
From Linear Logic to Type Systems for Sensitivity Analysis and Differential Privacy Verification

Victor Sannier
September 21 2026

Supervisor. Patrick Baillot
Advisor. Marco Gaboardi



Context

How to use **logical methods** to reason about (probabilistic) programs and statically prove their properties such as **differential privacy and sensitivity?**

- ▶ Introduction
 - ▶ Introduction to Formal Logic
 - ▶ Introduction to Differential Privacy
 - ▶ Fuzz: Linear Types for Sensitivity Analysis
- ▶ Contributions

Context

How to use **logical methods** to reason about (probabilistic) programs and statically prove their properties such as **differential privacy and sensitivity**?

- ▶ Introduction
 - ▶ Introduction to Formal Logic
 - ▶ Introduction to Differential Privacy
 - ▶ Fuzz: Linear Types for Sensitivity Analysis
- ▶ Contributions

1. Introduction to Formal Logic

Proofs, Programs and Arrows

What is a Proof?

informal proofs

a cogent paragraph written in natural language

Proof. Let n be an odd number. Suppose for the sake of contradiction that n^2 is even. Then we have $2 \mid n^2$ and since 2 is prime, we have $2 \mid n$. QEA. ■

formal proofs

a sequence of deduction rules applied to formulas

$$\frac{\frac{\frac{\vdots}{2 \mid a^2 \vdash 2 \mid a} W}{2 \nmid a, 2 \mid a^2 \vdash 2 \mid a} \rightarrow E \quad \frac{2 \nmid a, 2 \mid a^2 \vdash \perp}{2 \nmid a \vdash 2 \nmid a^2} \rightarrow I}{\vdash 2 \nmid a \rightarrow 2 \nmid a^2} \forall I$$
$$\vdash \forall n, 2 \nmid n \rightarrow 2 \nmid n^2$$

Reasoning on Proofs

A formal proof is a **mathematical object**, we can ask:

- ▶ is a proof correct in logic L ?
- ▶ does a proof of P exist in logic L ? in logic L' ?
(*incompleteness, (relative) consistency*, etc.)
- ▶ can we transform π into π' such that...?

What is a (Typed Functional) Program?

A program in the λ -calculus is a term constructed using:

constructs

constants

functions

applications

constructors and destructors

examples

$0 : \text{Nat}, \text{true} : \text{Bool}$

$\text{square} := \lambda x. x \times x : \text{Nat} \rightarrow \text{Nat}$

$\text{not true} : \text{Bool}, \text{square } 5 : \text{Nat}$

pairs, projections, conditionals

This language can be seen as a fragment of OCaml or Haskell.

What is a (Typed Functional) Program?

A program in the λ -calculus is a term constructed using:

constructs

constants

functions

applications

constructors and destructors

examples

$0 : \text{Nat}, \text{true} : \text{Bool}$

$\text{square} := \lambda x. x \times x : \text{Nat} \rightarrow \text{Nat}$

$\text{not true} : \text{Bool}, \text{square } 5 : \text{Nat}$

pairs, projections, conditionals

This language can be seen as a fragment of OCaml or Haskell.

Reasoning on Programs

Programs as previously defined are **mathematical objects**.

type checking does program P have type τ ?

program evaluation if $\lambda x.s : \sigma \rightarrow \tau$ and $t : \sigma$, then $(\lambda x.s) t : \tau \rightarrow_{\beta} s[t/x] : \tau$,

program synthesis does there exist a program of type τ ?

Answers to these questions depend on the type system we choose.

Reasoning on Programs

Programs as previously defined are **mathematical objects**.

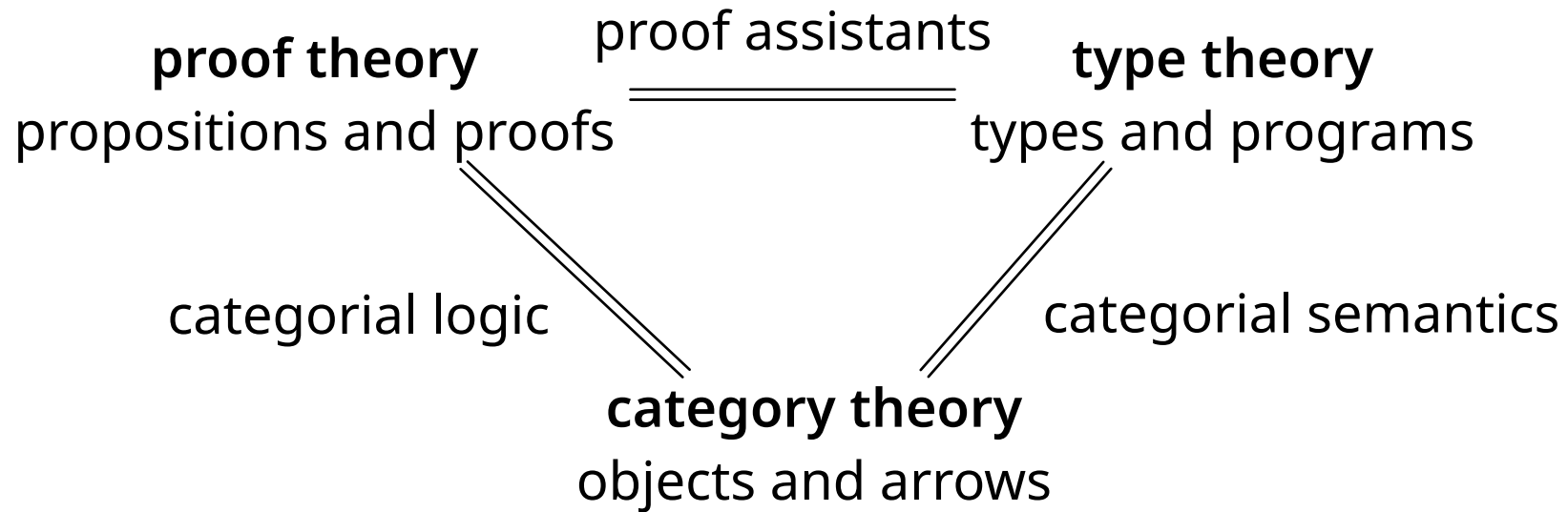
type checking does program P have type τ ?

program evaluation if $\lambda x.s : \sigma \rightarrow \tau$ and $t : \sigma$, then $(\lambda x.s) t : \tau \rightarrow_{\beta} s[t/x] : \tau$,

program synthesis does there exist a program of type τ ?

Answers to these questions depend on the type system we choose.

The Curry-Howard-Lambek correspondence



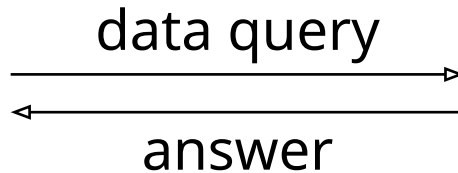
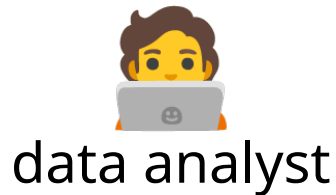
2. Introduction to Differential Privacy

Motivation, Definition and Properties

Risks and Benefits of Data Analysis

Many databases hold vast amounts of sensitive data:

- ▶ hospitals and agencies hold medical data.
- ▶ private companies hold consumer data, etc.



John Doe	59	chronic pain	...
Jane Doe	62	heart failure	...
Fred Bloggs	75	neck injury	...
...			

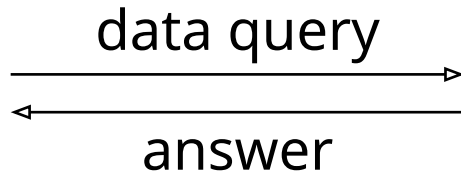
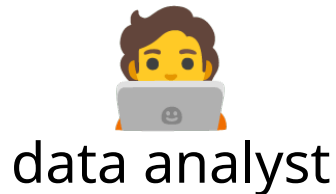
How to analyse private data without compromising privacy?

Anonymisation is not sufficient as one can leverage auxiliary information.

Risks and Benefits of Data Analysis

Many databases hold vast amounts of sensitive data:

- ▶ hospitals and agencies hold medical data.
- ▶ private companies hold consumer data, etc.



John Doe	59	chronic pain	...
Jane Doe	62	heart failure	...
Fred Bloggs	75	neck injury	...
...			

How to analyse private data without compromising privacy?

Anonymisation is not sufficient as one can leverage auxiliary information.

Definition of Differential Privacy

Intuition: queries on adjacent datasets should be indistinguishable.

Definition. (Dwork et al. (2006)) $\mathcal{M}$ preserves (ϵ, δ) -differential privacy if for all sets S and all adjacent datasets x, x' , we have:

$$\Pr[\mathcal{M}(x) \in S] \leq e^\epsilon \Pr[\mathcal{M}(x') \in S] + \delta.$$

Lemma. (ϵ, δ) -differential privacy is a compositional property.

Definition of Differential Privacy

Intuition: queries on adjacent datasets should be indistinguishable.

Definition. (Dwork et al. (2006)) $\mathcal{M}$ preserves (ϵ, δ) -differential privacy if for all sets S and all adjacent datasets x, x' , we have:

$$\Pr[\mathcal{M}(x) \in S] \leq e^\epsilon \Pr[\mathcal{M}(x') \in S] + \delta.$$

Lemma. (ϵ, δ) -differential privacy is a compositional property.

Definition of Differential Privacy

Intuition: queries on adjacent datasets should be indistinguishable.

Definition. (Dwork et al. (2006)) $\mathcal{M}$ preserves (ϵ, δ) -differential privacy if for all sets S and all adjacent datasets x, x' , we have:

$$\Pr[\mathcal{M}(x) \in S] \leq e^\epsilon \Pr[\mathcal{M}(x') \in S] + \delta.$$

Lemma. (ϵ, δ) -differential privacy is a compositional property.

Formal Methods for Differential Privacy Verification

Proving by hand that a given query preserves diff. priv. is error-prone.

deductive program verification

- ▶ probabilistic Hoare logic,
- ▶ separation logic, etc.

type systems

1. Automatically prove a simpler property.
2. Use a privacy-preserving theorem.

We are going to focus on the second approach.

Formal Methods for Differential Privacy Verification

Proving by hand that a given query preserves diff. priv. is error-prone.

deductive program verification

- ▶ probabilistic Hoare logic,
- ▶ separation logic, etc.

type systems

1. Automatically prove a simpler property.
2. Use a privacy-preserving theorem.

We are going to focus on the second approach.

Formal Methods for Differential Privacy Verification

Proving by hand that a given query preserves diff. priv. is error-prone.

deductive program verification

- ▶ probabilistic Hoare logic,
- ▶ separation logic, etc.

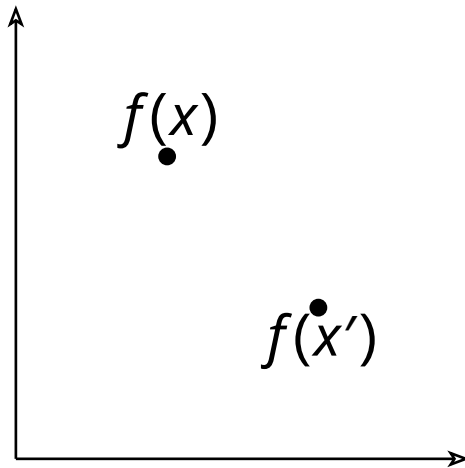
type systems

1. Automatically prove a simpler property.
2. Use a privacy-preserving theorem.

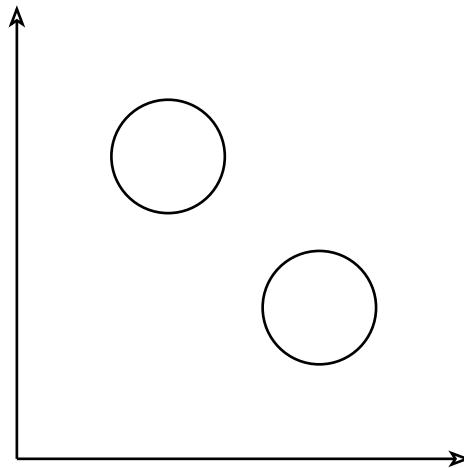
We are going to focus on **the second approach**.

Noise-Addition Mechanisms

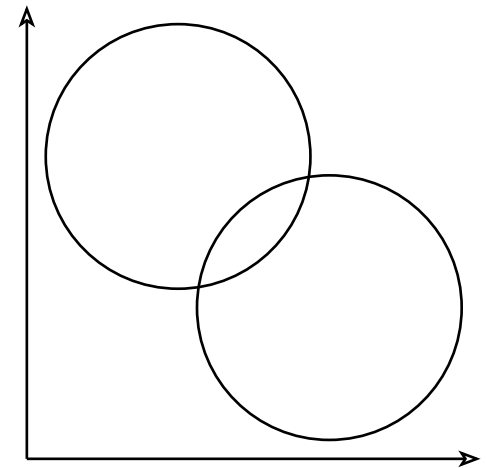
How much noise to add to preserve diff. priv.?



raw results



not enough noise



enough noise

Definition of Function Sensitivity

Definition. *The global sensitivity $\text{GS}(f)$ of a function f between metric spaces $X := (|X|, d_x)$ and $Y := (|Y|, d_y)$ is the smallest $s \in (0, +\infty]$ such that*

$$\forall x, x' \in X, d_y(f(x), f(x')) \leq s \cdot d_x(x, x')$$

A function f is

- ▶ *s-sensitive* whenever $\text{GS}(f) \leq s$, and
- ▶ *non-expansive* whenever $\text{GS}(f) \leq 1$.

Definition of Function Sensitivity

Definition. *The global sensitivity $\text{GS}(f)$ of a function f between metric spaces $X := (|X|, d_x)$ and $Y := (|Y|, d_y)$ is the smallest $s \in (0, +\infty]$ such that*

$$\forall x, x' \in X, d_y(f(x), f(x')) \leq s \cdot d_x(x, x')$$

A function f is

- ▶ *s-sensitive* whenever $\text{GS}(f) \leq s$, and
- ▶ *non-expansive* whenever $\text{GS}(f) \leq 1$.

Example of a Non-Expansive Function

Example. $f: (x_1, \dots, x_n) \mapsto \sum_{i=1}^n x_i$ is non-expansive from $(\mathbb{R}^n, d_1)$ to $(\mathbb{R}, d_1)$.

Proof. For all tuples x and x' of size n , we have

$$|f(x) - f(x')| = \left| \sum_{i=1}^n x_i - \sum_{i=1}^n x'_i \right| = \left| \sum_{i=1}^n (x_i - x'_i) \right| \leq \sum_{i=1}^n |x_i - x'_i| = d_1(x, x')$$

■

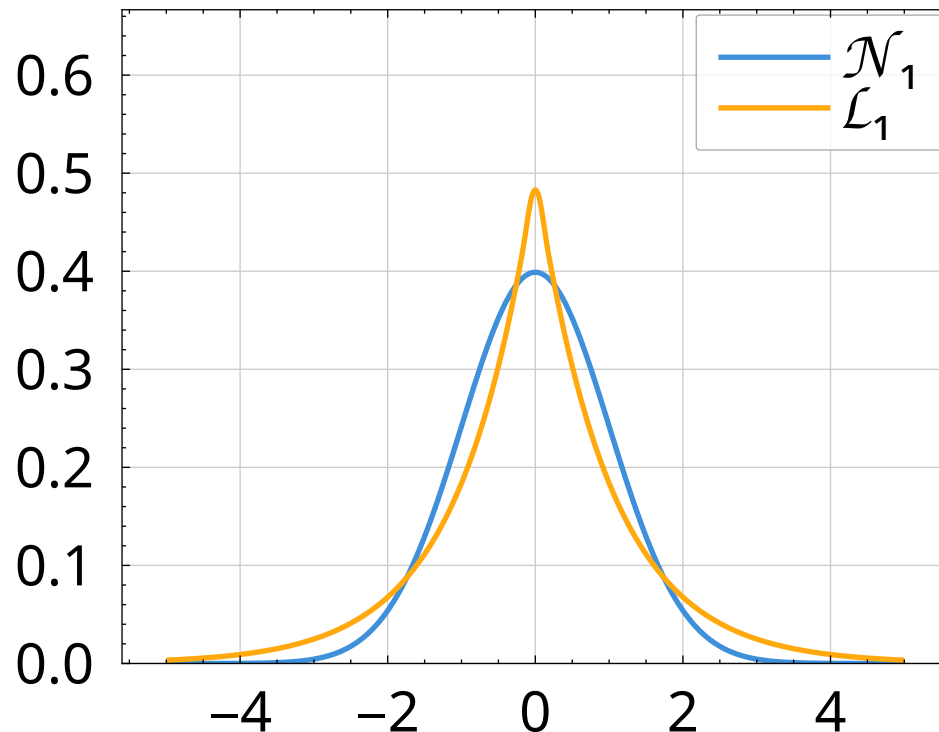
The Laplace Mechanism

Theorem. For all programs $f: \mathbb{N}^d \rightarrow (\mathbb{R}^n, d_1)$ with sensitivity s , for all $\varepsilon > 0$, and $Y_i \sim \mathcal{L}_{\text{GS}(f)/\varepsilon}$, we have that

$$x \mapsto f(x) + (Y_1, \dots, Y_n)$$

provides ε -differential privacy.

see also the Gaussian mechanism,
the exp. mechanism, etc.



Roadmap

Given a query $q: D \rightarrow X$,

1. **Formally derive a bound on the sensitivity of q .**
2. Run q on the database D to obtain a value x .
3. Add noise to x using a privacy mechanism.
4. **Compose!**

We will study and design type systems for steps 1 and 4.

3. Fuzz: Linear Types for Sensitivity Analysis

Reed and Pierce (2010)

Linear logic and Computational Resources

logical viewpoint

- ▶ intuitionistic logic without contraction or weakening
- ▶ every hypothesis must be used exactly once

computational viewpoint

- ▶ simple types without duplication or discarding
- ▶ every resource must be used exactly once

Girard (1987) introduced new logical connectives:

- ▶ a linear arrow ($\multimap$),
- ▶ two conjunctions ($\otimes$) 'tensor' and ($\&$) 'with', and
- ▶ a disjunction ($\oplus$) 'plus'.

Linear logic and Computational Resources

logical viewpoint

- ▶ intuitionistic logic without contraction or weakening
- ▶ every hypothesis must be used exactly once

computational viewpoint

- ▶ simple types without duplication or discarding
- ▶ every resource must be used exactly once

Girard (1987) introduced new logical connectives:

- ▶ a linear arrow ($\multimap$),
- ▶ two conjunctions ($\otimes$) 'tensor' and ($\&$) 'with', and
- ▶ a disjunction ($\oplus$) 'plus'.

Terms and Types

We consider a simply-typed linear λ -calculus.

$$\begin{aligned} a, b, \dots, e ::= & x \mid c \mid () \mid \langle e, e \rangle \mid (e, e) \mid \mathbf{let} (x, y) = e \mathbf{in} e \mid \mathbf{proj}_i e \\ & \mid \lambda x. e \mid e e \mid \mathbf{inj}_i e \mid \mathbf{case} e \mathbf{of} x \rightarrow e \mathbf{or} x \rightarrow e \\ & \mid !e \mid \mathbf{let}_! x = e \mathbf{in} e \\ \sigma, \tau, \dots ::= & \mathbf{Unit} \mid \sigma \multimap \tau \mid \tau_1 \otimes \tau_2 \mid \tau_1 \& \tau_2 \mid \tau_1 \oplus \tau_2 \mid !_s \tau \mid \dots \end{aligned}$$

(In practice, we also include recursive types.)

Typing Judgements

Typing contexts are sets of triples: $\Gamma ::= \emptyset \mid \Gamma, x :_s \tau$.

Example. $x :_1 \text{Nat}, y :_2 \text{Nat} \vdash (x + y, y) : \text{Nat} \otimes \text{Nat}$ will mean that the term is 1-sensitive in x and 2-sensitive in y .

Contexts can be added and scaled pointwise:

$$(\Gamma, x :_s \tau) + (\Delta, x :_r \tau) := (\Gamma + \Delta), x :_{r+s} \tau$$

$$s \cdot (\Gamma, x :_r \tau) := s \cdot \Gamma, x :_{rs} \tau$$

Typing Judgements

Typing contexts are sets of triples: $\Gamma ::= \emptyset \mid \Gamma, x :_s \tau$.

Example. $x :_1 \text{Nat}, y :_2 \text{Nat} \vdash (x + y, y) : \text{Nat} \otimes \text{Nat}$ will mean that the term is 1-sensitive in x and 2-sensitive in y .

Contexts can be added and scaled pointwise:

$$(\Gamma, x :_s \tau) + (\Delta, x :_r \tau) := (\Gamma + \Delta), x :_{r+s} \tau$$

$$s \cdot (\Gamma, x :_r \tau) := s \cdot \Gamma, x :_{rs} \tau$$

Typing Judgements

Typing contexts are sets of triples: $\Gamma ::= \emptyset \mid \Gamma, x :_s \tau$.

Example. $x :_1 \text{Nat}, y :_2 \text{Nat} \vdash (x + y, y) : \text{Nat} \otimes \text{Nat}$ will mean that the term is 1-sensitive in x and 2-sensitive in y .

Contexts can be added and scaled pointwise:

$$(\Gamma, x :_s \tau) + (\Delta, x :_r \tau) := (\Gamma + \Delta), x :_{r+s} \tau$$

$$s \cdot (\Gamma, x :_r \tau) := s \cdot \Gamma, x :_{rs} \tau$$

Typing Rules (subset)

Logical rules

$$\begin{array}{c} \frac{s \geq 1}{\Gamma, x :_s \tau \vdash x : \tau} \quad \frac{\Delta \vdash e_1 : \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Delta + \Gamma \vdash (e_1, e_2) : \tau_1 \otimes \tau_2} \otimes I \\[2ex] \frac{\Gamma, x :_s \sigma \vdash e : \tau}{\Gamma \vdash \lambda x. e : !_s \sigma \multimap \tau} \multimap I \quad \frac{\Delta \vdash f : !_s \sigma \multimap \tau \quad \Gamma \vdash e : \sigma}{\Delta + s\Gamma \vdash f e : \tau} \multimap E \end{array}$$

Primitives

$$\begin{array}{c} \frac{\Gamma_1 \vdash n_1 : \text{Int} \quad \Gamma_2 \vdash n_2 : \text{Int}}{\Gamma_1 + \Gamma_2 \vdash n_1 + n_2 : \text{Int}} (+) \quad \frac{\Gamma_1 \vdash n_1 : \text{Int} \quad \Gamma_2 \vdash n_2 : \text{Int}}{\infty \cdot (\Gamma_1 + \Gamma_2) \vdash n_1 \times n_2 : \text{Int}} (\times) \end{array}$$

Typing Rules (subset)

Logical rules

$$\begin{array}{c} \frac{s \geq 1}{\Gamma, x :_s \tau \vdash x : \tau} \quad \frac{\Delta \vdash e_1 : \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Delta + \Gamma \vdash (e_1, e_2) : \tau_1 \otimes \tau_2} \otimes I \\[10pt] \frac{\Gamma, x :_s \sigma \vdash e : \tau}{\Gamma \vdash \lambda x. e : !_s \sigma \multimap \tau} \multimap I \quad \frac{\Delta \vdash f : !_s \sigma \multimap \tau \quad \Gamma \vdash e : \sigma}{\Delta + s\Gamma \vdash f e : \tau} \multimap E \end{array}$$

Primitives

$$\frac{\Gamma_1 \vdash n_1 : \text{Int} \quad \Gamma_2 \vdash n_2 : \text{Int}}{\Gamma_1 + \Gamma_2 \vdash n_1 + n_2 : \text{Int}} (+) \quad \frac{\Gamma_1 \vdash n_1 : \text{Int} \quad \Gamma_2 \vdash n_2 : \text{Int}}{\infty \cdot (\Gamma_1 + \Gamma_2) \vdash n_1 \times n_2 : \text{Int}} (\times)$$

Soundness of Fuzz

Theorem. (soundness) If $\vdash f : !_s \sigma \multimap \tau$, then $\text{GS}(f) \leq s$.

Example. $\vdash \text{fold } 0 \ (+) : \text{List}\langle \text{Real} \rangle \multimap \text{Real}$, so *sum* is 1-sensitive.

One can prove this theorem using

- ▶ an operational semantics via logical relations¹
- ▶ a denotational semantics²
 - ▶ in the category **Met** of metric spaces for the finite fragment
 - ▶ in the category **MetCPO**_⊥ of metric cpos for the whole calculus

¹Reed and Pierce (2010)

²Azevedo de Amorim et al. (2017)

Soundness of Fuzz

Theorem. (soundness) If $\vdash f : !_s \sigma \multimap \tau$, then $\text{GS}(f) \leq s$.

Example. $\vdash \text{fold } 0 \ (+) : \text{List}\langle \text{Real} \rangle \multimap \text{Real}$, so *sum* is 1-sensitive.

One can prove this theorem using

- ▶ an operational semantics via logical relations¹
- ▶ a denotational semantics²
 - ▶ in the category **Met** of metric spaces for the finite fragment
 - ▶ in the category **MetCPO**_⊥ of metric cpos for the whole calculus

¹Reed and Pierce (2010)

²Azevedo de Amorim et al. (2017)

Soundness of Fuzz

Theorem. (soundness) If $\vdash f : !_s \sigma \multimap \tau$, then $\text{GS}(f) \leq s$.

Example. $\vdash \text{fold } 0 \ (+) : \text{List} \langle \text{Real} \rangle \multimap \text{Real}$, so *sum* is 1-sensitive.

One can prove this theorem using

- ▶ an operational semantics via logical relations¹
- ▶ a denotational semantics²
 - ▶ in the category **Met** of metric spaces for the finite fragment
 - ▶ in the category **MetCPO**_⊥ of metric cpos for the whole calculus

¹Reed and Pierce (2010)

²Azevedo de Amorim et al. (2017)

Key Takeaways

- ▶ Fuzz is a linear type system for sensitivity analysis.
 - ▶ **Recall.** Sensitivity is a proxy for differential privacy.
- ▶ Fuzz enjoys the expected metatheoretical properties.
 - ▶ weakening*, substitution, subject reduction, etc.
- ▶ Fuzz has a denotational semantics in **Met**.
- ▶ Type checking can be performed efficiently³.

³D'Antoni et al. (2013)

Key Takeaways

- ▶ Fuzz is a linear type system for sensitivity analysis.
 - ▶ **Recall.** Sensitivity is a proxy for differential privacy.
- ▶ Fuzz enjoys the expected metatheoretical properties.
 - ▶ weakening*, substitution, subject reduction, etc.
- ▶ Fuzz has a denotational semantics in **Met**.
- ▶ Type checking can be performed efficiently³.

³D'Antoni et al. (2013)

Key Takeaways

- ▶ Fuzz is a linear type system for sensitivity analysis.
 - ▶ **Recall.** Sensitivity is a proxy for differential privacy.
- ▶ Fuzz enjoys the expected metatheoretical properties.
 - ▶ weakening*, substitution, subject reduction, etc.
- ▶ Fuzz has a denotational semantics in **Met**.
- ▶ Type checking can be performed efficiently³.

³D'Antoni et al. (2013)

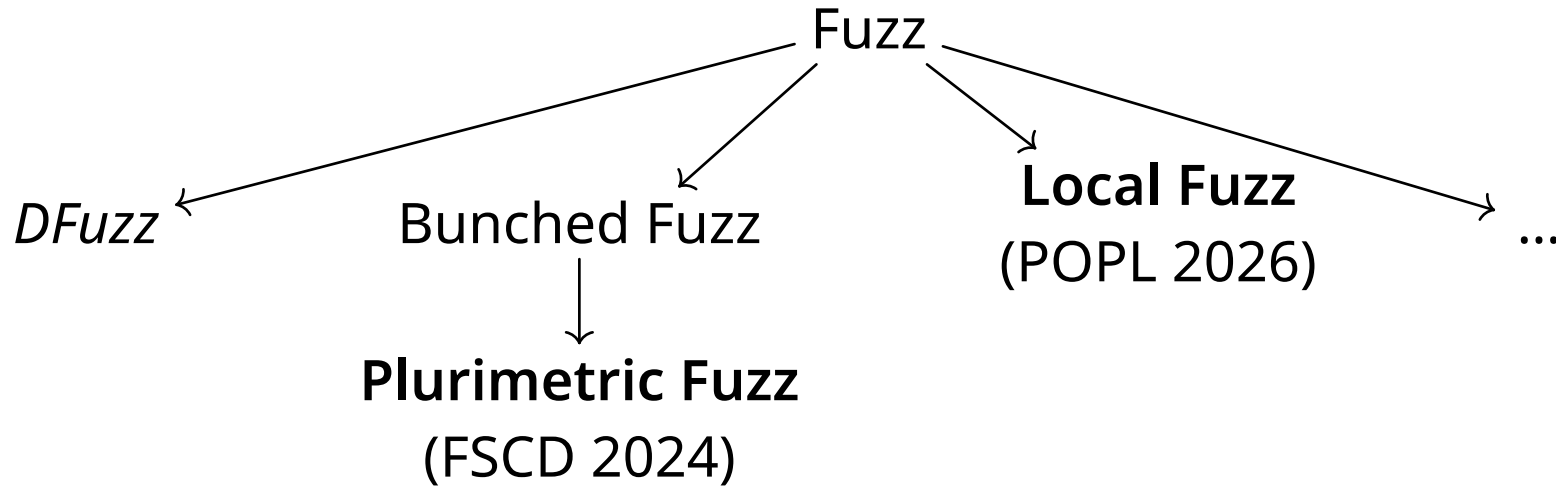
Key Takeaways

- ▶ Fuzz is a linear type system for sensitivity analysis.
 - ▶ **Recall.** Sensitivity is a proxy for differential privacy.
- ▶ Fuzz enjoys the expected metatheoretical properties.
 - ▶ weakening*, substitution, subject reduction, etc.
- ▶ Fuzz has a denotational semantics in **Met**.
- ▶ Type checking can be performed efficiently³.

³D'Antoni et al. (2013)

Contributions

Sensitivity analysis

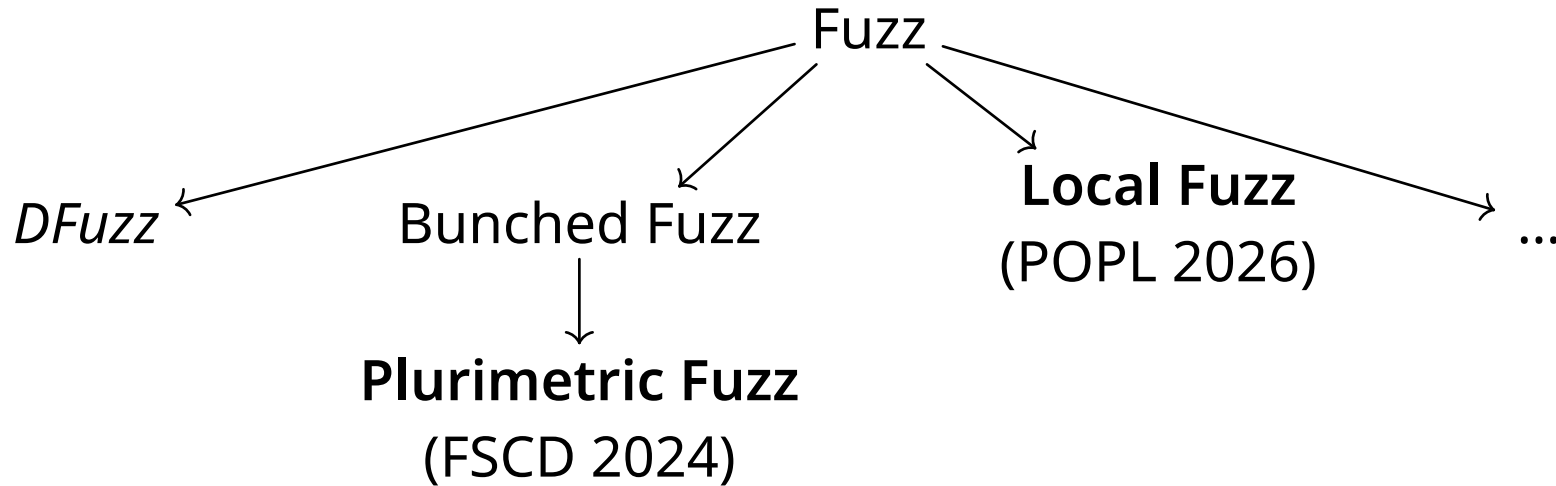


Differential privacy verification

Fuzz can be used as a sensitivity layer in a large system (CSF 2025).

Contributions

Sensitivity analysis



Differential privacy verification

Fuzz can be used as a sensitivity layer in a large system (CSF 2025).

4. Plurimetric Fuzz: Sensitivity Analysis with respect to Vector Metrics

Sannier and Baillot (2024)

Beyond the L^1 metric

Fuzz is biased towards the L^1 metric d_1 :

- ▶ $\llbracket \Gamma, \Delta \rrbracket = \llbracket \Gamma \rrbracket \times_1 \llbracket \Delta \rrbracket$, and
- ▶ $\llbracket \tau_1 \otimes \tau_2 \rrbracket = \llbracket \tau_1 \rrbracket \times_1 \llbracket \tau_2 \rrbracket$.

For all $p \geq 1$, we define the L_p metric as follows:

$$d_p((x, y), (x', y')) := \sqrt[p]{d(x, x')^p + d(y, y')^p}$$

- ▶ The Gaussian mechanism depends on L^2 -sensitivity analysis.
- ▶ Geographical data is best represented in the Euclidean plane.
- ▶ L^p -sensitivity might be less than L^1 sensitivity.

Beyond the L^1 metric

Fuzz is biased towards the L^1 metric d_1 :

- ▶ $\llbracket \Gamma, \Delta \rrbracket = \llbracket \Gamma \rrbracket \times_1 \llbracket \Delta \rrbracket$, and
- ▶ $\llbracket \tau_1 \otimes \tau_2 \rrbracket = \llbracket \tau_1 \rrbracket \times_1 \llbracket \tau_2 \rrbracket$.

For all $p \geq 1$, we define the L_p metric as follows:

$$d_p((x, y), (x', y')) := \sqrt[p]{d(x, x')^p + d(y, y')^p}$$

- ▶ The Gaussian mechanism depends on L^2 -sensitivity analysis.
- ▶ Geographical data is best represented in the Euclidean plane.
- ▶ L^p -sensitivity might be less than L^1 sensitivity.

Beyond the L^1 metric

Fuzz is biased towards the L^1 metric d_1 :

- ▶ $\llbracket \Gamma, \Delta \rrbracket = \llbracket \Gamma \rrbracket \times_1 \llbracket \Delta \rrbracket$, and
- ▶ $\llbracket \tau_1 \otimes \tau_2 \rrbracket = \llbracket \tau_1 \rrbracket \times_1 \llbracket \tau_2 \rrbracket$.

For all $p \geq 1$, we define the L_p metric as follows:

$$d_p((x, y), (x', y')) := \sqrt[p]{d(x, x')^p + d(y, y')^p}$$

- ▶ The Gaussian mechanism depends on L^2 -sensitivity analysis.
- ▶ Geographical data is best represented in the Euclidean plane.
- ▶ L^p -sensitivity might be less than L^1 sensitivity.

Terms, Types and Judgements

We consider:

- ▶ the same terms as Fuzz, and
- ▶ the same types as Bunched Fuzz⁴, including $(\otimes_p)_{p \geq 1}$ and $(\multimap_p)_{p \geq 1}$.

Example. $(\text{Real} \otimes_2 \text{Real}) \multimap_2 (\text{Real} \otimes_2 \text{Real})$ is the type of distance-preserving maps in the Euclidean plane $\mathbb{R}^2$.

Every judgement is annotated with a parameter $p \geq 1$: $(p) \Gamma \vdash e : \tau$.

⁴wunder et al. (2023)

Terms, Types and Judgements

We consider:

- ▶ the same terms as Fuzz, and
- ▶ the same types as Bunched Fuzz⁴, including $(\otimes_p)_{p \geq 1}$ and $(\multimap_p)_{p \geq 1}$.

Example. $(\text{Real} \otimes_2 \text{Real}) \multimap_2 (\text{Real} \otimes_2 \text{Real})$ is the type of distance-preserving maps in the Euclidean plane $\mathbb{R}^2$.

Every judgement is annotated with a parameter $p \geq 1$: $(p) \Gamma \vdash e : \tau$.

⁴wunder et al. (2023)

Typing Rules (subset)

$$\frac{(p) \Gamma \vdash e_1 : \tau_1 \quad (p) \Delta \vdash e_2 : \tau_2}{(p) \text{Contr}_p(\Delta; \Gamma) \vdash (e_1, e_2) : \tau_1 \otimes_p \tau_2} \otimes I$$

$$\frac{(p) \Gamma \vdash e : \tau \quad \Gamma \leq \Delta \quad p \geq q}{(q) \Delta \vdash e : \tau} \geq W \quad \frac{(p) \Gamma \vdash e : \tau \quad \Gamma \leq \Delta \quad p \leq q}{(q) c(p, q) \cdot \Delta \vdash e : \tau} \leq W$$

where $\text{Contr}_p(x :_s \tau; x :_r \tau) := x :_{\sqrt[p]{s^p + r^p}} \tau$.

We have the expected metatheoretical properties:
substitution, subject reduction, a form of subtyping, etc.

Typing Rules (subset)

$$\frac{(p) \Gamma \vdash e_1 : \tau_1 \quad (p) \Delta \vdash e_2 : \tau_2}{(p) \text{Contr}_p(\Delta; \Gamma) \vdash (e_1, e_2) : \tau_1 \otimes_p \tau_2} \otimes I$$

$$\frac{(p) \Gamma \vdash e : \tau \quad \Gamma \leq \Delta \quad p \geq q}{(q) \Delta \vdash e : \tau} \geq W \quad \frac{(p) \Gamma \vdash e : \tau \quad \Gamma \leq \Delta \quad p \leq q}{(q) c(p, q) \cdot \Delta \vdash e : \tau} \leq W$$

where $\text{Contr}_p(x :_s \tau; x :_r \tau) := x :_{\sqrt[p]{s^p + r^p}} \tau$.

We have the expected metatheoretical properties:
substitution, subject reduction, a form of subtyping, etc.

Soundness of Plurimetric Fuzz

Theorem. (soundness) If $(p) \vdash f : !_s \sigma \multimap_p \tau_1 \otimes_p \dots \otimes_p \tau_n$, then $\text{GS}_p(f) \leq s$.

Example. *translate* and *rotateLeft* are provably non-expansive in the Euclidean plane $\llbracket \text{Real} \otimes_2 \text{Real} \rrbracket := (\mathbb{R}^2, d_2)$.

We prove soundness by extending Fuzz's denotational semantics:

- ▶ $\llbracket \tau_1 \otimes_p \tau_2 \rrbracket := \llbracket \tau_1 \rrbracket \times_p \llbracket \tau_2 \rrbracket$, and
- ▶ $\llbracket (p) x :_s \tau, y :_r \sigma \rrbracket := \llbracket \tau \rrbracket \times_p \llbracket \sigma \rrbracket$.
- ▶ **Met** and **MetCPO**_⊥ are stable by $\otimes_p$ and $\multimap_p$ for all p .

Soundness of Plurimetric Fuzz

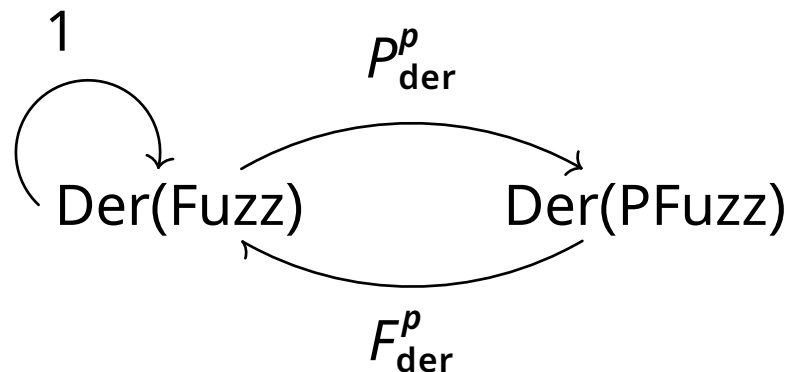
Theorem. (soundness) If $(p) \vdash f : !_s \sigma \multimap_p \tau_1 \otimes_p \dots \otimes_p \tau_n$, then $\text{GS}_p(f) \leq s$.

Example. *translate* and *rotateLeft* are provably non-expansive in the Euclidean plane $\llbracket \text{Real} \otimes_2 \text{Real} \rrbracket := (\mathbb{R}^2, d_2)$.

We prove soundness by extending Fuzz's denotational semantics:

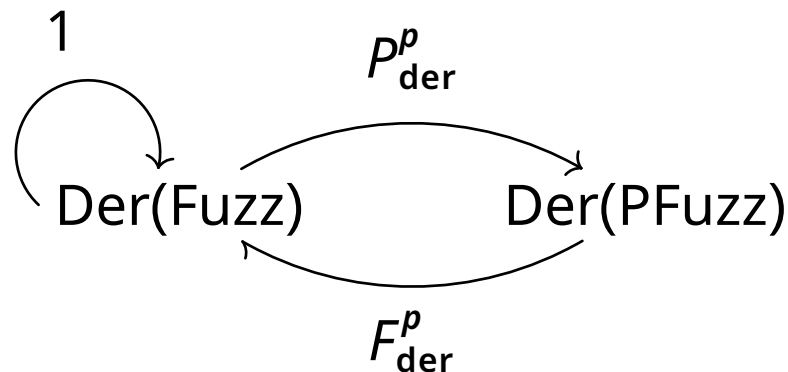
- ▶ $\llbracket \tau_1 \otimes_p \tau_2 \rrbracket := \llbracket \tau_1 \rrbracket \times_p \llbracket \tau_2 \rrbracket$, and
- ▶ $\llbracket (p) x :_s \tau, y :_r \sigma \rrbracket := \llbracket \tau \rrbracket \times_p \llbracket \sigma \rrbracket$.
- ▶ **Met** and **MetCPO**_⊥ are stable by $\otimes_p$ and $\multimap_p$ for all p .

Plurimetric Fuzz $\leftrightarrow$ Fuzz



Theorem. For all p , the p -image of a (minimal) judgement $\Gamma \vdash e : \tau$ in Fuzz is a (minimal) judgement $(p) P_{\text{ctx}}^p(\Gamma) \vdash e : P_{\text{type}}^p(\tau)$ in Plurimetric Fuzz.

Plurimetric Fuzz $\leftrightarrow$ Fuzz



Theorem. For all p , the p -image of a (minimal) judgement $\Gamma \vdash e : \tau$ in Fuzz is a (minimal) judgement $(p) P_{\text{ctx}}^p(\Gamma) \vdash e : P_{\text{type}}^p(\tau)$ in Plurimetric Fuzz.

5. Local Fuzz: Dependent Coeffects for Local Sensitivity Analysis

Sannier and Baillot (2026)

Beyond Global Sensitivity

- ▶ Global sensitivity is *often* infinite.
 - ▶ e.g., $x \mapsto x^2$, $l \mapsto \text{mean}(l)$, etc.
- ▶ In these cases, we cannot apply the Laplace mechanism.
- ▶ Intuitively, asking for a uniform bound is too strong.
 - ▶ We are not interested in *any* two inputs x and x' .

Definition. The local sensitivity of a function $f: X \rightarrow Y$ at point $x \in X$ is

$$\text{LS}(f, x) := \max \{ d_Y(f(x), f(x')) \mid x' \in X, d_X(x, x') = 1 \}.$$

Beyond Global Sensitivity

- ▶ Global sensitivity is *often* infinite.
 - ▶ e.g., $x \mapsto x^2$, $l \mapsto \text{mean}(l)$, etc.
- ▶ In these cases, we cannot apply the Laplace mechanism.
- ▶ Intuitively, asking for a uniform bound is too strong.
 - ▶ We are not interested in *any* two inputs x and x' .

Definition. The local sensitivity of a function $f: X \rightarrow Y$ at point $x \in X$ is

$$\text{LS}(f, x) := \max \{ d_Y(f(x), f(x')) \mid x' \in X, d_X(x, x') = 1 \}.$$

Local Sensitivity and Differential Privacy

We can leverage an upper bound on local sensitivity to obtain diff. priv.:

- ▶ smooth sensitivity⁵, and
- ▶ the Propose-Test-Release (PTR) framework⁶.

Can we design a type system to analyse local sensitivity?

Fuzz : global sensitivity :: ? : local sensitivity

Sensitivity annotations need to be functions.

⁵Nissim et al. (2007)

⁶Dwork and Lei (2009)

Local Sensitivity and Differential Privacy

We can leverage an upper bound on local sensitivity to obtain diff. priv.:

- ▶ smooth sensitivity⁵, and
- ▶ the Propose-Test-Release (PTR) framework⁶.

Can we design a type system to analyse local sensitivity?

Fuzz : global sensitivity :: ? : local sensitivity

Sensitivity annotations need to be functions.

⁵Nissim et al. (2007)

⁶Dwork and Lei (2009)

Design Choices

The naive extension of Fuzz is not expressive enough.

$x :_s \sigma \vdash e : \tau$ means $\forall x, x' \in \llbracket \tau \rrbracket, d_\tau(e(x), e(x')) \leq s(x) \cdot d(x, x')$.

Instead, we make the following choices:

- ▶ we define a composable version of local sensitivity

$$LS_k(f, x) := \max \{ d_Y(f(x), f(x')) \mid x' \in X, d_X(x, x') \leq k \}$$

- ▶ we use flat coeffects: annotations can depend on all variables.

Design Choices

The naive extension of Fuzz is not expressive enough.

$x :_s \sigma \vdash e : \tau$ means $\forall x, x' \in \llbracket \tau \rrbracket, d_\tau(e(x), e(x')) \leq s(x) \cdot d(x, x')$.

Instead, we make the following choices:

- ▶ we define a composable version of local sensitivity

$$LS_k(f, x) := \max \{ d_Y(f(x), f(x')) \mid x' \in X, d_X(x, x') \leq k \}$$

- ▶ we use flat coeffacts: annotations can depend on all variables.

Sensitivity Moduli

We write $\boxed{\Gamma @ \mathbf{s} \vdash e : \tau}$, where $\mathbf{s}$ is a *sensitivity modulus*.

$$C(\Gamma) := \left\{ \mathbf{s} : \Gamma \rightarrow (\overline{\text{Nat}}^{|\Gamma|} \rightarrow \overline{\text{Nat}}) \mid \left\{ \begin{array}{l} \forall \mathbf{x} : \Gamma, \mathbf{s}(\mathbf{x}, \mathbf{0}) = 0 \\ \mathbf{s}(\mathbf{x}, -) \text{ is non-decreasing} \end{array} \right\} \right\}$$

where $\overline{\text{Nat}} := \mathbb{N} \cup \{+\infty\}$.

Example. $x : \text{Int} @ \lambda x. \lambda k. 2|x|k + k^2 \vdash x^2 : \text{Int}$

Typing Rules (subset)

$$\frac{}{x : \tau @ \lambda x. \lambda k. k \vdash x : \tau} \text{var}$$

$$\frac{\Gamma, x : \sigma @ \mathbf{s} + \mathbf{r} \vdash e : \tau}{\Gamma @ \mathbf{s} \vdash \lambda x. e : \sigma \xrightarrow{\mathbf{r}} \tau} \rightarrow I$$

$$\frac{\Gamma @ \mathbf{s} \vdash e : \tau \quad \Gamma \subseteq \Gamma' \quad \mathbf{s} \leq \mathbf{s}' \quad \tau <: \tau'}{\Gamma' @ \mathbf{s}' \vdash e : \tau'} W$$

$$\frac{\Delta @ \mathbf{r} \vdash f : \sigma \xrightarrow{t} \tau \quad \Gamma @ \mathbf{s} \vdash e : \sigma}{\Delta \cup \Gamma @ \mathbf{r} + t \times_e \mathbf{s} \vdash f e : \tau} \rightarrow E$$

Typing Rules (continued)

We can write tight typing rules for many primitives.

Example. The following rule lets us derive the exact local sensitivity of $\lambda x. x^2$:

$$\frac{\Gamma @ \mathbf{s} \vdash a : \text{Int} \quad \Delta @ \mathbf{r} \vdash b : \text{Int}}{\Gamma @ \lambda \mathbf{x}. \lambda \mathbf{k}. \mathbf{r}(\mathbf{x}_\Delta, \mathbf{k}_\Delta) \cdot |a(\mathbf{x}_\Gamma)| + \mathbf{s}(\mathbf{x}_\Gamma, \mathbf{k}_\Gamma) \cdot |b(\mathbf{x}_\Delta)| + \mathbf{s}(\mathbf{x}_\Gamma, \mathbf{k}_\Gamma) \cdot \mathbf{r}(\mathbf{x}_\Delta, \mathbf{k}_\Delta) \vdash a \times b : \text{Int}} \times$$

Two Classes of Use Cases

Global sensitivity

If $\mathbf{s}$ is constant in its first coordinate, then $\Gamma @ \mathbf{s} \vdash e : \tau$ is a bound on the *global sensitivity* of e . **Local Fuzz is strictly more expressive than Fuzz.**

→ Laplace mechanism, Gaussian mechanism, etc.

Local sensitivity

We can analyse the *local sensitivity* of queries using $(\times)$, *median*, etc.

→ PTR framework, smooth sensitivity, etc.

Two Classes of Use Cases

Global sensitivity

If $\mathbf{s}$ is constant in its first coordinate, then $\Gamma @ \mathbf{s} \vdash e : \tau$ is a bound on the *global sensitivity* of e . **Local Fuzz is strictly more expressive than Fuzz.**

→ Laplace mechanism, Gaussian mechanism, etc.

Local sensitivity

We can analyse the *local sensitivity* of queries using $(\times)$, *median*, etc.

→ PTR framework, smooth sensitivity, etc.

Soundness of Local Fuzz

Theorem. (soundness) If $\vdash f : \sigma \xrightarrow{s} \tau$, then $LS(f, x) \leq s(1, x)$.

In categorical terms, to interpret Local Fuzz we need:

- ▶ a semicartesian closed category with finite products and coproducts,
- ▶ a dependently graded comonad⁷, rather than a graded comonad.

The category **Met** of metric spaces provides a model of Local Fuzz.

Open problem: Can this semantics be extended to recursive types?

⁷Capucci and Myers (2024)

Soundness of Local Fuzz

Theorem. (soundness) If $\vdash f : \sigma \xrightarrow{s} \tau$, then $LS(f, x) \leq s(1, x)$.

In categorical terms, to interpret Local Fuzz we need:

- ▶ a semicartesian closed category with finite products and coproducts,
- ▶ a dependently graded comonad⁷, rather than a graded comonad.

The category **Met** of metric spaces provides a model of Local Fuzz.

Open problem: **Can this semantics be extended to recursive types?**

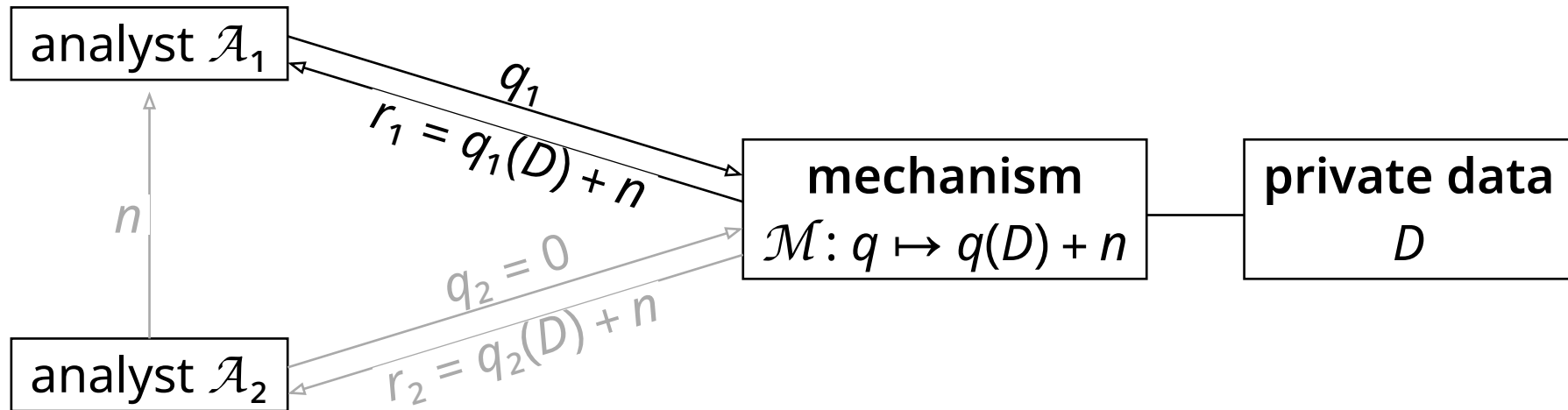
⁷Capucci and Myers (2024)

6. Session Types for the Composition of Interactive Differential Privacy

Sannier et al. (2025)

Beyond the One-Shot Setting

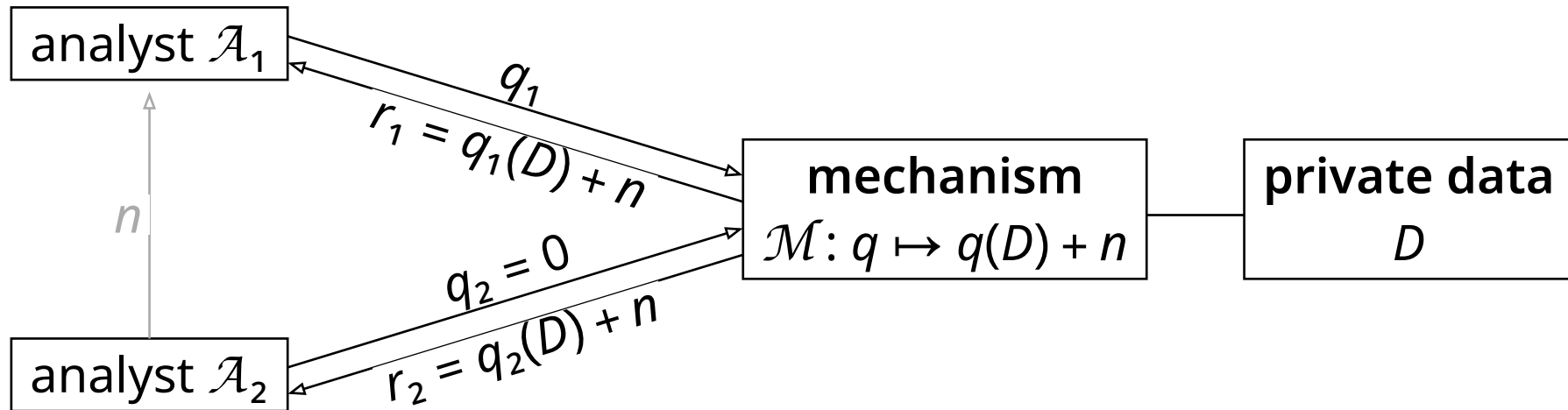
Several analysts may query the same mechanism.



The analysts can collude to compute $r_1 - r_2 = q_1(D) + n - (0 + n) = q_1(D)$.

Beyond the One-Shot Setting

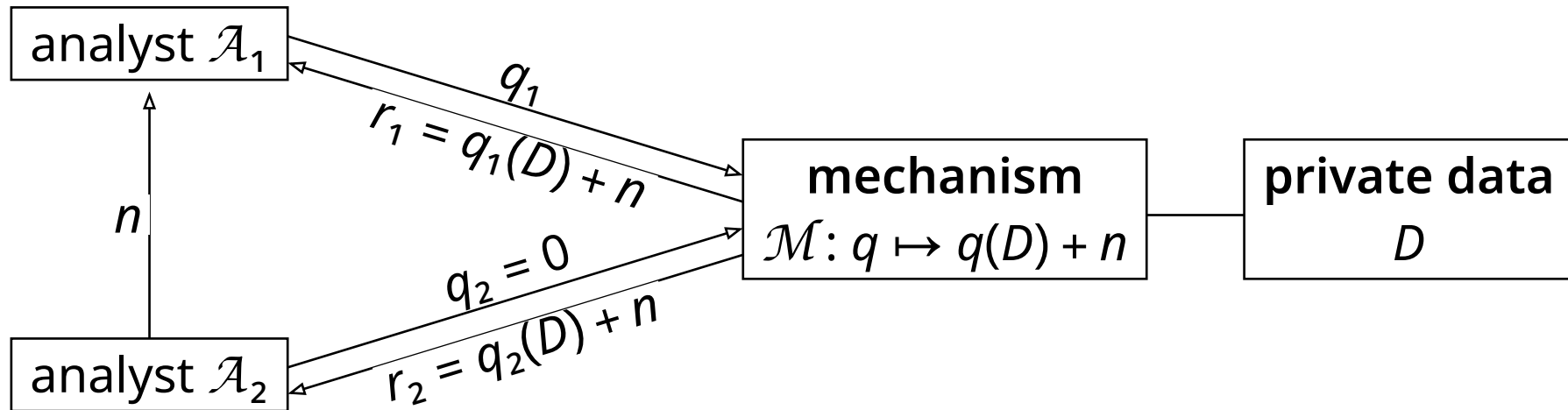
Several analysts may query the same mechanism.



The analysts can collude to compute $r_1 - r_2 = q_1(D) + n - (0 + n) = q_1(D)$.

Beyond the One-Shot Setting

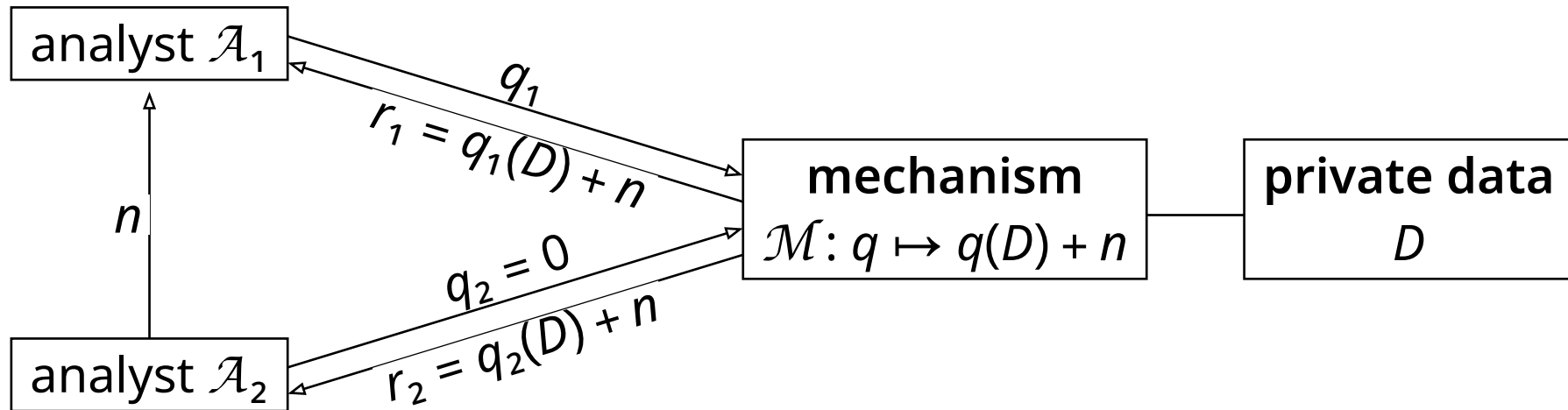
Several analysts may query the same mechanism.



The analysts can collude to compute $r_1 - r_2 = q_1(D) + n - (0 + n) = q_1(D)$.

Beyond the One-Shot Setting

Several analysts may query the same mechanism.



The analysts can collude to compute $r_1 - r_2 = q_1(D) + n - (0 + n) = q_1(D)$.

Interactive Differential Privacy

Definition. $\mathcal{M}$ preserves (ϵ, δ) -interactive diff. priv. if for all adversaries $\mathcal{A}$, all sets X and all adjacent inputs x, x' , we have

$$\Pr[\text{View}(\mathcal{A} \parallel \mathcal{M}(x)) \in X] \leq e^\epsilon \Pr[\text{View}(\mathcal{A} \parallel \mathcal{M}(x')) \in X] + \delta.$$

The *view* of a party is everything it sees during the execution.

Theorem. (Vadhan and Wang (2021))
Interactive differential privacy is composable.

Interactive Differential Privacy

Definition. $\mathcal{M}$ preserves (ϵ, δ) -interactive diff. priv. if for all adversaries $\mathcal{A}$, all sets X and all adjacent inputs x, x' , we have

$$\Pr[\text{View}(\mathcal{A} \parallel \mathcal{M}(x)) \in X] \leq e^\epsilon \Pr[\text{View}(\mathcal{A} \parallel \mathcal{M}(x')) \in X] + \delta.$$

The *view* of a party is everything it sees during the execution.

Theorem. (Vadhan and Wang (2021))
Interactive differential privacy is composable.

The π -calculus

λ -calculus : functional programs :: π -calculus : processes

Terms are generated by the following grammar:

$P, Q, \dots ::= k![e] . P$	value sending
$ k?(x) . P$	value reception
$ s \triangleright \{ l_i : P_i \mid i \in I \}$	label branching
$ s \triangleleft l . P$	label selection
$ P \parallel Q$	parallel composition
$ \dots$	

The π -calculus

λ -calculus : functional programs :: π -calculus : processes
--

Terms are generated by the following grammar:

$P, Q, \dots ::= k![e] . P$	value sending
$ k?(x) . P$	value reception
$ s \triangleright \{l_i : P_i \mid i \in I\}$	label branching
$ s \triangleleft l . P$	label selection
$ P \parallel Q$	parallel composition
$ \dots$	

Session Types

Types are generated by the following grammar:

$$a ::= ?\tau . a \mid !\tau . a \mid \&_i \{l_i : a_i\} \mid \oplus_i \{l_i : a_i\} \mid \text{end}$$

and judgements are written $\Gamma \vdash P \triangleright \Delta$.

Example. A doubling server and its client can be typed as follows:

$$\begin{aligned} n : \text{Int} &\vdash k![n] . k?(x) . 0 \triangleright k : !\text{Int} . ?\text{Int} . \text{end} \\ &\vdash k?(x) . k![x + x] . 0 \triangleright k : ?\text{Int} . !\text{Int} . \text{end} \end{aligned}$$

Session Types

Types are generated by the following grammar:

$$\alpha ::= ?\tau . \alpha \mid !\tau . \alpha \mid \&_i \{l_i : \alpha_i\} \mid \oplus_i \{l_i : \alpha_i\} \mid \text{end}$$

and judgements are written $\Gamma \vdash P \triangleright \Delta$.

Example. A doubling server and its client can be typed as follows:

$$\begin{aligned} n : \text{Int} &\vdash k![n] . k?(x) . 0 \triangleright k : !\text{Int} . ?\text{Int} . \text{end} \\ &\vdash k?(x) . k![x + x] . 0 \triangleright k : ?\text{Int} . !\text{Int} . \text{end} \end{aligned}$$

Additional Constructs

We extend the syntax of terms and types to account for

- ▶ random number generation, and
- ▶ finite server replication.

$$\begin{aligned} P, Q, \dots ::= & \dots \mid *_n k?(x) . P \\ & \mid \mathbf{Lap}_b?(x) . P \mid \mathbf{Gauss}_\sigma?(x) . P \\ \alpha ::= & \dots \mid *_n ?\tau . \alpha \end{aligned}$$

We write typing judgements of the form $\Gamma \vdash M \triangleright \Delta; (\varepsilon, \delta)$.

Additional Constructs

We extend the syntax of terms and types to account for

- ▶ random number generation, and
- ▶ finite server replication.

$$\begin{aligned} P, Q, \dots ::= & \dots \mid *_n k?(x) . P \\ & \mid \mathbf{Lap}_b?(x) . P \mid \mathbf{Gauss}_\sigma?(x) . P \\ \alpha ::= & \dots \mid *_n ?\tau . \alpha \end{aligned}$$

We write typing judgements of the form $\Gamma \vdash M \triangleright \Delta; (\varepsilon, \delta)$.

Typing Rules (subset)

Compare the usual rule in process calculi with session types

$$\frac{\Gamma \vdash e : \tau \quad \Gamma \vdash P \triangleright \Delta, k : \alpha}{\Gamma \vdash k![e] . P \triangleright \Delta, k : !\tau . \alpha}$$

with the non privacy-preserving rule

$$\frac{[\Gamma]_s \vdash_{\text{FUZZ}} e : \tau \quad \Gamma \vdash P \triangleright \Delta, k : \alpha; (\varepsilon, \delta)}{\Gamma \vdash k![e] . P \triangleright \Delta, k : !\tau . \alpha; (0, 1)} \text{ [T-Send]}$$

and, with the one implementing the Laplace mechanism

$$\frac{[\Gamma]_s \vdash_{\text{FUZZ}} e : \text{Int} \quad \Gamma \vdash P \triangleright \Delta, k : \alpha; (\varepsilon, \delta)}{\Gamma \vdash \mathbf{Lap}_{s/\varepsilon'}?(x) . k![e + x] . P \triangleright \Delta, k : !\text{Int} . \alpha; (\varepsilon + \varepsilon', \delta)} \text{ [T-NSend-Lap]}$$

Typing Rules (subset)

Compare the usual rule in process calculi with session types

$$\frac{\Gamma \vdash e : \tau \quad \Gamma \vdash P \triangleright \Delta, k : \alpha}{\Gamma \vdash k![e] . P \triangleright \Delta, k : !\tau . \alpha}$$

with the non privacy-preserving rule

$$\frac{[\Gamma]_s \vdash_{\text{FUZZ}} e : \tau \quad \Gamma \vdash P \triangleright \Delta, k : \alpha; (\varepsilon, \delta)}{\Gamma \vdash k![e] . P \triangleright \Delta, k : !\tau . \alpha; (0, 1)} \text{ [T-Send]}$$

and, with the one implementing the Laplace mechanism

$$\frac{[\Gamma]_s \vdash_{\text{FUZZ}} e : \text{Int} \quad \Gamma \vdash P \triangleright \Delta, k : \alpha; (\varepsilon, \delta)}{\Gamma \vdash \mathbf{Lap}_{s/\varepsilon'}?(x) . k![e + x] . P \triangleright \Delta, k : !\text{Int} . \alpha; (\varepsilon + \varepsilon', \delta)} \text{ [T-NSend-Lap]}$$

Typing Rules (subset)

Compare the usual rule in process calculi with session types

$$\frac{\Gamma \vdash e : \tau \quad \Gamma \vdash P \triangleright \Delta, k : \alpha}{\Gamma \vdash k![e] . P \triangleright \Delta, k : !\tau . \alpha}$$

with the non privacy-preserving rule

$$\frac{[\Gamma]_s \vdash_{\text{Fuzz}} e : \tau \quad \Gamma \vdash P \triangleright \Delta, k : \alpha; (\varepsilon, \delta)}{\Gamma \vdash k![e] . P \triangleright \Delta, k : !\tau . \alpha; (0, 1)} [\text{T-Send}]$$

and, with the one implementing the Laplace mechanism

$$\frac{[\Gamma]_s \vdash_{\text{Fuzz}} e : \text{Int} \quad \Gamma \vdash P \triangleright \Delta, k : \alpha; (\varepsilon, \delta)}{\Gamma \vdash \mathbf{Lap}_{s/\varepsilon'}?(x) . k![e + x] . P \triangleright \Delta, k : !\text{Int} . \alpha; (\varepsilon + \varepsilon', \delta)} [\text{T-NSend-Lap}]$$

Typing Rules (subset)

A mechanism can freely receive information, including private data:

$$\frac{\Gamma, x : \tau \vdash P \triangleright \Delta, k : \alpha; (\varepsilon, \delta)}{\Gamma \vdash k?(x) . P \triangleright \Delta, k : ?\tau . \alpha; (\varepsilon, \delta)} \text{ [T-Rcv]}$$

Mechanisms can be composed à la Vadhan and Wang (2021):

$$\frac{\Gamma \vdash P_1 \triangleright \Delta_1; (\varepsilon_1, \delta_1) \quad \Gamma \vdash P_2 \triangleright \Delta_2; (\varepsilon_2, \delta_2) \quad \Delta_1 \text{ and } \Delta_2 \text{ compatible}}{\Gamma \vdash P_1 \parallel P_2 \triangleright \Delta_1 \circ \Delta_2; \underbrace{(\varepsilon_1, \delta_1) \star (\varepsilon_2, \delta_2)}_{:= (\varepsilon_1 + \varepsilon_2, \delta_1 + e^{\varepsilon_1} \delta_2)}} \text{ [T-Conc]}$$

Typing Rules (subset)

A mechanism can freely receive information, including private data:

$$\frac{\Gamma, x : \tau \vdash P \triangleright \Delta, k : \alpha; (\varepsilon, \delta)}{\Gamma \vdash k?(x) . P \triangleright \Delta, k : ?\tau . \alpha; (\varepsilon, \delta)} \text{ [T-Rcv]}$$

Mechanisms can be composed à la Vadhan and Wang (2021):

$$\frac{\Gamma \vdash P_1 \triangleright \Delta_1; (\varepsilon_1, \delta_1) \quad \Gamma \vdash P_2 \triangleright \Delta_2; (\varepsilon_2, \delta_2) \quad \Delta_1 \text{ and } \Delta_2 \text{ compatible}}{\Gamma \vdash P_1 \parallel P_2 \triangleright \Delta_1 \circ \Delta_2; \underbrace{(\varepsilon_1, \delta_1) \star (\varepsilon_2, \delta_2)}_{:= (\varepsilon_1 + \varepsilon_2, \delta_1 + \mathbf{e}^{\varepsilon_1} \delta_2)}} \text{ [T-Conc]}$$

Operational Semantics

- ▶ A trace is a binary tree labelled at the leaves with lists of atoms.
- ▶ The view of a party is a subtree of the trace:
 $\text{View}(A \parallel M) := \text{Left}(\text{Tr}(A \parallel M)).$
- ▶ We define differential privacy as approximate view equivalence.

We obtain a purely syntactical proof of concurrent composition.

Theorem. (soundness) If $\Gamma \vdash M \triangleright \Delta; (\varepsilon, \delta)$, then M preserves interactive (ε, δ) -differential privacy.

Operational Semantics

- ▶ A trace is a binary tree labelled at the leaves with lists of atoms.
- ▶ The view of a party is a subtree of the trace:
 $\text{View}(A \parallel M) := \text{Left}(\text{Tr}(A \parallel M)).$
- ▶ We define differential privacy as approximate view equivalence.

We obtain a purely syntactical proof of concurrent composition.

Theorem. (soundness) If $\Gamma \vdash M \triangleright \Delta; (\varepsilon, \delta)$, then M preserves interactive (ε, δ) -differential privacy.

7. Conclusion

Summary

We have made three contributions to use of *formal methods* for sensitivity analysis and diff. priv. verification.

Specifically, we have

- ▶ extended Fuzz to account for vector metrics (Plurimetric Fuzz);
- ▶ extended Fuzz to account for local sensitivity (Local Fuzz); and
- ▶ shown how to use a process calculus to account for interactive diff. priv.

Summary

We have made three contributions to use of *formal methods* for sensitivity analysis and diff. priv. verification.

Specifically, we have

- ▶ extended Fuzz to account for vector metrics (Plurimetric Fuzz);
- ▶ extended Fuzz to account for local sensitivity (Local Fuzz); and
- ▶ shown how to use a process calculus to account for interactive diff. priv.

Open questions

Type Systems for Sensitivity Analysis

- ▶ Are type-checking and type-inference for Plurimetric Fuzz and Local Fuzz decidable? (conjecture: yes, modulo theory)
- ▶ Can we give a denotational semantics of recursive types for Local Fuzz?
- ▶ Can dependent coeffects be used beyond local sensitivity analysis?

Session Types for Interactive Differential Privacy

- ▶ Can interactive diff. priv. be restated as a form of bisimulation?
- ▶ How much probabilistic reasoning can be conducted within a typed process calculus?

Open questions

Type Systems for Sensitivity Analysis

- ▶ Are type-checking and type-inference for Plurimetric Fuzz and Local Fuzz decidable? (conjecture: yes, modulo theory)
- ▶ Can we give a denotational semantics of recursive types for Local Fuzz?
- ▶ Can dependent coeffects be used beyond local sensitivity analysis?

Session Types for Interactive Differential Privacy

- ▶ Can interactive diff. priv. be restated as a form of bisimulation?
- ▶ How much probabilistic reasoning can be conducted within a typed process calculus?

Open questions

Type Systems for Sensitivity Analysis

- ▶ Are type-checking and type-inference for Plurimetric Fuzz and Local Fuzz decidable? (conjecture: yes, modulo theory)
- ▶ Can we give a denotational semantics of recursive types for Local Fuzz?
- ▶ Can dependent coeffects be used beyond local sensitivity analysis?

Session Types for Interactive Differential Privacy

- ▶ Can interactive diff. priv. be restated as a form of bisimulation?
- ▶ How much probabilistic reasoning can be conducted within a typed process calculus?

Open questions

Type Systems for Sensitivity Analysis

- ▶ Are type-checking and type-inference for Plurimetric Fuzz and Local Fuzz decidable? (conjecture: yes, modulo theory)
- ▶ Can we give a denotational semantics of recursive types for Local Fuzz?
- ▶ Can dependent coeffects be used beyond local sensitivity analysis?

Session Types for Interactive Differential Privacy

- ▶ Can interactive diff. priv. be restated as a form of bisimulation?
- ▶ How much probabilistic reasoning can be conducted within a typed process calculus?

Open questions

Type Systems for Sensitivity Analysis

- ▶ Are type-checking and type-inference for Plurimetric Fuzz and Local Fuzz decidable? (conjecture: yes, modulo theory)
- ▶ Can we give a denotational semantics of recursive types for Local Fuzz?
- ▶ Can dependent coeffects be used beyond local sensitivity analysis?

Session Types for Interactive Differential Privacy

- ▶ Can interactive diff. priv. be restated as a form of bisimulation?
- ▶ How much probabilistic reasoning can be conducted within a typed process calculus?

Bibliography

- Azevedo de Amorim, A., Gaboardi, M., Hsu, J., Katsumata, S.-y., Cherigui, I., 2017. A semantic account of metric preservation, in: Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages (POPL '17). pp. 545–556.. <https://doi.org/10.1145/3009837.3009890>
- Capucci, M., Myers, D.J., 2024. Contextads as Wreaths; Kleisli, Para, and Span Constructions as Wreath Products [WWW Document].. URL <https://arxiv.org/abs/2410.21889>
- D'Antoni, L., Gaboardi, M., Arias, E.J.G., Haeberlen, A., Pierce, B., 2013. Sensitivity analysis using type-based constraints, in: Proceedings of the 1st Annual Workshop on Functional Programming Concepts in Domain-Specific Language (FPCDSL '13). pp. 43–50.. <https://doi.org/10.1145/2505351.2505353>
- Dwork, C., Lei, J., 2009. Differential privacy and robust statistics, in: Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing (STOC '09). pp. 371–380.. <https://doi.org/10.1145/1536414.1536466>

- Dwork, C., McSherry, F., Nissim, K., Smith, A., 2006. Calibrating Noise to Sensitivity in Private Data Analysis, in: Theory of Cryptography (TCC '06), Lecture Notes in Computer Science (LNCS). pp. 265–284.. https://doi.org/10.1007/11681878_14
- Girard, J.-Y., 1987. Linear logic. Theoretical Computer Science 50, 1–101.. [https://doi.org/10.1016/0304-3975\(87\)90045-4](https://doi.org/10.1016/0304-3975(87)90045-4)
- Nissim, K., Raskhodnikova, S., Smith, A., 2007. Smooth sensitivity and sampling in private data analysis, in: Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing (STOC '07). pp. 75–84.. <https://doi.org/10.1145/1250790.1250803>
- Reed, J., Pierce, B.C., 2010. Distance makes the types grow stronger: A calculus for differential privacy, in: Proceedings of the 15th ACM SIGPLAN International Conference on Functional Programming (ICFP '10). pp. 157–168.. <https://doi.org/10.1145/1863543.1863568>
- Sannier, V., Baillot, P., 2026. Dependent Coeffects for Local Sensitivity Analysis. Proceedings of the ACM on Programming Languages (PACMPL) 10, 806–832.. <https://doi.org/10.1145/3776670>

- Sannier, V., Baillot, P., 2024. A Linear Type System for L^p -Metric Sensitivity Analysis, in: 9th International Conference on Formal Structures for Computation and Deduction (FSCD '24), Leibniz International Proceedings in Informatics (LIPIcs).. <https://doi.org/10.4230/LIPIcs.FSCD.2024.12>
- Sannier, V., Baillot, P., Gaboardi, M., 2025. Session Types for the Concurrent Composition of Interactive Differential Privacy, in: 2025 IEEE 38th Computer Security Foundations Symposium (CSF '25). pp. 113–126.. <https://doi.org/10.1109/CSF64896.2025.00020>
- Vadhan, S., Wang, T., 2021. Concurrent Composition of Differential Privacy, in: Theory of Cryptography (TCC '21), Lecture Notes in Computer Science (LNCS). pp. 582–604.. https://doi.org/10.1007/978-3-030-90453-1_20
- wunder, j., Azevedo de Amorim, A., Baillot, P., Gaboardi, M., 2023. Bunched Fuzz: Sensitivity for Vector Metrics, in: Programming Languages and Systems: 32nd European Symposium on Programming (ESOP '23). pp. 451–478.. https://doi.org/10.1007/978-3-031-30044-8_17