mathsf{e}|\!\}(x_2)$. Analogous for the other rule by Corollary 3. Note that, in order to be precise we should have to write other two axioms for b; but they are almost the same by considering $\{\!|\mathsf{b}|\!\}$ when computing, respectively, the input and the output observations.
Rule **R2**: In this case we can observe that, $[\eta]\mathbf{skip}(\rho)$ holds iff $\forall x_1, x_2$ we have that $\eta(x_1) = \eta(x_2)$ implies $\rho[\![\mathbf{skip}]\!](x_1) = \rho(x_1) = \rho(x_2) = \rho[\![\mathbf{skip}]\!](x_2)$, and this trivially holds if $\eta \sqsubseteq \rho$.
Rule **R3**: In this case we need the precondition $[\eta]\mathsf{e}(\rho)$, which means that the expression semantics does not change the property, i.e., $\eta(x_1) = \eta(x_2) \;\Rightarrow$

$\rho(\{\!|\mathtt{e}|\!\}(x_1)) = \rho(\{\!|\mathtt{e}|\!\}(x_2))$. Hence, the assignment is complete if the expression is complete, but if there is more than one variable we need $\eta \sqsubseteq \rho$ for guaranteeing the implication (the assignment behaves like **skip**; on the other potential program viariables). Indeed, $[\![x := \mathtt{e}]\!](x_1) = x_1[x \mapsto \{\!|\mathtt{e}|\!\}(x_1)]$ and $[\![x := \mathtt{e}]\!](x_2) = x_2[x \mapsto \{\!|\mathtt{e}|\!\}(x_2)]$, provides results with the same ρ property since all the variables $y \neq x$, due to the hypotheses $\eta(x_1) = \eta(x_2)$ and $\eta \sqsubseteq \rho$, have values sharing the same ρ property, while for x returns the evaluations of the expression on the two different input memories. These evaluations share precisely the same ρ property by the rule precondition.
Rule **R4**: It is trivial since the semantics of the basic transfer function $\mathtt{b}?$ is precisely the semantics of the boolean expression $\mathtt{b}$.
Rule **R5**: In this case, the proof is obtained by using rule **R2**, **R6** and **R8**. Indeed, when we do not execute $\mathtt{C}$ ($n = 0$) we need in output to observe η (**R2**). When we execute $\mathtt{C}$ one or more times, by induction on $n \geq 1$, by hypotheses and by **R6**, we prove ANI with ρ in output, and therefore by **R8** we prove ANI observing $\eta \sqcup \rho = \eta$.
Rule **R6**: If $\forall x_1, x_2. \eta(x_1) = \eta(x_2) \Rightarrow \rho[\![\mathtt{C}_1]\!](x_1) = \rho[\![\mathtt{C}_1]\!](x_2)$ and $\forall y_1, y_2. \eta_1(y_1) = \eta_1(y_2) \Rightarrow \rho_1[\![\mathtt{C}_2]\!](y_1) = \rho_1[\![\mathtt{C}_2]\!](y_2)$, then we have that $\forall x_1, x_2.\ \eta(x_1) = \eta(x_2) \Rightarrow \rho_1[\![\mathtt{C}_2]\!]([\![\mathtt{C}_1]\!](x_1)) = \rho_1[\![\mathtt{C}_2]\!]([\![\mathtt{C}_1]\!](x_2))$. At this point, since $\rho[\![\mathtt{C}_1]\!](x_1) = \rho[\![\mathtt{C}_1]\!](x_2)$ implies $\eta_1[\![\mathtt{C}_1]\!](x_1) = \eta_1[\![\mathtt{C}_1]\!](x_2)$, then we have the thesis.
Rule **R7**: If $\forall x_1, x_2$ we have $\eta_1(x_1) = \eta_1(x_2) \Rightarrow \rho_1[\![\mathtt{C}_1]\!](x_1) = \rho_1[\![\mathtt{C}_1]\!](x_2)$ and $\forall y_1, y_2$ we have $\eta_2(y_1) = \eta_2(y_2) \Rightarrow \rho_2[\![\mathtt{C}_2]\!](y_1) = \rho_2[\![\mathtt{C}_2]\!](y_2)$, then $\forall x_1, x_2$ we have that . $(\eta_1 \sqcap \eta_2)(x_1) = (\eta_1 \sqcap \eta_2)(x_2)$ implies both the equalities $\eta_1(x_1) = \eta_1(x_2)$ and $\eta_2(x_1) = \eta_2(x_2)$, hence we have both $\rho_1[\![\mathtt{C}_1]\!](x_1) = \rho_1[\![\mathtt{C}_1]\!](x_2)$ and $\rho_2[\![\mathtt{C}_2]\!](x_1) = \rho_2[\![\mathtt{C}_2]\!](x_2)$. This implies that, being $[\![\mathtt{C}_1 \oplus \mathtt{C}_2]\!] = [\![\mathtt{C}_1]\!] \cup [\![\mathtt{C}_2]\!]$, $(\rho_1 \sqcup \rho_2)[\![\mathtt{C}_1 \oplus \mathtt{C}_2]\!](x_1) = (\rho_1 \sqcup \rho_2)[\![\mathtt{C}_1 \oplus \mathtt{C}_2]\!](x_2)$.
Rule **R8**: Trivial. Indeed, η implies η_1 and ρ_1 implies ρ.
Rule **R9**: By definition of $\sqcup$ of partitioning closures [28], we have that $\eta_1 \sqcup \eta_2(x_1) = \eta_1 \sqcup \eta_2(x_2)$ implies that either $\eta_1(x_1) = \eta_1(x_2)$ or $\eta_2(x_1) = \eta_2(x_2)$. then by hypothesis, in both cases we have that $\rho[\![\mathtt{C}]\!](x_1) = \rho[\![\mathtt{C}]\!](x_2)$, namely we have the thesis. We can trivially extend the proof to any set I.
Rule **R10**: Trivial by rule **R7**.
Rule **R11**: By definition of $\sqcap$ we have that $\sqcap_i \rho_i[\![\mathtt{C}]\!](x_1) = \bigwedge_i \rho_i[\![\mathtt{C}]\!](x_1)$. By hypothesis if $\eta(x_)) = \eta(x_2)$ then for each $i \in I$ we have $\rho_i[\![\mathtt{C}]\!](x_1) = \rho_i[\![\mathtt{C}]\!](x_2)$, but then $\bigwedge_i \rho_i[\![\mathtt{C}]\!](x_1) = \bigwedge_i \rho_i[\![\mathtt{C}]\!](x_2) = \sqcap_i \rho_i[\![\mathtt{C}]\!](x_2)$, namely we have the thesis.

References

1. Albarghouthi, A.: Introduction to neural network verification (2021). https://doi.org/10.48550/ARXIV.2109.10317, https://arxiv.org/abs/2109.10317
2. Arceri, V., Mastroeni, I.: Analyzing dynamic code: A sound abstract interpreter for evil eval. ACM Trans. Priv. Secur. **24**(2), 10:1–10:38 (2021)
3. Arceri, V., Mastroeni, I., Xu, S.: Static analysis for ecmascript string manipulation programs. Appl. Sci. **10**, 3525 (2020). https://doi.org/10.3390/app10103525
4. Assaf, M., Naumann, D.A., Signoles, J., Totel, E., Tronel, F.: Hypercollecting semantics and its application to static analysis of information flow. In: Proc. of POPL. pp. 874–887 (2017)

5. Bruni, R., Giacobazzi, R., Gori, R., Ranzato, F.: A logic for locally complete abstract interpretations. In: Symposium on Logic in Computer Science, LICS. pp. 1–13. IEEE (2021)
6. Campion, M., Dalla Preda, M., Giacobazzi, R.: Partial (in)completeness in abstract interpretation: limiting the imprecision in program analysis. Proc. ACM Program. Lang. **6**(POPL), 1–31 (2022). https://doi.org/10.1145/3498721, https://doi.org/10.1145/3498721
7. Clarkson, M.R., Schneider, F.B.: Hyperproperties. Journal of Computer Security **18**(6), 1157–1210 (2010)
8. Cohen, E.S.: Information transmission in sequential programs. In: et al., D. (ed.) Foundations of Secure Computation. pp. 297–335. Academic Press, New York (1978)
9. Cousot, P.: Constructive design of a hierarchy of semantics of a transition system by abstract interpretation. Theor. Comput. Sci. **277**(1-2), 47–103 (2002)
10. Cousot, P., Cousot, R.: Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In: Conference Record of the 4th ACM Symposium on Principles of Programming Languages (*POPL '77*). pp. 238–252. ACM Press (1977)
11. Cousot, P., Cousot, R.: Systematic design of program analysis frameworks. In: Conference Record of the 6th ACM Symposium on Principles of Programming Languages (*POPL '79*). pp. 269–282. ACM Press (1979)
12. Cousot, P., Cousot, R.: Abstract interpretation frameworks. J. Logic and Comput. **2**(4), 511–547 (1992)
13. Cousot, P., Halbwachs, N.: Automatic discovery of linear restraints among variables of a program. In: POPL '78: Proceedings of the 5th ACM SIGACT-SIGPLAN symposium on Principles of programming languages. pp. 84–96. ACM Press (1978). https://doi.org/http://doi.acm.org/10.1145/512760.512770
14. Cousot, P., Giacobazzi, R., Ranzato, F.: A^{2}i: abstract^2 interpretation. Proc. ACM Program. Lang. **3**(POPL), 42:1–42:31 (2019)
15. Dijkstra, E.W.: The humble programmer. Commun. ACM **15**(10), 859–866 (oct 1972). https://doi.org/10.1145/355604.361591, https://doi.org/10.1145/355604.361591
16. Filé, G., Giacobazzi, R., Ranzato, F.: A unifying view of abstract domain design. ACM Comput. Surv. **28**(2), 333–336 (1996)
17. Gehr, T., Mirman, M., Drachsler-Cohen, D., Tsankov, P., Chaudhuri, S., Vechev, M.: Ai2: Safety and robustness certification of neural networks with abstract interpretation. In: 2018 IEEE Symposium on Security and Privacy (SP). pp. 3–18 (2018). https://doi.org/10.1109/SP.2018.00058
18. Giacobazzi, R., Mastroeni, I.: Abstract non-interference: Parameterizing non-interference by abstract interpretation. In: Proc. of the 31st Annual ACM SIGPLAN-SIGACT Symp. on Principles of Programming Languages (POPL '04). pp. 186–197. ACM-Press (2004)
19. Giacobazzi, R., Mastroeni, I.: Proving abstract non-interference. In: J. Marcinkowski, A.T. (ed.) Annual Conf. of the European Association for Computer Science Logic (CSL '04). vol. 3210, pp. 280–294. Springer-Verlag (2004)
20. Giacobazzi, R., Mastroeni, I.: Adjoining declassification and attack models by abstract interpretation. In: Sagiv, S. (ed.) Proc. of the European Symp. on Programming (ESOP '05). Lecture Notes in Computer Science, vol. 3444, pp. 295–310. Springer-Verlag (2005)
21. Giacobazzi, R., Mastroeni, I.: Adjoining classified and unclassified information by abstract interpretation. Journal of Computer Security **18**(5), 751 – 797 (2010)

22. Giacobazzi, R., Mastroeni, I.: A proof system for abstract non-interference. Journal of Logic and Computation **20**, 449 – 479 (2010)
23. Giacobazzi, R., Mastroeni, I.: Abstract non-interference: A unifying framework for weakening information-flow. ACM Trans. Priv. Secur. **21**(2), 1–31 (2018)
24. Giacobazzi, R., Ranzato, F.: Refining and compressing abstract domains. In: Degano, P., Gorrieri, R., Marchetti-Spaccamela, A. (eds.) Proc. of the 24th Internat. Colloq. on Automata, Languages and Programming (*ICALP '97*). Lecture Notes in Computer Science, vol. 1256, pp. 771–781. Springer-Verlag (1997)
25. Giacobazzi, R., Ranzato, F., Scozzari., F.: Making abstract interpretation complete. Journal of the ACM **47**(2), 361–416 (March 2000)
26. Girard, A.: Reachability of uncertain linear systems using zonotopes. In: Proceedings of the 8th International Conference on Hybrid Systems: Computation and Control. p. 291–305. HSCC'05, Springer-Verlag, Berlin, Heidelberg (2005)
27. Goguen, J.A., Meseguer, J.: Security policies and security models. In: Proc. IEEE Symp. on Security and Privacy. pp. 11–20. IEEE Comp. Soc. Press (1982)
28. Hunt, S., Mastroeni, I.: The PER model of abstract non-interference. In: Hankin, C., Siveroni, I. (eds.) Proc. of The 12th Internat. Static Analysis Symp. (SAS '05). Lecture Notes in Computer Science, vol. 3672, pp. 171–185. Springer-Verlag (2005)
29. Mastroeni, I., Pasqua, M.: Hyperhierarchy of semantics - A formal framework for hyperproperties verification. In: Proc. of SAS. pp. 232–252 (2017)
30. Mastroeni, I., Pasqua, M.: Verifying bounded subset-closed hyperproperties. In: Proc. of SAS. pp. 1–20 (2018)
31. Mastroeni, I.: Abstract interpretation-based approaches to security - A survey on abstract non-interference and its challenging applications. In: Banerjee, A., Danvy, O., Doh, K., Hatcliff, J. (eds.) Semantics, Abstract Interpretation, and Reasoning about Programs: Essays Dedicated to David A. Schmidt on the Occasion of his Sixtieth Birthday, Manhattan, Kansas, USA, 19-20th September 2013. EPTCS, vol. 129, pp. 41–65 (2013)
32. Mastroeni, I., Pasqua, M.: Statically analyzing information flows: An abstract interpretation-based hyperanalysis for non-interference. In: Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing. pp. 2215–2223. Association for Computing Machinery (2019). https://doi.org/10.1145/3297280.3297498
33. Mastroeni, I., Pasqua, M.: Verifying opacity by abstract interpretation. In: Hong, J., Bures, M., Park, J.W., Cerný, T. (eds.) SAC '22: The 37th ACM/SIGAPP Symposium on Applied Computing, Virtual Event, April 25 - 29, 2022. pp. 1817–1826. ACM (2022). https://doi.org/10.1145/3477314.3507119, https://doi.org/10.1145/3477314.3507119
34. O'Hearn, P.W.: Incorrectness logic. Proceedings of the ACM on Programming Languages (POPL) **4**(10) (2020)
35. Ranzato, F., Tapparo, F.: Strong preservation as completeness in abstract interpretation. In: Schmidt, D. (ed.) Proc. of the 13th European Symp. on Programming (ESOP '04). Lecture Notes in Computer Science, vol. 2986, pp. 18–32. Springer-Verlag (2004)
36. Sabelfeld, A., Myers, A.: Language-based information-flow security. IEEE J. on selected ares in communications **21**(1), 5–19 (2003)
37. Winskel, G.: The formal semantics of programming languages: an introduction. MIT press (1993)