Plurimetric Fuzz: A Linear Type System for Bounding the Sensitivity of Vector Functions

2023 Days of the Scalp Working Group
Formal Structures for Computation and Proofs

Victor Sannier
Joint work with Patrick Baillot

Laboratoire CRIStAL de Lille

1. Introduction

What is the sensitivity of a function?
What is differential privacy?

Setting: privacy-preserving queries to a database

An analyst makes queries to a database.

Example

- What is the average salary of employees?
- How many patients have the flu?
- How old is Mr. Doe?
- What is the salary of Mrs. Smith?

How to protect the privacy of individuals?

Anonymising the data is not enough

```
let query (db : database) : float =
    let dupont = filter
        (fun p -> p.birthyear = 1970
                  && p.postcode = 75000
                  && p.weight >= 70)
        db |> head
    in dupont.salary
```

One can find an individual by cross-referencing information.

What is differential privacy?

Let

- D be a database,
- x be an individual,
- q be a query.

We want $q(D)$ and $q\left(D \cup \{x\}\right)$ to be indistinguishable.

Formal definition of differential privacy (Dwork et al. 2006)

Let

- $\mathcal{D}$ be the set of databases,
- X be an arbitrary space (usually R^n),
- $f : \mathcal{D} \to X$ be a probabilistic algorithm.

Definition

f is (ε, δ)-*differentially private* if for all $D, D' \in \mathcal{D}$ such that $d(D, D') \leq 1$ and all $S \subseteq X$:

$$\Pr\big(f(D) \in S\big) \leq e^{\epsilon} \Pr\big(f(D') \in S\big) + \delta$$

The distance between two databases is the number of rows that differ between them.

Query sensitivity

Definition

A function $f\colon X \to Y$ between two metric spaces (X, d_X) and (Y, d_Y) is said to be *k-sensitive* if for all $x, x' \in X$, we have $d_Y\big(f(x), f(x')\big) \le k \cdot d_X(x, x')$.

It is a measure of the *specificity*, that is of the *continuity* of the function.

Query sensitivity

Definition

A function $f \colon X \to Y$ between two metric spaces (X, d_X) and (Y, d_Y) is said to be *k-sensitive* if for all $x, x' \in X$, we have $d_Y\big(f(x), f(x')\big) \leq k \cdot d_X(x, x')$.

It is a measure of the *specificity*, that is of the *continuity* of the function.

Remark

The sensitivity of a query depends on the metric chosen on the source and target spaces.

Adding noise

In order to guarantee differential privacy *automatically*, one can add noise to the result of the query.

Theorem (Dwork and Roth 2014)

Let $f : X \to \mathbb{R}^k$ *be a* k*-sensitive function for the* L^1 *distance. The function* $\mathcal{M}_{Laplace}(f, \epsilon)$ *defined by:*

$$\mathcal{M}_{Laplace}(f, \epsilon)(x) = \left(\pi_1(f(x)) + Y_1, \ldots, \pi_n(f(x)) + Y_n\right)$$

where the Y_i *are i.i.d. according to a Laplace distribution of parameter* k/ϵ *is* $(\epsilon, 0)$*-DP.*

Reminder. $d_1(x, y) = \sum_{i=1}^{k} |x_i - y_i|$.

The role of logic

Program on a database

↓ **Type system**

Program whose sensitivity is known

↓ Noise addition mechanism

Differentially private program

2. The Fuzz functional language and its extensions

Sensitivity is an affine resource
which can be tracked by a type system.

Linear logic (Girard 1987)

Linear logic allows to reason about resources, state changes, etc.

Example

The $\otimes$ connective allows to model the simultaneous presence of two resources, and $\multimap$ their transformation.

$$H_2 \otimes H_2 \otimes O_2 \multimap H_2O \otimes H_2O$$

Deduction rules model consumption, distribution, etc.

Fuzz (Reed and Pierce 2010) is a functional language with affine types to reason about the sensitivity of programs.

Example

$$\frac{[x : A]_{s_1} \vdash b : B \quad [x : A]_{s_2} \vdash c : C}{[x : A]_{s_1+s_2} \vdash (b, c) : B \otimes C}$$

If $x \mapsto f(x)$ is s_1-sensitive and $x \mapsto g(x)$ is s_2-sensitive, then $x \mapsto \big(f(x), g(x)\big)$ is $(s_1 + s_2)$-sensitive.

Semantic soundness theorem (Azevedo de Amorim et al. 2017)

Types are interpreted as metric spaces,

Interpretation of types

- $A \otimes B$ is interpreted by the space $[\![A]\!] \times [\![B]\!]$ endowed with the metric d_1;
- $A \multimap B$ is interpreted by the space of 1-sensitive functions from $[\![A]\!]$ to $[\![B]\!]$ endowed with some metric $d_{\multimap}$.

and terms as functions between them.

Theorem

If $[x : A]_s \vdash e : B$ is derivable, then the following function is s-sensitive:

$$\begin{array}{rccc} [\![e]\!] & : & [\![A]\!] & \longrightarrow & [\![B]\!] \\ & & x & \longmapsto & [\![e]\!](x) \end{array}$$

Motivation for extending Fuzz

Motivation. Measure and track sensitivities for arbitrary L^p metrics, in particular to add noise according to different distributions:

- Laplace distribution for L^1,
- Gaussian distribution for L^2.

Reminder. For all $x, y \in \mathrm{R}^k$, $d_p(x, y) = \sqrt[p]{\sum_{i=1}^{k} |x_i - y_i|^p}$. For $p = 2$, this is the Euclidean metric.

Bunched Fuzz (Wunder et al. 2023)

The main idea is to introduce type constructors $\otimes_p$ and $\multimap_p$ for every parameter $p \in [1, +\infty]$. If $(a, b), (a', b') \in A \otimes_p B$, then

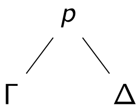
$$d_{\otimes_p}((a, b), (a', b')) = \sqrt[p]{d_A(a, a')^p + d_B(b, b')^p}.$$

Example

Semantically, we have rotate : Real $\otimes_2$ Real $\multimap_2$ Real $\otimes_2$ Real
but we do *not* have rotate : Real $\otimes_1$ Real $\multimap_1$ Real $\otimes_1$ Real

Bunched Fuzz contexts

The contexts are no longer lists of annotated variables, but trees.
For example, $\Gamma ,_p \Delta$ is the following tree:



and it is interpreted as $[\![\Gamma]\!] \otimes_p [\![\Delta]\!]$.

Example (Tensor introduction rule)

$$\frac{\Gamma \vdash a : A \qquad \Delta \vdash b : B}{\Gamma ,_p \Delta \vdash (a, b) : A \otimes_p B} \otimes I$$

Subject reduction property

Let $\downarrow$ be the evaluation relation for a standard big-step operational semantics.

~~Property~~ (subject reduction)

If $\emptyset \vdash a : A$ and $a \downarrow v$, then $\emptyset \vdash v : A$.

Bunched Fuzz does not satisfy the subject reduction property.

3. Plurimetric Fuzz

We extend Fuzz to L^p metrics using ordinary contexts to recover the subject reduction property.

Plurimetric Fuzz Types

Plurimetric Fuzz has:

- primitives types: Unit, Real, etc.;
- sum types: $A \oplus B$;
- product types: $A \otimes_p B$ for $p \in [1, +\infty]$;
- function types: $A \multimap_p B$ for $p \in [1, +\infty]$;
- exponential types: $!_s A$ for $s \in (0, +\infty]$;

Plurimetric Fuzz Types

Plurimetric Fuzz has:

- primitives types: Unit, Real, etc.;
- sum types: $A \oplus B$;
- product types: $A \otimes_p B$ for $p \in [1, +\infty]$;
- function types: $A \multimap_p B$ for $p \in [1, +\infty]$;
- exponential types: $!_s A$ for $s \in (0, +\infty]$;
- distribution types: $\bigcirc A$;

Plurimetric Fuzz Types

Plurimetric Fuzz has:

- primitives types: Unit, Real, etc.;
- sum types: $A \oplus B$;
- product types: $A \otimes_p B$ for $p \in [1, +\infty]$;
- function types: $A \multimap_p B$ for $p \in [1, +\infty]$;
- exponential types: $!_s A$ for $s \in (0, +\infty]$;
- distribution types: $\bigcirc A$;
- recursive types: $\mu\alpha.A$.

Plurimetric Fuzz Contexts

Idea. We only allow one parameter per context, so these are lists again, but with an additional annotation: $(p)\ \Gamma$.
We lose some flexibility, but we gain the subject reduction property.

Example

Tensor introduction rule

$$\frac{(p)\ \Gamma \vdash a : A \qquad (p)\ \Delta \vdash b : B}{(p)\ \mathrm{Contr}\,(p; \Gamma; \Delta) \vdash (a, b) : A \otimes_p B} \otimes I$$

Definition

$$\mathrm{Contr}\left(p; [x : A]_{s1}; [x : A]_{s2}\right) = [x : A]_{\sqrt[p]{s_1^p + s_2^p}}$$

Weakening rules

Two rules allow for changing the parameter of a context:

$$\frac{(p)\ \Gamma \vdash a : A \quad \Gamma \leq \Delta \quad p \geq q}{(q)\ \Delta \vdash a : A} \geq W \qquad \frac{(p)\ \Gamma \vdash a : A \quad \Gamma \leq \Delta \quad p \leq q}{(q)\ 2^{1/p - 1/q} \cdot \Delta \vdash a : A} \leq W$$

Lemma (Subtyping)

For all types A, B and parameters $p \leq q$, there exists two terms le *and* ge *such that*

$$\emptyset \vdash \mathsf{le} : A \otimes_p B \multimap_p A \otimes_q B \qquad \textit{and} \qquad \emptyset \vdash \mathsf{ge} : !_{2^{1/p-1/q}}(A \otimes_q B) \multimap_q A \otimes_p B$$

Examples

- lists, defined as $\mu\alpha.\mathsf{Unit} \oplus (A \otimes_p \alpha)$ generalising the list type constructor in Fuzz, and functions on them (map, fold, take, etc.);
- matrices, defined as $(A \otimes_p \ldots \otimes_p A) \otimes_1 \ldots \otimes_1 (A \otimes_p \ldots \otimes_p A)$ and functions on them;
- some machine-learning algorithms (gradient descent, k-means, k-nn clustering, etc.).

Translation of Fuzz judgements

Theorem

For all parameters p, the image by the mapping $P(p)$ of a (minimal) derivable judgement in Fuzz is a (minimal) derivable judgement in Plurimetric Fuzz.

Example

$$[n : \mathsf{Nat}]_2 \vdash (1, n, n) : \mathsf{Nat} \otimes \mathsf{Nat} \otimes \mathsf{Nat}$$

translates to the following judgement for $p = 2$:

$$(2)\ [n : \mathsf{Nat}]_{\sqrt{2}} \vdash (1, n, n) : \mathsf{Nat} \otimes_2 \mathsf{Nat} \otimes_2 \mathsf{Nat}$$

Translation of Fuzz judgements

Theorem

For all parameters p, the image by the mapping $P(p)$ of a (minimal) derivable judgement in Fuzz is a (minimal) derivable judgement in Plurimetric Fuzz.

Example

$$[n : \mathsf{Nat}]_2 \vdash (1, n, n) : \mathsf{Nat} \otimes \mathsf{Nat} \otimes \mathsf{Nat}$$

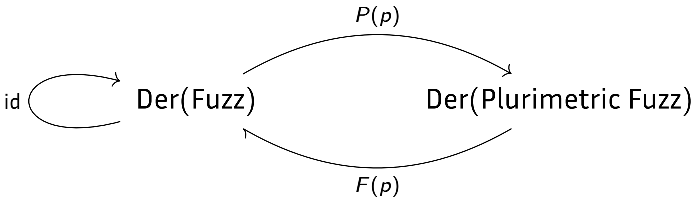
translates to the following judgement for $p = 2$:

$$(2)\ \ [n : \mathsf{Nat}]_{\sqrt{2}} \vdash (1, n, n) : \mathsf{Nat} \otimes_2 \mathsf{Nat} \otimes_2 \mathsf{Nat}$$

Remark

Not all Plurimetric Fuzz judgements can be obtained this way.

For all parameters p, the following diagram commutes:



Translation of primitive operations

The translation of the logical connectives does not extend to the primitive operations.

Example

For $p = 2$,

$$\frac{[x : \mathsf{Real}]_1 \vdash x : \mathsf{Real} \qquad [y : \mathsf{Real}]_1 \vdash y : \mathsf{Real}}{[x : \mathsf{Real}]_1, [y : \mathsf{Real}]_1 \vdash x + y : \mathsf{Real}} +$$

does not soundly translate to

$$\frac{(2)\ [x : \mathsf{Real}]_1 \vdash x : \mathsf{Real} \qquad (2)\ [y : \mathsf{Real}]_1 \vdash y : \mathsf{Real}}{(2)\ [x : \mathsf{Real}]_{\sqrt{1}}, [y : \mathsf{Real}]_{\sqrt{1}} \vdash x + y : \mathsf{Real}} +$$

Type safety

Proposition (substitution)

If we have

- $(p)\ \Delta \vdash a : A$;
- $(p)\ \Gamma, [x : A]_s \vdash b : B$;

then we have

$$(p)\ \text{Contr}\,(p; \Gamma; s\Delta) \vdash b[x \mapsto a] : B$$

Theorem (subject reduction)

If $(p)\ \emptyset \vdash a : A$ and $a \downarrow v$, then $(p)\ \emptyset \vdash v : A$.

Conclusion

Contribution: a standard (i.e., not bunched) type system that extends Fuzz to L^p metrics:

- such that the subject reduction property is satisfied
- with recursive types and a form of subtyping;
- that can be translated from and to Fuzz.

Conclusion

Contribution: a standard (i.e., not bunched) type system that extends Fuzz to L^p metrics:

- such that the subject reduction property is satisfied
- with recursive types and a form of subtyping;
- that can be translated from and to Fuzz.

Future work may include:

- generalisation to other metrics on distributions;
- automatic type checking and type inference;

References I

- Azevedo de Amorim, Arthur, Marco Gaboardi, Justin Hsu, Shin ya Katsumata, and Ikram Cherigui (Jan. 2017). "A semantic account of metric preservation". In: *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages*. ACM. DOI: 10.1145/3009837.3009890.
- Dwork, Cynthia, Frank McSherry, Kobbi Nissim, and Adam Smith (2006). "Calibrating Noise to Sensitivity in Private Data Analysis". In: *Theory of Cryptography*. Springer Berlin Heidelberg, pp. 265–284. DOI: 10.1007/11681878_14.
- Dwork, Cynthia and Aaron Roth (2014). "The Algorithmic Foundations of Differential Privacy". In: *Foundations and Trends in Theoretical Computer Science* 9.3–4, pp. 211–407. DOI: 10.1561/0400000042.
- Girard, Jean-Yves (1987). "Linear Logic". In: *Theoretical Computer Science* 50.1, pp. 1–102.

Reed, Jason and Benjamin C. Pierce (Sept. 2010). “Distance Makes the Types Grow Stronger: A Calculus for Differential Privacy”. In: *Proceedings of the 15th ACM SIGPLAN International Conference on Functional Programming*. ACM. DOI: 10.1145/1863543.1863568.

Wunder, June, Arthur Azevedo de Amorim, Patrick Baillot, and Marco Gaboardi (2023). “Bunched Fuzz: Sensitivity for Vector Metrics”. In: *Programming Languages and Systems*, pp. 451–478. DOI: 10.1007/978-3-031-30044-8_17.