Abstract Lipschitz Continuity

Combining Semantic and Quantitative Approximations

Marco Campion[1,2], Isabella Mastroeni[3], Michele Pasqua[3], and Caterina Urban[2]

[1] LIP6, Sorbonne Université, Paris, France
marco.campion@lip6.fr
[2] Inria & ENS | PSL, Paris, France
caterina.urban@inria.fr
[3] University of Verona, Department of Computer Science, Verona, Italy
{isabella.mastroeni,michele.pasqua}@univr.it

Abstract. We introduce *Abstract Lipschitz Continuity* (ALC), an extensional (i.e., input/output) property that ensures proportionally bounded differences in the *semantic approximations* of the output of a function (e.g., a program semantics) when the *semantic approximations* of the input differ slightly. ALC explicitly discerns between two complementary notions of approximation: *quantitative* differences, expressed via pre-metrics, and *qualitative* (or *semantic*) differences, captured through upper closure operators. This explicit separation of approximations has two main advantages. First, it enables ALC to be related to other important extensional *program properties*, including partial abstract non-interference in language-based security, partial completeness in abstract interpretation, and abstract robustness in machine learning. Second, ALC enables reasoning about its validity for programs through inductive reasoning on their syntax and on the chosen semantic abstractions. To this end, we propose a sound deductive system, parameterized by the quantitative and semantic approximations of interest, for proving ALC of programs. This proof system makes explicit the assumptions required for ALC, thereby ensuring a compositional proof approach.

Keywords: Abstract Lipschitz Continuity, Abstract Interpretation, Partial Abstract Non-Interference, Partial Completeness, Abstract Robustness

1 Introduction

Lipschitz continuity (LC) is a fundamental property in calculus and computational analysis, providing a strong guarantee: A Lipschitz continuous function is one whose output changes at most linearly with respect to changes in its input. Its utility is wide-ranging, serving as a cornerstone for convergence guarantees in optimization [46] and a key tool for analyzing robustness and stability in machine learning and program analysis, particularly in adversarial settings [21,30,31,56].

When reasoning about software programs, LC provides an essential measure of *program robustness* for code operating on uncertain input [10,11,12].

The standard definition of LC elegantly captures variations through metrics or other weaker forms of distances on the raw input and output spaces. In this work, we argue that making the *semantic dimension* of data explicit provides a natural and complementary perspective. Indeed, in complex systems such as machine learning models or programs, small changes in the *syntactic representation* of data (e.g., quantitative differences such as raw bit flips) may correspond to negligible or irrelevant *semantic differences*. By reasoning directly in terms of semantic properties, we can capture robustness and stability in a way that more closely aligns with the intended behavior of such systems. This enriched view of LC opens new possibilities for applications in areas ranging from program verification [16,25,52,53] to machine learning robustness [1,26,32].

For instance, in the context of security, there are several areas of application, such as code/SQL injection or file integrity, that motivate such explicit discrimination between quantitative and semantic properties of data. More concretely, SQL injection vulnerabilities arise from an unchecked interaction between untrusted input and the execution of a SQL query, allowing an attacker to execute arbitrary queries on the database. A trivial example is given on the left of Fig. 1 where the attacker is able to dump the whole `users` table by injecting the value `"3 OR 1 = 1"` in the (unsanitized) input `argv[0]`, forcing the query `q` condition into a tautology. In such a scenario, we could require that strings with a *similar* number of boolean operators do not yield a *difference* in the number of selected tuples. In other words, we aim at checking the distance between *properties* of inputs and outputs, rather than between concrete values. As another example, consider the scenario of malware detection, where detection tools typically aim to locate malicious patterns (e.g., signatures or byte sequences) in the files under inspection, or identify pattern presence to pinpoint potential malware infections in the files. A simple example is given by the code on the right of Fig. 1, which counts the occurrences of *"a"*, e.g., representing a malicious pattern, in a file. In this context, we could be interested in an integrity check: small changes in file dimension yield proportionally small changes in *"a"* (the pattern) count range, consistent with controlled, non-malicious file modifications. Dually, a small change in file dimension that yields a large change in the number of patterns, potentially out of the allowed range, may indicate a malicious infection in the file. Thus, we again need to compare distances between properties of data, rather than distances of data. Ordinary LC does not suit these scenarios, as distance metrics (e.g., the Levenshtein string distance) can only model quantitative data variations. One could define a tailored distance function that operates directly on semantic properties; however, this approach would embed the semantic abstraction within the distance calculation itself, making the resulting property less comparable with other established notions in the literature.

We introduce *Abstract Lipschitz Continuity* (ALC for short), a novel property that generalizes the notion of LC to the semantic domain (Sec. 3). ALC formally ensures that *differences in the semantic approximations* of inputs lead

```
<?php

$id = $argv[0];
$q = "SELECT * FROM users WHERE id = $id;";
$result = pg_query($conn, $q);

?>
```

```
int count-a(FILE *F) {
  int c, nc = 0;
  while ((c=fgetc(F)) != EOF) {
    if (c == 'a') nc++;
  }
  return nc;
}
```

Fig. 1: Example of SQL injection (left) and counting pattern program (right).

to *proportionally bounded differences in the semantic approximations* of outputs. In other words, ALC represents a generalized continuity-like requirement that explicitly incorporates semantic approximations of input and output spaces.

Our formalization is grounded in the theory of *abstract interpretation* [14,15]. We thus model *semantic* (or *qualitative*) approximations using upper closure operators, which effectively abstract concrete values (such as a string) to sets of values sharing a common property (e.g., all strings having the same length). ALC is defined by combining these semantic approximations with *quantitative* approximations via a distance function over a pre-metric space. Unlike the definition of LC, the definition of ALC is parametrized by these two forms of approximations.

Thanks to this explicit and parametrized distinction of approximations, we establish a formal correlation between ALC in the context of program semantics, and other foundational program properties (Sec. 4). We show how ALC relates to (*partial*) *completeness* [5,9,27] in abstract interpretation, a property essential for bounding imprecision in static analysis. By connecting quantitative stability to qualitative precision, we provide new insights into the interplay between these domains. For instance, we prove that, by fixing a program semantics of interest, any ALC program admits a *complete* best correct approximation over the abstract domain corresponding to the semantic approximation, when the chosen distance satisfies a structural property. Similarly, we demonstrate the utility of ALC in machine learning by relating it to (*abstract*) *robustness* [26,36], where ALC provides a stronger, quantified measure of semantic stability, generalizing simple qualitative robustness guarantees, but also in language-based security by relating ALC to (*partial*) *abstract non-interference* [8,25]. Overall, these results demonstrate that ALC is a stronger quantitative program property than the notions considered above. Properties such as abstract robustness are weaker as they can capture quantitative behaviors beyond those enforced by continuity-like properties; conversely, ALC excludes functions that satisfy quantitative constraints only in a weaker, non-continuous sense.

Finally, we propose a novel sound *deductive system* for verifying ALC for programs through inductive reasoning on their syntax (Sec. 5). Our system is parametric with respect to the chosen input and output semantic approximations and distance functions. It is designed to make explicit the assumptions required for ALC, thereby ensuring a compositional proof approach.

Related Work. We formally discuss the relationship between ALC and (partial) abstract non-interference [8,28,25] or approximate non-interference [48], (par-

$$\begin{array}{ll} \mathsf{Stm} \ni \mathsf{c} ::= \mathbf{skip} \mid x := \mathsf{a} \mid \mathsf{b}? & [\![\mathsf{P}_1 \,;\, \mathsf{P}_2]\!]c \overset{\text{def}}{=} [\![\mathsf{P}_2]\!] \circ [\![\mathsf{P}_1]\!]c \\ \mathsf{Prog} \ni \mathsf{P} ::= \mathsf{c} \mid \mathsf{P} \,;\, \mathsf{P} \mid \mathsf{P} \oplus \mathsf{P} \mid \mathsf{P}^* & [\![\mathsf{P}_1 \oplus \mathsf{P}_2]\!]c \overset{\text{def}}{=} [\![\mathsf{P}_1]\!]c \vee [\![\mathsf{P}_2]\!]c \\ \mathsf{a} \in \mathsf{AExp},\ x \in \mathbb{X},\ \mathsf{b} \in \mathsf{BExp} & [\![\mathsf{P}^*]\!]c \overset{\text{def}}{=} \bigvee\{[\![\mathsf{P}]\!]^n c \mid n \in \mathbb{N}\} \end{array}$$

Fig. 2: Syntax (left) and semantics (right) of Prog.

tial) completeness [5,9,27], and (abstract) robustness [26,36] in Sec. 4. The idea of software doping, technically introduced in [17], looks for a similar effect as ALC with the aim to detect intentionally developed ill-programs. The authors propose a Lipschitz-style condition for defining software doping which is tailored to the Hausdorff lifting. This is then further generalized via a functional modulus of continuity. Their work, like ours, discerns a property of interest, expressed through the Hausdorff lifting, while, in our approach, we are discerning a semantic property that can be formalized as a closure operator (or Galois connection). Other potential related works are cited in Sec. 6, since a better understanding of the relation with ALC deserves further research.

Our deductive system for verifying ALC in Sec. 5 generalizes and extends the one proposed by Chaudhuri et al. [11,12] for proving program robustness [26]. Specifically, when the semantic approximations are the identity function (namely, no semantic approximation is involved), our system recovers (an extension, due to the corresponding (**star**) rule, of) the deductive system in [11].

2 Background

In the following, we introduce the relevant background on programs, as well as approximations for simplifying reasoning about properties of their semantics.

2.1 Program Syntax and Semantics

We consider programs written in the language Prog of regular commands [4,47], which encompasses deterministic as well as non-deterministic and probabilistic computations. The syntax of Prog is on the left of Fig. 2, where $\oplus$ denotes non-deterministic choice and * is the Kleene closure. For our purposes, we instantiate the basic commands $\mathsf{c} \in \mathsf{Stm}$ with **skip**, variable assignments, and Boolean tests. We assume a standard grammar for arithmetic expressions in AExp and Boolean expressions in BExp. Variables range from a denumerable set $\mathbb{X}$ while values range from a denumerable set $\mathbb{V}$ (e.g., integer or natural numbers).

Suppose we have a semantics $[\![\mathsf{c}]\!] \colon C \to C$ for basic commands $\mathsf{c} \in \mathsf{Stm}$ on a complete lattice $\langle C, \preceq, \vee, \wedge, \top, \bot \rangle$, where $\preceq$ is the partial order, $\vee$ is the least upper bound, $\wedge$ the greatest lower bound, $\top$ is the supremum of C and $\bot$ is the infimum of C. Then, the *semantics for programs* $[\![\cdot]\!] \colon \mathsf{Prog} \to C \to C$ is inductively defined on program syntax as on the right of Fig 2.

A notable instance of such concrete semantics is the *collecting semantics*, central in abstract interpretation [14,15], expressing the set of all possible program states that could occur at each program point. Formally, consider a complete lattice $\langle \wp(\mathbb{M}), \subseteq, \cup, \cap, \mathbb{M}, \varnothing \rangle$ of program memories, where $\mathtt{m} \in \mathbb{M}$ maps variables to values, namely $\mathtt{m} \colon \mathbb{X} \to \mathbb{V}$. We can define a collecting big-step semantics $[\![\mathsf{P}]\!] \colon \wp(\mathbb{M}) \to \wp(\mathbb{M})$ for a program $\mathsf{P} \in \mathsf{Prog}$ as the standard predicate transformer semantics on sets of program memories $\wp(\mathbb{M})$, collecting all possible program memories that reach the end of the program. Assume a big-step evaluation semantics $\Downarrow_{\mathsf{a}}$ for arithmetic expressions and $\Downarrow_{\mathsf{b}}$ for Boolean expressions. Given $S \in \wp(\mathbb{M})$, the semantics of basic commands is defined as:

$$[\![\mathbf{skip}]\!]S \overset{\text{def}}{=} S \quad [\![x := \mathsf{a}]\!]S \overset{\text{def}}{=} \{\mathtt{m}[x \hookleftarrow v] \mid \mathtt{m} \in S \wedge \mathtt{m} \Downarrow_{\mathsf{a}} v\} \quad [\![\mathsf{b}?]\!]S \overset{\text{def}}{=} \{\mathtt{m} \in S \mid \mathtt{m} \Downarrow_{\mathsf{b}} \mathtt{tt}\}$$

The collecting semantics for basic commands is monotone by construction on the powerset lattice of program memories, and so $[\![\mathsf{P}]\!]$ is also monotone for any program in Prog. We will extensively use this semantics in the examples of Sec. 4, 5.

2.2 Abstractions and Distances

In program analysis approximations are fundamental for simplifying reasoning while preserving essential properties. Following [8], we consider *qualitative* or *semantic* approximations, formalized by *upper closure operators*, and *quantitative* approximations, formalized by *pre-metrics.* Their combination yields a general approximation parametrized by both an abstraction and a distance.

Semantic Approximations via Upper Closure Operators. Qualitative or semantic approximations preserve certain *semantic properties* of the approximated data. Semantics approximations are at the hearth of abstract interpretation for approximating computations by evaluating functions (e.g., program semantics) over an abstract domain—a partially ordered set (poset, for short)—$\langle A, \preceq_{\mathsf{A}} \rangle$ instead of the concrete domain $\langle C, \preceq_{\mathsf{C}} \rangle$. A (monotone) concretization function $\gamma \colon A \to C$ relates abstract elements to their concrete counterparts, preserving the (partial) ordering of information. When paired with a (monotone) abstraction function $\alpha \colon C \to A$ such that $\alpha(c) \preceq_{\mathsf{A}} a \Leftrightarrow c \preceq_{\mathsf{C}} \gamma(a)$, $\forall a \in A$ and $\forall c \in C$, the pair forms a *Galois Connection* (GC) between the two domains. A GC is a *Galois Insertion* (GI) when $\alpha \circ \gamma = id$, where $id \overset{\text{def}}{=} \lambda x.\, x$.

Given a concrete domain $\langle C, \preceq_{\mathsf{C}} \rangle$, GIs can be equivalently formulated in terms of *upper closure operators* [15] (ucos or closures, for short), namely a function $\rho \colon C \to C$ with the following properties $\forall c, c' \in C$: monotonicity ($c \preceq_{\mathsf{C}} c' \Rightarrow \rho(c) \preceq_{\mathsf{C}} \rho(c')$), extensivity ($c \preceq_{\mathsf{C}} \rho(c)$), idempotence ($\rho(\rho(c)) = \rho(c)$). Ucos are uniquely determined by the set of their fixpoints: $\rho(C) = \{c \in C \mid \rho(c) = c\}$. For instance, the composition $\gamma \circ \alpha$ is an uco of C. In the following, we will often write ρf for the composition of any two functions $\rho \circ f$. The set of all ucos on a poset C is denoted by $uco(C)$. As an example, the closure $\mathsf{Sign} \in uco(\wp(\mathbb{Z}))$ abstracts a set of integers by discarding all information except the sign of its values, unless the set contains only the value 0. The closure is defined by the set of fixpoints $\mathsf{Sign}(\wp(\mathbb{Z})) \overset{\text{def}}{=} \{\varnothing, \{0\}, \{z \in \mathbb{Z} \mid z \leq 0\}, \{z \in \mathbb{Z} \mid z \geq 0\}, \mathbb{Z}\}$.

	pre-	**quasipseudo-**	**quasisemi-**	**semi-**	**quasi-**	**pseudo-**	**metric**
(if-identity)	✓	✓	✓	✓	✓	✓	✓
(iff-identity)	✗	✗	✓	✓	✓	✗	✓
(symmetry)	✗	✗	✗	✓	✗	✓	✓
(triangle-inequality)	✗	✓	✗	✗	✓	✓	✓
Example		$\delta_{\subseteq}^{\mathsf{Int}}$			$\delta_{\subseteq}$	$\delta_{siz}, \delta_{\mathrm{DIM}}, \delta_{-}$	δ_2
Reference		Ex. 5			Ex. 3	Ex. 2, 6, 11	

Fig. 3: Metrics and their weakenings.

Definition 1 (Semantic Approximation [8]). *Given a poset $\langle C, \preceq_{\mathsf{C}} \rangle$ and an uco $\rho \in uco(C)$, an element $x \in C$ is* semantically approximated *by $\rho(x)$.*

Example 1. Let $\mathsf{Int} \in uco(\wp(\mathbb{Z}))$ be the interval abstraction [13], mapping a set of integers $S \in \wp(\mathbb{Z})$ to the smallest interval $[l, u] \overset{\text{def}}{=} \{i \in \mathbb{Z} \mid l \leq i \leq u\}$ such that $S \subseteq [l, u]$, where $l \in \mathbb{Z} \cup \{-\infty\}$, $u \in \mathbb{Z} \cup \{+\infty\}$ and $l \leq u$. The set of integers $\{0, 1, 4\}$ can be semantically approximated by the interval $[0, 4]$ through Int. Moreover, the set $\{\{0,4\}, \{0,1,4\}, \{0,2,4\}, \{0,3,4\}, \{0,1,2,4\}, \{0,2,3,4\}, \{0,1,3,4\}, \{0,1,2,3,4\}\}$ contains all sets of integers S such that $\mathsf{Int}(S) = [0, 4]$. ■

Quantitative Approximations via Pre-Metrics. Quantitative approximations preserve *closeness* of the approximated data, here measured through a distance function between elements of a poset. We model distance functions using (pre-)metrics. Let $\mathbb{R}^{\infty} \overset{\text{def}}{=} \mathbb{R} \cup \{\infty\}$ such that for all $r \in \mathbb{R}$, $r < \infty$, and let $\mathbb{R}_{\geq 0}$ be the restriction of $\mathbb{R}$ to values greater or equal to 0.

Definition 2 (Pre-Metric [9]). *Given a non-empty set L, a* pre-metric *is a binary function $\delta \colon L \times L \to \mathbb{R}^{\infty}_{\geq 0}$ satisfying the only axiom:*

$$\textit{(if-identity)}\ \forall x, y \in L.\ x = y \ \Rightarrow\ \delta(x, y) = 0.$$

The pair $\langle L, \delta \rangle$ is called a pre-metric space.

We will occasionally use the subscript δ_{L} in cases where the set L may not be immediately clear from the context. The same convention will be adopted for orderings $\preceq$. Pre-metrics are among the less restrictive formal notion of distances. On the other hand, *metrics* could be considered as very restrictive formal notion of distance, since they further satisfy the following axioms:

$$\begin{aligned} \textit{(iff-identity)}\ & x = y \ \Leftrightarrow\ \delta(x, y) = 0; \\ \textit{(symmetry)}\ & \delta(x, y) = \delta(y, x); \\ \textit{(triangle-inequality)}\ & \delta(x, y) \leq \delta(x, z) + \delta(z, y). \end{aligned}$$

A classic example of metric is the Euclidean distance $\delta_2(x, y) \overset{\text{def}}{=} |x - y|$. Fig. 3 summarizes other distance notions lying between pre-metrics and metrics in terms of the properties that they satisfy. The last two rows display the distance symbol and the example in which the distance is defined and used for the first time. Understanding the type of distance function we are manipulating is essential for proving some implications between properties of programs (cf. Sec. 4). From this point forward, whenever we say that a function δ is a distance, we assume that it satisfies, at least, the axiom of a pre-metric.

Def. 2 is general enough to be instantiated with several practical notions of distance proposed in the abstract interpretation literature (e.g., [5,33,34,49,54]). The following examples illustrate two of these notions, used extensively throughout the rest of the paper. For additional examples of pre-metrics and their applications in domains used within the context of program analysis, we refer to [9].

Example 2 (Size Distance). Consider the powerset $\wp(L)$ of a set L. We write $\mathit{size}(S)$ for the number of elements in $S \in \wp(L)$. We define the size distance $\delta_{siz} : \wp(L) \times \wp(L) \to \mathbb{R}^{\infty}_{\geq 0}$ between two sets $S_1, S_2 \in \wp(L)$ as follows: $\delta_{siz}(S_1, S_2) \stackrel{\text{def}}{=} 0$ if $S_1 = S_2$, $\delta_{siz}(S_1, S_2) \stackrel{\text{def}}{=} |\mathit{size}(S_2) - \mathit{size}(S_1)|$ otherwise where $|\infty - \infty| = \infty$. In other words, δ_{siz} calculates the absolute value of the difference in their size. Note that δ_{siz} is a pseudo-metric: two sets may have the same size yet being different. In program analysis, δ_{siz} could be used to count, for instance, the number of spurious elements added by an abstract computation with respect to the abstraction of a concrete computation. For instance, if $[0, 0]$ is the (interval abstraction of the) strongest numerical invariant of a program variable x at certain program point, while $[0, 10]$ is the abstract invariant generated by an abstract interpretation over Int, then $\delta_{siz}(\{0\}, \{0, 1, \ldots, 10\}) = 10$ indicates that the abstract interpretation added 10 more spurious values with respect to the (interval abstraction of the) concrete execution. ■

Example 3 (Inclusion Distance). Given a poset $\langle \wp(\mathbb{Z}), \subseteq \rangle$, we define the inclusion distance $\delta_{\subseteq} : \wp(\mathbb{Z}) \times \wp(\mathbb{Z}) \to \mathbb{N}^{\infty}$ such that $\delta_{\subseteq}(S_1, S_2) \stackrel{\text{def}}{=} k$ with $k \in \mathbb{N}$ if $S_1 \subseteq S_2$ and S_2 has k more elements than S_1. For all other cases, the distance is ∞. For instance, $\delta_{\subseteq}(\{0, 1, 4\}, \{0, 1, 4, 10\}) = 1$ while $\delta_{\subseteq}(\{0, 1, 4\}, \{1, 4, 10\}) = \infty$ because $\{0, 1, 4\} \not\subseteq \{1, 4, 10\}$. This is another distance, like δ_{siz}, that could be used in program analysis to count the number of spurious elements added by an abstract computation with respect to the abstraction of the concrete execution. Note that $\delta_{\subseteq}$ may differ from δ_{siz} even between comparable sets: $\delta_{\subseteq}(\mathbb{Z}_{>0}, \mathbb{Z}_{\geq 0}) = 1 \neq \infty = \delta_{siz}(\mathbb{Z}_{>0}, \mathbb{Z}_{\geq 0})$. The pair $\langle \wp(\mathbb{Z}), \delta_{\subseteq} \rangle$ forms a quasi-metric space. It is not a metric space because $\delta_{\subseteq}$ does not satisfy (*symmetry*). ■

Definition 3 (Quantitative Approximation [8]). *Given a pre-metric space $\langle C, \delta \rangle$ and a fixed constant $\varepsilon \in \mathbb{R}^{\infty}_{\geq 0}$, an element $x \in C$ is* quantitatively approximated *by any element $y \in C$ such that $\delta(x, y) \leq \varepsilon$.*

Example 4. Continuing Ex. 1, we may approximate sets of integer numbers by the size distance δ_{siz} defined in Ex. 2. For instance, $\{0, 1, 4\}$ can be quantitatively approximated by any set of integers whose maximum distance from it is at most $\varepsilon = 1$. Examples of such approximations include sets $\{0, 1\}$ and $\{5, 6, 8, 10\}$. ■

Here, the admitted noise concerns elements that are "close", according to the chosen δ, to the original one but that may share no semantic property.

Combining Semantic and Quantitative Approximations. By *combining* the two forms of approximation, we obtain a general approximation that incorporates a quantitative error within a qualitative abstraction. Given $\rho \in uco(C)$ and a pre-metric space $\langle C, \delta \rangle$, we define the distance $\delta^\rho \colon C \times C \to \mathbb{R}^{\infty}_{\geq 0}$ as

$$\delta^\rho(x, y) \overset{\text{def}}{=} \delta(\rho(x), \rho(y))$$

which calculates the distance between the semantic approximations of x and y.

Definition 4 (General Approximation [8]). *Let $\langle C, \preceq \rangle$ be a poset and $\langle C, \delta \rangle$ be a pre-metric space, and let $\rho \in uco(C)$. An element $x \in C$ is semantically approximated with ρ* and *quantitatively approximated up to $\varepsilon \in \mathbb{R}^{\infty}_{\geq 0}$ by δ, by any element $y \in C$ such that $\delta^\rho(x, y) \leq \varepsilon$.*

Example 5. By considering the interval abstraction $\mathsf{Int} \in uco(\wp(\mathbb{Z}))$, we can combine the two forms of approximation, namely $\delta_\subseteq$ and Int, into $\delta^{\mathsf{Int}}_\subseteq$: this new distance calculates the number of more elements between two comparable interval abstractions rather than considering the original input sets. Note that $\delta^{\mathsf{Int}}_\subseteq$ loses the (*iff-identity*) axiom as one interval might represent more than one set in $\wp(\mathbb{Z})$, thus $\langle \wp(\mathbb{Z}), \delta^{\mathsf{Int}}_\subseteq \rangle$ forms a quasipseudo-metric space.

Continuing Ex. 3, the set $\{0, 1, 4\}$ can be semantically and quantitatively approximated by $\delta^{\mathsf{Int}}_\subseteq$ and $\varepsilon = 1$ in any set in $\{S \in \wp(\mathbb{Z}) \mid \delta^{\mathsf{Int}}_\subseteq(\{0, 1, 4\}, S) \leq 1\}$. ■

3 Abstract Lipschitz Continuity

In mathematical analysis and calculus, Lipschitz continuity is a strong form of uniform continuity of functions that establishes a quantitative relationship between changes to the input of a function and the resulting changes in its output. Specifically, it imposes that perturbations to the input of a function lead to at most linear changes to its output. Lipschitz continuity is usually defined assuming that the input and output domains coincide and that both are metric spaces. In our setting, we allow for *distinct pre-metric input and output spaces.*

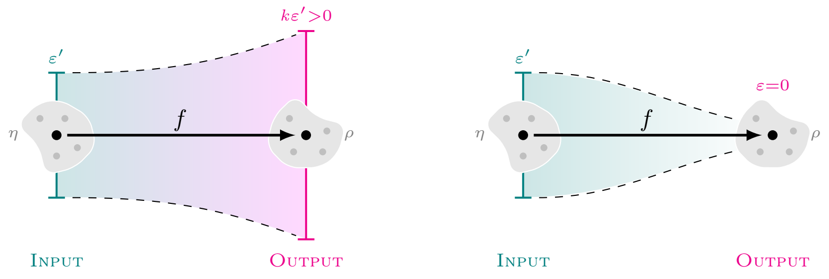
Definition 5 (Lipschitz Continuity). *Let $\langle C, \delta_{\mathtt{C}} \rangle$ and $\langle D, \delta_{\mathtt{D}} \rangle$ be pre-metric spaces. Let $k \in \mathbb{R}_{\geq 0}$. A function $f : C \to D$ satisfies k-*Lipschitz continuity *(k-LC for short) w.r.t. $\langle \delta_{\mathtt{C}}, \delta_{\mathtt{D}} \rangle$ if and only if:*

$$\forall x, y \in C.\ \delta_{\mathtt{D}}(f(x), f(y)) \ \leq \ k\delta_{\mathtt{C}}(x, y).$$

A function f satisfies Lipschitz continuity *(LC) w.r.t. $\langle \delta_{\mathtt{C}}, \delta_{\mathtt{D}} \rangle$ if and only if there exists $k \in \mathbb{R}_{\geq 0}$ such that f satisfies k-Lipschitz continuity w.r.t. $\langle \delta_{\mathtt{C}}, \delta_{\mathtt{D}} \rangle$.*

The Lipschitz constant k provides an upper bound on the rate of change for the output of the function f, i.e., ${}^{\delta_{\mathtt{D}}(f(x), f(y))}/_{\delta_{\mathtt{C}}(x,y)} \leq k$. Note that, k-Lipschitz continuity can be equivalently formulated as follows:

$$\forall x, y \in C.\ \forall \varepsilon' \geq 0.\ \delta_{\mathtt{C}}(x, y) \leq \varepsilon' \Rightarrow \ \delta_{\mathtt{D}}(f(x), f(y)) \leq k\varepsilon'$$



(a) Controlled input/output approximations. (b) Suppression of input approximation.

Fig. 4: Abstract Lipschitz Continuity.

When approximations are introduced to the input of a function, they propagate through its computations, affecting the output. Understanding how approximations evolve during computations provides insight into the behavior of the function and, consequently, into program executions when the function represents program semantics [10,11,57,35,7].

In this work, we generalize the definition of Lipschitz continuity (Def. 5), which relies on quantitative approximations (Def. 3), to work with general approximations (Def. 4), *explicitly discerning semantic approximations (i.e., ucos) from quantitative approximations (i.e., distances)*. This yields the novel notion of *abstract Lipschitz continuity*, which enforces a *controlled (linear) error propagation* from a *general approximation of the input* to a *general approximation of the output* of a function computation (Fig. 4a).

Definition 6 (Abstract Lipschitz Continuity). *Let $\langle C, \preceq_{\mathtt{C}} \rangle$ and $\langle D, \preceq_{\mathtt{D}} \rangle$ be posets, $\langle C, \delta_{\mathtt{C}} \rangle$ and $\langle D, \delta_{\mathtt{D}} \rangle$ be pre-metric spaces, and let $\eta \in \mathrm{uco}(C)$, $\rho \in \mathrm{uco}(D)$, and $k \in \mathbb{R}_{\geq 0}$. A function $f\colon C \to D$ satisfies* Abstract k-Lipschitz Continuity *(k-ALC, for short) w.r.t. $\langle \delta^{\eta}_{\mathtt{C}}, \delta^{\rho}_{\mathtt{D}} \rangle$ when:*

$$\forall x, y \in C.\ \delta^{\rho}_{\mathtt{D}}(f(x), f(y)) \ \leq\ k\delta^{\eta}_{\mathtt{C}}(x, y)$$

A function f satisfies Abstract Lipschitz Continuity *(ALC) if and only if there exists $k \in \mathbb{R}_{\geq 0}$ such that f satisfies abstract k-Lipschitz continuity.*

In other words, f satisfies ALC when it satisfies LC w.r.t. $\langle \delta^{\eta}_{\mathtt{C}}, \delta^{\rho}_{\mathtt{D}} \rangle$. When k-ALC holds, k will be called the *abstract Lipschitz constant*. 0-ALC represents a special case in which the function computation completely suppresses the input semantic approximation (Fig. 4b).

It is worth remarking that Def. 5 can be instantiated to model Def. 6 by incorporating the distance calculus between semantic approximations within $\delta_{\mathtt{C}}$ and $\delta_{\mathtt{D}}$. However, explicitly discerning semantic approximations (i.e., ucos) from quantitative approximations (i.e., distances) as in Def. 6, has two main advantages. First it allows us to relate ALC to other extensional program properties (i.e., relative to the input/output behavior of programs) studied in the program analysis literature (Sec. 4). Second, it allows reasoning on its validity over programs by an inductive reasoning on their syntax and on the chosen ucos (Sec. 5).

Similarly to k-LC, k-ALC can be equivalently reformulated as follows:

Proposition 1. *Consider the premises of Def. 6. A function $f\colon C \to D$ satisfies k-ALC w.r.t. $\langle \delta^{\eta}_{\mathtt{C}}, \delta^{\rho}_{\mathtt{D}} \rangle$ if and only if:*

$$\forall x, y \in C.\, \forall \varepsilon' \geq 0.\ \delta^{\eta}_{\mathtt{C}}(x, y) \leq \varepsilon' \ \Rightarrow\ \delta^{\rho}_{\mathtt{D}}(f(x), f(y)) \leq k\varepsilon'.$$

Example 6. Consider the fragment of C code in Fig. 1 (on the right), which counts the occurrences of *"a"* in a file $F \in \textsc{Files}$. Let $[\![\texttt{count-a}]\!]$ denote its semantics. As described in the introduction, *"a"* could stand for a specific malicious pattern (e.g., signature or byte sequence) that a malware detection tool may look for. In this context, ALC could serve as an integrity check: small changes in file dimension yield proportionally small changes in the *"a"* (pattern) count range, consistent with controlled, non-malicious modifications. On the other hand, it is suspicious if a small change in file dimension causes a big change in the number of patterns. More formally, let $\textsc{Dim}(F)\colon \textsc{Files} \to \mathbb{N}$ be the function extracting the dimension of a file $F \in \textsc{Files}$. We define an (input) abstraction η by additive lift to sets of files: $\eta \overset{\text{def}}{=} \lambda F.\ \{F' \mid \textsc{Dim}(F) = \textsc{Dim}(F')\}$. Moreover, given sets of files $\mathbf{F}$,$\mathbf{F'}$, we define the pseudo-metric $\delta_{\textsc{Dim}}(\mathbf{F}, \mathbf{F'}) = |\max\{\textsc{Dim}(F) \mid F \in \mathbf{F}\} - \max\{\textsc{Dim}(F) \mid F \in \mathbf{F'}\}|$, measuring the difference between file dimensions. Finally, we define an (output) abstraction ρ that only keeps track of reasonable output values, say, between -1000 and 1000 pattern occurrences: $\rho \overset{\text{def}}{=} \lambda n.\ n$ if $-1000 \leq n \leq 1000$, and $\rho \overset{\text{def}}{=} \lambda n. \top$ otherwise. We formalize our integrity property of interest using ALC:

$$\forall F_1, F_2.\ \delta_2(\rho[\![\texttt{count-a}]\!](F_1), \rho[\![\texttt{count-a}]\!](F_2)) \leq k\delta_{\textsc{Dim}}(\eta(F_1), \eta(F_2))$$

where δ_2 is the Euclidean distance. Now, if a slight change in the file dimension caused a non-linear increase of *"a"*, i.e., if $\exists F_1, F_2 \in \textsc{Files}$, and $\varepsilon \in \mathbb{R}_{\geq 0}$ such that $\delta_{\textsc{Dim}}(\eta(F_1), \eta(F_2)) \leq \varepsilon$ but $\delta_2(\rho[\![\texttt{count-a}]\!](F_1), \rho[\![\texttt{count-a}]\!](F_2)) \not\leq k\varepsilon$, then something suspicious has happened, e.g., a malware attack transforming file F_1 into file F_2 by replicating the malicious pattern *"a"*. ■

4 ALC and Other Extensional Program Properties

If we consider the set of all programs whose semantics (to be specified) satisfy Def. 6, then, ALC represents an extensional property of programs, i.e., relative to their input/output behavior. By explicitly discerning the role of qualitative and quantitative approximations in its definition, we can relate ALC to other important extensional properties in the literature that capture input/output relations in comparable yet distinct ways: *partial abstract non-interference* (Eq. 1), *partial completeness* (Eq. 2), and *abstract robustness* (Eq. 3).

Partial Abstract Non-Interference. Partial Abstract Non-Interference [8] (PANI for short) is a relaxation of (abstract) non-interference [28,25] that combines both semantic and quantitative approximations on the output of a function computation. This notion can be seen as a generalization of approximate non-interference [48], originally introduced in a probabilistic process algebra, requiring the *observable* behaviors of two agents to be under a similarity threshold ε. Specifically, PANI observes indistinguishable (distance-wise) *properties* of input data, while allowing a *bounded distance* between the observed output *properties*. This enables a more nuanced treatment of security policies, especially when small, bounded differences in outputs are tolerable. In the same premises of Def. 6, ε-PANI$[\delta^{\eta}_{\mathsf{C}}, \delta^{\rho}_{\mathsf{D}}]$ has been formalized as [8]:

$$\forall x, y \in C.\ \delta^{\eta}_{\mathsf{C}}(x, y) = 0 \ \Rightarrow\ \delta^{\rho}_{\mathsf{D}}(f(x), f(y)) \leq \varepsilon \tag{1}$$

Given inputs with zero property distance under $\delta^{\eta}_{\mathsf{C}}$, ε-PANI$[\delta^{\eta}_{\mathsf{C}}, \delta^{\rho}_{\mathsf{D}}]$ allows the function to produce different outputs, potentially with different properties under $\delta^{\rho}_{\mathsf{D}}$, as long as the variation remains bounded, i.e., not exceeding the threshold ε.

It turns out that ALC is a strictly stronger property than PANI, as stated by the following theorem.

Theorem 1. *Let $\langle C, \delta_{\mathsf{C}}\rangle$ and $\langle D, \delta_{\mathsf{D}}\rangle$ be pre-metric spaces. If $f\colon C \to D$ satisfies ALC w.r.t. $\langle \delta^{\eta}_{\mathsf{C}}, \delta^{\rho}_{\mathsf{D}}\rangle$, then, for any $\varepsilon \in \mathbb{R}_{\geq 0}$, f also satisfies ε-PANI$[\delta^{\eta}_{\mathsf{C}}, \delta^{\rho}_{\mathsf{D}}]$.*

In the context of programs where f represents the semantics $[\![\mathsf{P}]\!]$ of a program P, requiring $[\![\mathsf{P}]\!]$ to satisfy ALC is a stronger condition than requiring ε-PANI.

Example 7. Let $\mathsf{R} = (x > 0?\,;\, x := x - 1) \oplus (x \leq 0?\,;\, x := x + 1)$ be a program which increments all non-negative values by 1 and decrements all non-positive values by 1. Let us consider the pseudo-metric space $\langle \wp(\mathbb{Z}), \delta_{siz}\rangle$ as the input and output domain, and the interval closure $\mathsf{Int} \in uco(\wp(\mathbb{Z}))$. The collecting semantics $[\![\mathsf{R}^*]\!] : \wp(\mathbb{Z}) \to \wp(\mathbb{Z})$ of the Kleene closure of R, R^*, satisfies 1-ALC w.r.t. $\langle \delta^{\mathsf{Int}}_{siz}, \delta^{\mathsf{Int}}_{siz}\rangle$. Indeed, $[\![\mathsf{R}^*]\!]$ is monotone by definition, thus preserving the inclusion relation, and either reduces the distance of input intervals or leaves them unchanged. For instance:

$$\delta^{\mathsf{Int}}_{siz}([\![\mathsf{R}^*]\!]([2,6]), [\![\mathsf{R}^*]\!]([0,7])) = \delta^{\mathsf{Int}}_{siz}([0,6],[0,7]) = 1 \leq 3 = \delta^{\mathsf{Int}}_{siz}([2,6],[0,7])$$
$$\delta^{\mathsf{Int}}_{siz}([\![\mathsf{R}^*]\!]([-5,-2]), [\![\mathsf{R}^*]\!]([-7,0])) = \delta^{\mathsf{Int}}_{siz}([-5,1],[-7,1]) = 2 \leq 5 = \delta^{\mathsf{Int}}_{siz}([-5,-2],[-7,0])$$
$$\delta^{\mathsf{Int}}_{siz}([\![\mathsf{R}^*]\!]([-2,3]), [\![\mathsf{R}^*]\!]([-5,3])) = \delta^{\mathsf{Int}}_{siz}([-2,3],[-5,3]) = 3 \leq 3 = \delta^{\mathsf{Int}}_{siz}([-2,3],[-5,3])$$

By Thm. 1, the program semantics $[\![\mathsf{R}^*]\!]$ of R^* also satisfies 0-PANI$[\delta^{\mathsf{Int}}_{siz}, \delta^{\mathsf{Int}}_{siz}]$. ■

As a corollary result, when δ_{D} additionally satisfies the (*iff-identity*), ALC implies the Abstract Non-Interference (ANI) property, where distances are converted as equality requirements [25]: $\forall x, y \in C.\ \eta(x) = \eta(y) \ \Rightarrow\ \rho f(x) = \rho f(y)$.

Corollary 1. *Let $\langle C, \delta_{\mathsf{C}}\rangle$ and $\langle D, \delta_{\mathsf{D}}\rangle$ be quasisemi-metric spaces. If $f\colon C \to D$ satisfies ALC w.r.t. $\langle \delta^{\eta}_{\mathsf{C}}, \delta^{\rho}_{\mathsf{D}}\rangle$, then f satisfies also ANI.*

Partial Completeness. Partial completeness [5,9] is a weakening of the standard notion of completeness in abstract interpretation. In the classical abstract interpretation framework [14], the computation of a concrete (possibly uncomputable) monotone function $f\colon C \to C$ over a poset $\langle C, \preceq_{\mathsf{C}}\rangle$ is replaced by an abstract (possibly computable) sound computation $f^\natural\colon \rho(C) \to \rho(C)$ over an abstract domain $\rho(C)$, where $\rho \in \mathit{uco}(C)$. For example, if P is a program with a single variable then f could be the collecting big-step semantics $[\![\mathsf{P}]\!]\colon \wp(\mathbb{Z}) \to \wp(\mathbb{Z})$ defined in Sec. 2.1 over $\langle \wp(\mathbb{Z}), \subseteq\rangle$, $\rho = \mathsf{Int}$ the interval abstract domain (Ex. 1) and $[\![\mathsf{P}]\!]^\natural\colon \mathsf{Int}(\wp(\mathbb{M})) \to \mathsf{Int}(\wp(\mathbb{M}))$ is the abstract interval semantics soundly computed on intervals. An abstract function $f^\natural\colon \rho(C) \to \rho(C)$ is *sound* when $\rho f \preceq_{\mathsf{C}} f^\natural \rho$, and is said to be *complete* when $\rho f = f^\natural \rho$, i.e., $f^\natural$ coincides with the *best correct approximation* (bca) [15] $\bar{f} \overset{\text{def}}{=} \rho f \rho$ of f. Completeness is the best possible scenario where no imprecision is introduced by the abstract computation of $f^\natural$. In practice, however, completeness is rarely achieved. For this reason, Campion et al. [5,9] introduced a weaker notion of completeness, called *ε-partial completeness* which allows some degree of imprecision limited by ε. This imprecision is measured by pre-metrics $\delta_{\rho(\mathsf{C})}\colon \rho(C) \times \rho(C) \to \mathbb{R}^{\infty}_{\geq 0}$ that are *$\preceq_{\mathsf{C}}$-compatible* with the abstract domain, i.e., pre-metrics additionally satisfying [9]:

$$x \preceq_{\mathsf{C}} y \preceq_{\mathsf{C}} z \;\Rightarrow\; \delta_{\rho(\mathsf{C})}(x, y) \leq \delta_{\rho(\mathsf{C})}(x, z) \,\wedge\, \delta_{\rho(\mathsf{C})}(y, z) \leq \delta_{\rho(\mathsf{C})}(x, z)$$

Partial completeness was originally defined as a local property (namely, defined on a strict subset of the input domain) [5]. Here, for our purposes, we consider a *global* version of ε-partial completeness:

$$\forall x \in C.\ \delta_{\rho(\mathsf{C})}(\rho f(x), f^\natural \rho(x)) \leq \varepsilon \tag{2}$$

To relate ALC w.r.t. $\langle \delta^\rho_{\mathsf{C}}, \delta^\rho_{\mathsf{C}}\rangle$ for a monotone function $f\colon C \to C$, to partial completeness, we observe that, given a pre-metric $\preceq_{\mathsf{C}}$-compatible δ_{C} defined over C, since $\rho(C) \subseteq C$ we can always restrict δ_{C} to elements of $\rho(C)$, i.e., $\forall x, y \in \rho(C)\colon \delta_{\rho(\mathsf{C})}(x, y) \overset{\text{def}}{=} \delta_{\mathsf{C}}(x, y)$ without losing the $\preceq_{\mathsf{C}}$-compatible property.

Theorem 2. *Let $\langle C, \delta_{\mathsf{C}}\rangle$ be a $\preceq_{\mathsf{C}}$-compatible pre-metric space. If $f\colon C \to C$ satisfies ALC w.r.t. $\langle \delta^\rho_{\mathsf{C}}, \delta^\rho_{\mathsf{C}}\rangle$, then the bca $\bar{f}$ of f satisfies ε-partial completeness w.r.t. f for any $\varepsilon \in \mathbb{R}_{\geq 0}$.*

Corollary 2. *Let $\delta_{\rho(\mathsf{C})}$ be a quasisemi-metric. If $f\colon C \to C$ satisfies ALC w.r.t. $\langle \delta^\rho_{\mathsf{C}}, \delta^\rho_{\mathsf{C}}\rangle$, then the bca $\bar{f}$ of f satisfies completeness w.r.t. f.*

Thm. 2 and Cor. 2 reveal a novel relationship between the ALC of a function f and the partial completeness of its bca, particularly in the context of program analysis where f represents a (monotone) semantics $[\![\mathsf{P}]\!]$ of a given program P. Notably, a proof of ALC of $[\![\mathsf{P}]\!]$ w.r.t. the distance δ^ρ_{C} implies that the imprecision introduced by approximating $[\![\mathsf{P}]\!]$ with its bca $\rho[\![\mathsf{P}]\!]\rho$ is bounded by any constant factor. Consequently, $\rho[\![\mathsf{P}]\!]\rho$ also satisfies 0-partial completeness, namely *no imprecision is generated by the bca according to δ_{C}*. Thus, ALC of a program $[\![\mathsf{P}]\!]$ *ensures the existence of an abstract interpreter capable of approximating the* *computation of* $[\![\mathsf{P}]\!]$ *over* $\rho(C)$ *with no imprecision* according to the imprecision measured by δ_{C}. Furthermore, when the distance $\delta_{\rho(\mathsf{C})}$ is a quasisemi-metric, then the bca $\rho[\![\mathsf{P}]\!]\rho$ is guaranteed to satisfy completeness, namely, *no imprecision is generated by the bca.* Another way to interpret Cor. 2 (and, analogously, Thm. 2) is as follows: if a program $[\![\mathsf{P}]\!]$ does not admit a complete bca $\rho[\![\mathsf{P}]\!]\rho$, then $[\![\mathsf{P}]\!]$ *cannot* satisfy ALC for any quasisemi-metric.

Example 8. Consider again the collecting semantics $[\![\mathsf{R}^*]\!]$ of Ex. 7 which has been proved to satisfy 1-ALC w.r.t. $\langle \delta^{\mathsf{Int}}_{siz}, \delta^{\mathsf{Int}}_{siz}\rangle$. It is easy to note that $[\![\mathsf{R}^*]\!]$ also satisfies 1-ALC w.r.t. $\langle \delta^{\mathsf{Int}}_{\subseteq}, \delta^{\mathsf{Int}}_{\subseteq}\rangle$, where $\delta_{\subseteq}$ is the quasi-metric defined in Ex. 3. Then by Thm. 2, the bca $\mathsf{Int}[\![\mathsf{R}^*]\!]\mathsf{Int}$ also satisfies 0-partial completeness w.r.t. $\delta^{\mathsf{Int}}_{\subseteq}$, i.e., $\delta^{\mathsf{Int}}_{\subseteq}([\![\mathsf{R}^*]\!]S, [\![\mathsf{R}^*]\!]\mathsf{Int}(S)) \leq 0$, for any $S \in \wp(\mathbb{Z})$. Moreover, since $\delta_{\subseteq}$ is also a quasisemi-metric, we can conclude that $\mathsf{Int}[\![\mathsf{R}^*]\!]\mathsf{Int}$ is complete, namely it does not add any imprecision when approximating $[\![\mathsf{R}^*]\!]$.

Consider instead the pseudo-metric space $\langle \wp(\mathbb{Z}), \delta_{siz}\rangle$ and the semantics $[\![\mathsf{R}]\!]$ of program R. The bca $\mathsf{Int}[\![\mathsf{R}]\!]\mathsf{Int}$ does not satisfy 0-partial completeness w.r.t. $\delta^{\mathsf{Int}}_{siz}$: given $X = \{-1, 1\}$, we have that $[\![\mathsf{R}]\!]$ cannot satisfy ALC for $\langle \delta^{\mathsf{Int}}_{siz}, \delta^{\mathsf{Int}}_{siz}\rangle$ since $\delta^{\mathsf{Int}}_{siz}([\![\mathsf{R}]\!]X, [\![R]\!]\mathsf{Int}(X)) = \delta^{\mathsf{Int}}_{siz}([0,0],[0,1]) = 1 \neq 0$. In fact, it is easy to note that $[\![\mathsf{R}]\!]$ satisfies 1-partial completeness for all inputs. ■

Abstract Robustness. The idea of abstract robustness is to formally characterize the amount of error that an analyst may tolerate in a machine learning classification result [26]. This error, generated by an input perturbation, either a semantic or a quantitative approximation [26], is modeled as a semantic approximation of the classification outcome, expressed through a uco ρ. Given an admissible margin of error $\varepsilon \in \mathbb{R}_{\geq 0}$, and assuming f to be a classifier, abstract robustness with a quantitative input perturbation is defined as [26]:

$$\forall x, y \in C.\ \delta_{\mathsf{C}}(x,y) \leq \varepsilon \ \Rightarrow\ \rho f(x) = \rho f(y) \tag{3}$$

If we look at this definition in terms of ALC, then we observe that this notion requires that the error introduced by the perturbation of the input must be neutralized, at least for the ρ observation of the output classification. In fact, if we consider in output a quasi-metric the equality means distance smaller than 0, and this corresponds precisely to 0-ALC.

Theorem 3. *Let $\langle D, \delta_{\mathsf{D}}\rangle$ be a quasisemi-metric space. 0-ALC w.r.t. $\langle \delta^{id}_{\mathsf{C}}, \delta^{\rho}_{\mathsf{D}}\rangle$ holds if and only if abstract robustness holds.*

Setting the abstract Lipschitz constant to zero is necessary to stabilize the output of the classifier under the same semantic abstraction ρ. The distance δ_{D} must also be a quasisemi-metric to ensure that equal elements can be distinguished when the distance evaluates to zero. What is interesting in this result is that abstract robustness, and therefore standard robustness, is a specific instance of ALC, the one where the potential error in input does not affect in any way the classification of the output (at least in the observed output property). In this sense, ALC becomes a generalization of robustness by *admitting* an effect in the output, but this effect must be linearly bounded to the input error. From this perspective, ALC serves as a model for further weakening robustness in ML, enabling us to better adapt it to specific contexts or fields of application.

5 Proving ALC for Programs

Deductive systems for the verification of completeness [23], partial completeness [7] and Lipschitz continuity [11,12] properties of programs have already been formalized in the literature. In this section we introduce a novel deductive system, inductively defined on the program's syntax, that is able to soundly prove the new ALC notion of a program semantics w.r.t. the input and output abstractions $\langle \eta, \rho \rangle$ and a given pre-metric δ. Our objective in designing this deductive system is to identify and track the assumptions necessary to establish a proof of ALC. In a Hoare-style approach, the proof system also allows to weaken or strengthen both the input and output abstractions when attempting to complete a proof of ALC for a program. Soundness here means that when the semantics $[\![\mathsf{P}]\!] \colon C \to C$ of a program P is typed as k-ALC w.r.t. $\langle \delta^\eta, \delta^\rho \rangle$ by the deductive system, then $[\![\mathsf{P}]\!]$ is certainly k-ALC for $\langle \delta^\eta, \delta^\rho \rangle$.

Given $k \in \mathbb{R}^\infty_{\geq 0}$, $\eta, \rho \in uco(C), \delta \colon C^2 \to \mathbb{R}^\infty_{\geq 0}$, we introduce the following set

$$k\text{-}ALip\langle \delta^\eta, \delta^\rho \rangle \overset{\text{def}}{=} \{ f \in C \to C \mid f \text{ is } k\text{-ALC w.r.t. } \langle \delta^\eta, \delta^\rho \rangle \}$$

of all abstract k-Lipschitz continuous functions on a complete lattice $\langle C, \preceq \rangle$ for $\langle \delta^\eta, \delta^\rho \rangle$. In order to preserve the soundness of derivations in the proof system, we allow k to take the value ∞, since this is an admissible sound (albeit possibly imprecise) result. Clearly, any function is ∞-ALC. The following lemma outlines some basic properties of $k\text{-}ALip\langle \delta^\eta, \delta^\rho \rangle$ that will be later exploited.

Lemma 1. *The following hold for all functions $f \in C \to C$, closures $\eta, \rho \in uco(C)$, pre-metric δ and $k \in \mathbb{R}^\infty_{\geq 0}$:*

(i) $k \geq 1 \;\Rightarrow\; \rho \in k\text{-}ALip\langle \delta^\rho, \delta^\rho \rangle$

(ii) f *is k-LP w.r.t.* $\langle \delta, \delta \rangle \;\Leftrightarrow\; f \in k\text{-}ALip\langle \delta^{id}, \delta^{id} \rangle$

(iii) $\rho \in k\text{-}ALip\langle \delta^{id}, \delta^{id} \rangle \;\Leftrightarrow\; \rho \in k\text{-}ALip\langle \delta^{id}, \delta^\rho \rangle$

(i) states that, when considering the same input-output abstractions (i.e. $\eta = \rho$), then the abstraction function is k-ALC for any $k \geq 1$. Moreover, for the statement (ii), when both input-output abstractions are the identity function id, then the class $k\text{-}ALip\langle \delta^{id}, \delta^{id} \rangle$ corresponds precisely to the set of all k-LP functions. Finally, (iii) shows that, when a closure ρ satisfies ALC w.r.t. $\langle \delta^{id}, \delta^{id} \rangle$, then ρ also satisfies k-ALC for $\langle \delta^{id}, \delta^\rho \rangle$, and vice-versa. This is due to the idempotence property of closure operators.

From now on, we fix a program semantics of interest $[\![\cdot]\!] \colon \mathsf{Prog} \to C \to C$ as well as a complete lattice $\langle C, \preceq, \vee, \wedge, \top, \bot \rangle$, and we will also use the statement "P is k-ALC w.r.t. $\langle \delta^\eta, \delta^\rho \rangle$" to indicate that the semantics $[\![\mathsf{P}]\!]$ is abstract k-Lipschitz continuous w.r.t. $\langle \delta^\eta, \delta^\rho \rangle$, i.e. $[\![\mathsf{P}]\!] \in k\text{-}ALip\langle \delta^\eta, \delta^\rho \rangle$.

$$\frac{\llbracket \mathsf{c} \rrbracket \in k\text{-}ALip\langle \delta^\eta, \delta^\rho \rangle}{k \vdash [\delta^\eta]\, \mathsf{c}\, [\delta^\rho]}\ (\textbf{base})$$

$$\frac{k' \vdash [\delta^{\eta'}]\, \mathsf{P}\, [\delta^{\rho'}] \quad k' \leq k \quad \eta' \in t\text{-}ALip\langle \delta^\eta, \delta^{\eta'} \rangle \quad \rho \in s\text{-}ALip\langle \delta^{\rho'}, \delta^\rho \rangle}{stk \vdash [\delta^\eta]\, \mathsf{P}\, [\delta^\rho]}\ (\textbf{weaken})$$

$$\frac{k_1 \vdash [\delta^\eta]\, \mathsf{P}_1\, [\delta^\rho] \quad k_2 \vdash [\delta^\eta]\, \mathsf{P}_2\, [\delta^\rho] \quad \eta \in t\text{-}ALip\langle \delta^\rho, \delta^\eta \rangle}{k_1 t k_2 \vdash [\delta^\eta]\, \mathsf{P}_1 ; \mathsf{P}_2\, [\delta^\rho]}\ (\textbf{seq})$$

$$\frac{k_1 \vdash [\delta^\eta]\, \mathsf{P}_1\, [\delta^\rho] \quad k_2 \vdash [\delta^\eta]\, \mathsf{P}_2\, [\delta^\rho]}{k_1 \uplus k_2 \vdash [\delta^\eta]\, \mathsf{P}_1 \oplus \mathsf{P}_2\, [\delta^\rho]}\ (\textbf{join})$$

$$\frac{k \vdash [\delta^\eta]\, \mathsf{P}\, [\delta^\rho] \quad \begin{array}{c} \eta \in t\text{-}ALip\langle \delta^\rho, \delta^\eta \rangle \\ \rho \in v\text{-}ALip\langle \delta^\eta, \delta^\rho \rangle \end{array} \quad *\text{-Bound}(\mathsf{P}^*, m)}{K_m \vdash [\delta^\eta]\, \mathsf{P}^*\, [\delta^\rho]}\ (\textbf{bound-star})$$

$$\frac{k \vdash [\delta^\eta]\, \mathsf{P}\, [\delta^\rho] \quad \eta \in t\text{-}ALip\langle \delta^\rho, \delta^\eta \rangle \quad \rho \in v\text{-}ALip\langle \delta^\eta, \delta^\rho \rangle}{K_\infty \vdash [\delta^\eta]\, \mathsf{P}^*\, [\delta^\rho]}\ (\textbf{star})$$

Fig. 5: A deductive system for proving ALC for Prog.

The deductive rules are provided in Fig. 5. The judgments take the form:

$$k \vdash [\delta^\eta]\, \mathsf{P}\, [\delta^\rho]$$

We will later show that deriving a judgment $k \vdash [\delta^\eta]\, \mathsf{P}\, [\delta^\rho]$ through the deductive rules in Fig. 5, implies that $\llbracket \mathsf{P} \rrbracket \in k\text{-}ALip\langle \delta^\eta, \delta^\rho \rangle$. Let us examine each rule and provide an intuitive, informal explanation.

The (**base**) rule allows deriving the triple $k \vdash [\delta^\eta]\, \mathsf{c}\, [\delta^\rho]$ for all basic commands $\mathsf{c} \in \mathsf{Stm}$ (i.e., for **skip**, assignments and Boolean guards) assuming that we have a proof of k-ALC of them, encoded by the predicate $\llbracket \mathsf{c} \rrbracket \in k\text{-}ALip\langle \delta^\eta, \delta^\rho \rangle$.

The (**weaken**) rule allows weakening both the abstract Lipschitz constant and the abstractions considered. In particular, when we are able to derive the k'-ALC for program P w.r.t. $\langle \delta^{\eta'}, \delta^{\rho'} \rangle$, then we can always deduce a higher abstract Lipschitz constant $k \geq k'$ without changing the validity of the triple. For the input abstraction η', we can consider a new input abstraction η whenever η' is proved to be t-ALC w.r.t. $\langle \delta^\eta, \delta^{\eta'} \rangle$ with η as input abstraction. This weakening comes at the cost of multiplying the already deduced constant k' with the new constant t. This could happen, for instance, when η is in fact widening the distance $\delta^{\eta'}(c_1, c_2)$ between any two elements $c_1, c_2 \in C$, by a constant factor of t, namely by $t\delta^\eta(c_1, c_2)$. On the other hand, we can weaken the output abstraction ρ' by a new abstraction ρ whenever ρ is proved to be ALC for $\langle \delta^{\rho'}, \delta^\rho \rangle$ namely with ρ' as input abstraction. Here ρ could represent a narrow output abstraction in terms of distance δ between elements in C with respect to ρ', namely having distance $\delta^\rho(c_1, c_2) \leq s\delta^{\rho'}(c_1, c_2)$ and thus introducing a new abstract Lipschitz constant s. Note that (**weaken**) allows also for selecting which weakening we want to apply. For instance, if we want to weaken the abstract Lipschitz constant k' only, then we can set $\eta' \in 1\text{-}ALip\langle\delta^{\eta'}, \delta^{\eta'}\rangle$ and $\rho' \in 1\text{-}ALip\langle\delta^{\rho'}, \delta^{\rho'}\rangle$ in the premises as they always hold (by Lem. 1(i)) without modifying any abstraction.

Composition of programs is treated by the (**seq**) rule. Although it is well known that composing two functions f_1 and f_2 which are k_1-LP and k_2-LP, respectively, gives as result a new k_1k_2-LP function $f_2 \circ f_1$, this in general does not always hold for ALC as abstractions come into play. However, when we have a derivation for P_1 and P_2 with abstract Lipschitz constants k_1 and k_2, respectively, and we are able to prove that the input abstraction η is t-ALC w.r.t. $\langle\delta^\rho, \delta^\eta\rangle$, then this is a sufficient condition for deriving the k_2tk_1-ALC of the composition $\mathsf{P}_1; \mathsf{P}_2$. Requiring $\eta \in t\text{-}ALip\langle\delta^\rho, \delta^\eta\rangle$ corresponds to require $\delta^\eta(c_1, c_2) \leq t\delta^\rho(c_1, c_2)$, namely that we have a linear relation between their distances: when $t \geq 1$ then ρ is widening the distance, while when $0 < t < 1$ then ρ is narrowing their distances, both cases with a constant factor of t. Note that, when the input and output abstractions coincide, i.e. $\eta = \rho$, then $\rho \in 1\text{-}ALip\langle\delta^\rho, \delta^\rho\rangle$ holds trivially (by Lem. 1(i)). As a consequence, for the case $\eta = \rho$, the ALC property is closed under composition, analogously to the LP property.

The rule (**join**) involves the join operator. Similarly for the composition, the join of two ALC functions is not necessarily ALC. The problem here stems in the fact that the resulting abstract Lipschitz constant bound could not be determined by knowing only the abstract Lipschitz constants of both P_1 and P_2. This is because the distance between the execution of $\mathsf{P}_1 \oplus \mathsf{P}_2$ and the join of the two post-conditions, relies on the underlying structure of the input and output abstractions considered. Our solution, inspired by [5,7], consists in parametrizing the proof system with a binary operation $\uplus : \mathbb{R}^\infty_{\geq 0} \times \mathbb{R}^\infty_{\geq 0} \to \mathbb{R}^\infty_{\geq 0}$, called an $\oplus$-bound, which yields a new abstract Lipschitz constant.

Definition 7 ($\oplus$-Bound). *Let $\uplus : \mathbb{R}^\infty_{\geq 0} \times \mathbb{R}^\infty_{\geq 0} \to \mathbb{R}^\infty_{\geq 0}$ be a binary operator such that $\langle\mathbb{R}^\infty_{\geq 0}, \uplus, \cdot\rangle$ forms a complete semiring, where $\cdot$ is the standard multiplication. $\uplus$ is a* $\oplus$-bound *when the following holds for all $\mathsf{P}_1, \mathsf{P}_2 \in \mathsf{Prog}$ and $k_1, k_2 \in \mathbb{R}^\infty_{\geq 0}$:*

$$\begin{array}{r}\llbracket\mathsf{P}_1\rrbracket \in k_1\text{-}ALip\langle\delta^\eta, \delta^\rho\rangle \\ \textit{and}\ \ \llbracket\mathsf{P}_2\rrbracket \in k_2\text{-}ALip\langle\delta^\eta, \delta^\rho\rangle\end{array} \ \Rightarrow\ \rho\llbracket\mathsf{P}_1\rrbracket \vee \rho\llbracket\mathsf{P}_2\rrbracket \in k_1 \uplus k_2\text{-}ALip\langle\delta^\eta, \delta^\rho\rangle$$

Example 9. Consider the pseudo-metric space $\langle\wp(\mathbb{Z}), \delta_{siz}\rangle$ and the collecting semantics $\llbracket\cdot\rrbracket$. Let the input and output abstractions be $\rho = \eta = \mathsf{Int}$. Then the standard addition $+$ is a $\oplus$-bound. In other words, having an ALC proof for both P_1 and P_2, with abstract Lipschitz constants k_1, k_2, respectively, gives:

$$\begin{aligned}&\delta_{siz}((\mathsf{Int}\llbracket\mathsf{P}_1\rrbracket \vee \mathsf{Int}\llbracket\mathsf{P}_2\rrbracket)c_1, (\mathsf{Int}\llbracket\mathsf{P}_1\rrbracket \vee \mathsf{Int}\llbracket\mathsf{P}_2\rrbracket)c_2) \leq \\ &\quad k_1\delta_{siz}(\mathsf{Int}(c_1), \mathsf{Int}(c_2)) + k_2\delta_{siz}(\mathsf{Int}(c_1), \mathsf{Int}(c_2)) = (k_1 + k_2)\delta_{siz}(\mathsf{Int}(c_1), \mathsf{Int}(c_2))\end{aligned}$$

This is because, when considering δ_{siz} as distance and Int as input and output abstractions, the size of the join of two intervals can be over-approximated by the sum of the number of the elements inside the two intervals. A similar reasoning holds for the quasisemi-metric space $\langle\wp(\mathbb{Z}), \delta_{\subseteq}\rangle$ defined in Ex. 3. ∎

The premise of the (**join**) rule asks for the validity of the following predicates: assume we have an ALC derivation $k_1 \vdash [\delta^\eta]\, \mathsf{P}_1\, [\delta^\rho]$ for P_1 and $k_2 \vdash [\delta^\eta]\, \mathsf{P}_2\, [\delta^\rho]$ for P_2, then we can soundly conclude that the join $\mathsf{P}_1 \oplus \mathsf{P}_2$ is $k_1 \uplus k_2$-ALC.

For the Kleene star operator, we introduce two rules: (**bound-star**) and (**star**). Both rules require that the program P satisfies k-ALC w.r.t. $\langle \delta^\eta, \delta^\rho \rangle$. Moreover, in order to compose programs, the input abstraction η is required to satisfy t-ALC w.r.t. $\langle \delta^\rho, \delta^\eta \rangle$ (as required by (**seq**)). The additional requirement $\rho \in v\text{-}ALip\langle \delta^\eta, \delta^\rho \rangle$ enforces a linear relationship between η and ρ which is necessary to handle the case of zero iterations. Note that these requirements are trivially satisfied when $\eta = \rho$ (by Lem. 1(i)). The (**bound-star**) rule further requires the validity of the assertion $*$-Bound(P^*, m), with $m \in \mathbb{N}$, which holds when $[\![\mathsf{P}^*]\!]c = \bigvee\{[\![\mathsf{P}]\!]^n c \mid n \in \mathbb{N}_{\leq m}\}$ for any $c \in C$. In other words, given any input $c \in C$, the result of $[\![\mathsf{P}^*]\!]c$ can be obtained by executing P at most m times. (**star**) rule, instead, has no constraints on the number of iterations of P. Rules (**bound-star**) and (**star**) rely on the following definitions:

$$K_0 \overset{\text{def}}{=} v\ , \qquad K_n \overset{\text{def}}{=} v \uplus \biguplus_{i=1}^{n} k^i t^{i-1} \quad (n \geq 1)\ , \qquad K_\infty \overset{\text{def}}{=} \sup_{n \geq 0} K_n$$

where the supremum is with respect to the standard order induced by $\uplus$ (i.e., $x \preceq_\uplus y$ iff $x \uplus y = y$). For rule (**star**), the completeness of the semiring guarantees that K_∞ always exists and, depending on the definition of $\uplus$, as well as the values of k and t (see, e.g., Ex. 11), it is either a finite value in $\mathbb{R}_{\geq 0}$ or ∞.

The following theorem shows that our proposed deductive system is sound, namely, if $k \vdash [\delta^\eta]\, \mathsf{P}\, [\delta^\rho]$ can be derived by applying the rules of Fig. 5, then $[\![\mathsf{P}]\!]$ satisfies k-ALC w.r.t. $\langle \delta^\eta, \delta^\rho \rangle$.

Theorem 4 (Soundness). $k \vdash [\delta^\eta]\, \mathsf{P}\, [\delta^\rho] \;\Rightarrow\; [\![\mathsf{P}]\!] \in k\text{-}ALip\langle \delta^\eta, \delta^\rho \rangle$

Example 10. Consider the program $(x < 0?\,;\, x := 0) \oplus (x \geq 0?\,;\, \textbf{skip})$ implementing the ReLU function used in artificial neural networks [45], that filters the input below 0. Consider the quasi-metric space $\langle \wp(\mathbb{Z}), \delta_\subseteq \rangle$ and the input and output abstraction $\eta = \rho = \mathsf{Int}$. We want to prove that the collecting semantics $[\![\mathsf{ReLU}]\!] : \wp(\mathbb{Z}) \to \wp(\mathbb{Z})$ satisfies 1-ALC for $\langle \delta_\subseteq^{\mathsf{Int}}, \delta_\subseteq^{\mathsf{Int}} \rangle$. Let us start by analyzing the base commands on the left of $\oplus$. Because the Boolean guard $x < 0?$ is either preserving or removing values from the input, by the (**base**) rule, we can derive $1 \vdash [\delta_\subseteq^{\mathsf{Int}}]\, x < 0?\, [\delta_\subseteq^{\mathsf{Int}}]$. The command $x := 0$ is neutralizing any distance between input sets since $\delta_\subseteq^{\mathsf{Int}}([\![x := 0]\!]S_1, [\![x := 0]\!]S_2) = 0$ for any $S_1, S_2 \in \wp(\mathbb{Z})$. We can derive $0 \vdash [\delta_\subseteq^{\mathsf{Int}}]\, x := 0\, [\delta_\subseteq^{\mathsf{Int}}]$ by the (**base**) rule. Since $\mathsf{Int} \in 1\text{-}ALip\langle \delta_\subseteq^{\mathsf{Int}}, \delta_\subseteq^{\mathsf{Int}} \rangle$ follows from Lem. 1, we can infer $0 \vdash [\delta_\subseteq^{\mathsf{Int}}]\, x < 0?\,;\, x := 0\, [\delta_\subseteq^{\mathsf{Int}}]$ by the (**seq**) rule. For the base commands on the right of $\oplus$, we get $1 \vdash [\delta_\subseteq^{\mathsf{Int}}]\, x \geq 0?\, [\delta_\subseteq^{\mathsf{Int}}]$ with rule (**base**). The **skip** command does not modify the distance of the input sets, so $1 \vdash [\delta_\subseteq^{\mathsf{Int}}]\, \textbf{skip}\, [\delta_\subseteq^{\mathsf{Int}}]$ can be derived by (**base**). Since $\mathsf{Int} \in 1\text{-}ALip\langle \delta_\subseteq^{\mathsf{Int}}, \delta_\subseteq^{\mathsf{Int}} \rangle$ holds, the rule (**seq**) derives $1 \vdash [\delta_\subseteq^{\mathsf{Int}}]\, x \geq 0?\,;\, \textbf{skip}\, [\delta_\subseteq^{\mathsf{Int}}]$. Now for the $\oplus$ operation, we consider $\uplus = +$ (as in Ex. 9), thus guaranteeing a sound upper bound for the abstract Lipschitz constants on the program join. By the two derivations on the left and right parts of $\oplus$, we can conclude by the (**join**) rule: $1 \vdash [\delta_{\subseteq}^{\mathsf{Int}}]\ \mathsf{ReLU}\ [\delta_{\subseteq}^{\mathsf{Int}}]$. By Thm. 4, $[\![\mathsf{ReLU}]\!]$ satisfies 1-ALC for $\langle \delta_{\subseteq}^{\mathsf{Int}}, \delta_{\subseteq}^{\mathsf{Int}} \rangle$. ■

Example 11. Consider the program $\mathsf{F}\colon (x > 0?\ x := x/4) \oplus (x \leq 0?\,; x := 0)$ corresponding to ReLU with a scaling factor of $\frac{1}{4}$, and the psueudo-metric space $\langle \wp(\mathbb{R}), \delta_{-} \rangle$, where $\delta_{-}(S_1, S_2) \overset{\text{def}}{=} 0$ if $S_1 = S_2$, $\delta_{-}(S_1, S_2) \overset{\text{def}}{=} |(\max(S_2) - \min(S_2)) - (\max(S_1) - \min(S_1))|$ otherwise. Let $\eta = \rho = \mathsf{Int}$. Intuitively, when calculating the distance δ_{-} over two sets in $\mathsf{Int}(\wp(\mathbb{R}))$, the distance returns the absolute value of the difference between the length of the two intervals. We derive the abstract Lipschitz constant of the collecting semantics $[\![\mathsf{F}^*]\!]\colon \wp(\mathbb{R}) \to \wp(\mathbb{R})$. By following derivations similar to those in Ex. 10, we derive $\frac{1}{4} \vdash [\delta_{-}^{\mathsf{Int}}]\ \mathsf{F}\ [\delta_{-}^{\mathsf{Int}}]$ for F. To apply rule (**star**), we consider the valid premises $\mathsf{Int} \in 1\text{-}ALip\langle \delta^{\mathsf{Int}}, \delta^{\mathsf{Int}} \rangle$ and $\mathsf{Int} \in 1\text{-}ALip\langle \delta^{\mathsf{Int}}, \delta^{\mathsf{Int}} \rangle$, and we choose $\uplus = +$ as a valid $\oplus$-bound. Then, K_∞ turns into the infinite series $1 + \sum_{i=0}^{\infty} (\frac{1}{4})^i$ which is a geometric series converging to $\frac{7}{3}$. Thus, by rule (**star**), we derive the judgment $\frac{7}{3} \vdash [\delta_{-}^{\mathsf{Int}}]\ \mathsf{F}^*\ [\delta_{-}^{\mathsf{Int}}]$ and, by Thm. 4, we can conclude $[\![\mathsf{F}^*]\!] \in \frac{7}{3}\text{-}ALip\langle \delta_{-}^{\mathsf{Int}}, \delta_{-}^{\mathsf{Int}} \rangle$. Note that (**bound-star**) cannot by applied since there is no m such that $*$-Bound(F^*, m) holds. ■

As a direct consequence of Thm. 4, if we instantiate the abstractions with $\eta = \rho = id$, then the deductive rules of Fig. 5 derive judgments for the LP of programs (Def. 5). This is because all the predicates on abstractions, such as $\eta \in t\text{-}ALip\langle \delta^{\rho}, \delta^{\eta} \rangle$, becomes trivially true by Lem. 1.

Corollary 3. $k \vdash [\delta^{id}]\ \mathsf{P}\ [\delta^{id}] \ \Rightarrow\ [\![\mathsf{P}]\!]$ *is k-LP w.r.t.* $\langle \delta, \delta \rangle$

6 Conclusion

Our work is not the first to adapt the LC notion to specific user-needs. For instance, ϵ-*sensitivity* [51] can be seen as the probabilistic counterpart of LC, requiring an ϵ-bounded maximum divergence of output distributions for probabilistic functions. Sensitivity allows to model and reason about security-relevant properties like ϵ-differential privacy [20], while it does not suite more complex scenarios, like (ϵ, δ)-differential privacy [19], requiring distance metrics not satisfying the triangle inequality. Thus, de Amorin et al. [18] proposed a categorical framework to encode relational properties, like (ϵ, δ)-differential privacy, into equivalent sensitivity-like properties over a suitable metric space. In other words, the framework lifts sensitivity from the standard Max Divergence metric to more complex, non-metric divergences like Skew Divergence. These generalizations account for probabilistic computations but not for qualitative aspects of (deterministic) computations like ALC. Qualitative aspects are considered by Bonchi et al. [2], who proved the soundness of *up-to techniques* [50] from the qualitative bisimilarity setting to the quantitative metric setting. Specifically, the paper gives sufficient conditions for the soundness of the quantitative version of up-to contexts in systems that have a (co-)algebraic structure (e.g., a bialgebra). That is, contexts are *non-expansive* with respect to the behavioral metric, which is the quantitative analogue of behavioral equivalence being a congruence. This mixed qualitative and quantitative formalization is obtained by adopting quantale-based metric spaces, thus generalizing the ordinary LC setting. Still, non-expansiveness requires a non-increasing distance change, thus potentially generalizing LC only when $k = 1$. We plan to formally investigate the relation between ALC and the aforementioned approaches as a future work.

As stated in Sec. 2.1, basic commands could be instantiated differently, for instance with probabilistic or nondeterministic assignments. In this work we considered only deterministic assignments and $\oplus$ is defined simply as the semantic join. Extending ALC to a richer probabilistic language is an interesting direction, and suitable adjustments to the join rule should make this possible. In this context we would expect standard metrics for probabilistic programs (e.g., the total variation or Kantorovich metrics) to work as well.

The proposed ALC notion is a *global* property, in the sense that it is universally quantified over all inputs. As a future work, we plan to formalize its *local* version, namely requiring ALC over a strict subset of inputs, and study its relation with other local properties in the context of abstract interpretation [3,4,6]. Dropping the universal quantification may invalidate the correlation already established between the global counterparts. Also, reasoning about local properties may be more challenging, as the proposed deductive system requires nontrivial modifications to be used for proving ALC on a subset of executions.

Another interesting future direction consists in considering weaker abstraction notions able to formalize properties that do not necessarily admit a best abstraction function, such as the domain of convex polyhedra [29], and more generally, as the weak closures defined in [41]. Finally, in [41] it is proved that, under certain assumptions, there is a correspondence between the completeness property in abstract interpretation and ANI in language-based security [24,25,37,22]. Such a connection suggests a potential relation between ALC and ANI, as well as to other quantitative program properties [38,39,40,42,43,44,55], which deserves further investigation in several directions, both theoretical and practical.

Finally, in Thm. 4, we proved that the proof system presented in Fig. 5 is sound. However, in its current form, the proof system is incomplete. As a consequence, the tightest Lipschitz constant derivable may be strictly greater than the actual one. The main source of this limitation lies in the treatment of the Kleene star when the body of a program P is not contracting (i.e., when $k \geq 1$). In such cases, even the (**bound-star**) rule may be ineffective, since the $*$-Bound predicate is not guaranteed to hold (see, e.g., Ex. 11). A more thorough investigation is required and we leave it for a future extension of this work.

Acknowledgements We wish to thank the anonymous reviewers of FoSSaCS 2026 for their detailed comments. This work was partially supported by the project SERICS (PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU; by the project PRIN2022PNRR "RAP-ARA" (PE6) - codice MUR: P2022HXNSC; by the SAIF project, funded by the "France 2030" government investment plan managed by the French National Research Agency, under the reference ANR-23-PEIA-0006.

References

1. Kumail Alhamoud, Hasan Abed Al Kader Hammoud, Motasem Alfarra, and Bernard Ghanem. Generalizability of adversarial robustness under distribution shifts. *Trans. Mach. Learn. Res.*, 2023, 2023. URL: https://openreview.net/forum?id=XNFo3dQiCJ.
2. Filippo Bonchi, Barbara König, and Daniela Petrisan. Up-to techniques for behavioural metrics via fibrations. *Math. Struct. Comput. Sci.*, 33(4-5):182–221, 2023. doi:10.1017/S0960129523000166.
3. Roberto Bruni, Roberto Giacobazzi, Roberta Gori, and Francesco Ranzato. 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*, pages 1–13. IEEE, 2021. doi:10.1109/LICS52264.2021.9470608.
4. Roberto Bruni, Roberto Giacobazzi, Roberta Gori, and Francesco Ranzato. A correctness and incorrectness program logic. *J. ACM*, 70(2):15:1–15:45, 2023. doi:10.1145/3582267.
5. Marco Campion, Mila Dalla Preda, and Roberto Giacobazzi. Partial (in)completeness in abstract interpretation: limiting the imprecision in program analysis. *Proc. ACM Program. Lang.*, 6(POPL):1–31, 2022. doi:10.1145/3498721.
6. Marco Campion, Mila Dalla Preda, Roberto Giacobazzi, and Caterina Urban. Monotonicity and the precision of program analysis. *Proc. ACM Program. Lang.*, 8(POPL):1629–1662, 2024. doi:10.1145/3632897.
7. Marco Campion, Mila Dalla Preda, Roberto Giacobazzi, and Caterina Urban. A logic for the imprecision of abstract interpretations. *Proc. ACM Program. Lang.*, 10(POPL), 2026. doi:10.1145/3776707.
8. Marco Campion, Isabella Mastroeni, and Caterina Urban. Relating distances and abstractions - an abstract interpretation perspective. In Hakjoo Oh and Yulei Sui, editors, *Static Analysis - 32nd International Symposium, SAS 2025, Singapore, October 13-14, 2025, Proceedings*, volume 16100 of *Lecture Notes in Computer Science*, pages 249–277. Springer, 2025. doi:10.1007/978-3-032-07106-4_11.
9. Marco Campion, Caterina Urban, Mila Dalla Preda, and Roberto Giacobazzi. A formal framework to measure the incompleteness of abstract interpretations. In Manuel V. Hermenegildo and José F. Morales, editors, *Static Analysis - 30th International Symposium, SAS 2023, Cascais, Portugal, October 22-24, 2023, Proceedings*, volume 14284 of *Lecture Notes in Computer Science*, pages 114–138. Springer, 2023. doi:10.1007/978-3-031-44245-2_7.
10. Swarat Chaudhuri, Sumit Gulwani, and Roberto Lublinerman. Continuity Analysis of Programs. In *Proceedings of the 37th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '10, page 57–70, New York, NY, USA, 2010. Association for Computing Machinery. doi:10.1145/1706299.1706308.
11. Swarat Chaudhuri, Sumit Gulwani, and Roberto Lublinerman. Continuity and Robustness of Programs. 55(8):107–115, 2012. doi:10.1145/2240236.2240262.
12. Swarat Chaudhuri, Sumit Gulwani, Roberto Lublinerman, and Sara NavidPour. Proving Programs Robust. In Tibor Gyimóthy and Andreas Zeller, editors, *SIGSOFT/FSE'11 19th ACM SIGSOFT Symposium on the Foundations of Software Engineering (FSE-19) and ESEC'11: 13th European Software Engineering Conference (ESEC-13), Szeged, Hungary, September 5-9, 2011*, pages 102–112. ACM, 2011. doi:10.1145/2025113.2025131.

13. Patrick Cousot. *Principles of Abstract Interpretation*. The MIT Press, Cambridge, Mass., 2021.
14. Patrick Cousot and Radhia Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In Robert M. Graham, Michael A. Harrison, and Ravi Sethi, editors, *Conference Record of the Fourth ACM Symposium on Principles of Programming Languages, Los Angeles, California, USA, January 1977*, pages 238–252. ACM, 1977. doi:10.1145/512950.512973.
15. Patrick Cousot and Radhia Cousot. Systematic design of program analysis frameworks. In Alfred V. Aho, Stephen N. Zilles, and Barry K. Rosen, editors, *Conference Record of the Sixth Annual ACM Symposium on Principles of Programming Languages, San Antonio, Texas, USA, January 1979*, pages 269–282. ACM Press, 1979. doi:10.1145/567752.567778.
16. Patrick Cousot and Radhia Cousot. An abstract interpretation-based framework for software watermarking. In *Proceedings of the 31st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '04, pages 173–185, New York, NY, USA, 2004. Association for Computing Machinery. doi:10.1145/964001.964016.
17. Pedro R. D'Argenio, Gilles Barthe, Sebastian Biewer, Bernd Finkbeiner, and Holger Hermanns. Is your software on dope? - formal analysis of surreptitiously "enhanced" programs. In Hongseok Yang, editor, *Programming Languages and Systems - 26th European Symposium on Programming, ESOP 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22-29, 2017, Proceedings*, volume 10201 of *Lecture Notes in Computer Science*, pages 83–110. Springer, 2017. doi:10.1007/978-3-662-54434-1_4.
18. Arthur Azevedo de Amorim, Marco Gaboardi, Justin Hsu, and Shin-ya Katsumata. Probabilistic relational reasoning via metrics. In *34th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2019, Vancouver, BC, Canada, June 24-27, 2019*, pages 1–19. IEEE, 2019. doi:10.1109/LICS.2019.8785715.
19. Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: privacy via distributed noise generation. In *Proceedings of the 24th Annual International Conference on The Theory and Applications of Cryptographic Techniques*, EUROCRYPT'06, pages 486–503, Berlin, Heidelberg, 2006. Springer-Verlag. doi:10.1007/11761679_29.
20. Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In Shai Halevi and Tal Rabin, editors, *Theory of Cryptography*, pages 265–284, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.
21. Mahyar Fazlyab, Alexander Robey, Hamed Hassani, Manfred Morari, and George J. Pappas. Efficient and accurate estimation of lipschitz constants for deep neural networks. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 11423–11434, 2019. URL: https://proceedings.neurips.cc/paper/2019/hash/95e1533eb1b20a97777749fb94fdb944-Abstract.html.
22. R. Giacobazzi and I. Mastroeni. Adjoining classified and unclassified information by abstract interpretation. *Journal of Computer Security*, 18(5):751 – 797, 2010.

23. Roberto Giacobazzi, Francesco Logozzo, and Francesco Ranzato. Analyzing program analyses. In Sriram K. Rajamani and David Walker, editors, *Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2015, Mumbai, India, January 15-17, 2015*, pages 261–273. ACM, 2015. doi:10.1145/2676726.2676987.
24. Roberto Giacobazzi and Isabella Mastroeni. Abstract non-interference: parameterizing non-interference by abstract interpretation. In Neil D. Jones and Xavier Leroy, editors, *Proceedings of the 31st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2004, Venice, Italy, January 14-16, 2004*, pages 186–197. ACM, 2004. doi:10.1145/964001.964017.
25. Roberto Giacobazzi and Isabella Mastroeni. Abstract non-interference: A unifying framework for weakening information-flow. *ACM Trans. Priv. Secur.*, 21(2):9:1–9:31, 2018. doi:10.1145/3175660.
26. Roberto Giacobazzi, Isabella Mastroeni, and Elia Perantoni. Adversities in abstract interpretation - accommodating robustness by abstract interpretation. *ACM Trans. Program. Lang. Syst.*, 46(2):1–31, 2024. doi:10.1145/3649309.
27. Roberto Giacobazzi, Francesco Ranzato, and Francesca Scozzari. Making abstract interpretations complete. *J. ACM*, 47(2):361–416, 2000. doi:10.1145/333979.333989.
28. Joseph A. Goguen and José Meseguer. Security policies and security models. In *1982 IEEE Symposium on Security and Privacy, Oakland, CA, USA, April 26-28, 1982*, pages 11–20. IEEE Computer Society, 1982. doi:10.1109/SP.1982.10014.
29. Branko Grünbaum, Victor Klee, Micha A Perles, and Geoffrey Colin Shephard. *Convex polytopes*, volume 16. Springer, 1967.
30. Yujia Huang, Huan Zhang, Yuanyuan Shi, J. Zico Kolter, and Anima Anandkumar. Training certifiably robust neural networks with efficient local lipschitz bounds. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 22745–22757, 2021. URL: https://proceedings.neurips.cc/paper/2021/hash/c055dcc749c2632fd4dd806301f05ba6-Abstract.html.
31. Ziwei Ji and Matus Telgarsky. Directional convergence and alignment in deep learning. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL: https://proceedings.neurips.cc/paper/2020/hash/c76e4b2fa54f8506719a5c0dc14c2eb9-Abstract.html.
32. Lin Li, Yifei Wang, Chawin Sitawarin, and Michael W. Spratling. Oodrobustbench: a benchmark and large-scale analysis of adversarial robustness under distribution shift. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. URL: https://openreview.net/forum?id=kAFevjEYsz.
33. Dennis Liew, Tiago Cogumbreiro, and Julien Lange. Sound and partially-complete static analysis of data-races in GPU programs. *Proc. ACM Program. Lang.*, 8(OOPSLA2):2434–2461, 2024. doi:10.1145/3689797.
34. Francesco Logozzo. Towards a quantitative estimation of abstract interpretations. In *Workshop on Quantitative Analysis of Software*. Microsoft, June 2009. URL: https://www.microsoft.com/en-us/research/publication/towards-a-quantitative-estimation-of-abstract-interpretations/.

35. Ulrike von Luxburg and Olivier Bousquet. Distance-based classification with lipschitz functions. *Journal of Machine Learning Research*, 5(Jun):669–695, 2004.
36. Luca Marzari, Isabella Mastroeni, and Alessandro Farinelli. Advancing neural network verification through hierarchical safety abstract interpretation. *CoRR*, abs/2505.05235, 2025. URL: https://doi.org/10.48550/arXiv.2505.05235, arXiv:2505.05235, doi:10.48550/ARXIV.2505.05235.
37. I. Mastroeni. On the rôle of abstract non-interference in language-based security. In K. Yi, editor, *Third Asian Symp. on Programming Languages and Systems (APLAS '05)*, volume 3780 of *Lecture Notes in Computer Science*, pages 418–433. Springer-Verlag, 2005.
38. Isabella Mastroeni. Abstract domain adequacy. *Int. J. Softw. Tools Technol. Transf.*, 26(6):747–765, 2024. URL: https://doi.org/10.1007/s10009-024-00774-x, doi:10.1007/S10009-024-00774-X.
39. Isabella Mastroeni. Abstract local completeness. In Krishna Shankaranarayanan, Sriram Sankaranarayanan, and Ashutosh Trivedi, editors, *Verification, Model Checking, and Abstract Interpretation*, pages 3–25, Cham, 2025. Springer Nature Switzerland.
40. Isabella Mastroeni and Michele Pasqua. Statically analyzing information flows: an abstract interpretation-based hyperanalysis for non-interference. In Chih-Cheng Hung and George A. Papadopoulos, editors, *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing, SAC 2019*, pages 2215–2223. ACM, 2019. doi:10.1145/3297280.3297498.
41. Isabella Mastroeni and Michele Pasqua. Domain precision in galois connection-less abstract interpretation. In Manuel V. Hermenegildo and José F. Morales, editors, *Static Analysis - 30th International Symposium, SAS 2023, Cascais, Portugal, October 22-24, 2023, Proceedings*, volume 14284 of *Lecture Notes in Computer Science*, pages 434–459. Springer, 2023. doi:10.1007/978-3-031-44245-2_19.
42. Isabella Mastroeni and Michele Pasqua. Abstract interpretation-based verification for confidentiality: Information hiding and code protection by abstract interpretation. *ACM Trans. on Privacy and Security*, pages 1–28, 2025. doi:10.1145/3786347.
43. Denis Mazzucato, Marco Campion, and Caterina Urban. Quantitative input usage static analysis. In Nathaniel Benz, Divya Gopinath, and Nija Shi, editors, *NASA Formal Methods - 16th International Symposium, NFM 2024, Moffett Field, CA, USA, June 4-6, 2024, Proceedings*, volume 14627 of *Lecture Notes in Computer Science*, pages 79–98. Springer, 2024. doi:10.1007/978-3-031-60698-4_5.
44. Denis Mazzucato, Marco Campion, and Caterina Urban. Quantitative static timing analysis. In Roberto Giacobazzi and Alessandra Gorla, editors, *Static Analysis - 31st International Symposium, SAS 2024, Pasadena, CA, USA, October 20-22, 2024, Proceedings*, volume 14995 of *Lecture Notes in Computer Science*, pages 268–299. Springer, 2024. doi:10.1007/978-3-031-74776-2_11.
45. Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In Johannes Fürnkranz and Thorsten Joachims, editors, *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, pages 807–814. Omnipress, 2010. URL: https://icml.cc/Conferences/2010/papers/432.pdf.
46. Yurii Nesterov. *Lectures on Convex Optimization*. Springer Publishing Company, Incorporated, 2nd edition, 2018. doi:10.1007/978-3-319-91578-4.
47. Peter W. O'Hearn. Incorrectness logic. *Proc. ACM Program. Lang.*, 4(POPL):10:1–10:32, 2020. doi:10.1145/3371078.

48. Alessandra Di Pierro, Chris Hankin, and Herbert Wiklicky. Approximate non-interference. *J. Comput. Secur.*, 12(1):37–82, 2004. doi:10.3233/JCS-2004-12103.
49. Alessandra Di Pierro and Herbert Wiklicky. Measuring the precision of abstract interpretations. In Kung-Kiu Lau, editor, *Logic Based Program Synthesis and Transformation, 10th International Workshop, LOPSTR 2000 London, UK, July 24-28, 2000, Selected Papers*, volume 2042 of *Lecture Notes in Computer Science*, pages 147–164. Springer, 2000. doi:10.1007/3-540-45142-0_9.
50. Damien Pous. Complete lattices and up-to techniques. In *Proceedings of the 5th Asian Conference on Programming Languages and Systems*, APLAS'07, pages 351–366, Berlin, Heidelberg, 2007. Springer-Verlag.
51. Jason Reed and Benjamin C. Pierce. Distance makes the types grow stronger: a calculus for differential privacy. *SIGPLAN Not.*, 45(9):157–168, September 2010. doi:10.1145/1932681.1863568.
52. Xavier Rival and Kwangkeun Yi. *Introduction to static analysis: an abstract interpretation perspective*. Mit Press, 2020.
53. Daniel Schoepe and Andrei Sabelfeld. Understanding and enforcing opacity. In *2015 IEEE 28th Computer Security Foundations Symposium*, pages 539–553, 2015. doi:10.1109/CSF.2015.41.
54. Pascal Sotin. Quantifying the Precision of Numerical Abstract Domains. Research report, February 2010. URL: https://inria.hal.science/inria-00457324.
55. Caterina Urban and Peter Müller. An abstract interpretation framework for input data usage. In Amal Ahmed, editor, *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*, volume 10801 of *Lecture Notes in Computer Science*, pages 683–710. Springer, 2018. doi:10.1007/978-3-319-89884-1_24.
56. Bohang Zhang, Du Jiang, Di He, and Liwei Wang. Rethinking lipschitz neural networks and certified robustness: A boolean function perspective. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL: http://papers.nips.cc/paper_files/paper/2022/hash/7b04ec5f2b89d7f601382c422dfe07af-Abstract-Conference.html.
57. Bohang Zhang, Du Jiang, Di He, and Liwei Wang. Rethinking lipschitz neural networks and certified robustness: A boolean function perspective. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 19398–19413. Curran Associates, Inc., 2022. URL: https://proceedings.neurips.cc/paper_files/paper/2022/file/7b04ec5f2b89d7f601382c422dfe07af-Paper-Conference.pdf.