Generator-Dense Galois Connections

Marco Campion[1,⋆], Mila Dalla Preda[2], Roberto Giacobazzi[3], Antoine Miné[1], and Caterina Urban[4]

[1] Sorbonne Université, LIP6, Paris, France
{marco.campion, antoine.mine}@lip6.fr
[2] University of Verona, Department of Computer Science, Italy
mila.dallapreda@univr.it
[3] University of Arizona, Department of Computer Science, Tucson, AZ, USA
giacobazzi@arizona.edu
[4] Inria & ENS | PSL, Paris, France
caterina.urban@inria.fr

Abstract. The (im)precision of an abstract interpretation is typically studied from the *abstract* side of a Galois connection (GC): either through the abstract domain or the abstract semantics itself. This paper instead studies a structural property of the *concrete* domain: how the *generators*—the minimal concrete witnesses of each abstract property through the abstraction α—are distributed inside the preimages of α, and how this distribution governs the (im)precision, i.e., the (partial) completeness, of an abstract interpretation. We introduce the class of *Generator-Dense Galois Connections* (GDGC), capturing the case in which every concrete property lies above some generator of its abstraction, so that the generators form a dense "basis" for the entire concrete domain. We study GDGCs from two angles. On the structural side, generator-density is preserved by products and pointwise liftings but not by composition in general; we identify two sufficient conditions recovering closure under composition. On the (im)precision side, generator-density enables *reducing* the proof of an abstract interpreter's (partial) completeness from the entire concrete domain to the generators alone. In other words, the abstract interpreter's (im)precision is *fully characterized* by its behavior on generators, and any imprecision the analyzer incurs at a concrete input can be *localized* at a specific generator below it, providing a simple target on which to focus precision-improving refinements.

Keywords: Abstract Interpretation · Galois Connection · Generator · Generator-Density · Partial Completeness.

1 Introduction

In the theory of abstract interpretation [19,20,18], Galois connections lie at the heart of program analysis design: a typically very large or even infinite concrete domain $\mathcal{C}$ of (possibly undecidable) program properties is paired with a simpler

⋆ Corresponding author.

abstract domain of (possibly decidable) program properties $\mathcal{A}$ via an abstraction map $\alpha : \mathcal{C} \to \mathcal{A}$ and a concretization $\gamma : \mathcal{A} \to \mathcal{C}$. A concrete semantic function $[\![\mathtt{P}]\!] : \mathcal{C} \to \mathcal{C}$—computing the behavior of a program $\mathtt{P}$ on concrete input properties, and typically computationally intractable—is then approximated by a sound abstract counterpart $[\![\mathtt{P}]\!]^\sharp : \mathcal{A} \to \mathcal{A}$ that operates on the simpler abstract domain. The abstract result is always a sound over-approximation of the concrete program behavior, formally $[\![\mathtt{P}]\!] \circ \gamma \sqsubseteq_{\mathcal{C}} \gamma \circ [\![\mathtt{P}]\!]^\sharp$, where $\sqsubseteq_{\mathcal{C}}$ is the partial order of the concrete domain. The fundamental question is then: *how close is the abstract result* $[\![\mathtt{P}]\!]^\sharp(\alpha(c))$ *to* $\alpha([\![\mathtt{P}]\!](c))$*, the abstraction of the concrete computation, on every input* $c \in \mathcal{C}$*?* The strongest possible answer is *completeness*, requiring exact equality $\alpha \circ [\![\mathtt{P}]\!] = [\![\mathtt{P}]\!]^\sharp \circ \alpha$. Since completeness is hard to achieve in practice [27], recent work has weakened it in two ways: *local* completeness [7] requires equality only on a subset of inputs $S \subseteq \mathcal{C}$, and *partial* completeness [14,8] admits a bounded amount of imprecision measured by an order-compatible distance on the abstract domain. A common approach to study precision in either form is to focus on the *abstract* side of the GC, for instance by characterizing program properties that determine when an abstraction is precise [27,5,10], or by refining the abstract domain or its semantics [22,21,41,18].

In this paper, we instead approach the study of the (im)precision of an abstract interpretation from the *concrete* side of the Galois connection. We identify a structural property of the concrete domain that, when it holds, lets the *(im)precision be characterized via a set of canonical concrete representatives.*

A first step in this direction was taken in [11] with the notion of *generators* of an abstract property $a \in \mathcal{A}$: a $\sqsubseteq_{\mathcal{C}}$-minimal concrete witness of a through the abstraction α, carrying just enough concrete information to produce a and nothing more. For instance, in the GC $\langle \wp(\mathbb{Z}), \subseteq \rangle \underset{\alpha_{\mathsf{Int}}}{\overset{\gamma_{\mathsf{Int}}}{\leftrightarrows}} \langle \mathsf{Int}, \sqsubseteq_{\mathsf{Int}} \rangle$ of the standard interval abstract domain Int [19], the generator of an interval $[c, d]$ (with $c, d \in \mathbb{Z}$, $c \leq d$) is the two-element set $\{c, d\}$, the $\subseteq$-smallest integer set whose abstraction is $[c, d]$. As observed in [11], studying the imprecision of an abstract interpretation at generators automatically yields worst-case imprecision bounds at some specific concrete inputs—in the interval example, at every S with $\{c, d\} \subseteq S \subseteq \gamma_{\mathsf{Int}}([c, d])$.

Whether this argument from generators scales to *all* concrete elements in $\mathcal{C}$ depends on a structural property of the *concrete* domain of the GC: how the generators are *distributed* within the fiber (i.e., the preimage) $\alpha^{-1}(a)$ of any $a \in \mathcal{A}$. In some GCs, there could exist concrete elements that lie above (in terms of $\sqsubseteq_{\mathcal{C}}$) no generator, which therefore escape any generator-based argument. This is the case of $\alpha^{-1}([c, +\infty])$ for unbounded intervals since $[c, +\infty]$ does not admit any generator. In other GCs, every concrete element of $\alpha^{-1}(a)$ sits above some generator of a, and thus reasoning at generators automatically lifts to global guarantees on the full concrete domain. We call this latter property *generator-density*, and the GCs satisfying it, *Generator-Dense Galois Connections* (GDGC). GDGCs are the central object of study of this paper. The next paragraph illustrates the utility of generator-density on two simple programs.

A illustrative example. Fix a bounded integer range $J = [\mathrm{MIN}, \mathrm{MAX}] \cap \mathbb{Z}$ and consider the bounded-interval GC $\langle \wp(J), \subseteq \rangle \underset{\alpha_{\mathsf{Int}}}{\overset{\gamma_{\mathsf{Int}}}{\leftrightarrows}} \langle \mathsf{Int}_J, \sqsubseteq_{\mathsf{Int}} \rangle$, where $\alpha_{\mathsf{Int}}(S)$ is the smallest interval $[min(S), max(S)]$ containing a non-empty $S \subseteq J$. This GC is generator-dense: every interval $[a, b] \in \mathsf{Int}_J$ has the pair $\{a, b\}$ as its (unique) generator, and every $S \in \alpha^{-1}_{\mathsf{Int}}([a, b])$ contains $\{a, b\}$ by definition of α_{Int}. Now take the absolute-value program

$$\texttt{ABS} \ : \ \texttt{if} \ \ x \geq 0 \ \ \texttt{then} \ \ x := x \ \ \texttt{else} \ \ x := -x$$

with its standard collecting semantics $[\![\texttt{ABS}]\!] \colon \wp(J) \to \wp(J)$ and the standard abstract interval semantics $[\![\texttt{ABS}]\!]^\sharp_{\mathsf{Int}} \colon \mathsf{Int}_J \to \mathsf{Int}_J$, which on an input $[a, b]$ with $a \leq 0 \leq b$ returns $[\![\texttt{ABS}]\!]^\sharp_{\mathsf{Int}}([a, b]) = [0, max(|a|, b)]$. Consider, for $n > 0$, the sign-symmetric interval $[-n, n]$, whose generator is $\{-n, n\}$. At the generator, the concrete computation yields $[\![\texttt{ABS}]\!](\{-n, n\}) = \{n\}$, which abstracts to the singleton interval $[n, n]$; the abstract semantics instead returns $[\![\texttt{ABS}]\!]^\sharp_{\mathsf{Int}}([-n, n]) = [0, n]$, an over-approximation containing n spurious integer values $0, 1, \ldots, n-1$ not produced by the concrete computation. The imprecision of $[\![\texttt{ABS}]\!]^\sharp_{\mathsf{Int}}$ at the generator $\{-n, n\}$ is therefore exactly n, as measured by the *counting distance* formally introduced in Sec. 6 (Ex. 15). This bound n propagates to the entire fiber $\alpha^{-1}_{\mathsf{Int}}([-n, n])$: every $S \in \alpha^{-1}_{\mathsf{Int}}([-n, n])$ contains the generator $\{-n, n\}$ as a subset (its minimum and maximum are forced to be $-n$ and n), and the bound at $\{-n, n\}$ therefore lifts to every such S. Take the specific instance $n = 4$: the preimage $\alpha^{-1}_{\mathsf{Int}}([-4, 4])$ contains exponentially many integer sets—$\{-4, 4\}$, $\{-4, 0, 4\}$, $\{-4, -3, \ldots, 3, 4\}$, and so on—yet on every one of them the imprecision of $[\![\texttt{ABS}]\!]^\sharp_{\mathsf{Int}}$ with respect to $[\![\texttt{ABS}]\!]$ is bounded by 4, the value computed at the generator. A single computation at a structurally simple, two-value generator therefore certifies an imprecision bound across an exponential family of concrete inputs. This is the GDGC reduction at work: bounding the imprecision of $[\![\texttt{ABS}]\!]^\sharp_{\mathsf{Int}}$ at every generator $\{a, b\} \in \mathsf{Int}_J$ yields a worst-case imprecision bound on every concrete input in $\wp(J)$, and reasoning on the parametric family of two-element generators thus replaces reasoning on the full powerset domain. Note that this principle fails on the original GC $\langle \wp(\mathbb{Z}), \subseteq \rangle \underset{\alpha_{\mathsf{Int}}}{\overset{\gamma_{\mathsf{Int}}}{\leftrightarrows}} \langle \mathsf{Int}, \sqsubseteq_{\mathsf{Int}} \rangle$, which is not generator-dense: for instance, the fiber $\alpha^{-1}_{\mathsf{Int}}([0, +\infty])$ admits no minimal element, hence no generator, so no set in it contains a generator.

The same reduction principle handles completeness itself. Take the program

$$\texttt{ReLU} \ : \ \texttt{if} \ \ x < 0 \ \ \texttt{then} \ \ x := 0 \ \ \texttt{else} \ \ x := x$$

that maps negative values to zero, and the abstract interval semantics which on an interval $[a, b]$ returns $[\![\texttt{ReLU}]\!]^\sharp_{\mathsf{Int}}([a, b]) = [max(0, a), max(0, b)]$. At every generator $\{a, b\} \in \wp(J)$, the concrete and abstract semantics agree by the equality: $\alpha_{\mathsf{Int}}([\![\texttt{ReLU}]\!](\{a, b\})) = [\![\texttt{ReLU}]\!]^\sharp_{\mathsf{Int}}([a, b])$. Hence, $[\![\texttt{ReLU}]\!]^\sharp_{\mathsf{Int}}$ is locally complete at every generator and, by generator-density, *complete on the entire concrete domain* $\wp(J)$: a check at the parametric family of two-element generators certifies that the abstract semantics is precise on every input set.

Contributions. The two examples above instantiate a principle that this paper develops in full generality: *when a GC is generator-dense, the (partial) completeness of an abstract interpreter is fully governed by its behavior on generators.* Moreover, any imprecision the analyzer incurs at a concrete input can be *localized* at a specific generator below it, providing a simple target on which to focus precision-improving refinements.

In summary, this paper makes the following contributions:

- We introduce the class of *Generator-Dense Galois Connections* and give equivalent order-theoretic characterizations together with examples and counterexamples drawn from the abstract interpretation literature (Sec. 4).
- We characterize the closure properties of generator-density under the standard combinations of GCs: it is preserved by product and pointwise lifting, but *not* by composition. We identify two sufficient conditions under which generator-density is preserved by composition (Sec. 5).
- We prove our main result (Thm. 6): under generator-density, $\boldsymbol{e}$-partial local completeness of an abstract interpreter at every generator lifts to $\hat{\boldsymbol{e}}$-partial completeness on the entire concrete domain, with the imprecision function $\hat{\boldsymbol{e}}$ propagated automatically from per-generator bounds. Cor. 2 specializes this to completeness. We finally illustrate the resulting methodology on four GDGCs from the literature: bounded intervals, trace-to-reachability, allocation-site abstractions, and a dependency abstraction (Sec. 6).

Sections 2 and 3 introduce the background needed for the rest of the paper. All proofs can be found in the extended version available on HAL [9].

2 Background

We recall the basic notions of order theory, Galois connections, and abstract interpretation. For a comprehensive treatment, we refer to [18].

Order theory. Given two sets S and T, we denote by $\wp(S)$ the powerset of S, by $\varnothing$ the empty set, by $S \smallsetminus T$ the set-difference, and by $|S|$ the cardinality of S. Set inclusion and strict inclusion are denoted by $S \subseteq T$ and $S \subset T$, respectively. We write $\mathbb{N}$, $\mathbb{Z}$, and $\mathbb{R}$ for the sets of natural, integer, and real numbers. We use a subscript on them to limit their range, such as $\mathbb{R}_{\geq 0}$ for the non-negative real numbers, and the superscript ∞ for adding the infinite symbol, e.g., $\mathbb{R}^{\infty}_{\geq 0} \stackrel{\text{def}}{=} \{i \mid i \in \mathbb{R} \wedge i \geq 0\} \cup \{\infty\}$ such that $i < \infty$ for all $i \in \mathbb{R}$.

A *partially ordered set*, or *poset*, $\langle \mathcal{L}, \sqsubseteq_{\mathcal{L}} \rangle$ consists of a set $\mathcal{L}$ endowed with a partial order relation $\sqsubseteq_{\mathcal{L}} \subseteq \mathcal{L} \times \mathcal{L}$, namely a reflexive, antisymmetric, and transitive relation. We will highlight posets by using a calligraphic font, e.g., $\mathcal{L}$, to distinguish them from plain sets. The *strict version* of $\sqsubseteq_{\mathcal{L}}$ is denoted by $\sqsubset_{\mathcal{L}}$ and is defined by $x \sqsubset_{\mathcal{L}} y \stackrel{\text{def}}{\Leftrightarrow} x \sqsubseteq_{\mathcal{L}} y \wedge x \neq y$, for all $x, y \in \mathcal{L}$. The *reversed* (or *dual*) order of $\sqsubseteq_{\mathcal{L}}$ is denoted by $\sqsupseteq_{\mathcal{L}}$, and its strict version by $\sqsupset_{\mathcal{L}}$. $\langle \mathcal{L}, \sqsubseteq_{\mathcal{L}} \rangle$ satisfies the *descending chain condition* (DCC for short) when every descending chain $l_0 \sqsupseteq_{\mathcal{L}} l_1 \sqsupseteq_{\mathcal{L}} l_2 \sqsupseteq_{\mathcal{L}} \cdots$ of elements of $\mathcal{L}$ eventually stabilizes, that is, there

exists $n \in \mathbb{N}$ such that $l_n = l_{n+k}$ for all $k \in \mathbb{N}$. We may omit the subscript $\mathcal{L}$ from $\sqsubseteq_{\mathcal{L}}$ whenever the underlying poset is clear from context.

A function $f\colon \mathcal{L}_1 \to \mathcal{L}_2$ between two posets is *order-preserving* (or *monotone*) when $\forall x, y \in \mathcal{L}_1.\, x \sqsubseteq_{\mathcal{L}_1} y \Rightarrow f(x) \sqsubseteq_{\mathcal{L}_2} f(y)$. A function is *injective* when $\forall x, y \in \mathcal{L}_1.\, f(x) = f(y) \Rightarrow x = y$, and *surjective* when $\forall y \in \mathcal{L}_2.\, \exists x \in \mathcal{L}_1.\, f(x) = y$. The composition of two functions $f_1\colon \mathcal{L}_1 \to \mathcal{L}_2$ and $f_2\colon \mathcal{L}_2 \to \mathcal{L}_3$ is denoted by $f_2 \circ f_1\colon \mathcal{L}_1 \to \mathcal{L}_3$. The *identity* function over $\mathcal{L}$ is $id \stackrel{\text{def}}{=} \lambda x.\, x$.

Given a function $f\colon \mathcal{L}_1 \to \mathcal{L}_2$ and an element $y \in \mathcal{L}_2$, the *fiber* of f at y is the subset of $\mathcal{L}_1$ that f maps to y: $f^{-1}(y) \stackrel{\text{def}}{=} \{x \in \mathcal{L}_1 \mid f(x) = y\}$. The family $\{f^{-1}(y)\}_{y \in \mathcal{L}_2}$ forms a partition of the domain of f (up to empty fibers): every $x \in \mathcal{L}_1$ belongs to exactly one fiber, namely $f^{-1}(f(x))$. A subset $S \subseteq \mathcal{L}$ of a poset admits a (possibly empty) set of *minimal elements*, defined as:

$$Min(S) \stackrel{\text{def}}{=} \{s \in S \mid \forall s' \in S.\, s' \sqsubseteq_{\mathcal{L}} s \Rightarrow s' = s\}\,.$$

Galois connections. Galois connections are the basic algebraic structure of the abstract interpretation framework [20,21], and formalize the correspondence between a *concrete domain* $\mathcal{C}$ of exact properties and an *abstract domain* $\mathcal{A}$ of approximate (or computable) properties.

Definition 1 (Galois connection). *Let $\langle \mathcal{C}, \sqsubseteq_{\mathcal{C}} \rangle$ and $\langle \mathcal{A}, \sqsubseteq_{\mathcal{A}} \rangle$ be two posets. A pair of order-preserving functions $\alpha\colon \mathcal{C} \to \mathcal{A}$ (the* abstraction *or* lower adjoint*) and $\gamma\colon \mathcal{A} \to \mathcal{C}$ (the* concretization *or* upper adjoint*) forms a* Galois connection *(GC), denoted $\langle \mathcal{C}, \sqsubseteq_{\mathcal{C}} \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \mathcal{A}, \sqsubseteq_{\mathcal{A}} \rangle$, when the following adjunction holds:*

$$\forall c \in \mathcal{C}.\, \forall a \in \mathcal{A}.\, \alpha(c) \sqsubseteq_{\mathcal{A}} a \;\Leftrightarrow\; c \sqsubseteq_{\mathcal{C}} \gamma(a)\,.$$

We say that $a \in \mathcal{A}$ is an *(over-)approximation* of $c \in \mathcal{C}$ when $c \sqsubseteq_{\mathcal{C}} \gamma(a)$, or equivalently $\alpha(c) \sqsubseteq_{\mathcal{A}} a$, and the best such a is exactly $\alpha(c)$ itself. Intuitively, $\gamma(a)$ represents the *weakest* concrete property, with respect to $\sqsubseteq_{\mathcal{C}}$, which is over-approximated by a, whereas $\alpha(c)$ is the *strongest* (most precise) abstract property in $\mathcal{A}$, with respect to $\sqsubseteq_{\mathcal{A}}$, that still over-approximates c. A GC is a *Galois insertion* (GI) when γ is injective (or, equivalently, α is surjective). Intuitively, a GI contains no *spurious* abstract elements: every $a \in \mathcal{A}$ is the abstraction of some $c \in \mathcal{C}$. Every GC can be turned into a GI by removing the spurious elements of $\mathcal{A}$, without losing any information.

Abstract interpretation. Abstract interpretation [20,21,18] is a theory of sound-by-construction approximation of program semantics, obtained by replacing a concrete, possibly uncomputable, semantic function with a computable abstract counterpart that operates on a simpler domain of properties. Formally, fix a GC $\langle \mathcal{C}, \sqsubseteq_{\mathcal{C}} \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \mathcal{A}, \sqsubseteq_{\mathcal{A}} \rangle$, a concrete order-preserving function $f\colon \mathcal{C} \to \mathcal{C}$, and an abstract function $f^\sharp\colon \mathcal{A} \to \mathcal{A}$. Note that $f^\sharp$ is *not* required to be order-preserving. The different degrees of precision of $f^\sharp$ with respect to f are captured by the following two definitions.

Definition 2 (Soundness). *The abstract function $f^\sharp$ is a* sound *(or* correct*) (over-)approximation of f when the following inequality holds: $f \circ \gamma \sqsubseteq_{\mathcal{C}} \gamma \circ f^\sharp$.*

Soundness is the minimal requirement of any abstract interpretation: it ensures that no concrete behavior of f is overlooked by its abstract counterpart $f^\sharp$.

Definition 3 ((Local) Completeness). *We say that $f^\sharp$ is a* complete *approximation of f when the following equality holds: $\alpha \circ f = f^\sharp \circ \alpha$.*

It is a locally complete *approximation of f at a set of inputs $S \subseteq \mathcal{C}$ when the equality holds on the elements of S, formally: $\forall c \in S.\, \alpha(f(c)) = f^\sharp(\alpha(c))$.*

Completeness is the strongest form of precision achievable in abstract interpretation: it requires the abstract behavior of $f^\sharp$ to coincide with the abstraction of the concrete behavior of f on *all* concrete inputs. Local completeness, introduced in [6,7], is a weaker notion that relaxes the (global) equation to a prescribed set of inputs $S \subseteq \mathcal{C}$, thus enabling precise analysis on the inputs of interest while tolerating imprecision elsewhere. Note that completeness coincides with local completeness when $S = \mathcal{C}$.

Completeness is also known in the literature as α-*completeness* or *backward completeness* [30,18]. The adjoint-dual notion of Def. 3 is the γ-*completeness*, or *forward completeness* [30,18]: $f \circ \gamma = \gamma \circ f^\sharp$, with the equality now checked in the concrete domain. The two notions play symmetric roles inside the GC framework, but they address different verification scenarios: γ-completeness compares $f(\gamma(a))$ against $\gamma(f^\sharp(a))$ in $\mathcal{C}$, whereas α-completeness is checked in $\mathcal{A}$ (typically a domain of decidable properties) and is sufficient for verifying specifications expressible in the abstract domain, the prevalent setting in GC-based static program analysis. Clearly, in the case that the concrete and abstract domains do not admit a GC, it is only possible to use the γ-completeness as a precision notion. Since we are assuming the existence of a GC, we accordingly focus on α-completeness throughout the paper.

3 Generators of an Abstract Property

In this section we recall from [11] the notion of *generator* of an abstract property, which is the building block of the present paper: generator-density, introduced in Sec. 4, is a structural property about the distribution of generators inside the fibers of the abstraction function.

For the rest of the paper, we fix a GC $\langle \mathcal{C}, \sqsubseteq_\mathcal{C} \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \mathcal{A}, \sqsubseteq_\mathcal{A} \rangle$ between a concrete and an abstract poset.

Definition 4 (Generator [11]). *The generator function $\mathscr{G} \colon \mathcal{A} \to \wp(\mathcal{C})$ associates to each abstract property $a \in \mathcal{A}$ the set $\mathscr{G}(a) \in \wp(\mathcal{C})$ defined by:*

$$\mathscr{G}(a) \overset{\text{def}}{=} \left\{ g \in \mathcal{C} \,\middle|\, (i)\ \alpha(g) = a \ \wedge\ (ii)\ \forall c \in \mathcal{C}.\, c \sqsubset_\mathcal{C} g \Rightarrow \alpha(c) \sqsubset_\mathcal{A} \alpha(g) \right\}$$

Each element of $\mathscr{G}(a)$ is called a generator *of a.*

Intuitively, a generator g of an abstract property $a \in \mathcal{A}$ is a concrete property that α maps exactly to a (condition (i)) while carrying the *least amount of concrete information* needed to represent a via α (condition (ii)). Note that (ii)

is stronger than the order-preservation of α, which guarantees only $c \sqsubseteq_{\mathcal{C}} g \Rightarrow \alpha(c) \sqsubseteq_{\mathcal{A}} \alpha(g)$; the strictness is precisely what characterizes generators as minimal witnesses of a in the fiber $\alpha^{-1}(a)$. This minimality intuition is made formal by the following characterization.

Proposition 1. *For every $a \in \mathcal{A}$, $\mathscr{G}(a) = Min(\alpha^{-1}(a))$.*

An abstract property $a \in \mathcal{A}$ is said to be *generable* when it admits at least one generator, i.e., $\mathscr{G}(a) \neq \varnothing$. By Prop. 1, generability of a is equivalent to the existence of a minimal element in the fiber $\alpha^{-1}(a)$. Generability may fail for two distinct reasons:

(1) $\alpha^{-1}(a) = \varnothing$, that is, a is not in the image of α. This case arises exactly when the GC is not a GI, and captures the spurious abstract properties.

(2) $\alpha^{-1}(a) \neq \varnothing$ but the fiber has no minimal element. This is the genuinely order-theoretic obstruction: a is the image of some concrete property, yet none of them is minimal.

Example 1 (Intervals over integers [19]). Consider the interval over integers GC $\langle \wp(\mathbb{Z}), \subseteq \rangle \underset{\alpha_{\mathsf{Int}}}{\overset{\gamma_{\mathsf{Int}}}{\leftrightarrows}} \langle \mathsf{Int}, \sqsubseteq_{\mathsf{Int}} \rangle$, where $\mathsf{Int} \overset{\text{def}}{=} \{[a,b] \mid a \in \mathbb{Z} \cup \{-\infty\} \wedge b \in \mathbb{Z} \cup \{+\infty\} \wedge a \leq b\} \cup \{\bot_{\mathsf{Int}}\}$ with the standard ordering $\sqsubseteq_{\mathsf{Int}}$ induced by the interval containment. For every closed interval $[a,b] \in \mathsf{Int}$ we get: $\mathscr{G}([a,b]) = \{\{a,b\}\}$, namely the unique two-element subset consisting of the two endpoints. In contrast, the unbounded intervals $[a,+\infty]$, $[-\infty,b]$, $[-\infty,+\infty]$ admit *no* generator for reason (2) above: their fibers are non-empty, but any witness would require an infinite set of integers with no minimal subset reproducing the same abstraction. ⧫

Example 2 (Trace to reachability property [18]). Let $\langle \Sigma, \tau \rangle$ be a transition system, where Σ is a (potentially infinite) set of states and $\tau \subseteq \Sigma \times \Sigma$ is a transition relation. A *trace* is a non-empty sequence of states $\sigma = s_0 s_1 \cdots$, finite or infinite, such that $(s_i, s_{i+1}) \in \tau$ whenever s_{i+1} is defined. We write $\Sigma^{+\infty}$ for the set of all traces, $\Sigma^{+} \subseteq \Sigma^{+\infty}$ for the set of the finite ones, and $\sigma_\omega \in \Sigma$ for the last state of a finite trace $\sigma \in \Sigma^{+}$. Consider the GC $\langle \wp(\Sigma^{+\infty}), \subseteq \rangle \underset{\alpha_{\mathcal{I}}}{\overset{\gamma_{\mathcal{I}}}{\leftrightarrows}} \langle \wp(\Sigma), \subseteq \rangle$ abstracting a trace property $P \in \wp(\Sigma^{+\infty})$ into the reachability invariant $\alpha_{\mathcal{I}}(P) \in \wp(\Sigma)$ collecting the final states of all finite traces in P. For any $S \in \wp(\Sigma)$, the generators of S are exactly

$$\mathscr{G}(S) = \left\{ G \in \wp(\Sigma^{+}) \,\middle|\, \forall \sigma \in G.\, \sigma_\omega \in S \;\wedge\; \forall s \in S.\, \exists! \sigma \in G.\, \sigma_\omega = s \right\},$$

i.e., the trace sets containing, for each state $s \in S$, exactly one finite trace ending in s. Unlike the interval example, here a single abstract property may admit *multiple* generators, reflecting the fact that distinct finite traces can lead to the same final state. Moreover, since for each $s \in S$ the one-state trace $\langle s \rangle$ is a finite trace ending in s, $\mathscr{G}(S) \neq \varnothing$ for any set of states $S \in \wp(\Sigma)$, thus all the abstract properties in this GC are generable. ⧫

When restricting attention to the generators of $a \in \mathcal{A}$ approximated by (or simply *below*) a concrete property $c \in \mathcal{C}$, we write $\mathscr{G}(a)_{\sqsubseteq c} \overset{\text{def}}{=} \{g \in \mathscr{G}(a) \mid g \sqsubseteq_{\mathcal{C}} c\}$.

4 Generator-Dense Galois Connections

The distribution of generators within fibers can vary substantially. Examples 1 and 2 show two different scenarios: in Ex. 1 some abstract properties are non-generable, so concrete elements in their fibers have no generator below them; in contrast, in Ex. 2 all abstract properties admit at least one generator and any concrete property approximates at least one generator. In this section we focus on GCs with the latter kind of distribution: every concrete property of each fiber sits *above* some generator of the corresponding abstract property. This condition, strictly stronger than asking every abstract property to admit a generator, is the defining property of the class we study next: the *generator-dense* GCs.

We start by recalling the standard order-theoretic notion of *dense subset* of a poset [34] (originally called *coinitial* subset in [1]).

Definition 5 (Dense set [34]). *Let $\langle \mathcal{L}, \sqsubseteq_{\mathcal{L}} \rangle$ be a poset and $S \subseteq \mathcal{L}$ a subset. We say that S is* dense in $\mathcal{L}$ *(or* $\mathcal{L}$-dense*) when:* $\forall l \in \mathcal{L}. \exists s \in S. s \sqsubseteq_{\mathcal{L}} l$.

Intuitively, every element of $\mathcal{L}$ admits a lower bound inside S, so that S acts as an "approximating basis from below" for $\mathcal{L}$. In our setting the role of $\mathcal{L}$ will be played by the fibers $\alpha^{-1}(a)$ of the abstraction function, and the role of S by the generators $\mathscr{G}(a)$ of the corresponding abstract property.

Definition 6 (Generator-dense GC). *A GC $\langle \mathcal{C}, \sqsubseteq_{\mathcal{C}} \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \mathcal{A}, \sqsubseteq_{\mathcal{A}} \rangle$ is said to be* generator-dense *(*GDGC *for short) when, for every abstract property $a \in \mathcal{A}$:*

$$\mathscr{G}(a) \text{ is } \alpha^{-1}(a)\text{-dense}$$

that is, the generators of each fiber are dense in that fiber.

Unfolding Def. 6 and invoking Prop. 1, a GC is generator-dense exactly when, for every abstract property $a \in \mathcal{A}$ and every concrete property $c \in \alpha^{-1}(a)$, there exists a generator $g \in \mathscr{G}(a)$ with $g \sqsubseteq_{\mathcal{C}} c$. In purely order-theoretic terms, the minimal elements of every fiber of α are dense in the fiber. The following proposition collects three equivalent reformulations of this condition that will be used throughout the paper.

Proposition 2. *The following statements are equivalent:*

(i) $\langle \mathcal{C}, \sqsubseteq_{\mathcal{C}} \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \mathcal{A}, \sqsubseteq_{\mathcal{A}} \rangle$ *is generator-dense;*

(ii) $\forall a \in \mathcal{A}. \forall c \in \alpha^{-1}(a). \mathscr{G}(\alpha(c))_{\sqsubseteq c} \neq \varnothing$;

(iii) for every $a \in \mathcal{A}$, $Min(\alpha^{-1}(a))$ *is* $\alpha^{-1}(a)$*-dense.*

Item (i) is the definition itself, item (ii) is its local form: generator-density is equivalent to requiring that every concrete property c already contain a generator of its own abstraction $\alpha(c)$ below itself, i.e., $\mathscr{G}(\alpha(c))_{\sqsubseteq c} \neq \varnothing$. Item (iii) rephrases the condition in purely order-theoretic terms—the minimal elements of each fiber are dense in it—without explicit reference to the generator terminology.

There are well-known GCs in the abstract interpretation literature (several widely used in practice) that are generator-dense. We illustrate a few below.

Example 3 (Bounded intervals). Fix integers MIN $\leq$ MAX and let $J = \mathbb{Z}_{\geq \mathrm{MIN}} \cap \mathbb{Z}_{\leq \mathrm{MAX}}$. Consider the restriction of the interval GC (Ex. 1) to J:

$$\langle \wp(J), \subseteq \rangle \underset{\alpha_{\mathsf{Int}}}{\overset{\gamma_{\mathsf{Int}}}{\leftrightarrows}} \langle \mathsf{Int}_J, \sqsubseteq_{\mathsf{Int}} \rangle$$

where $\mathsf{Int}_J \overset{\text{def}}{=} \{[a,b] \mid a,b \in J,\ a \leq b\} \cup \{\bot_{\mathsf{Int}}\}$, with $\bot_{\mathsf{Int}}$ as bottom, and $\alpha_{\mathsf{Int}}(\varnothing) = \bot_{\mathsf{Int}}$, $\alpha_{\mathsf{Int}}(S) = [min(S), max(S)]$ for $S \neq \varnothing$. For every $S \in \alpha^{-1}_{\mathsf{Int}}([a,b])$ one has $\{a,b\} \subseteq S$ as a generator, witnessing $\alpha^{-1}_{\mathsf{Int}}([a,b])$-density. Hence the GC is generator-dense. ♦

Example 4 (Sign [19]). Consider the sign GC $\langle \wp(\mathbb{Z}), \subseteq \rangle \underset{\alpha_{\mathsf{Sign}}}{\overset{\gamma_{\mathsf{Sign}}}{\leftrightarrows}} \langle \mathsf{Sign}, \sqsubseteq_{\mathsf{Sign}} \rangle$, where $\mathsf{Sign} \overset{\text{def}}{=} \{\bot, \mathsf{neg}, \mathsf{zero}, \mathsf{pos}, \leq 0, \geq 0, \neq 0, \top\}$ with the usual lattice structure, and α_{Sign} maps a set of integers to the least sign that covers it. Each fiber is witnessed from below by a finite subset. For instance, $\mathscr{G}(\mathsf{neg}) = \{\{n\} \mid n < 0\}$, and for every $S \in \alpha^{-1}_{\mathsf{Sign}}(\mathsf{neg})$ (i.e., every non-empty $S \subseteq \mathbb{Z}_{<0}$) any singleton $\{n\} \subseteq S$ with $n \in S$ gives a generator below S. The argument is analogous for the other non-bottom abstract elements: for signs covering k atomic classes, a generator picks one integer from each class, yielding a set of size k; for every concrete property in the fiber, such a generator can be chosen below it. Hence the GC is generator-dense. ♦

Example 5 (Trace to reachability). Consider the trace to reachability GC defined in Ex. 2 $\langle \wp(\Sigma^{+\infty}), \subseteq \rangle \underset{\alpha_{\mathcal{I}}}{\overset{\gamma_{\mathcal{I}}}{\leftrightarrows}} \langle \wp(\Sigma), \subseteq \rangle$. For every $R \subseteq \Sigma$ and every $T \in \alpha^{-1}_{\mathcal{I}}(R)$, for each $s \in R$ pick a trace $\sigma \in T$ such that $\sigma_\omega = s$ (such a trace exists because $s \in \alpha_{\mathcal{I}}(T) = R$); the set $T_R = \{\sigma \in T \mid \sigma_\omega \in R\}$ satisfies $T_R \subseteq T$ and $\alpha_{\mathcal{I}}(T_R) = R$. Shrinking T_R to a subset-minimal element of $\alpha^{-1}_{\mathcal{I}}(R)$ yields a generator $g \in \mathscr{G}(R)$ with $g \subseteq T$. Hence $\mathscr{G}(R)$ is $\alpha^{-1}_{\mathcal{I}}(R)$-dense for every R. ♦

Example 6 (Allocation-site abstraction [4]). Informally, let Loc be the (potentially infinite) set of *concrete memory locations*, namely, the run-time memory cells that a program allocates during execution. Following the standard memory model of points-to analysis [4], every successful invocation of an allocation primitive (e.g., `malloc(...)` in C) produces a fresh, distinct element of Loc that is never reused, so that Loc collects allocation instances rather than physical addresses and distinct allocation sites yield disjoint sets of locations. The set AllocSite, by contrast, contains the *allocation sites*, namely the syntactic positions in the source code where allocations occur (e.g., each occurrence of `malloc(...)` in the program text). AllocSite is bounded by the size of the program text, hence finite. A standard points-to abstraction [4] quotients Loc by a surjective function $\eta : \mathrm{Loc} \to \mathrm{AllocSite}$ that maps each concrete location to the unique allocation site where it was created (e.g., all the locations returned by repeated executions of the same `malloc(...)` statement collapse to the single site corresponding to that statement). This induces the GI

$$\langle \wp(\mathrm{Loc}), \subseteq \rangle \underset{\alpha_\eta}{\overset{\gamma_\eta}{\leftrightarrows}} \langle \wp(\mathrm{AllocSite}), \subseteq \rangle$$

with $\alpha_\eta(L) = \{\eta(\ell) \mid \ell \in L\}$. For every $S \subseteq \text{AllocSite}$ and every $L \in \alpha_\eta^{-1}(S)$, choosing for each $s \in S$ a location $\ell_s \in L \cap \eta^{-1}(s)$ (non-empty because $s \in \alpha_\eta(L) = S$) yields a set $\{\ell_s \mid s \in S\} \subseteq L$ that is minimal in $\alpha_\eta^{-1}(S)$, hence a generator. So $\mathscr{G}(S)$ is $\alpha_\eta^{-1}(S)$-dense, and the GC is generator-dense. ⧫

Example 7 (Dependency abstraction [17,44,39,38]). Let $\mathbb{X}$ be a finite set of variables, and let $\Sigma = \mathbb{Z}^{\mathbb{X}}$ be the set of all possible states $s \colon \mathbb{X} \to \mathbb{Z}$ mapping each variable to an integer value. We consider a GC that captures variables dependencies between initial and final states of (finite) program executions. Concretely, we consider sets $T \subseteq \Sigma \times \Sigma$ where each pair $(s_0, s_\omega) \in T$ represents an input-output behavior. We write $\text{DEP}_{x,y}(T)$ when a variable dependency $(x, y) \in D \subseteq \mathbb{X} \times \mathbb{X}$ holds in T, formally:

$$\text{DEP}_{x,y}(T) \overset{def}{\Leftrightarrow} \exists (s_0, s_\omega) \in T. \forall (s'_0, s'_\omega) \in T. s_0 \equiv_{\diagdown x} s'_0 \Rightarrow s_\omega(y) \neq s'_\omega(y)$$

where $s \equiv_{\diagdown x} s' \overset{def}{\Leftrightarrow} s(x) \neq s'(x) \wedge \forall y \neq x. s(y) = s'(y)$. Informally: any change to the input value of x (keeping all other input values unchanged) is guaranteed to change the output value of y. This is the standard *must non-interference dependency*: a guaranteed causal link between input x and output y, robust to non-deterministic choices the program may make. We lift this notion to capture variable dependencies that *must hold* over collections $S \subseteq \wp(\Sigma \times \Sigma)$ of program executions:

$$\langle \wp(\wp(\Sigma \times \Sigma)), \subseteq \rangle \underset{\alpha_{\rightsquigarrow}}{\overset{\gamma_{\rightsquigarrow}}{\leftrightarrows}} \langle \wp(\mathbb{X} \times \mathbb{X}), \supseteq \rangle$$

where $\alpha_{\rightsquigarrow}(S) \overset{\text{def}}{=} \{(x, y) \mid \forall T \in S \colon \text{DEP}_{x,y}(T)\}$ and $\gamma_{\rightsquigarrow}(D) \overset{\text{def}}{=} \{T \mid \forall (x, y) \in D \colon \text{DEP}_{x,y}(T)\}$. The above GC is generator-dense: let $D \in \wp(\mathbb{X} \times \mathbb{X})$ and $S \in \alpha_{\rightsquigarrow}^{-1}(D)$. For each $(x, y) \notin D$ (dependency that must not hold), define $F_{x,y} = \{T \in S \mid \neg \text{DEP}_{x,y}(T)\}$ (which is non-empty since $\alpha_{\rightsquigarrow}(S) = D$). A subset $S' \subseteq S$ has $\alpha_{\rightsquigarrow}(S') = D$ iff it intersects (hits) every $F_{x,y}$ (to exclude unwanted dependencies). Since $\mathbb{X}$ is finite, there are finitely many $F_{x,y}$, hence a minimal hitting set $G \subseteq S$ exists. By minimality, no strict subset of G still hits all families, so G is a generator of D. ⧫

On the other hand, there are also many well-known GCs that are not generator-dense. The interval GC shown in Ex. 1 and the octagon GC [40] fail it because abstract elements with $\pm\infty$ bounds have fibers admitting strictly $\subseteq$-descending chains with no minimum; the same problem rules out the Galois Connection $\langle \wp(\wp(\Sigma^{+\infty})), \subseteq \rangle \underset{\alpha_{\mathcal{T}}}{\overset{\gamma_{\mathcal{T}}}{\leftrightarrows}} \langle \wp(\Sigma^{+\infty}), \subseteq \rangle$ between semantic (hyper)properties (sets of sets of traces) and trace properties [18]. In Sec. 8 we discuss how the notion of GDGC could be relaxed to accommodate some of these cases.

Two remarks about the interplay between generator-density of a GC and generability of abstract properties (i.e., when $\mathscr{G}(a) \neq \varnothing$) are in order. First, a GDGC does *not* ensure that every abstract element is generable: when the GC is not a GI, α is not surjective and, thus, there exists $\hat{a} \in \mathcal{A}$ with an empty fiber $\alpha^{-1}(\hat{a}) = \varnothing$ which yields $\mathscr{G}(\hat{a}) = \varnothing$ by Prop. 1, while the density condition is vacuously satisfied on such fiber. Second, the converse also fails: even when the GC is a GI and every abstract element is generable, density may still break. The

typical obstruction is the presence, in some fiber $\alpha^{-1}(a)$, of an *infinite strictly descending chain of concrete properties* sitting above no generator of a, as the next example illustrates.

Example 8. Consider the GC $\langle \mathcal{C}, \sqsubseteq_{\mathcal{C}} \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \mathcal{A}, \sqsubseteq_{\mathcal{A}} \rangle$ defined as follows. The concrete domain is $\mathcal{C} = \mathbb{Z} \cup \{\bot_{\mathcal{C}}, \$, \top_{\$}, \top_{\mathcal{C}}\}$, ordered by extending the usual order on $\mathbb{Z}$ with: $\bot_{\mathcal{C}} \sqsubseteq_{\mathcal{C}} x \sqsubseteq_{\mathcal{C}} \top_{\mathcal{C}}$ for every $x \in \mathcal{C}$; $n \sqsubseteq_{\mathcal{C}} \top_{\$}$ for every $n \in \mathbb{Z}$; $\$ \sqsubseteq_{\mathcal{C}} \top_{\$}$; and $\top_{\$} \sqsubseteq_{\mathcal{C}} \top_{\mathcal{C}}$. The element $\$$ is incomparable with every integer. The element $\top_{\$}$ is included exactly so that the subset $\mathbb{Z} \cup \{\$\}$ admits a join in $\mathcal{C}$, which is required for the abstraction below to preserve arbitrary joins. The abstract domain is $\mathcal{A} = \{\bot_{\mathcal{A}}, \$^\sharp, \top_{\mathcal{A}}\}$ with $\bot_{\mathcal{A}} \sqsubseteq_{\mathcal{A}} \$^\sharp \sqsubseteq_{\mathcal{A}} \top_{\mathcal{A}}$, and the abstraction is defined by $\alpha(\bot_{\mathcal{C}}) = \bot_{\mathcal{A}}$, $\alpha(\top_{\mathcal{C}}) = \top_{\mathcal{A}}$, and $\alpha(c) = \$^\sharp$ for every $c \in \mathbb{Z} \cup \{\$, \top_{\$}\}$. One checks that α preserves arbitrary joins, so the adjoint concretization γ exists and satisfies $\gamma(\bot_{\mathcal{A}}) = \bot_{\mathcal{C}}$, $\gamma(\$^\sharp) = \top_{\$}$, $\gamma(\top_{\mathcal{A}}) = \top_{\mathcal{C}}$. Moreover, every abstract element is generable, with: $\mathscr{G}(\bot_{\mathcal{A}}) = \{\bot_{\mathcal{C}}\}$, $\mathscr{G}(\$^\sharp) = \{\$\}$, $\mathscr{G}(\top_{\mathcal{A}}) = \{\top_{\mathcal{C}}\}$. However, the GC is *not* generator-dense: the fiber $\alpha^{-1}(\$^\sharp) = \mathbb{Z} \cup \{\$, \top_{\$}\}$ contains the strictly infinite descending chain $0 > -1 > -2 > \cdots$, and no element of this chain has a lower bound in $\mathscr{G}(\$^\sharp) = \{\$\}$, since $\$$ is incomparable with every integer. Hence $\mathscr{G}(\$^\sharp)$ fails to be $\alpha^{-1}(\$^\sharp)$-dense. ♦

5 Operations on GDGCs

Abstract interpretations are typically assembled from smaller GCs by algebraic operations, taking products, lifting pointwise over an index set, composing abstractions, and so on. It is therefore natural to ask which of these operations preserve the generator-dense property, so that the reduction-to-generators results of Sec. 6 continue to apply to composite abstract domains obtained by such constructions. In this section we study three such operations: product, pointwise lifting, and composition. Products and pointwise liftings preserve generator-density unconditionally; composition does *not*, as we exhibit with a counter-example.

Theorem 1 (Product). *Consider two GDGCs* $\langle \mathcal{C}_1, \sqsubseteq_1 \rangle \underset{\alpha_1}{\overset{\gamma_1}{\leftrightarrows}} \langle \mathcal{A}_1, \sqsubseteq_1^\sharp \rangle$ *and* $\langle \mathcal{C}_2, \sqsubseteq_2 \rangle \underset{\alpha_2}{\overset{\gamma_2}{\leftrightarrows}} \langle \mathcal{A}_2, \sqsubseteq_2^\sharp \rangle$. *The product GC* $\langle \mathcal{C}_1 \times \mathcal{C}_2, \sqsubseteq_\times \rangle \underset{\alpha_\times}{\overset{\gamma_\times}{\leftrightarrows}} \langle \mathcal{A}_1 \times \mathcal{A}_2, \sqsubseteq_\times^\sharp \rangle$, *with the componentwise orders*

$$(c_1, c_2) \sqsubseteq_\times (c_1', c_2') \overset{def}{\Leftrightarrow} c_1 \sqsubseteq_1 c_1' \wedge c_2 \sqsubseteq_2 c_2'$$
$$(a_1, a_2) \sqsubseteq_\times^\sharp (a_1', a_2') \overset{def}{\Leftrightarrow} a_1 \sqsubseteq_1^\sharp a_1' \wedge a_2 \sqsubseteq_2^\sharp a_2',$$

and componentwise adjoints

$$\alpha_\times(c_1, c_2) \overset{\text{def}}{=} (\alpha_1(c_1), \alpha_2(c_2)) \quad \gamma_\times(a_1, a_2) \overset{\text{def}}{=} (\gamma_1(a_1), \gamma_2(a_2))$$

is generator-dense.

The intuition is that the product order is componentwise: a minimal element of a product fiber is exactly a pair of componentwise minimal elements of the individual fibers. Hence generator-density lifts coordinate-by-coordinate.

Example 9 (Hyperrectangles of bounded intervals). Iterating Thm. 1 on the bounded intervals GC of Ex. 3 yields the n-dimensional hyperrectangle GC

$$\langle \wp(J)^n, \subseteq \rangle \underset{\alpha^n_{\mathsf{Int}}}{\overset{\gamma^n_{\mathsf{Int}}}{\leftrightarrows}} \langle \mathsf{Int}^n_J, \sqsubseteq_{\mathsf{Int}} \rangle$$

abstracting n-tuples of sets of integers into n-dimensional hyperrectangles of Int_J. By Thm. 1, this GC is generator-dense: for each non-empty hyperrectangle $[a_1, b_1] \times \cdots \times [a_n, b_n]$ its unique generator is the tuple $(\{a_1, b_1\}, \ldots, \{a_n, b_n\})$. ♦

Theorem 2 (Pointwise lifting). *Let* $\langle \mathcal{C}, \sqsubseteq_{\mathcal{C}} \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \mathcal{A}, \sqsubseteq_{\mathcal{A}} \rangle$ *be a GDGC and* X *a non-empty set. The pointwise lifting* $\langle X \to \mathcal{C}, \sqsubseteq_{pw} \rangle \underset{\alpha_{pw}}{\overset{\gamma_{pw}}{\leftrightarrows}} \langle X \to \mathcal{A}, \sqsubseteq^\sharp_{pw} \rangle$, *with the pointwise orders*

$$f \sqsubseteq_{pw} f' \overset{\text{def}}{\Leftrightarrow} \forall x \in X.\, f(x) \sqsubseteq_{\mathcal{C}} f'(x), \qquad g \sqsubseteq^\sharp_{pw} g' \overset{\text{def}}{\Leftrightarrow} \forall x \in X.\, g(x) \sqsubseteq_{\mathcal{A}} g'(x),$$

and adjoints $\alpha_{pw}(f)(x) \overset{\text{def}}{=} \alpha(f(x))$, $\gamma_{pw}(g)(x) \overset{\text{def}}{=} \gamma(g(x))$, *is generator-dense.*

The proof is analogous to that of Thm. 1: each fiber of α_{pw} is a Cartesian product of base fibers indexed by X, and minimality is pointwise.

Example 10 (Non-relational store abstraction). Let *Var* be a (possibly infinite) set of program variables. The non-relational *sign-store abstraction* is the pointwise lifting of the sign GC $\langle \wp(\mathbb{Z}), \subseteq \rangle \underset{\alpha_{\mathsf{Sign}}}{\overset{\gamma_{\mathsf{Sign}}}{\leftrightarrows}} \langle \mathsf{Sign}, \sqsubseteq_{\mathsf{Sign}} \rangle$ of Ex. 4 over *Var*:

$$\langle\, \mathit{Var} \to \wp(\mathbb{Z}), \sqsubseteq_{pw} \rangle \underset{\alpha_{pw}}{\overset{\gamma_{pw}}{\leftrightarrows}} \langle\, \mathit{Var} \to \mathsf{Sign}, \sqsubseteq^\sharp_{pw} \rangle\,,$$

with $\alpha_{pw}(\rho)(x) \overset{\text{def}}{=} \alpha_{\mathsf{Sign}}(\rho(x))$ and $\gamma_{pw}(\sigma)(x) \overset{\text{def}}{=} \gamma_{\mathsf{Sign}}(\sigma(x))$. Since $\langle \wp(\mathbb{Z}), \subseteq \rangle \underset{\alpha_{\mathsf{Sign}}}{\overset{\gamma_{\mathsf{Sign}}}{\leftrightarrows}} \langle \mathsf{Sign}, \sqsubseteq_{\mathsf{Sign}} \rangle$ is generator-dense (Ex. 4), so is its pointwise lifting by Thm. 2. The same argument applies to any non-relational abstraction built by lifting a generator-dense base GC over variables, e.g., intervals, parity, constants. ♦

We now turn to composition, the operation most frequently used to build a chain of abstractions $\mathcal{C} \to \mathcal{A}_1 \to \cdots \to \mathcal{A}_k$. Unfortunately, in contrast with products and pointwise liftings, composition does *not* preserve generator-density, even when both components are GCs between complete lattices.

Theorem 3 (Non-closure under composition). *Generator-density is* not *preserved by composition of GDGCs.*

Proof. Thm. 3 is witnessed by the following counter-example. Consider three ordered domains: $\mathcal{C} = \{c_n \mid n \in \mathbb{N}\} \cup \{\omega\}$ with $\omega \sqsubset \cdots \sqsubset c_{n+1} \sqsubset c_n \sqsubset \cdots \sqsubset c_0$; $\mathcal{A} = \{a_n \mid n \in \mathbb{N}\} \cup \{a^*, a_\omega\}$ with $a_\omega \sqsubset a^* \sqsubset \cdots \sqsubset a_{n+1} \sqsubset a_n \sqsubset \cdots \sqsubset a_0$; and $\mathcal{A}' = \{b_0, b_1\}$ with $b_1 \sqsubset b_0$. Define two GCs by:

$$\alpha_1(c_n) \overset{\text{def}}{=} a_n,\ \alpha_1(\omega) \overset{\text{def}}{=} a_\omega; \qquad \gamma_1(a_n) \overset{\text{def}}{=} c_n,\ \gamma_1(a^*) \overset{\text{def}}{=} \gamma_1(a_\omega) \overset{\text{def}}{=} \omega;$$
$$\alpha_2(a_n) \overset{\text{def}}{=} b_0,\ \alpha_2(a^*) \overset{\text{def}}{=} b_0,\ \alpha_2(a_\omega) \overset{\text{def}}{=} b_1; \qquad \gamma_2(b_0) \overset{\text{def}}{=} a_0,\ \gamma_2(b_1) \overset{\text{def}}{=} a_\omega.$$

Both GCs are generator-dense: every non-empty fiber of α_1 is a singleton (note that a^* has empty fiber), and the non-singleton fiber $\alpha_2^{-1}(b_0) = \{a^*\} \cup \{a_n \mid n \in \mathbb{N}\}$ admits a^* as its minimum, which is therefore its unique generator. However, the composed abstraction $\alpha_2 \circ \alpha_1 : \mathcal{C} \to \mathcal{A}'$ sends every c_n to b_0 and ω to b_1. The outer fiber $(\alpha_2 \circ \alpha_1)^{-1}(b_0) = \{c_n \mid n \in \mathbb{N}\}$ is an infinite strictly descending chain with no minimum. Consequently, $Min((\alpha_2 \circ \alpha_1)^{-1}(b_0)) = \varnothing$, and c_0 has no generator below it in the composed GC. □

Inspecting the proof of Thm. 3 reveals that the minimum a^* of $\alpha_2^{-1}(b_0)$ is absent from the image of α_1, so it cannot be pulled back through γ_1 to a minimum of the composed outer fiber in $\mathcal{C}$. The problem originates from the existence of an infinite strictly descending chain inside an outer fiber $((\alpha_2 \circ \alpha_1)^{-1}(b_0))$, with no lower bound in that fiber. Forbidding it amounts to requiring the descending chain condition (DCC) on every outer fiber—a sufficient structural condition not only for compositions, but for *any* GC to be generator-dense.

Theorem 4 (DCC on fibers). *If, for every $a \in \mathcal{A}$, the fiber $\alpha^{-1}(a)$ satisfies DCC with respect to $\sqsubseteq_{\mathcal{C}}$, then the GC is generator-dense.*

The proof proceeds by a standard DCC-descent argument: starting from any $c \in \alpha^{-1}(a)$, we descend in the sub-poset $\{c' \in \alpha^{-1}(a) \mid c' \sqsubseteq_{\mathcal{C}} c\}$; DCC guarantees this descent terminates at a minimal element of $\alpha^{-1}(a)$ which, by construction, lies below c.

Two notable corollaries follow. *(i)* When $\mathcal{C}$ itself satisfies DCC, every fiber inherits DCC (sub-posets of a DCC poset are DCC), so every GC over $\mathcal{C}$ is generator-dense. This is the case, for instance, of bounded intervals (Ex. 3). *(ii)* Applied to the composition $\alpha_2 \circ \alpha_1$ of two GCs, Thm. 4 yields a sufficient condition for closure under composition: *if every outer fiber $(\alpha_2 \circ \alpha_1)^{-1}(a')$ satisfies DCC then the composition is generator-dense*, regardless of whether the components are themselves generator-dense.

Example 11 (Binary floor abstraction). Fix $N \in \mathbb{N}$ and let $\mathcal{C} = \{0, 1, \ldots, N\} \times (\mathbb{Z} \cup \{-\infty, +\infty\})$ with the pointwise order. $\mathcal{C}$ fails DCC, since $(0,0) > (0,-1) > (0,-2) > \cdots$ is a strictly descending chain that never stabilizes. Consider the projection onto the second factor

$$\langle \mathcal{C}, \sqsubseteq \rangle \underset{\alpha_1}{\overset{\gamma_1}{\leftrightarrows}} \langle \mathbb{Z} \cup \{-\infty, +\infty\}, \leq \rangle, \quad \alpha_1(k,n) \stackrel{\text{def}}{=} n, \quad \gamma_1(n) \stackrel{\text{def}}{=} (N,n)\,.$$

For every n, the fiber $\{0, \ldots, N\} \times \{n\}$ is finite with minimum $(0,n)$, hence, by Thm. 4, the GC is generator-dense. Compose with the binary-floor abstraction

$$\langle \mathbb{Z} \cup \{-\infty, +\infty\}, \leq \rangle \underset{\alpha_2}{\overset{\gamma_2}{\leftrightarrows}} \langle \mathbb{Z} \cup \{-\infty, +\infty\}, \leq \rangle,$$

with $\alpha_2(n) \stackrel{\text{def}}{=} \lfloor n/2 \rfloor$ on integers, $\alpha_2(+\infty) \stackrel{\text{def}}{=} +\infty$, $\alpha_2(-\infty) \stackrel{\text{def}}{=} -\infty$, $\gamma_2(n) \stackrel{\text{def}}{=} 2n{+}1$, $\gamma_2(+\infty) \stackrel{\text{def}}{=} +\infty$, $\gamma_2(-\infty) \stackrel{\text{def}}{=} -\infty$, which is generator-dense (fiber of $m \in \mathbb{Z}$ is $\{2m, 2m{+}1\}$ with minimum $2m$; fibers of $\pm\infty$ are singletons). The composed abstraction sends $(k,n) \mapsto \lfloor n/2 \rfloor$, so the outer fiber of $m \in \mathbb{Z}$ is the finite rectangle

$\{0,\ldots,N\}\times\{2m,2m{+}1\}$ and the outer fibers of $\pm\infty$ are $\{0,\ldots,N\}\times\{\pm\infty\}$—all finite, hence DCC. Thm. 4, applied to the composed GC, gives generator-density of the composition. ⧫

A different, more structural way to recover composition is to specialize the left GC to a family for which the "minimum problem" shown in the counter-example of Thm. 3 cannot arise: the *elementwise* GCs introduced by Darais and Van Horn [24]. These lift an arbitrary function $\eta : X \to Y$ pointwise to powersets, making the abstraction additive; the minimal elements of each fiber $\alpha^{-1}(T)$ are then precisely the sets $\{\ell_y \mid y \in T\}$ obtained by picking one η-preimage $\ell_y \in \eta^{-1}(y)$ for every $y \in T$.

Definition 7 (Elementwise GC [24]). *A GC* $\langle \wp(X), \subseteq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \wp(Y), \subseteq\rangle$ *is* elementwise *if there exists a function* $\eta : X \to Y$ *such that* $\alpha(S) = \{\eta(x) \mid x \in S\}$ *for all* $S \subseteq X$.

Elementwise GCs subsume many standard abstractions: the allocation-site abstraction of Ex. 6 is elementwise via η defined as the allocation-site function, and the trace-to-reachability GC of Ex. 5 is elementwise via $\eta(\sigma) = \sigma_\omega$ (defined on finite traces, with infinite traces excluded from the domain). The next theorem shows that elementwise GCs, unlike arbitrary GDGCs, enjoy a clean closure property under composition with any GDGC.

Theorem 5 (Elementwise and generator-density). *If* GC_1 *is elementwise and* GC_2 *is generator-dense, then* $GC_2 \circ GC_1$ *is generator-dense.*

Example 12 (Allocation-site composed with region abstraction). It is easy to see that the allocation-site GI of Ex. 6 is elementwise via the surjection $\eta_1 : \mathrm{Loc} \to \mathrm{AllocSite}$. We can now further abstract allocation sites into *memory regions* (e.g., heap, stack, globals) via a surjective function $\eta_2 : \mathrm{AllocSite} \to \mathrm{Region}$, which induces the elementwise GI:

$$\langle \wp(\mathrm{AllocSite}), \subseteq\rangle \underset{\alpha_{\eta_2}}{\overset{\gamma_{\eta_2}}{\leftrightarrows}} \langle \wp(\mathrm{Region}), \subseteq\rangle .$$

This second GC is generator-dense by the selection argument. By Thm. 5 (applied to GC_1 = allocation-site and GC_2 = region), the composition between $\langle \wp(\mathrm{Loc}), \subseteq\rangle$ and $\langle \wp(\mathrm{Region}), \subseteq\rangle$ is generator-dense. This conclusion is not available from Thm. 4: when Loc is infinite, the outer fibers inherit arbitrarily long descending chains of the form $S \supset S \setminus \{\ell_1\} \supset S \setminus \{\ell_1, \ell_2\} \supset \cdots$ whenever they contain an infinite set S, hence they fail DCC.

The standard per-variable points-to map abstraction $\langle \mathrm{Var} \to \wp(\mathrm{Loc}), \sqsubseteq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \mathrm{Var} \to \wp(\mathrm{AllocSite}), \sqsubseteq\rangle$ used in Andersen-style analyses—which tracks, for each program variable, the set of allocation sites it may point to—arises as the pointwise lifting (Thm. 2) of the GC of Ex. 6 over Var, and is therefore generator-dense as well, but not elementwise. ⧫

Since the composition of elementwise GCs via $\eta_1 : X_1 \to X_2, \ldots, \eta_{k-1} : X_{k-1} \to X_k$ is itself elementwise—via the composed function $\eta_{k-1} \circ \cdots \circ \eta_1$—iterating Thm. 5 extends the closure property to arbitrary pipelines:

Corollary 1 (Pipelines of elementwise GCs). *Let* $\mathrm{GC}_i : \langle \wp(X_i), \subseteq \rangle \underset{\alpha_i}{\overset{\gamma_i}{\leftrightarrows}} \langle \wp(X_{i+1}), \subseteq \rangle$ *be elementwise for* $i = 1, \ldots, k-1$*, and* $\mathrm{GC}_k : \langle \wp(X_k), \subseteq \rangle \underset{\alpha_k}{\overset{\gamma_k}{\leftrightarrows}} \langle \mathcal{A}, \sqsubseteq_{\mathcal{A}} \rangle$ *be generator-dense. Then the full composition* $\mathrm{GC}_k \circ \cdots \circ \mathrm{GC}_1$ *is generator-dense.*

Cor. 1 covers many pipeline abstractions of the Cousot hierarchy [18], such as the pipeline $\wp(\Sigma^{+\infty}) \to \wp(\Sigma) \to \mathsf{Sign}$ from traces to reachable states to sign analysis, reducing the verification of generator-density to checking only the final, non-elementwise link.

6 Partial Completeness and Generator-Density

Completeness of an abstract interpretation is the strongest precision condition one can demand. Since this is hard to achieve in practice [27], *partial (local) completeness* [11,8] has been introduced to relax the equality into a bounded gap, measured by an order-compatible pre-metric on the abstract domain [14]. This reflects common practice: abstract interpretation-based program analyses are typically shown to have bounded imprecision over specific subsets of inputs of interest. This section first recalls the pre-metric framework (Sec. 6.1), then proves the main result of the paper: *under generator-density, partial completeness can be established by checking only the generators*, with the imprecision bound automatically propagated to the whole concrete domain (Sec. 6.2). Sec. 6.3 illustrates the theorem on four abstract domains.

6.1 Partial Completeness with Pre-metrics

Partial completeness [11,8] relaxes the completeness condition $\alpha \circ f = f^\sharp \circ \alpha$ by admitting a bounded amount of imprecision, measured by a distance between $\alpha(f(c))$ and $f^\sharp(\alpha(c))$ in the abstract domain. This distance is measured by *pre-metrics* that manifest a form of compatibility (axiom (*chain-order*)) with the underlying partial order of the abstract domain.

Definition 8 (Order-compatible pre-metric [14]). *Let* $\langle \mathcal{L}, \sqsubseteq_{\mathcal{L}} \rangle$ *be a poset. A function* $\delta_{\mathcal{L}} : \mathcal{L} \times \mathcal{L} \to \mathbb{R}^{\infty}_{\geq 0}$ *is a* pre-metric compatible with $\sqsubseteq_{\mathcal{L}}$ *($\sqsubseteq_{\mathcal{L}}$-compatible pre-metric, for short) when the following two axioms hold:*

(if-identity) $x = y \;\Rightarrow\; \delta_{\mathcal{L}}(x, y) = 0$*;*
(chain-order) $x \sqsubseteq_{\mathcal{L}} y \sqsubseteq_{\mathcal{L}} z \;\Rightarrow\; \delta_{\mathcal{L}}(x, y) \leq \delta_{\mathcal{L}}(x, z) \;\wedge\; \delta_{\mathcal{L}}(y, z) \leq \delta_{\mathcal{L}}(x, z)$*.*

When additionally $x \sqsubseteq_{\mathcal{L}} y \;\Rightarrow\; (\delta_{\mathcal{L}}(x, y) = 0 \;\Rightarrow\; x = y)$ *holds* (chain iff-identity)*,* $\delta_{\mathcal{L}}$ *is said to be a* strong $\sqsubseteq_{\mathcal{L}}$*-compatible pre-metric.*

Axiom *(if-identity)* makes $\delta_{\mathcal{L}}$ a pre-metric in the standard sense: equal elements are at zero distance, but the converse need not hold. Axiom *(chain-order)* imposes monotonicity along comparable elements: on any chain $x \sqsubseteq_{\mathcal{L}} y \sqsubseteq_{\mathcal{L}} z$,

both inner distances $\delta_{\mathcal{L}}(x, y)$ and $\delta_{\mathcal{L}}(y, z)$ are dominated by the distance between the two extremes $\delta_{\mathcal{L}}(x, z)$. In abstract interpretation terms, instantiating the chain with $x = \alpha(f(c))$, $y = f_1^\sharp(\alpha(c))$, and $z = f_2^\sharp(\alpha(c))$ for two sound abstract transformers $f_1^\sharp \sqsubseteq_{\mathcal{A}} f_2^\sharp$ (so that $f_1^\sharp$ is more precise than $f_2^\sharp$), this captures the expected monotonicity of imprecision with respect to precision: the more precise $f_1^\sharp$ is closer to the concrete evaluation than the coarser $f_2^\sharp$—$\delta_{\mathcal{A}}(\alpha(f(c)), f_1^\sharp(\alpha(c))) \leq \delta_{\mathcal{A}}(\alpha(f(c)), f_2^\sharp(\alpha(c)))$—and the gap between $f_1^\sharp$ and $f_2^\sharp$ cannot exceed the total imprecision of $f_2^\sharp$ relative to the concrete.

Essentially all the distances used in practice for measuring the imprecision of abstract interpretations fit Def. 8 (see, e.g., [8,35,36,25,43,15,13]); we recall three very simple ones that will be used in the examples of Sec. 6.3.

Example 13 (Equality distance [14]). For every poset $\langle \mathcal{L}, \sqsubseteq_{\mathcal{L}} \rangle$, the *equality distance* $\delta_{\mathcal{L}}^{=} : \mathcal{L} \times \mathcal{L} \to \mathbb{N}^\infty$ is defined by $\delta_{\mathcal{L}}^{=}(x, y) \stackrel{\text{def}}{=} 0$ if $x = y$ and $\delta_{\mathcal{L}}^{=}(x, y) \stackrel{\text{def}}{=} \infty$ otherwise. It is a strong $\sqsubseteq_{\mathcal{L}}$-compatible pre-metric. ⧫

Example 14 (Size distance [12]). For any powerset $\wp(L)$ of a set L, the size distance $\delta_{\wp(L)}^{siz} : \wp(L) \times \wp(L) \to \mathbb{N}^\infty$ between two sets $S_1, S_2 \in \wp(L)$ is the absolute value difference in their size, i.e., $\delta_{\wp(L)}^{siz}(S_1, S_2) \stackrel{\text{def}}{=} \mathsf{Abs}(|S_1| - |S_2|)$, where Abs is the absolute value function (we assume $\mathsf{Abs}(\infty - \infty) = \infty$). ⧫

Example 15 (Counting distance on intervals [11]). On the bounded-interval poset $\langle \mathsf{Int}_J, \sqsubseteq_{\mathsf{Int}} \rangle$ of Ex. 3, the *counting distance* $\delta_{\mathsf{Int}}^{\sim} : \mathsf{Int}_J \times \mathsf{Int}_J \to \mathbb{N}^\infty$ measures the absolute difference between the cardinalities of the two concretizations:

$$\delta_{\mathsf{Int}}^{\sim}([a_1, b_1], [a_2, b_2]) \stackrel{\text{def}}{=} \mathsf{Abs}(\, |\gamma_{\mathsf{Int}}([a_1, b_1])| - |\gamma_{\mathsf{Int}}([a_2, b_2])|\,)\,.$$

It is a strong $\sqsubseteq_{\mathsf{Int}}$-compatible pre-metric, quantifying the number of spurious integer values introduced by an over-approximation: when $[a_1, b_1] \sqsubseteq_{\mathsf{Int}} [a_2, b_2]$, $\delta_{\mathsf{Int}}^{\sim}([a_1, b_1], [a_2, b_2])$ is exactly the number of integer values in $[a_2, b_2]$ but not in $[a_1, b_1]$. A multi-dimensional generalization is the *volume distance* δ^{Vol} [14], which computes the difference of the volumes of two hyperrectangles. ⧫

Definition 9 ($\boldsymbol{e}$-Partial (local) completeness [11]). *Consider a GC* $\langle \mathcal{C}, \sqsubseteq_{\mathcal{C}} \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \mathcal{A}, \sqsubseteq_{\mathcal{A}} \rangle$*, a* $\sqsubseteq_{\mathcal{A}}$*-compatible pre-metric* $\delta_{\mathcal{A}}$*, an order-preserving concrete function* $f : \mathcal{C} \to \mathcal{C}$*, and a bounding function* $\boldsymbol{e} : \mathcal{C} \to \mathbb{R}_{\geq 0}^\infty$*. A sound abstract function* $f^\sharp : \mathcal{A} \to \mathcal{A}$ *is* $\boldsymbol{e}$-partial local complete *for* f *at* $S \subseteq \mathcal{C}$ *when:*

$$\forall c \in S.\, \delta_{\mathcal{A}}(\alpha(f(c)), f^\sharp(\alpha(c))) \leq \boldsymbol{e}(c)\,;$$

It is $\boldsymbol{e}$-partial complete *for* f *when* $S = \mathcal{C}$*.*

The bounding function $\boldsymbol{e} : \mathcal{C} \to \mathbb{R}_{\geq 0}^\infty$ assigns a budget of admissible imprecision to each concrete input c: the abstract function $f^\sharp$ is $\boldsymbol{e}$-partially complete for f when, measured by $\delta_{\mathcal{A}}$, its output on $\alpha(c)$ is at most $\boldsymbol{e}(c)$ away from the abstraction $\alpha(f(c))$ of the concrete output. Partial local completeness restricts the requirement to a subset $S \subseteq \mathcal{C}$ of interest. When the pre-metric chosen is also strong, $\mathbf{0}$-partial completeness, where $\mathbf{0} \stackrel{\text{def}}{=} \lambda x.\, 0$, corresponds to completeness (Def. 3) thanks to axiom *(chain iff-identity)*.

6.2 Main Result: Reducing to Generators

We write $\mathcal{G} \stackrel{\text{def}}{=} \bigcup_{a \in \mathcal{A}} \mathscr{G}(a)$ for the set of all generators of an abstract domain $\mathcal{A}$.

Theorem 6 (GDGC and partial completeness). *Assume* $\langle \mathcal{C}, \sqsubseteq_{\mathcal{C}} \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \mathcal{A}, \sqsubseteq_{\mathcal{A}} \rangle$ *is generator-dense. Let* $f : \mathcal{C} \to \mathcal{C}$ *be order-preserving,* $f^\sharp : \mathcal{A} \to \mathcal{A}$ *sound, and* $\delta_{\mathcal{A}}$ *a* $\sqsubseteq_{\mathcal{A}}$*-compatible pre-metric. The following implication holds:*

$$f^\sharp \ \boldsymbol{e}\text{-partial local complete at } \mathcal{G} \ \Rightarrow \ f^\sharp \text{ is } \hat{\boldsymbol{e}}\text{-partial complete}$$

where for all $c \in \mathcal{C}$*:* $\hat{\boldsymbol{e}}(c) \stackrel{\text{def}}{=} \inf\{\boldsymbol{e}(g) \mid g \in \mathscr{G}(\alpha(c))_{\sqsubseteq c}\}$.

Thm. 6 states that once $f^\sharp$ is $\boldsymbol{e}$-partial local complete at every generator $g \in \mathcal{G}$, it is automatically $\hat{\boldsymbol{e}}$-partial complete on the *entire* concrete domain, with $\hat{\boldsymbol{e}}(c)$ equal to the greatest lower bound among the $\boldsymbol{e}$-values of the generators of $\alpha(c)$ below c. Generator-density is the structural hypothesis that drives the reduction: it ensures $\mathscr{G}(\alpha(c))_{\sqsubseteq c}$ is non-empty for every c, so that the infimum is taken over a non-empty set. Moreover, when $\boldsymbol{e}$ takes values in a well-ordered set—as is the case when its values lie in $\mathbb{N}^\infty$, e.g., for the counting distance of Ex. 15—the infimum is attained at a specific $g^* \in \mathscr{G}(\alpha(c))_{\sqsubseteq c}$, making the bound *witnessed*: one can exhibit a canonical generator realizing it. Note that the reverse implication of Thm. 6, which would be written as

$$f^\sharp \ \boldsymbol{e}\text{-partial local complete at } \mathcal{G} \ \Leftarrow \ f^\sharp \text{ is } \boldsymbol{e}\text{-partial complete}$$

is trivially satisfied since $\mathcal{G} \subseteq \mathcal{C}$.

Thm. 6 has a clean consequence for completeness (Def. 3).

Corollary 2 (GDGC and completeness).

$$f^\sharp \text{ is locally complete at } \mathcal{G} \ \Leftrightarrow \ f^\sharp \text{ is complete.}$$

Cor. 2 is obtained as a direct instance of Thm. 6 by taking the strong pre-metric $\delta^{=}_{\mathcal{A}}$ (Ex. 13) and $\boldsymbol{e} \equiv 0$. Under these choices, $\hat{\boldsymbol{e}} \equiv 0$ and *(chain iff-identity)* of $\delta^{=}_{\mathcal{A}}$ upgrades $\delta^{=}_{\mathcal{A}}(\alpha(f(c)), f^\sharp(\alpha(c))) = 0$ to the equality $\alpha(f(c)) = f^\sharp(\alpha(c))$.

The value of Thm. 6 and Cor. 2 lies not primarily in a *cardinality* reduction—$\mathcal{G}$ is typically as large as $\mathcal{C}$ when $\mathcal{C}$ is infinite—but in a *structural* one. Not in cardinality reduction because, whenever α separates singletons (i.e., $\alpha(\{x\}) \neq \alpha(\{y\})$ for distinct atoms $x, y \in X$), each singleton $\{x\}$ is minimal in the fiber $\alpha^{-1}(\alpha(\{x\}))$, so $|\mathcal{G}| \geq |X|$; hence $\mathcal{G}$ is infinite whenever X is. This covers all standard abstractions—sign, intervals, allocation-site, points-to, trace-to-reachability—and precludes an "infinite-to-finite" collapse for any abstraction that distinguishes concrete atoms. Finite $\mathcal{G}$ with infinite $\mathcal{C}$ is possible only under "heavy forgetting" abstractions, such as the projection $S \mapsto S \cap E$ onto a fixed finite observable set E.

What the reduction *does* buy is structural, in three complementary ways. First, each generator $g \in \mathscr{G}(a)$ is a *canonical witness* for a: a minimal concrete property carrying only the information of a and nothing else.

Second, Thm. 6 and Cor. 2 highlight that, when working with GDGCs, both the precision and the worst-case imprecision of $f^\sharp$ with respect to f at any input in $\mathcal{C}$ are *fully characterized* by its behavior on generators. Moreover, Thm. 6 *localizes the origin of imprecision of an abstract interpretation*: when $f^\sharp$ is $\hat{\boldsymbol{e}}$-partial complete with $\hat{\boldsymbol{e}}(c) = \boldsymbol{e}(g^*)$, the worst-case imprecision at c is *entirely* accounted for by a single generator $g^* \sqsubseteq_{\mathcal{C}} c$. For the completeness case, dually, incompleteness at c is certified by incompleteness at some $g \sqsubseteq_{\mathcal{C}} c$, giving a concrete, structurally simple diagnostic target for precision improvements.

Third, the generator set $\mathcal{G}$ typically forms a *parametric family*—singletons $\{n\}$ for sign, pairs $\{a, b\}$ for bounded intervals, sets $\{\ell_s \mid s \in S\}$ picking one concrete representative per abstract element, for allocation-site and trace-to-reachability—which may allow inductive, symbolic, or case-based reasoning in a way arbitrary concrete elements may not.

6.3 Examples

We illustrate Thm. 6 and Cor. 2 on four GDGCs, each exhibiting a different aspect of the reduction.

Example 16 (Bounded intervals: worst-case imprecision of ABS*).* Take the bounded interval GC $\langle \wp(J), \subseteq \rangle \underset{\alpha_{\mathsf{Int}}}{\overset{\gamma_{\mathsf{Int}}}{\leftrightarrows}} \langle \mathsf{Int}_J, \sqsubseteq_{\mathsf{Int}} \rangle$ of Ex. 3, equipped with the counting distance $\delta^{\sim}_{\mathsf{Int}}$ of Ex. 15, and the absolute-value program

$$\texttt{ABS} \;:\; \texttt{if}\ x \geq 0\ \texttt{then}\ x := x\ \texttt{else}\ x := -x$$

with its standard collecting denotational semantics $[\![\texttt{ABS}]\!] \colon \wp(\mathbb{Z}) \to \wp(\mathbb{Z})$ collecting the outputs reachable from a set of inputs, and the standard abstract interval semantics $[\![\texttt{ABS}]\!]^\sharp_{\mathsf{Int}}$ (see, e.g., Chapter 4.5 of [41]). We have seen in Ex. 3 that the generator of every interval $[a, b] \in \mathsf{Int}_J$ is the pair $\{a, b\} \subseteq J$.

At the generator $\{a, b\}$ with $a < 0 < b$, direct computation yields

$$\begin{aligned} \alpha_{\mathsf{Int}}([\![\texttt{ABS}]\!]\{a, b\}) &= [\mathit{min}(|a|, b),\ \mathit{max}(|a|, b)] \\ [\![\texttt{ABS}]\!]^\sharp_{\mathsf{Int}}(\alpha_{\mathsf{Int}}(\{a, b\})) &= [0,\ \mathit{max}(|a|, b)], \end{aligned}$$

so $\delta^{\sim}_{\mathsf{Int}}(\alpha_{\mathsf{Int}}([\![\texttt{ABS}]\!]\{a, b\}), [\![\texttt{ABS}]\!]^\sharp_{\mathsf{Int}}([a, b])) = \mathit{min}(|a|, b)$. Setting $\boldsymbol{e}(\{a, b\}) = \mathit{min}(|a|, b)$ on generators with opposite signs (i.e., $a < 0 < b$) and $\boldsymbol{e} \equiv 0$ on sign-uniform ones yields $\boldsymbol{e}$-partial local completeness of $[\![\texttt{ABS}]\!]^\sharp_{\mathsf{Int}}$ on $\mathcal{G}$. By Thm. 6, since the GC is generator-dense, for every $S \in \wp(J)$ with $\alpha_{\mathsf{Int}}(S) = [a, b]$,

$$\delta^{\sim}_{\mathsf{Int}}(\alpha_{\mathsf{Int}}([\![\texttt{ABS}]\!]S),\ [\![\texttt{ABS}]\!]^\sharp_{\mathsf{Int}}(\alpha_{\mathsf{Int}}(S))) \leq \hat{\boldsymbol{e}}(S) = \boldsymbol{e}(\{a, b\})\,.$$

The worst-case imprecision of $[\![\texttt{ABS}]\!]^\sharp_{\mathsf{Int}}$ over every concrete input S is therefore controlled by a single pair of extreme values $\{a, b\} \subseteq S$: testing two integers per interval fiber certifies the bound over the entire $\wp(J)$. ♦

Example 17 (Trace to reachability: per-trace completeness). Consider the elementwise trace-to-last-state GC $\langle \wp(\Sigma^{+\infty}), \subseteq \rangle \underset{\alpha_\eta}{\overset{\gamma_\eta}{\leftrightarrows}} \langle \wp(\Sigma), \subseteq \rangle$ via $\eta(\sigma) = \sigma_\omega$ on finite traces (cf. Ex. 5 and Def. 7). Given $R \in \wp(\Sigma)$, its generators are the sets having the form $\{\sigma \mid \sigma_\omega \in R\}$ consisting of exactly one finite trace ending in each reached state $s \in R$; in particular, singleton abstract states $R = \{s\}$ admit singleton trace sets $\{\sigma\}$ as their generators. By Cor. 2, an abstract semantics $[\![\mathtt{P}]\!]^\sharp_{\wp(\Sigma)}$ is complete iff it is locally complete on every such one-trace-per-state generator: whole-set precision reduces to per-generator (indeed per-trace when $|R| = 1$) precision. This principle mirrors the idea underlying *trace partitioning* [42]: by splitting a set of traces along the partition induced by final states, completeness of the analyzer factors through completeness on each piece of the partition. ⧫

Example 18 (Allocation-site: precision at generators, imprecision at merging). Consider the allocation-site GI of Ex. 6, elementwise via $\eta : \mathrm{Loc} \to \mathrm{AllocSite}$. Its generators are the sets $\{\ell_s \mid s \in S\}$ consisting of exactly one concrete location per abstract allocation site $s \in S \subseteq \mathrm{AllocSite}$: configurations on which the abstraction η is injective when restricted to the generator, namely no two distinct concrete locations inside the generator share an allocation site. By Cor. 2, a sound abstract transfer function is complete on the entire $\wp(\mathrm{Loc})$ iff it is exact on every such generator. This formalizes the intuition that imprecision in this abstraction arises from *merging*—distinct concrete locations being abstracted to the same allocation site—rather than from configurations on which η is injective: such configurations already determine global precision, so an analyzer that handles them exactly is automatically precise. Partial completeness refines the picture via Thm. 6: given any imprecision measure of interest δ, for every concrete location set $L \subseteq \mathrm{Loc}$, the imprecision of the transfer function on L is bounded by $\hat{\boldsymbol{e}}(L) = \inf\{\boldsymbol{e}(g) \mid g \in \mathscr{G}(\alpha_\eta(L))_{\sqsubseteq L}\}$, i.e., by the tightest $\boldsymbol{e}$-bound achievable on a generator of $\alpha_\eta(L)$ contained in L. Generators with large $\boldsymbol{e}$-values pinpoint the configurations on which the analyzer introduces spurious allocation sites, providing targets for selective refinement (e.g., by allocation-site splitting or context sensitivity). The same conclusions hold for the points-to map abstraction $\langle \mathrm{Var} \to \wp(\mathrm{Loc}), \sqsubseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \mathrm{Var} \to \wp(\mathrm{AllocSite}), \sqsubseteq \rangle$ of Ex. 12: checking partial completeness at the maps assigning to each variable exactly one concrete location per allocation site certifies it on every points-to map. ⧫

Example 19. Let us consider again $\langle \wp(\wp(\Sigma \times \Sigma)), \subseteq \rangle \underset{\alpha_{\rightsquigarrow}}{\overset{\gamma_{\rightsquigarrow}}{\leftrightarrows}} \langle \wp(\mathbb{X} \times \mathbb{X}), \supseteq \rangle$ of Ex. 7 and let $\mathbb{X} = \{x, y\}$ and $\Sigma = \{\mathrm{T}, \mathrm{F}\}^{\mathbb{X}}$. We equip $\wp(\mathbb{X} \times \mathbb{X})$ with the size distance $\delta^{siz}_{\wp(\mathbb{X}\times\mathbb{X})}$ of Ex. 14. Given the program

$$\texttt{SWAP} \ : \ \texttt{if}\ x = \mathrm{T}\ \texttt{then}\ y := \mathrm{F}\ \texttt{else}\ y := \mathrm{T}$$

we define its collecting semantics $[\![\texttt{SWAP}]\!] \colon \wp(\wp(\Sigma \times \Sigma)) \to \wp(\wp(\Sigma \times \Sigma))$ as $[\![\texttt{SWAP}]\!]S \overset{\text{def}}{=} \{\{(s_0, s_\omega[y \leftarrow \neg s_\omega(x)]) \mid (s_0, s_\omega) \in T\} \mid T \in S\}$, and its abstract semantics $[\![\texttt{SWAP}]\!]^\sharp \colon \wp(\mathbb{X} \times \mathbb{X}) \to \wp(\mathbb{X} \times \mathbb{X})$ as $[\![\texttt{SWAP}]\!]^\sharp D \overset{\text{def}}{=} D \smallsetminus \{(y, y), (x, y)\}$ which abstracts away control dependencies. Recall from Ex. 7 that any generator $G \in \mathcal{G}$ is a minimal hitting set for the collections of input-output behaviors excluding

dependencies that must *not* hold. Given any generator $G \in \mathcal{G}$, by definition of $\alpha_{\rightsquigarrow}$ and $[\![\texttt{SWAP}]\!]^\sharp$, we have $(y,y) \notin [\![\texttt{SWAP}]\!]^\sharp \alpha_{\rightsquigarrow}(G)$ and $(x,y) \notin [\![\texttt{SWAP}]\!]^\sharp \alpha_{\rightsquigarrow}(G)$. Moreover, by the definition of $\alpha_{\rightsquigarrow} \circ [\![\texttt{SWAP}]\!]$, we have $(y,y) \notin \alpha_{\rightsquigarrow}([\![\texttt{SWAP}]\!]G)$ but $(x,y) \in \alpha_{\rightsquigarrow}([\![\texttt{SWAP}]\!]G)$. Thus, $\delta^{siz}_{\wp(\mathbb{X}\times\mathbb{X})}(\alpha_{\rightsquigarrow}([\![\texttt{SWAP}]\!]G), [\![\texttt{SWAP}]\!]^\sharp \alpha_{\rightsquigarrow}(G)) \leq \boldsymbol{e}(G)$, for $\boldsymbol{e} \equiv 1$ and for any $G \in \mathcal{G}$, meaning that $[\![\texttt{SWAP}]\!]^\sharp$ is $\boldsymbol{e}$-partial local complete at $\mathcal{G}$. Hence, thanks to Thm. 6, $[\![\texttt{SWAP}]\!]^\sharp$ is also $\boldsymbol{e}$-partial complete for any $S \in \wp(\wp(\Sigma \times \Sigma))$. This is the GDGC reduction at work: an $\boldsymbol{e}$-partial local completeness at generators lifts automatically to $\boldsymbol{e}$-partial completeness. ♦

7 Related Work

Generators in abstract interpretation. The notion of generator of an abstract property was introduced in [11] to support an inductive proof system for partial completeness whose judgments $\boldsymbol{e}$-$\textsc{Bound}(\mathtt{P}, g)_{\mathcal{A}}$ are parametrized by a single generator g. Thm. 6 is the ingredient that lifts such per-generator judgments to a *global* partial-completeness guarantee, provided the underlying GC is generator-dense. In this sense, GDGC is precisely the structural hypothesis that makes generator-parametric reasoning globally sound.

Completeness, local completeness, partial completeness. Completeness (Def. 3) was introduced by Cousot and Cousot [20] and further studied by Giacobazzi et al. in [31], who showed that it is inherently hard to achieve [27]. A line of recent work has therefore weakened it: Bruni et al. develop *local completeness* and the associated Local Completeness Logic (LCL) for program verification [6,7]; Campion et al. introduce *partial completeness* through order-compatible pre-metrics [14,13], used here in Def. 8; and [11] combines the two into a per-generator partial-completeness proof system for imperative programs. Our main result (Thm. 6) sits at the structural level beneath these efforts.

Constructive Galois connections. Darais and Van Horn [24] introduced *constructive Galois connections* (CGCs) as a structural construction principle for GCs over a powerset concrete domain: any function $\eta : X \to \mathcal{A}$ extends additively to a GC $\langle \wp(X), \subseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \mathcal{A}, \sqsubseteq_{\mathcal{A}} \rangle$ via $\alpha(S) = \bigvee_{\mathcal{A}} \{\eta(x) \mid x \in S\}$. Conversely, every join-preserving abstraction admits the canonical decomposition $\alpha(S) = \bigvee_{\mathcal{A}} \{\alpha(\{x\}) \mid x \in S\}$, so every GC with a powerset concrete domain is automatically a CGC, with $\eta(x) = \alpha(\{x\})$—the constructive structure comes for free in the powerset setting. A more restrictive sub-case studied in their Sec. 8.2 is the *elementwise abstraction*: the CGC where $\mathcal{A}$ is itself a powerset $\wp(Y)$ and $\eta : X \to Y$ maps into the underlying set Y (rather than into $\wp(Y)$), so that $\alpha(S) = \{\eta(x) \mid x \in S\}$. This is exactly the elementwise GC of Def. 7.

Generator-density and the CGC condition live in different worlds and are independent. CGC is a structural hypothesis on how an abstraction is built (element-by-element extraction from a powerset); generator-density is a fiber-level hypothesis on the existence of minimal witnesses inside $\alpha^{-1}(a)$. For instance, the interval GC over $\mathbb{Z} \cup \{\pm\infty\}$ has a powerset concrete domain, hence is a CGC, but fails generator-density (Ex. 1); conversely, any DCC concrete domain that

is not a powerset yields a generator-dense GC (Thm. 4) for which the CGC notion is undefined. The two notions collapse in the elementwise sub-case: every elementwise GC is generator-dense (resp. constructive), and arbitrary pipelines of elementwise GCs followed by any GDGC (resp. CGC) remain generator-dense (resp. constructive) (Thm. 5, Cor. 1).

Dense sets and fiber minimality. The order-theoretic notion of a dense (or coinitial) subset of a poset goes back at least to Abian [1] and was systematically developed by Höft [34]. Generator-density (Def. 6) instantiates this condition fiber-by-fiber on the abstraction map α. Similar structural conditions appear in the study of atomic lattices and algebraic lattices in domain theory [2,32]. The fiber-local perspective also isolates the source of non-density: as illustrated by the unbounded intervals, octagons [40], and semantic properties of traces, non-density is caused by infinite strictly descending chains within a single fiber.

8 Conclusion

We introduced the property *generator-density* of a GC as the structural property that turns the generators of an abstract domain into a "basis" of the concrete one: every imprecision an abstract interpreter incurs is already manifest on a structurally simple, generator-shaped witness. The methodological payoff is twofold: not only a reduction of the (partial) completeness check from the full concrete domain to the generator set alone, with imprecision bounds propagating automatically, but also a localization of the origin of imprecision at specific generators below the input—simple targets to focus precision-improving refinements on.

A distinctive feature of generator-density is that it is a property of the *concrete* domain—of how concrete elements distribute inside the fibers of α—rather than of the abstract domain. This sets it apart from most precision-preserving constructions in abstract interpretation, which refine the *abstract* side: the complete shell, complete kernel [31], disjunctive completion, and reduced product all extend an unsuitable abstract domain into one with the desired property. The concrete, by contrast, is typically fixed by the semantics being analyzed, so repairing it is less straightforward than repairing an abstract domain. It is, however, not immutable: static analyzers routinely adopt a more restrictive concrete semantics for soundness reasons—excluding, e.g., non-memory-safe executions or the use of reflection—and such restrictions shrink the fibers of α, which is precisely what can remove the infinite descending chains obstructing generator-density. This raises a natural open direction: when can a non-GDGC concrete domain be *restricted* or *extended* into a GDGC one? Possible routes include excluding behaviors, adjoining synthetic minima to fibers that lack them, fiberwise completions in the spirit of the constructions of [31] but applied on the concrete side, or completions analogous to the generator representation of convex polyhedra given by the Weyl–Minkowski theorem [23]. And what do such transformations represent semantically and topologically?

GDGCs form a stricter class than general GCs, and hence than many of those used in practice. This is the price of the reduction: Thm. 6 replaces a check over

the entire concrete domain by one on the generators alone, and a reduction this strong cannot be expected to hold for every GC. Generator-density does hold, however, for some GCs of practical interest—bounded intervals over a finite set of machine-representable values, the domain underlying interval bound propagation in neural-network verification [26,33], and the sign domain—as well as for the dependency and allocation-site abstractions, for which, to the best of our knowledge, no analogous reduction had been observed. When one wishes to abstract such domains further, Sec. 5 shows how generator-density can be preserved. One way to widen the scope is to relax the density requirement itself: a *partial* generator-density, admitting some (but not all) concrete properties lying above no generator, would keep Thm. 6 applicable on the subset of concrete properties that do cover a generator, weakening its conclusion from global partial completeness to partial local completeness on that subset. Beyond GCs, frameworks lacking a best abstraction—convex polyhedra being the standard example—could be accommodated by admitting a partial α, or via the weak-closure operators of the Galois-connection-less framework of [37].

Several other directions remain open. What does generator-density buy in the verification of assertions, contracts, or pre/post-conditions encoded as abstract elements? Cor. 2 suggests that, under GDGC, certifying an assertion at every generator is equivalent to certifying it on every concrete state, so that a check over arbitrary concrete states is replaced by one over structurally simple, canonical witnesses. But what can be said when a specification is *not* satisfied? A deeper investigation is needed in this direction.

We showed that the standard product preserves generator-density, but whether the reduced product, the disjunctive completion, or the open product of [16] preserves it is open and deserves a future investigation.

A further direction concerns *code obfuscation*, which can be read as the deliberate injection of incompleteness into a static analyzer [28,29], and whose effectiveness against abstract interpretation-based analyzers has recently been assessed empirically [3]. Under generator-density, Thm. 6 localizes the imprecision incurred at any concrete input at a generator below it: generators are therefore the natural target of such transformations, and a natural benchmark for measuring obfuscation potency.

A complementary, program-side direction is to study the class of programs that map generators to generators. When this property holds, the imprecision bound of Thm. 6 propagates through computation while remaining itself generator-based, suggesting an inductive characterization of partial completeness restricted to generator-preserving programs. Monotone programs [10] provide a natural setting where this preservation holds, and a systematic study of generator-preservation in connection with that work appears promising.

Acknowledgments. This work was supported by the SAIF project, managed by the ANR for France 2030 under the reference ANR-23-PEIA-0006; by the ForML project, funded by the ANR under reference ANR-23-CE25-0009; by the SecurEval project, managed by the ANR for France 2030 under reference ANR-22-PECY-0005; and by the Air Force Office of Scientific Research under award number FA9550-23-1-0544.

References

1. Abian, A.: Molecules of a partially ordered set. Algebra universalis **12**, 258–261 (1981). https://doi.org/10.1007/BF02483885
2. Abramsky, S., Jung, A.: Domain Theory, vol. 3. Oxford University Press (1994)
3. Altamura, N., Bragastini, E., Campion, M., Preda, M.D.: Assessing the effectiveness of the tigress obfuscator against mopsa and binaryninja. In: Proceedings of the 2025 Workshop on Research on Offensive and Defensive Techniques in the Context of Man At The End (MATE) Attacks. p. 29–37. CheckMATE '25, Association for Computing Machinery, New York, NY, USA (2025). https://doi.org/10.1145/3733817.3762702
4. Andersen, L.O.: Program analysis and specialization for the C programming language. PhD Thesis, University of Copenhagen (1994)
5. Bruni, R., Giacobazzi, R., Gori, R., Garcia-Contreras, I., Pavlovic, D.: Abstract extensionality: on the properties of incomplete abstract interpretations. Proc. ACM Program. Lang. **4**(POPL), 28:1–28:28 (2020). https://doi.org/10.1145/3371096
6. Bruni, R., Giacobazzi, R., Gori, R., Ranzato, F.: A logic for locally complete abstract interpretations. In: 36th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2021, Rome, Italy, June 29 - July 2, 2021. pp. 1–13. IEEE (2021). https://doi.org/10.1109/LICS52264.2021.9470608
7. Bruni, R., Giacobazzi, R., Gori, R., Ranzato, F.: A correctness and incorrectness program logic. J. ACM **70**(2), 15:1–15:45 (2023). https://doi.org/10.1145/3582267
8. 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
9. Campion, M., Dalla Preda, M., Giacobazzi, R., Miné, A., Urban, C.: Generator-Dense Galois Connections (2026), https://inria.hal.science/hal-05703393
10. Campion, M., Dalla Preda, M., Giacobazzi, R., Urban, C.: Monotonicity and the precision of program analysis. Proc. ACM Program. Lang. **8**(POPL), 1629–1662 (2024). https://doi.org/10.1145/3632897
11. Campion, M., Dalla Preda, M., Giacobazzi, R., Urban, C.: A logic for the imprecision of abstract interpretations. Proc. ACM Program. Lang. **10**(POPL), 1876–1904 (2026). https://doi.org/10.1145/3776707
12. Campion, M., Mastroeni, I., Pasqua, M., Urban, C.: Abstract lipschitz continuity: Combining semantic and quantitative approximations. In: Bertrand, N., Milius, S. (eds.) Foundations of Software Science and Computation Structures - 29th International Conference, FoSSaCS 2026, Held as Part of the International Joint Conferences on Theory and Practice of Software, ETAPS 2026, Turin, Italy, April 11-16, 2026, Proceedings. pp. 153–177. Lecture Notes in Computer Science, Springer (2026). https://doi.org/10.1007/978-3-032-22730-0_8
13. Campion, M., Mastroeni, I., Urban, C.: Relating distances and abstractions: An abstract interpretation perspective. In: Oh, H., Sui, Y. (eds.) Static Analysis - 32th International Symposium, SAS 2025, Singapore, October 13-14, 2025, Proceedings. Lecture Notes in Computer Science, vol. 16100, pp. 249–277. Springer (2025). https://doi.org/10.1007/978-3-032-07106-4_11
14. Campion, M., Urban, C., Dalla Preda, M., Giacobazzi, R.: A formal framework to measure the incompleteness of abstract interpretations. In: Hermenegildo, M.V., Morales, J.F. (eds.) Static Analysis - 30th International Symposium, SAS 2023, Cascais, Portugal, October 22-24, 2023, Proceedings. Lecture Notes in Computer

Science, vol. 14284, pp. 114–138. Springer (2023). https://doi.org/10.1007/978-3-031-44245-2_7
15. Casso, I., Morales, J.F., López-García, P., Giacobazzi, R., Hermenegildo, M.V.: Computing abstract distances in logic programs. In: International Symposium on Logic-Based Program Synthesis and Transformation. pp. 57–72. Springer (2019). https://doi.org/10.1007/978-3-030-45260-5_4
16. Cortesi, A., Charlier, B.L., Hentenryck, P.V.: Combinations of abstract domains for logic programming: open product and generic pattern construction. Sci. Comput. Program. **38**(1-3), 27–71 (2000). https://doi.org/10.1016/S0167-6423(99)00045-3
17. Cousot, P.: Abstract semantic dependency. In: Chang, B.E. (ed.) Static Analysis - 26th International Symposium, SAS 2019, Porto, Portugal, October 8-11, 2019, Proceedings. pp. 389–410. Lecture Notes in Computer Science, Springer (2019). https://doi.org/10.1007/978-3-030-32304-2_19
18. Cousot, P.: Principles of Abstract Interpretation. The MIT Press, Cambridge, Mass. (2021)
19. Cousot, P., Cousot, R.: Static determination of dynamic properties of programs. In: Proceedings of the 2nd International Symposium on Programming. pp. 106–130. Dunod, Paris (1976). https://doi.org/10.1145/390019.808314
20. Cousot, P., Cousot, R.: Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In: Graham, R.M., Harrison, M.A., Sethi, R. (eds.) Proceedings of the 4th ACM Symposium on Principles of Programming Languages, Los Angeles, California, USA, January 1977. pp. 238–252. ACM (1977). https://doi.org/10.1145/512950.512973
21. Cousot, P., Cousot, R.: Systematic design of program analysis frameworks. In: Aho, A.V., Zilles, S.N., Rosen, B.K. (eds.) Proceedings of the 6th ACM Symposium on Principles of Programming Languages, San Antonio, Texas, USA, January 1979. pp. 269–282. ACM Press (1979). https://doi.org/10.1145/567752.567778
22. Cousot, P., Cousot, R., Mauborgne, L.: The reduced product of abstract domains and the combination of decision procedures. In: Hofmann, M. (ed.) Foundations of Software Science and Computational Structures - 14th International Conference, FOSSACS 2011, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2011, Saarbrücken, Germany, March 26-April 3, 2011. Proceedings. pp. 456–472. Lecture Notes in Computer Science, Springer (2011). https://doi.org/10.1007/978-3-642-19805-2_31
23. Cousot, P., Halbwachs, N.: Automatic discovery of linear restraints among variables of a program. In: Aho, A.V., Zilles, S.N., Szymanski, T.G. (eds.) Conference Record of the Fifth Annual ACM Symposium on Principles of Programming Languages, Tucson, Arizona, USA, January 1978. pp. 84–96. ACM Press (1978). https://doi.org/10.1145/512760.512770
24. Darais, D., Horn, D.V.: Constructive galois connections. J. Funct. Program. **29**, e11 (2019). https://doi.org/10.1017/S0956796819000066
25. Di Pierro, A., Wiklicky, H.: Measuring the precision of abstract interpretations. In: Lau, K. (ed.) Logic Based Program Synthesis and Transformation, 10th International Workshop, LOPSTR 2000 London, UK, July 24-28, 2000, Selected Papers. Lecture Notes in Computer Science, vol. 2042, pp. 147–164. Springer (2000). https://doi.org/10.1007/3-540-45142-0_9
26. Gehr, T., Mirman, M., Drachsler-Cohen, D., Tsankov, P., Chaudhuri, S., Vechev, M.T.: AI2: safety and robustness certification of neural networks with abstract

interpretation. In: 2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA. pp. 3–18. IEEE Computer Society (2018). https://doi.org/10.1109/SP.2018.00058
27. Giacobazzi, R., Logozzo, F., Ranzato, F.: Analyzing program analyses. In: Rajamani, S.K., Walker, D. (eds.) Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2015, Mumbai, India, January 15-17, 2015. pp. 261–273. ACM (2015). https://doi.org/10.1145/2676726.2676987
28. Giacobazzi, R., Mastroeni, I.: Making abstract interpretation incomplete: Modeling the potency of obfuscation. In: Miné, A., Schmidt, D. (eds.) Static Analysis - 19th International Symposium, SAS 2012, Deauville, France, September 11-13, 2012. Proceedings. Lecture Notes in Computer Science, vol. 7460, pp. 129–145. Springer (2012). https://doi.org/10.1007/978-3-642-33125-1_11
29. Giacobazzi, R., Mastroeni, I., Dalla Preda, M.: Maximal incompleteness as obfuscation potency. Formal Aspects Comput. **29**(1), 3–31 (2017). https://doi.org/10.1007/s00165-016-0374-2
30. Giacobazzi, R., Quintarelli, E.: Incompleteness, counterexamples, and refinements in abstract model-checking. In: Cousot, P. (ed.) Static Analysis, 8th International Symposium, SAS 2001, Paris, France, July 16-18, 2001, Proceedings. Lecture Notes in Computer Science, vol. 2126, pp. 356–373. Springer (2001). https://doi.org/10.1007/3-540-47764-0_20
31. Giacobazzi, R., Ranzato, F., Scozzari, F.: Making abstract interpretations complete. J. ACM **47**(2), 361–416 (2000). https://doi.org/10.1145/333979.333989
32. Gierz, G., Hofmann, K.H., Keimel, K., Lawson, J.D., Mislove, M., Scott, D.S.: Continuous Lattices and Domains. Cambridge University Press (2003)
33. Gowal, S., Dvijotham, K., Stanforth, R., Bunel, R., Qin, C., Uesato, J., Arandjelovic, R., Mann, T.A., Kohli, P.: Scalable verified training for provably robust image classification. In: 2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019. pp. 4841–4850. IEEE (2019). https://doi.org/10.1109/ICCV.2019.00494, https://doi.org/10.1109/ICCV.2019.00494
34. Höft, H.F., Höft, M.H.: Coinitial sets and fixed points in partially ordered sets. Algebra universalis **16**, 258–260 (1983). https://doi.org/10.1007/BF01191777
35. Liew, D., Cogumbreiro, T., Lange, J.: Sound and partially-complete static analysis of data-races in GPU programs. Proc. ACM Program. Lang. **8**(OOPSLA2), 2434–2461 (2024). https://doi.org/10.1145/3689797
36. Logozzo, F.: Towards a quantitative estimation of abstract interpretations. In: Workshop on Quantitative Analysis of Software. Microsoft (June 2009), https://www.microsoft.com/en-us/research/publication/towards-a-quantitative-estimation-of-abstract-interpretations/
37. Mastroeni, I., Pasqua, M.: Domain precision in galois connection-less abstract interpretation. In: Hermenegildo, M.V., Morales, J.F. (eds.) Static Analysis - 30th International Symposium, SAS 2023, Cascais, Portugal, October 22-24, 2023, Proceedings. Lecture Notes in Computer Science, vol. 14284, pp. 434–459. Springer (2023). https://doi.org/10.1007/978-3-031-44245-2_19
38. Mazzucato, D., Campion, M., Urban, C.: Quantitative input usage static analysis. In: Benz, N., Gopinath, D., Shi, N. (eds.) NASA Formal Methods - 16th International Symposium, NFM 2024, Moffett Field, CA, USA, June 4-6, 2024, Proceedings. Lecture Notes in Computer Science, vol. 14627, pp. 79–98. Springer (2024). https://doi.org/10.1007/978-3-031-60698-4_5

39. Mazzucato, D., Campion, M., Urban, C.: Quantitative static timing analysis. In: Giacobazzi, R., Gorla, A. (eds.) Static Analysis - 31st International Symposium, SAS 2024, Pasadena, CA, USA, October 20-22, 2024, Proceedings. Lecture Notes in Computer Science, vol. 14995, pp. 268–299. Springer (2024), https://doi.org/10.1007/978-3-031-74776-2_11
40. Miné, A.: The octagon abstract domain. High. Order Symb. Comput. **19**(1), 31–100 (2006). https://doi.org/10.1007/s10990-006-8609-1
41. Miné, A.: Tutorial on static inference of numeric invariants by abstract interpretation. Foundations and Trends in Programming Languages **4**(3-4), 120–372 (2017). https://doi.org/10.1561/2500000034
42. Rival, X., Mauborgne, L.: The trace partitioning abstract domain. ACM Trans. Program. Lang. Syst. **29**(5), 26 (2007). https://doi.org/10.1145/1275497.1275501
43. Sotin, P.: Quantifying the precision of numerical abstract domains. Tech. Rep. HAL Id: inria-00457324, INRIA (2010), https://hal.inria.fr/inria-00457324
44. Urban, C., Müller, P.: An abstract interpretation framework for input data usage. In: Ahmed, A. (ed.) Programming Languages and Systems - 27th European Symposium on Programming, ESOP 2018, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2018, Thessaloniki, Greece, April 14-20, 2018, Proceedings. pp. 683–710. Lecture Notes in Computer Science, Springer (2018). https://doi.org/10.1007/978-3-319-89884-1_24