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?
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

```haskell
let query (db : database) : float =
    let dupont = filter
        (fun p -> p.birthday = 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 $q(D \cup \{x\})$ 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 \( \mathbb{R}^n \)),
- \( f : \mathcal{D} \rightarrow X \) be a probabilistic algorithm.

**Definition**

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

\[
\Pr(f(D) \in S) \leq e^\varepsilon \Pr(f(D') \in S) + \delta
\]

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

A function $f : X \to Y$ 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 $d_Y(f(x), f(x')) \leq k \cdot d_X(x, x')$.

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

A function \( f : X \to Y \) 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 \( d_Y(f(x), f(x')) \leq k \cdot d_X(x, x') \).

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 $f : X \rightarrow \mathbb{R}^k$ 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 $k/\epsilon$ 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 allows to reason about resources, state changes, etc.

Example

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

$$H_2 \otimes H_2 \otimes O_2 \rightarrow H_2O \otimes H_2O$$

Deduction rules model consumption, distribution, etc.
The Fuzz functional language

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 \( x \mapsto (f(x), g(x)) \) is \((s_1 + s_2)\)-sensitive.
Types are interpreted as metric spaces,

**Interpretation of types**

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

and terms as functions between them.

**Theorem**

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

$$[[e]] : [[A]] \rightarrow [[B]]$$

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

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

**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:

```
  p
 / \  \\
Γ   Δ
```

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

Example (Tensor introduction rule)

```
$\Gamma \vdash a : A \quad \Delta \vdash b : B$

$\Gamma, p \Delta \vdash (a, b) : A \otimes_p B$
```

\( \otimes I \)
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 has:

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

\[
\begin{align*}
(p) \Gamma \vdash a : A & \quad (p) \Delta \vdash b : B \\
\hline
(p) \text{Contr} (p; \Gamma; \Delta) \vdash (a, b) : A \otimes_B B
\end{align*}
\]

**Definition**

\[
\text{Contr} (p; [x : A]_{s1}; [x : A]_{s2}) = [x : A]_\sqrt{s_1^p + s_2^p}
\]
Two rules allow for changing the parameter of a context:

\[
\begin{align*}
(p) \quad & \Gamma \vdash a : A \quad \Gamma \leq \Delta \quad p \geq q \quad \geq W \\
(q) \quad & \Delta \vdash a : A
\end{align*}
\]

\[
\begin{align*}
(p) \quad & \Gamma \vdash a : A \quad \Gamma \leq \Delta \quad p \leq q \quad \leq W \\
(q) \quad & \Delta \vdash a : A
\end{align*}
\]

Lemma (Subtyping)

For all types \(A, B\) and parameters \(p \leq q\), there exists two terms \(\text{le}\) and \(\text{ge}\) such that

\[
\emptyset \vdash \text{le} : A \otimes_p B \rightharpoonup_p A \otimes_q B \quad \text{and} \quad \emptyset \vdash \text{ge} : \!_{2^{1/p-1/q}}(A \otimes_q B) \rightharpoonup_q A \otimes_p B
\]
Examples

- lists, defined as $\mu \alpha. \text{Unit } \oplus (A \otimes_p \alpha)$ 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**

\[
\left[n : \text{Nat}\right]_2 \vdash (1, n, n) : \text{Nat} \otimes \text{Nat} \otimes \text{Nat}
\]

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

\[
(2) \left[n : \text{Nat}\right]_{\sqrt{2}} \vdash (1, n, n) : \text{Nat} \otimes_2 \text{Nat} \otimes_2 \text{Nat}
\]

Remark

Not all Plurimetric Fuzz judgements can be obtained this way.
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**

\[
[n : Nat]_2 \vdash (1, n, n) : Nat \otimes Nat \otimes Nat
\]

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

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

**Remark**

Not all Plurimetric Fuzz judgements can be obtained this way.
For all parameters $p$, the following diagram commutes:
The translation of the logical connectives does not extend to the primitive operations.

Example

For \( p = 2 \),

\[
\frac{[x : \text{Real}]_1 \vdash x : \text{Real} \quad [y : \text{Real}]_1 \vdash y : \text{Real}}{[x : \text{Real}]_1, [y : \text{Real}]_1 \vdash x + y : \text{Real}} +
\]

does not soundly translate to

\[
\frac{(2) [x : \text{Real}]_1 \vdash x : \text{Real} \quad (2) [y : \text{Real}]_1 \vdash y : \text{Real}}{(2) [x : \text{Real}]_{\sqrt{1}}, [y : \text{Real}]_{\sqrt{1}} \vdash x + y \vdash : \text{Real}} +
\]
Proposition (substitution)

If we have

1. \((p) \Delta \vdash a : A;\)
2. \((p) \Gamma, [x : A]s \vdash b : B;\)

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.

Future work may include:
- generalisation to other metrics on distributions;
- automatic type checking and type inference;
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;


