Verifying Differentially Private Bayesian Inference

Marco Gaboardi
University of Dundee

Joint work with G. Barthe, G.P. Farina, E.J. Gallego Arias, A.Gordon,…

Differentially Private vs Probabilistic Inference

Differential Privacy

Probabilistic Inference

Differentially Private vs Probabilistic Inference

Differential Privacy	Probabilistic Inference

The goal in machine learning is very often similar to the goal in private data analysis. The learner typically wishes to learn some simple rule that explains a data set. However, she wishes this rule to generalize […] Generally, this means that she wants to learn a rule that captures distributional information about the data set on hand, in a way that does not depend too specifically on any single data point.

C. Dwork & A. Roth - The algorithmic Foundations of Differential privacy

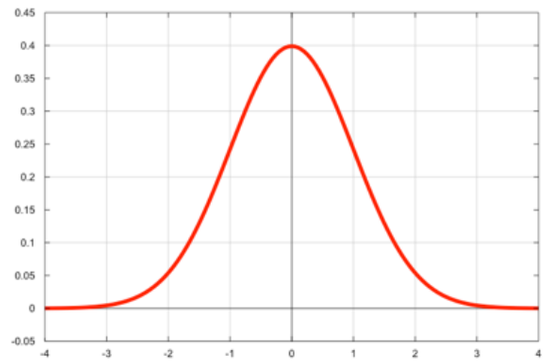
Outline

- Bayesian learning
- A language for bayesian learning
- Differential Privacy
- Combining differential privacy and bayesian learning

Bayesian Inference

Probabilistic Inference

Prior Distribution

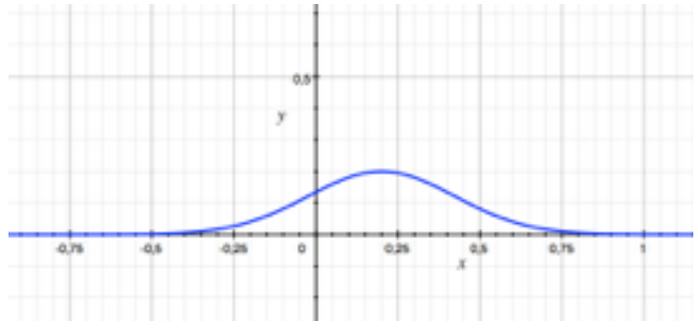


Posterior Distribution

Data

Probabilistic Program

Fun/Tabular



Bayes theorem

$$\mathrm{P}(y \mid x) = \frac{\mathrm{P}(x \mid y)\mathrm{P}(y)}{\mathrm{P}(x)}$$

Bayes theorem

Posterior

Model

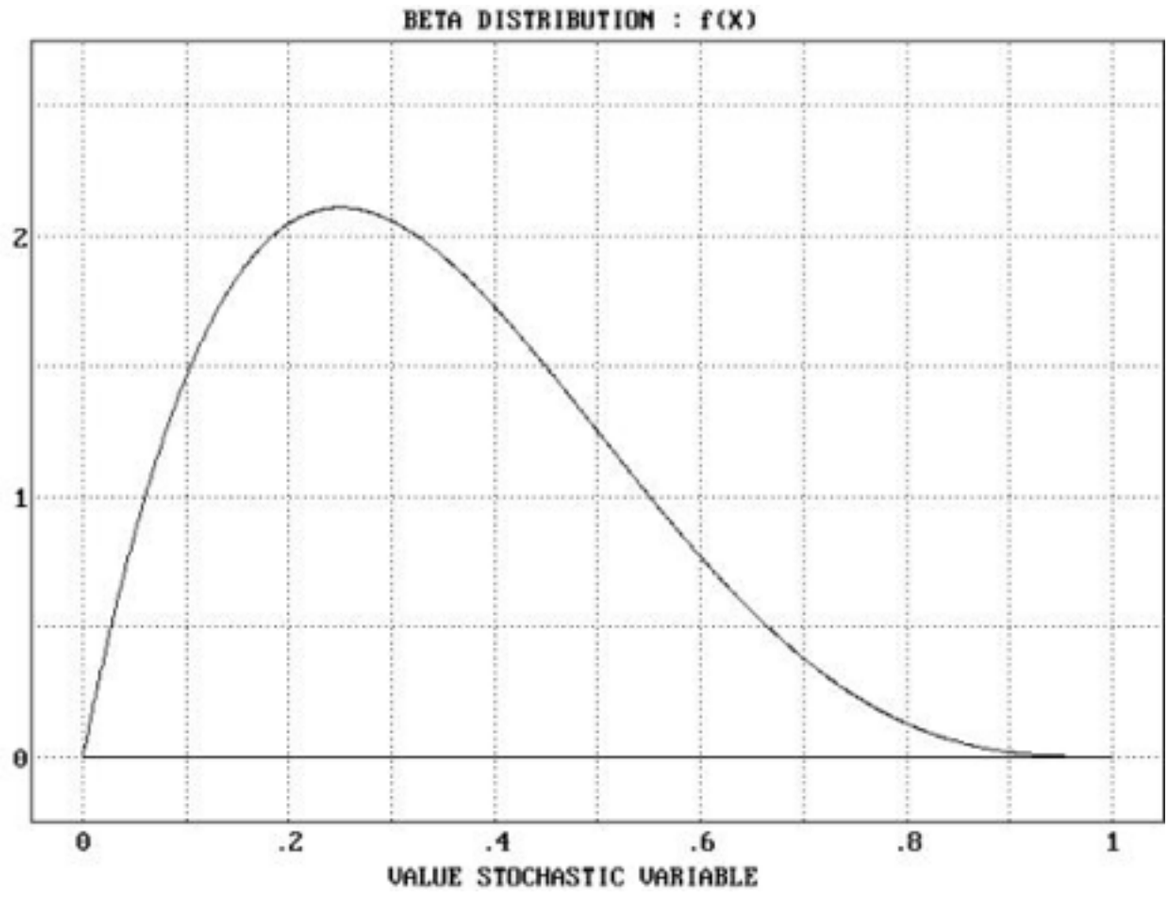
Prior

$$\mathrm{P}(y \mid x) = \frac{\mathrm{P}(x \mid y)\mathrm{P}(y)}{\mathrm{P}(x)}$$

Normalization Factor

A simple example: bias of a coin

ID	disease
1	0
2	1
3	0
4	1
5	1
6	1
7	0
8	1
9	…



Bayes theorem

Posterior

Model

Prior

$$\mathrm{P}(y \mid x) = \frac{\mathrm{P}(x \mid y)\mathrm{P}(y)}{\mathrm{P}(x)}$$

Model/Likelihood: Bernoulli

Observed values

y = (0,1,0,1,1,1,…,0)

Model/Likelihood: Bernoulli

Observed values

y = (0,1,0,1,1,1,…,0)

The probability mass function *f(s,Θ)* is

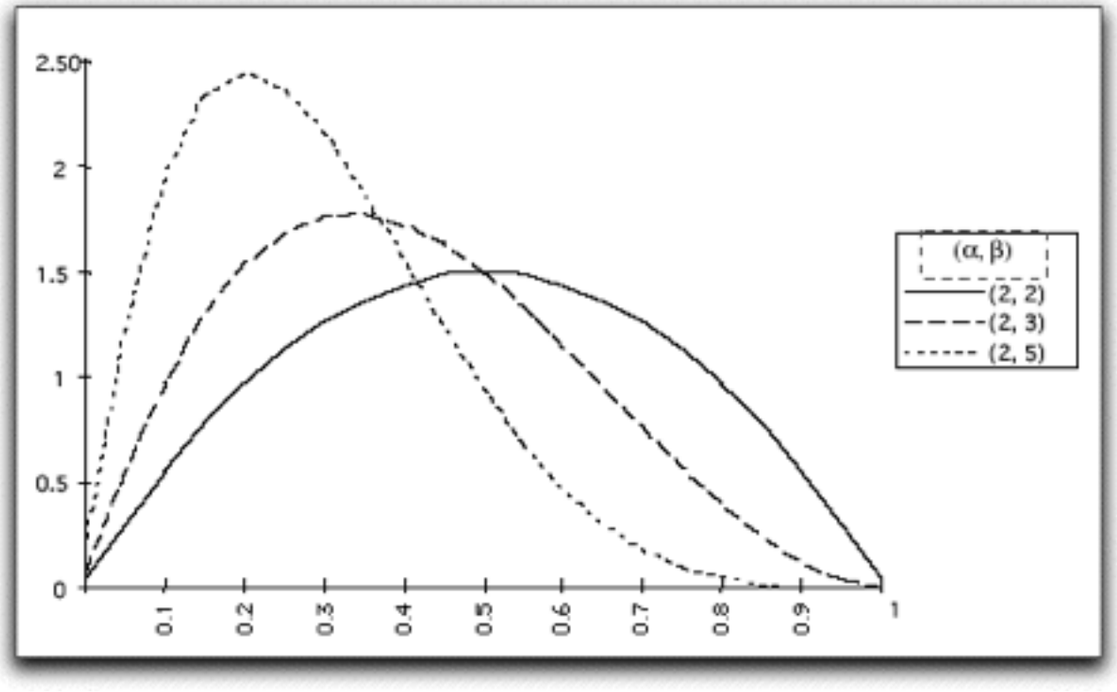
$$f(s,\Theta) = \begin{cases} \Theta & \text{if } s = 1 \\ 1 - \Theta & \text{if } s = 0 \end{cases}$$

where Θ is the “bias of the coin”. This can also be expressed as:

$$f(s,\Theta) = \Theta^{s}(1-\Theta)^{1-s}$$

Prior: Beta distribution

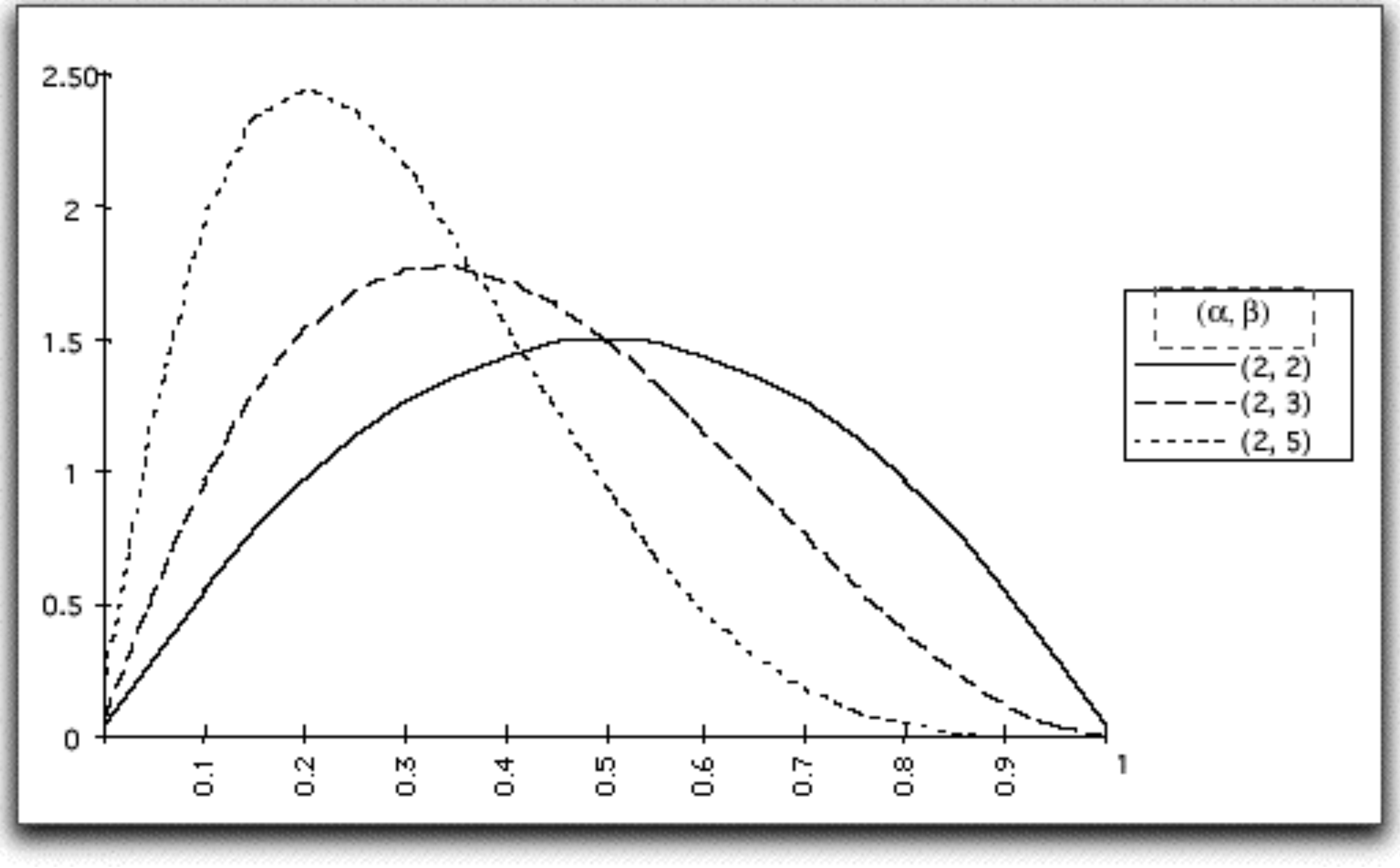
Probability Distribution Function



$$f(a,b,\Theta) = \frac{\Theta^a(1-\Theta)^b}{B(a,b)}$$

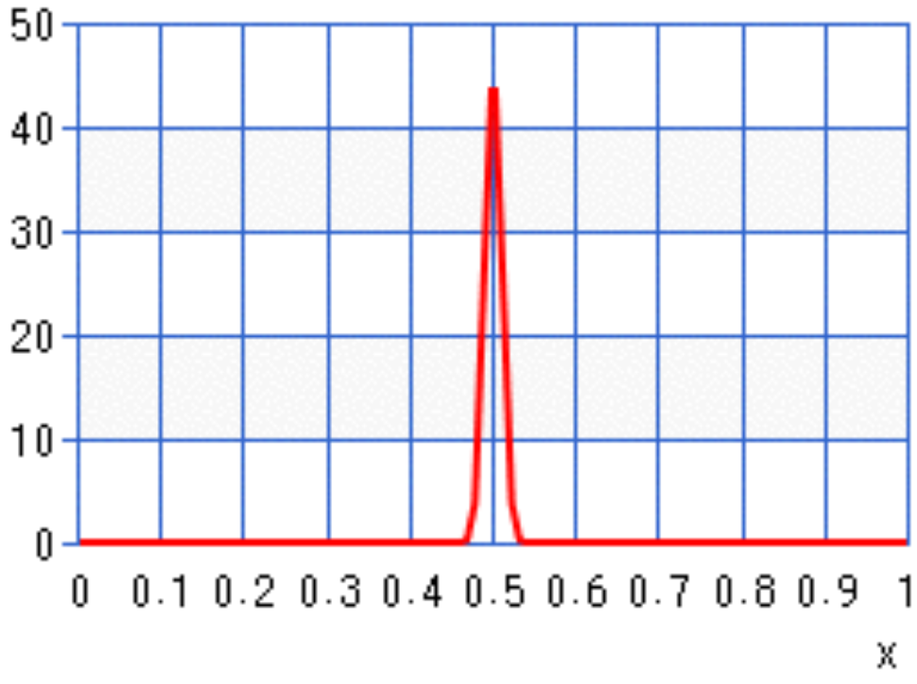
Conjugate to the Bernoulli distribution

Prior: Beta distribution

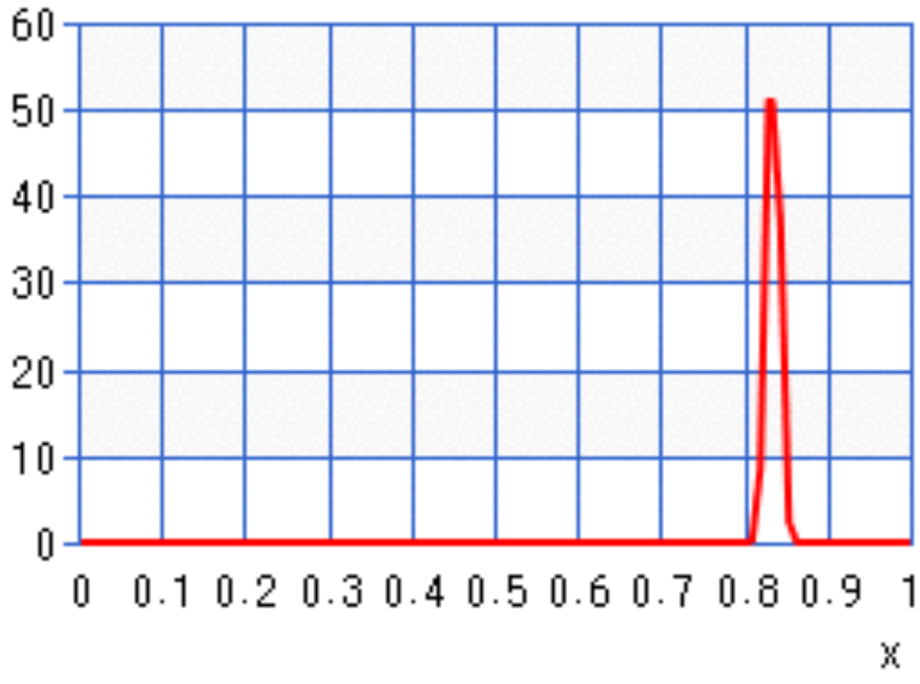


Bias of a Coin

Beta(1500,1500)

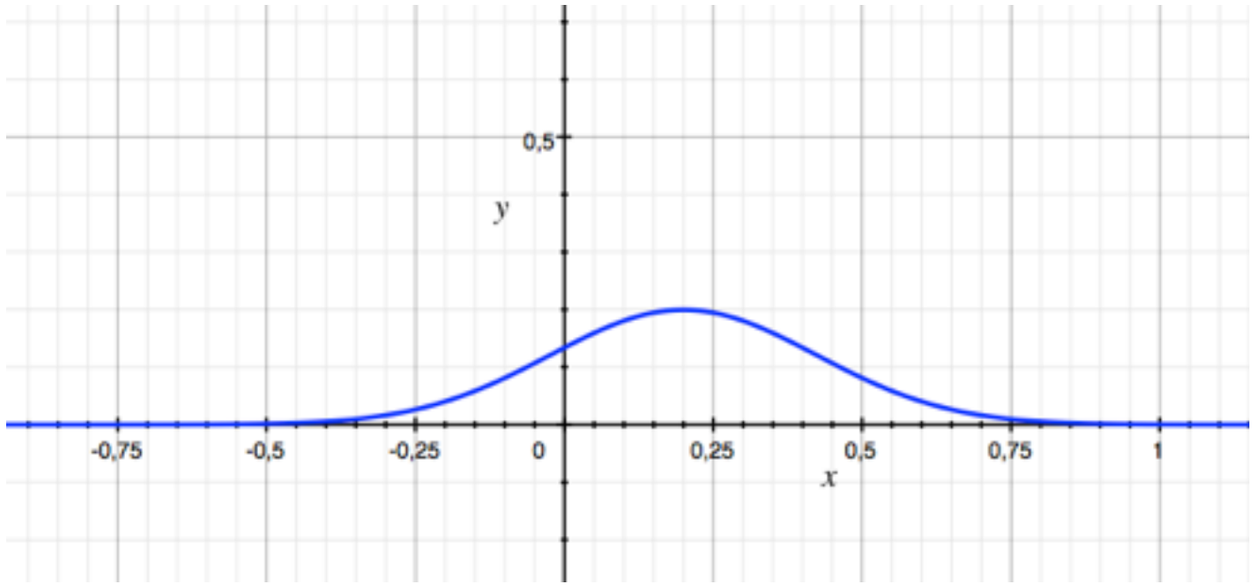


Beta(2500,500)



Another example

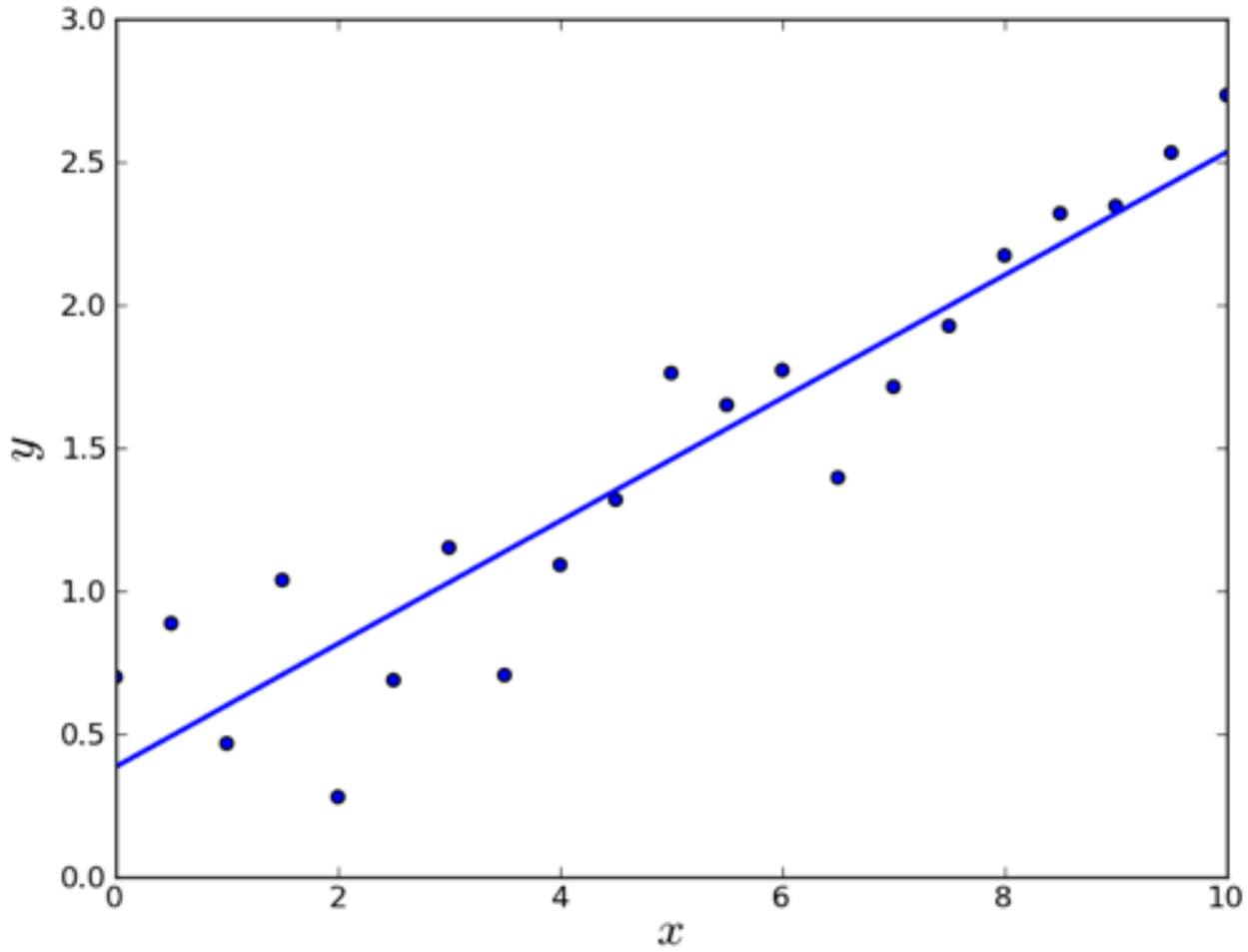
ID	Cholesterol
1	4.66
2	5.01
3	3.45
4	4.12
5	6.35
6	4.45
7	6.06
8	4.98
9	…



Known variance

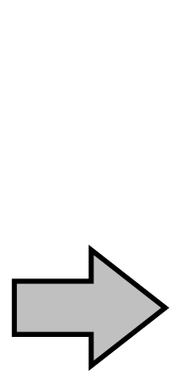
Another example

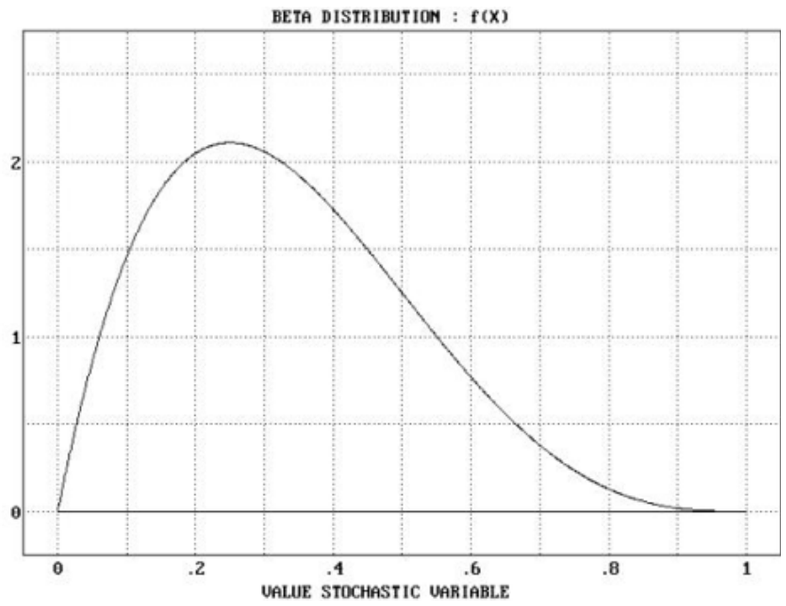
ID	Y	X
1	4.2	1.5
2	2.5	0.2
3	2.2	2
4	1.1	6.5
5	1.5	8.2
6	0.5	5.6
7	2.0	2.3
8	0.8	4.3
9	…	…

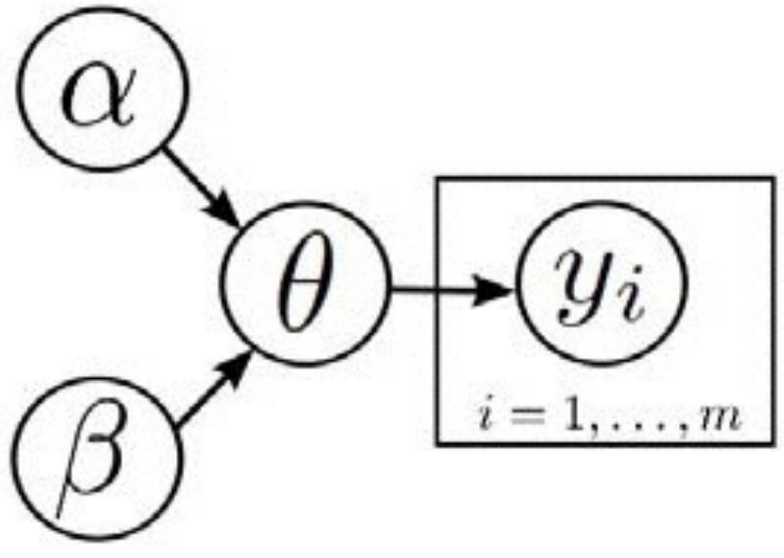


Graphical Models

I	disease
1	0
2	1
3	0
4	1
5	1
6	1
7	0
8	1
9	…







Probabilistic PCF

$$
\begin{array}{rcl}
e \in \mathbf{PCF}_p(\mathcal{X}) & ::= & x \mid c \mid e\ e \mid \lambda x.\, e \mid \texttt{let}\ x = e\ \texttt{in}\ e \\
& \mid & \texttt{letrec}\ f\ x = e \mid \texttt{case}\ e\ \texttt{with}\ \left[d_i\ \overline{x_i} \Rightarrow e_i\right]_i \\
& \mid & \texttt{unit}_{\mathrm{M}}\ e \mid \texttt{bind}_{\mathrm{M}}\ x = e\ \texttt{in}\ e \\
& \mid & \mathsf{observe}\, x \Rightarrow e\, \mathsf{in}\, e \mid \mathsf{infer}[e] \mid \mathsf{ran}(e) \\
& \mid & \mathsf{bernoulli}(e) \mid \mathsf{gauss}(e, e) \mid \mathsf{beta}(e, e) \mid \mathsf{uniform}() \\
& \mid & \mathsf{laplace}(e, e) \mid \mathsf{exponential}(e, e, e)
\end{array}
$$

Probabilistic Semantics

$$[\![\Gamma \vdash \mathtt{unit}_{\mathrm{M}}\ e : \mathfrak{M}[\tau]]\!]_\theta = x \mapsto \begin{cases} 1 & \text{if } x = [\![e]\!]_\theta \\ 0 & \text{otherwise} \end{cases}$$

$$[\![\Gamma \vdash \mathtt{bind}_{\mathrm{M}}\ x = e_1\ \mathtt{in}\ e_2 : \mathfrak{M}[\sigma]]\!]_\theta =$$

$$d \mapsto \sum_{g \in [\![\tau]\!]} ([\![e_1]\!]_\theta(g) \times [\![e_2]\!]_{\theta^g_x}(d))$$

$$[\![\Gamma \vdash \mathtt{letrec}^{\mathrm{m}}\ f\ x = e : \tau \to \sigma]\!]_\theta = \bigcup_{n \in \mathbb{N}} F^n(\bot_{[\![\tau \to \sigma]\!]})$$

$$\text{where } F(d) = [\![\Gamma, f : \tau \to \sigma \vdash \lambda x.\, e : \tau \to \sigma]\!]_{\theta^d_f}$$

Observe for Conditional Distributions

observe z ⇒ X(z) in Y

Observe for Conditional Distributions

observe z ⇒ X(z) in Y

$$P(y \mid x) = \frac{P(x \mid y)P(y)}{P(x)}$$

Semantics of Observe

$$[\![\Gamma \vdash \mathsf{observe}\, x \Rightarrow t \,\mathsf{in}\, u : \mathfrak{M}(A)]\!]_\rho =$$

$$d \mapsto \frac{[\![\Gamma \vdash u : \mathfrak{M}[A]]\!]_\rho\, d \times ([\![\Gamma, x : A \vdash t : \mathsf{bool}]\!]_{\rho_x^d} \mathsf{true})}{\sum_{a \in [\![A]\!]} [\![\Gamma \vdash u : \mathfrak{M}[A]]\!]_\rho\, a \times ([\![\Gamma, x : A \vdash t : \mathsf{bool}]\!]_{\rho_x^a} \mathsf{true})}$$

Semantics of Observe

$$[\![\Gamma \vdash \mathsf{observe}\, x \Rightarrow t \,\mathsf{in}\, u : \mathfrak{M}(A)]\!]_\rho =$$

$$d \mapsto \frac{[\![\Gamma \vdash u : \mathfrak{M}[A]]\!]_\rho\, d \times ([\![\Gamma, x : A \vdash t : \mathsf{bool}]\!]_{\rho_x^d} \mathsf{true})}{\sum_{a \in [\![A]\!]} [\![\Gamma \vdash u : \mathfrak{M}[A]]\!]_\rho\, a \times ([\![\Gamma, x : A \vdash t : \mathsf{bool}]\!]_{\rho_x^a} \mathsf{true})}$$

Filter and rescaling

Semantics of Observe

$$[\![\Gamma \vdash \mathsf{observe}\, x \Rightarrow t \,\mathsf{in}\, u : \mathfrak{M}(A)]\!]_\rho =$$

$$d \mapsto \frac{[\![\Gamma \vdash u : \mathfrak{M}[A]]\!]_\rho\, d \times ([\![\Gamma, x : A \vdash t : \mathsf{bool}]\!]_{\rho_x^d} \mathsf{true})}{\sum_{a \in [\![A]\!]} [\![\Gamma \vdash u : \mathfrak{M}[A]]\!]_\rho\, a \times ([\![\Gamma, x : A \vdash t : \mathsf{bool}]\!]_{\rho_x^a} \mathsf{true})}$$

Filter and rescaling

Differential Privacy

Private Queries?

Private Queries?

query

answer

Letter to the Editor

Methodological weakness in using correlation coefficients for assessing the interchangeability of analyte data between samples collected under different sampling conditions – the example of matrix metalloproteinase 9 determined in serum and plasma samples

Private Queries?

Does Critias have cancer?

query

answer

Private Queries?

Does Critias have cancer?

query

answer

I know he visited Atlantis

From Plato's Timaeus dialogue

Private Queries?

Letter to the Editor

Methodological weakness in using correlation coefficients for assessing the interchangeability of analyte data between samples collected under different sampling conditions – the example of matrix metalloproteinase 9 determined in serum and plasma samples

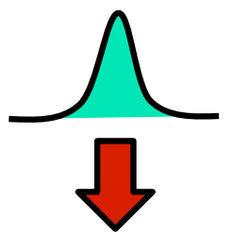
Does Critias have cancer?

I know he visited Atlantis

From Plato's Timaeus dialogue

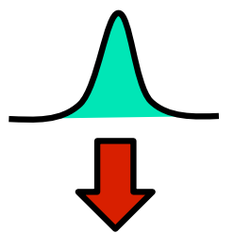
Differential Privacy

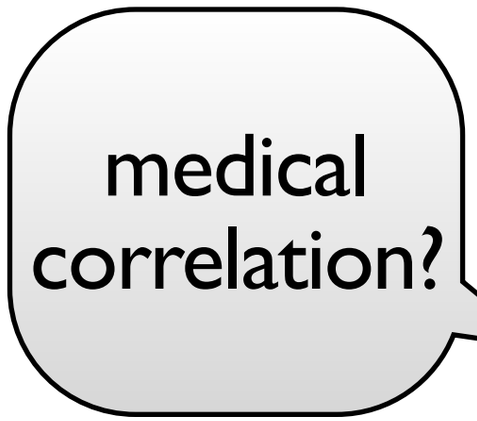
Noise



Differential Privacy

Noise





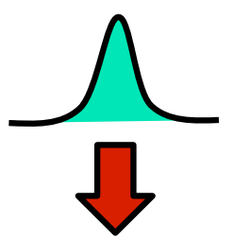
query

answer + noise

Methodological weakness in using correlation coefficients for assessing the interchangeability of analyte data between samples collected under different sampling conditions – the example of matrix metalloproteinase 9 determined in serum and plasma samples

Differential Privacy

Noise



Letter to the Editor

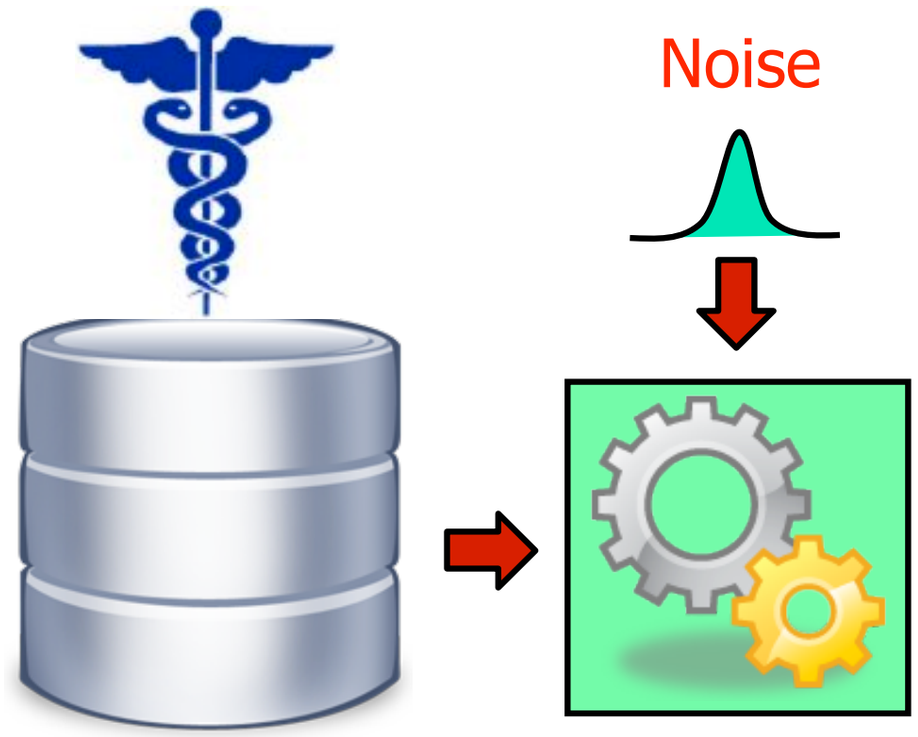
Methodological weakness in using correlation coefficients for assessing the interchangeability of analyte data between samples collected under different sampling conditions – the example of matrix metalloproteinase 9 determined in serum and plasma samples

?!?

query

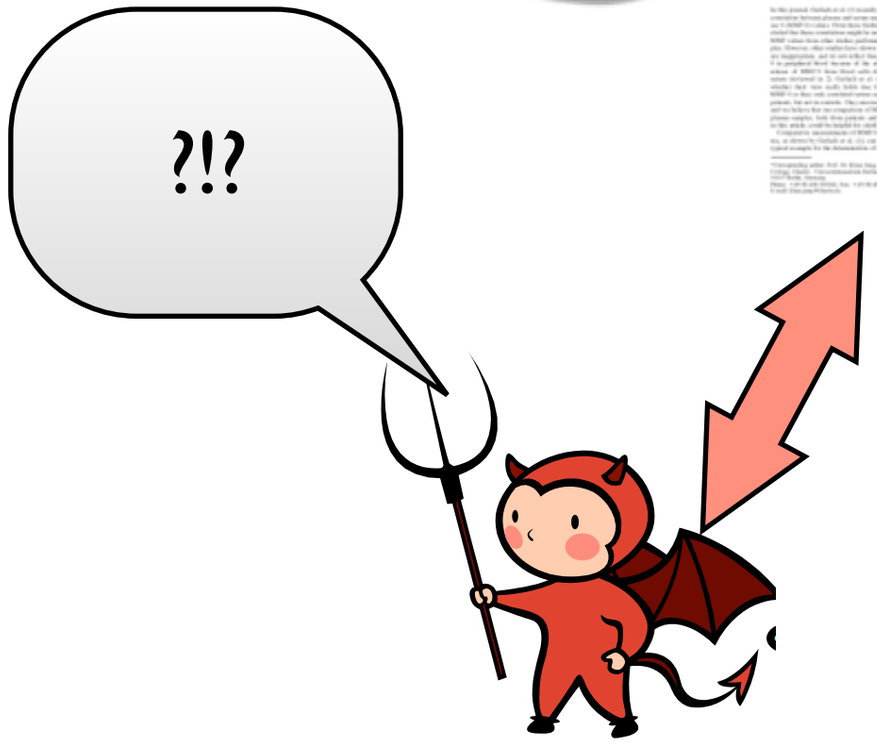
answer + noise

Differential Privacy



Letter to the Editor

Methodological weakness in using correlation coefficients for assessing the interchangeability of analyte data between samples collected under different sampling conditions – the example of matrix metalloproteinase 9 determined in serum and plasma samples



(ε,δ)-Differential Privacy

Definition

Given $\varepsilon,\delta \geq 0$, a probabilistic query Q: db $\rightarrow$ R is (ε,δ)-differentially private iff

$\forall b_1, b_2$:db differing in one row and for every $S \subseteq R$:

$$\Pr[Q(b_1) \in S] \leq \exp(\varepsilon) \cdot \Pr[Q(b_2) \in S] + \delta$$

(ε,δ)-Differential Privacy

A query returning a probability distribution

Definition

Given $\varepsilon,\delta \geq 0$, a probabilistic query Q: db → R is (ε,δ)-differentially private iff

$\forall b_1, b_2$:db differing in one row and for every $S \subseteq R$:

$$\Pr[Q(b_1) \in S] \leq \exp(\varepsilon) \cdot \Pr[Q(b_2) \in S] + \delta$$

(ε,δ)-Differential Privacy

Privacy parameters

Definition

Given $\varepsilon,\delta \geq 0$, a probabilistic query Q: db → R is (ε,δ)-differentially private iff

$\forall b_1, b_2$:db differing in one row and for every $S \subseteq R$:

$$\Pr[Q(b_1) \in S] \leq \exp(\varepsilon) \cdot \Pr[Q(b_2) \in S] + \delta$$

(ε,δ)-Differential Privacy

Definition

Given $\varepsilon,\delta \geq 0$, a probabilistic query Q: db $\rightarrow$ R is (ε,δ)-differentially private iff

$\forall b_1, b_2$:db differing in one row and for every $S \subseteq R$:

$$\Pr[Q(b_1) \in S] \leq \exp(\varepsilon) \cdot \Pr[Q(b_2) \in S] + \delta$$

a quantification over all the databases

(ε,δ)-Differential Privacy

Definition

Given $\varepsilon,\delta \geq 0$, a probabilistic query Q: db → R is (ε,δ)-differentially private iff

$\forall b_1, b_2$:db differing in one row and for every $S \subseteq R$:

$$\Pr[Q(b_1) \in S] \leq \exp(\varepsilon) \cdot \Pr[Q(b_2) \in S] + \delta$$

a notion of adjacency or distance

(ε,δ)-Differential Privacy

Definition

Given $\varepsilon,\delta \geq 0$, a probabilistic query Q: db → R is (ε,δ)-differentially private iff

$\forall b_1, b_2$:db differing in one row and for every $S \subseteq R$:

$$\Pr[Q(b_1) \in S] \leq \exp(\varepsilon) \cdot \Pr[Q(b_2) \in S] + \delta$$

and over all the possible outcomes

ε-Differential Privacy

Definition

Given $\varepsilon \geq 0$, a probabilistic query $Q: db \rightarrow R$ is ε-differentially private iff $\forall b_1, b_2$:db differing in one row and for every $S \subseteq R$:

$$Pr[Q(b_1) \in S] \leq \exp(\varepsilon) \cdot Pr[Q(b_2) \in S]$$

ε-Differential Privacy

Definition
Given $\varepsilon \geq 0$, a probabilistic query $Q: db \rightarrow R$ is ε-differentially private iff $\forall b_1, b_2$:db differing in one row and for every $S \subseteq R$:

$$Pr[Q(b_1) \in S] \leq \exp(\varepsilon) \cdot Pr[Q(b_2) \in S]$$

Let's substitute a concrete instance:

$$Pr[Q(b \cup \{x\}) \in S] \leq \exp(\varepsilon) \cdot Pr[Q(b) \in S]$$

Let's use the two quantifiers:

$$\exp(-\varepsilon) \cdot Pr[Q(b) \in S] \leq Pr[Q(b \cup \{x\}) \in S] \leq \exp(\varepsilon) \cdot Pr[Q(b) \in S]$$

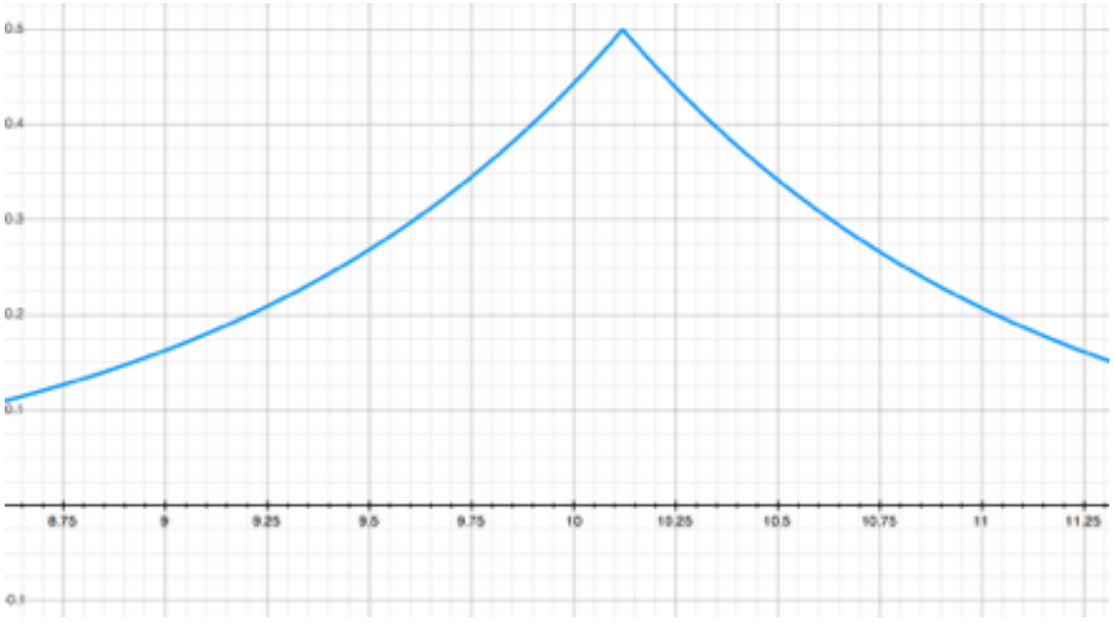
and so:

$$\log \left| \frac{Pr[Q(b \cup \{x\}) \in S]}{Pr[Q(b) \in S]} \right| \leq \varepsilon$$

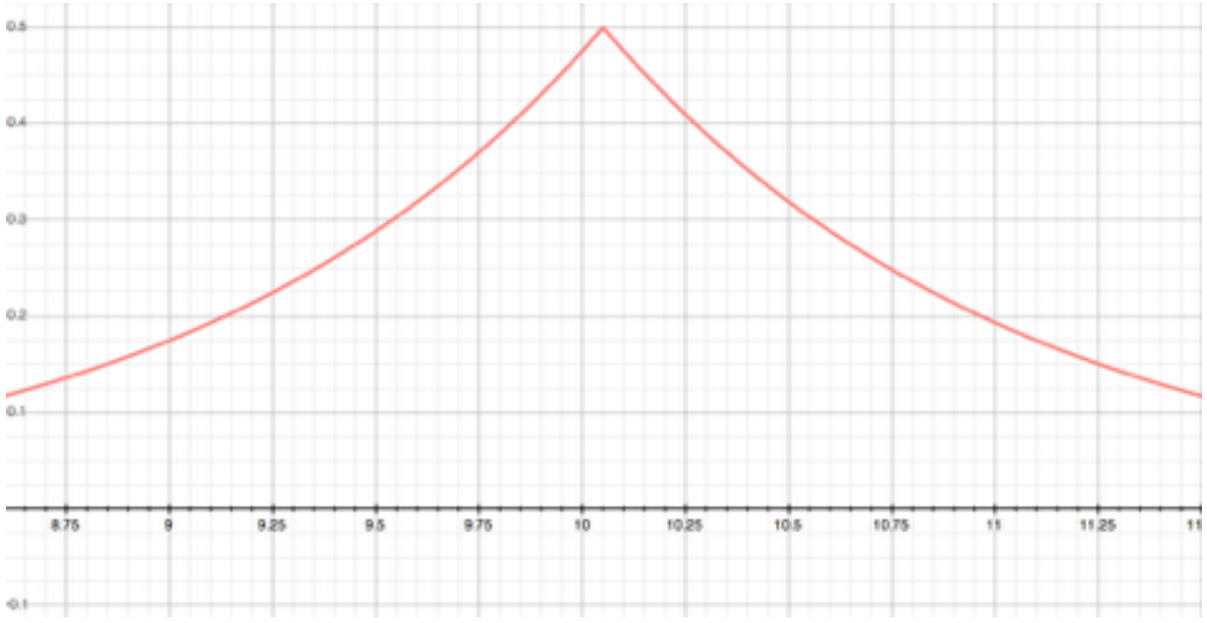
ε-Differential Privacy

Q : db => R probabilistic

Q(b)

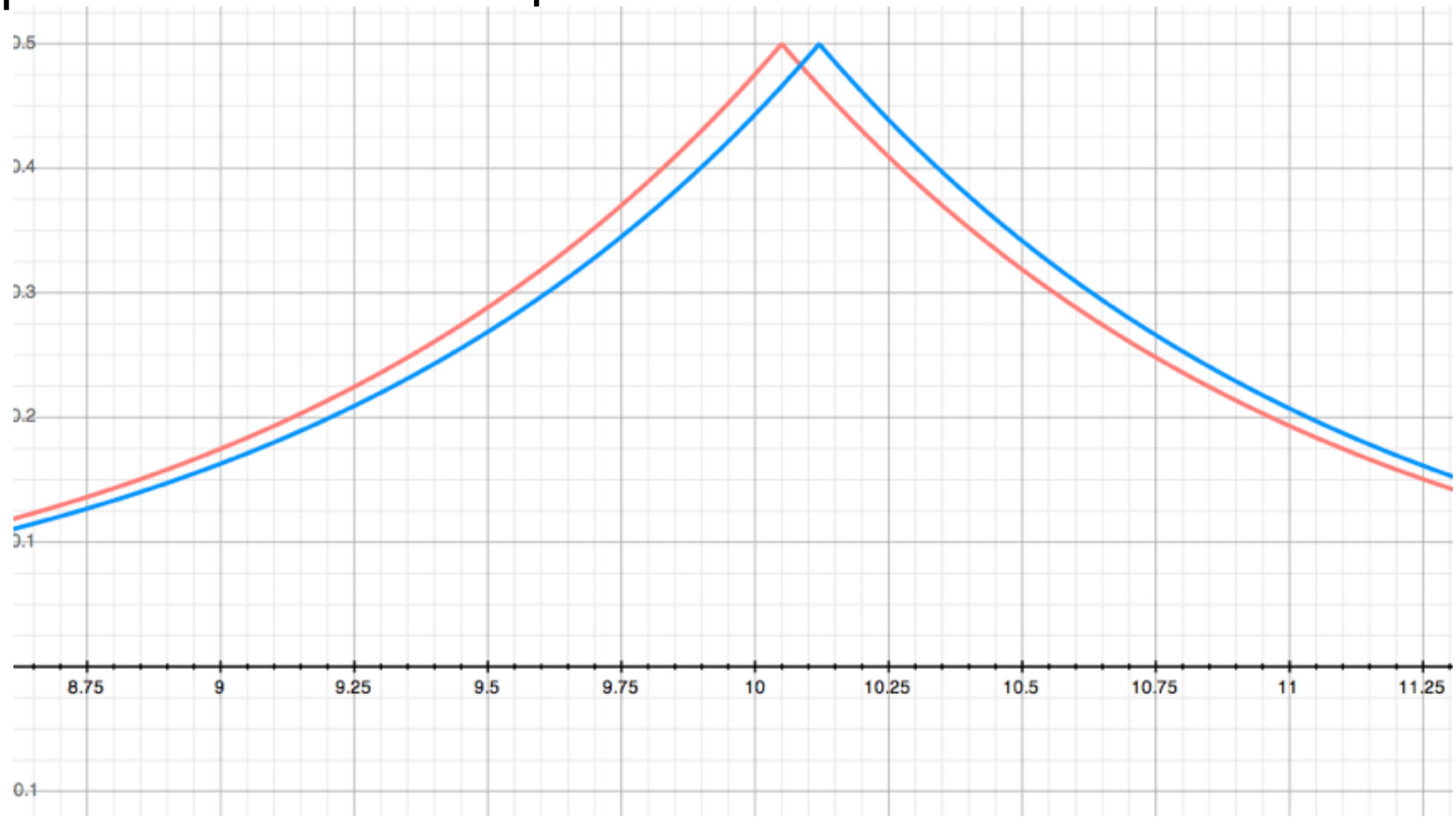


Q(b∪{x})



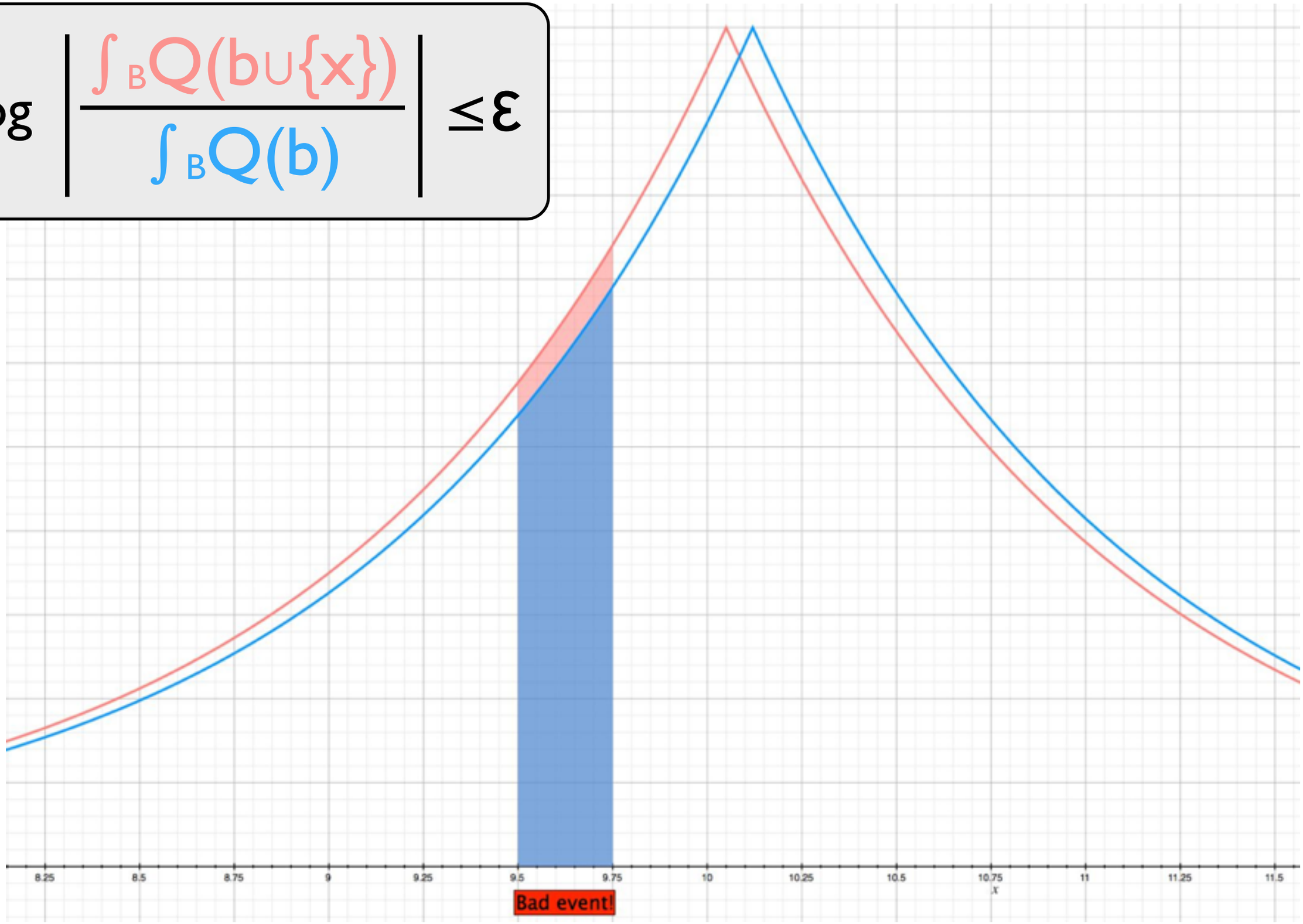
ε-Differential Privacy

$$\log \left| \frac{\Pr[Q(b \cup \{x\}) \in S]}{\Pr[Q(b) \in S]} \right| \leq \varepsilon$$



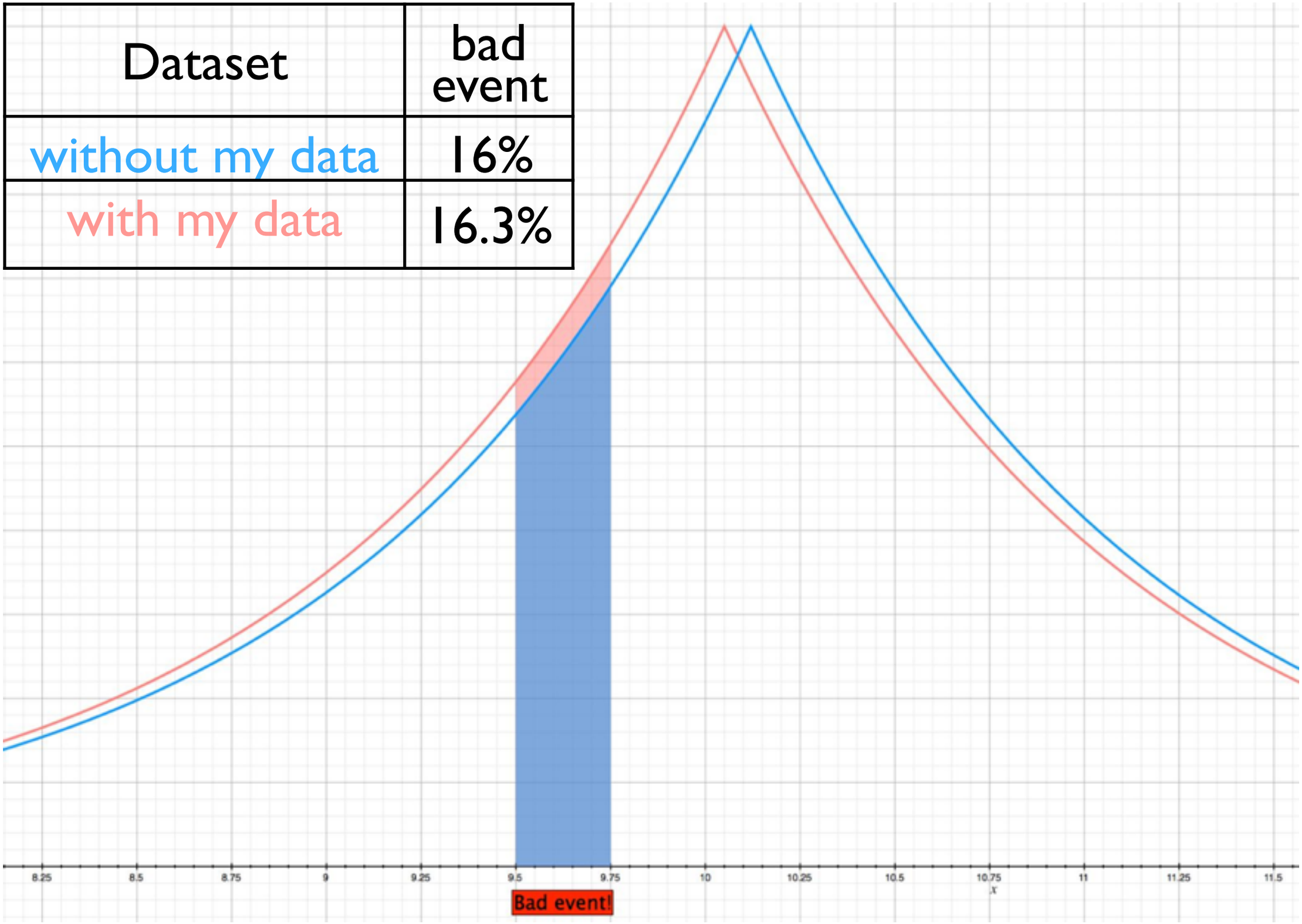
Probability of a bad event

$$\log \left| \frac{\int_B Q(b \cup \{x\})}{\int_B Q(b)} \right| \leq \varepsilon$$



Probability of a bad event

Dataset	bad event
without my data	16%
with my data	16.3%

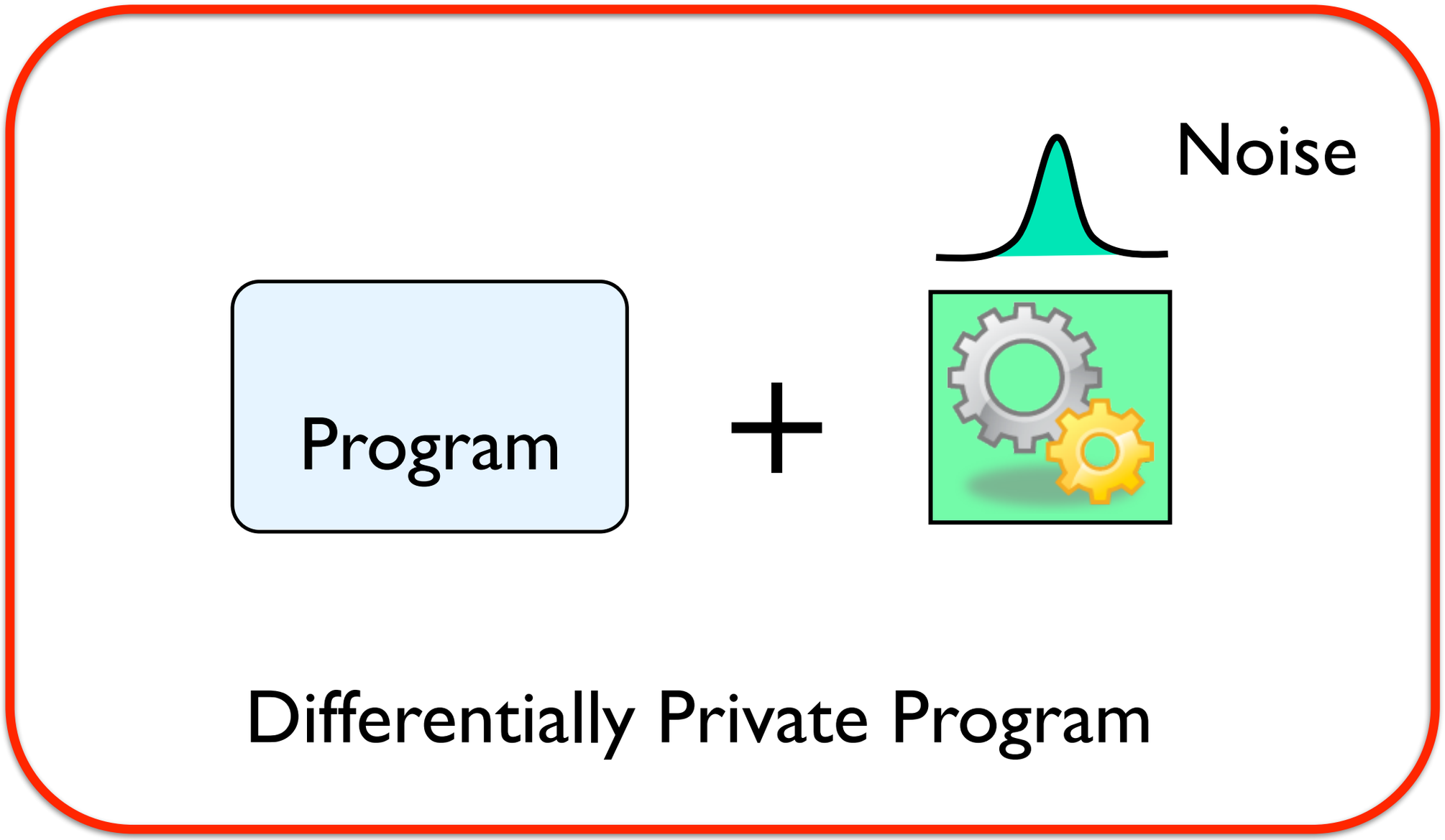


(ε,δ)-Differential privacy

for $b \sim_1 b'$ $\quad \Pr[Q(b') \in S] \leqq e^{\varepsilon} \Pr[Q(b) \in S] + \delta$

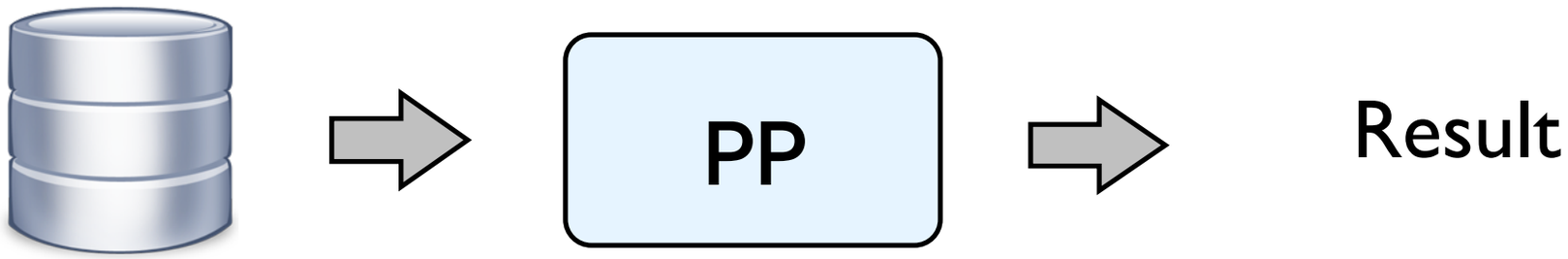
(ε,δ)-Differential privacy

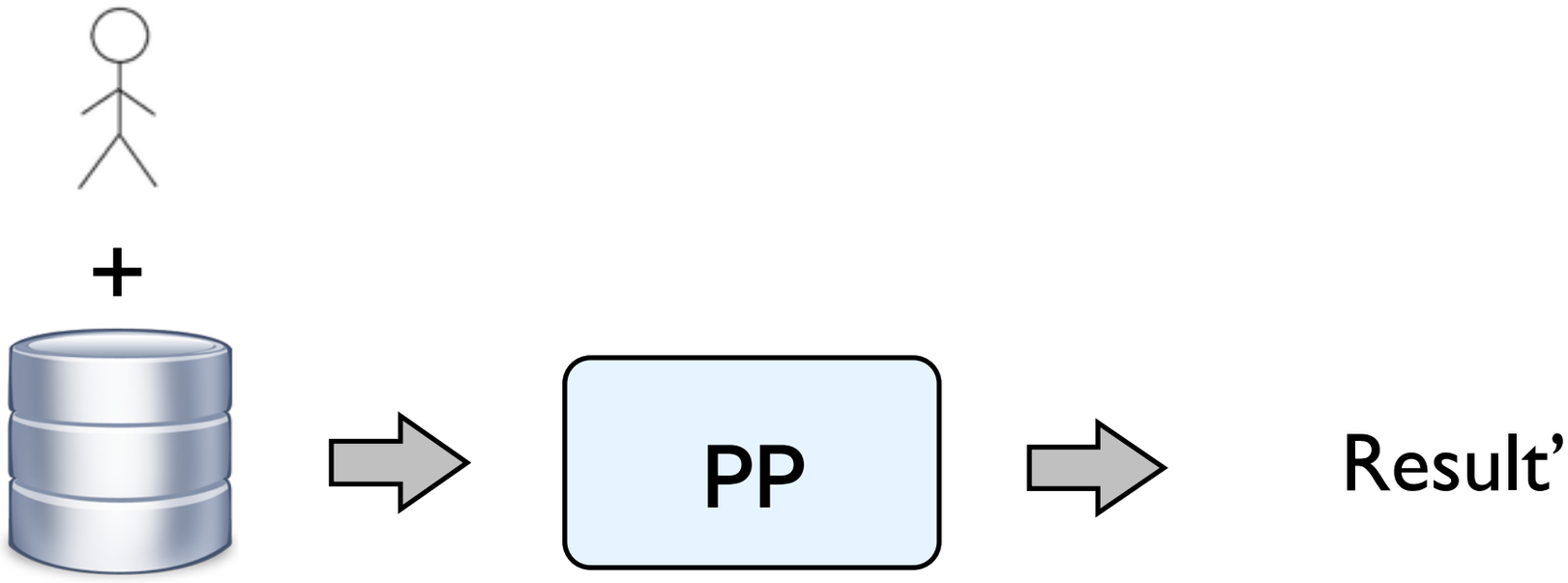
for $b \sim_1 b'$ $\quad \Pr[Q(b') \in S] \leqq e^{\varepsilon} \Pr[Q(b) \in S] + \delta$



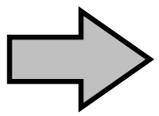
Differentially Private Program

Sensitivity

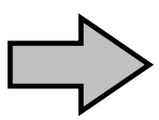




Sensitivity



PP

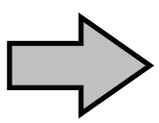


Result

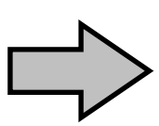
Sensitivity

≤K

+



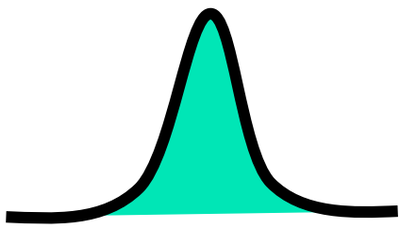
PP



Result'

Calibrating the Noise on the Sensitivity

Noise($\frac{k}{\varepsilon}$)



Program

k sensitive

+

ε-Differentially Private Program

HOARe^2: Higher Order Approximate Relational Refinement Types for DP

Components

- Relational Refinement Types
- Higher Order Refinements
- Partiality Monad
- Semantic Subtyping
- Approximate Equivalence for Distributions

Barthe et al, POPL'15

HOARe^2: Higher Order Approximate Relational Refinement Types for DP

$$
\begin{array}{rcl}
e \in \mathbf{PCF}_p(\mathcal{X}) & ::= & x \mid c \mid e\ e \mid \lambda x.\, e \mid \texttt{let}\ x = e\ \texttt{in}\ e \\
& \mid & \texttt{letrec}\ f\ x = e \mid \texttt{case}\ e\ \texttt{with}\ \left[d_i\ \overline{x_i} \Rightarrow e_i\right]_i \\
& \mid & \texttt{unit}_{\mathrm{M}}\ e \mid \texttt{bind}_{\mathrm{M}}\ x = e\ \texttt{in}\ e
\end{array}
$$

HOARe^2: Higher Order Approximate Relational Refinement Types for DP

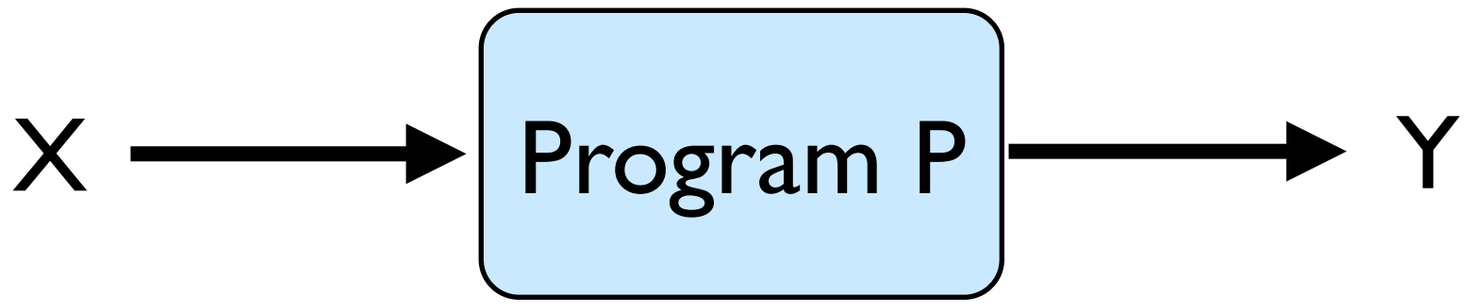
Components

- Relational Refinement Types
- Higher Order Refinements
- Partiality Monad
- Semantic Subtyping
- Approximate Equivalence for Distributions

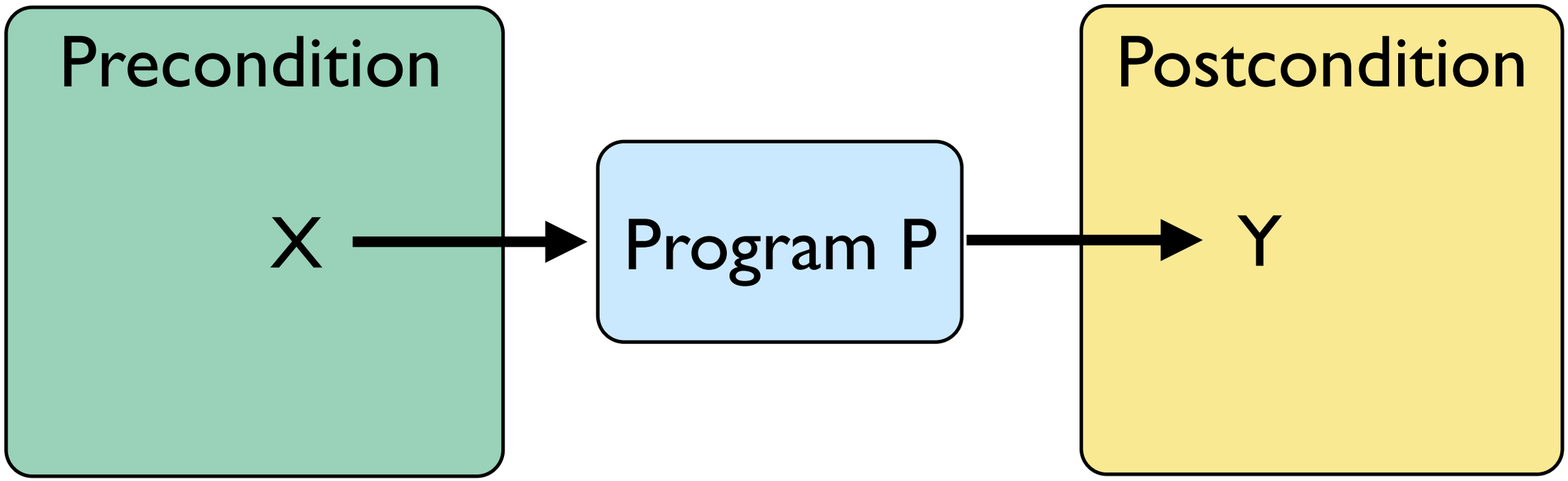
reasoning about two runs of a program

Barthe et al, POPL'15

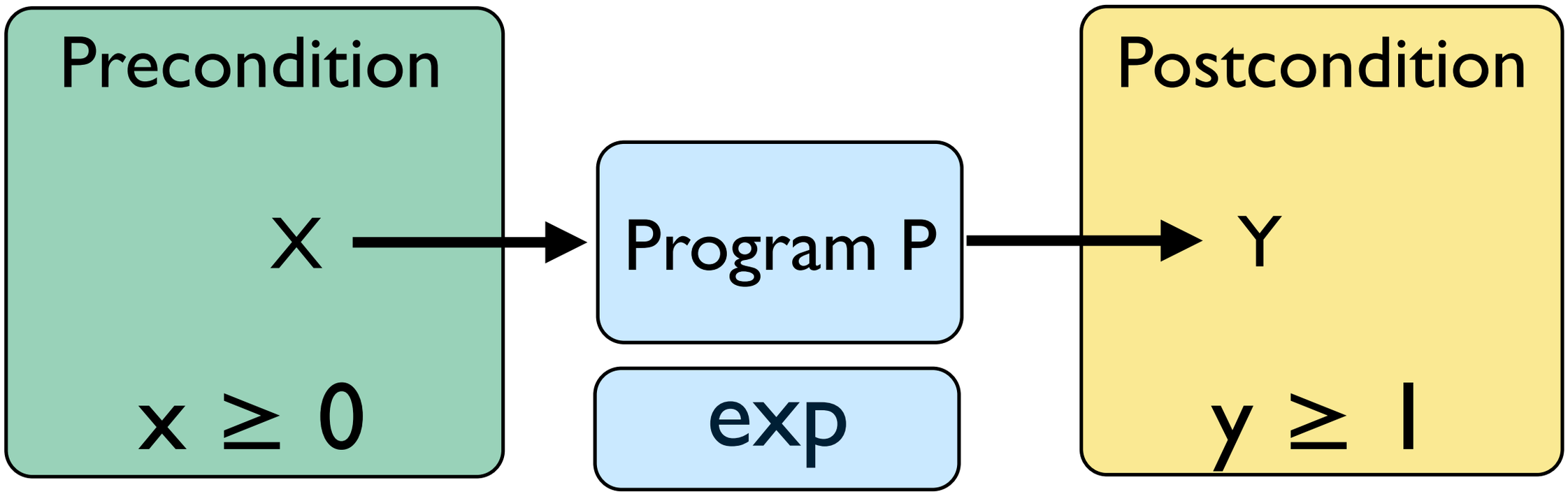
Program Logic



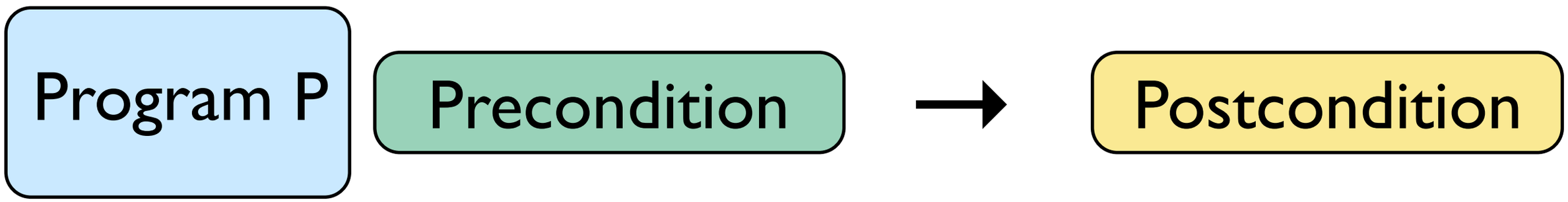
Program Logic



Program Logic



Refinement Type



$$P : \{x \mid Pre(x)\} \rightarrow \{y \mid Post(y)\}$$

Refinement Type

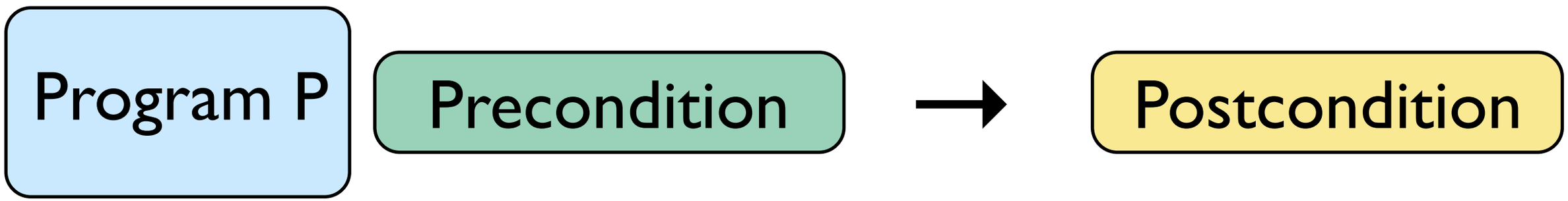
Program P

Precondition → Postcondition

$$P : \{x \mid Pre(x)\} \rightarrow \{y \mid Post(y)\}$$

Logical Predicates

Refinement Type

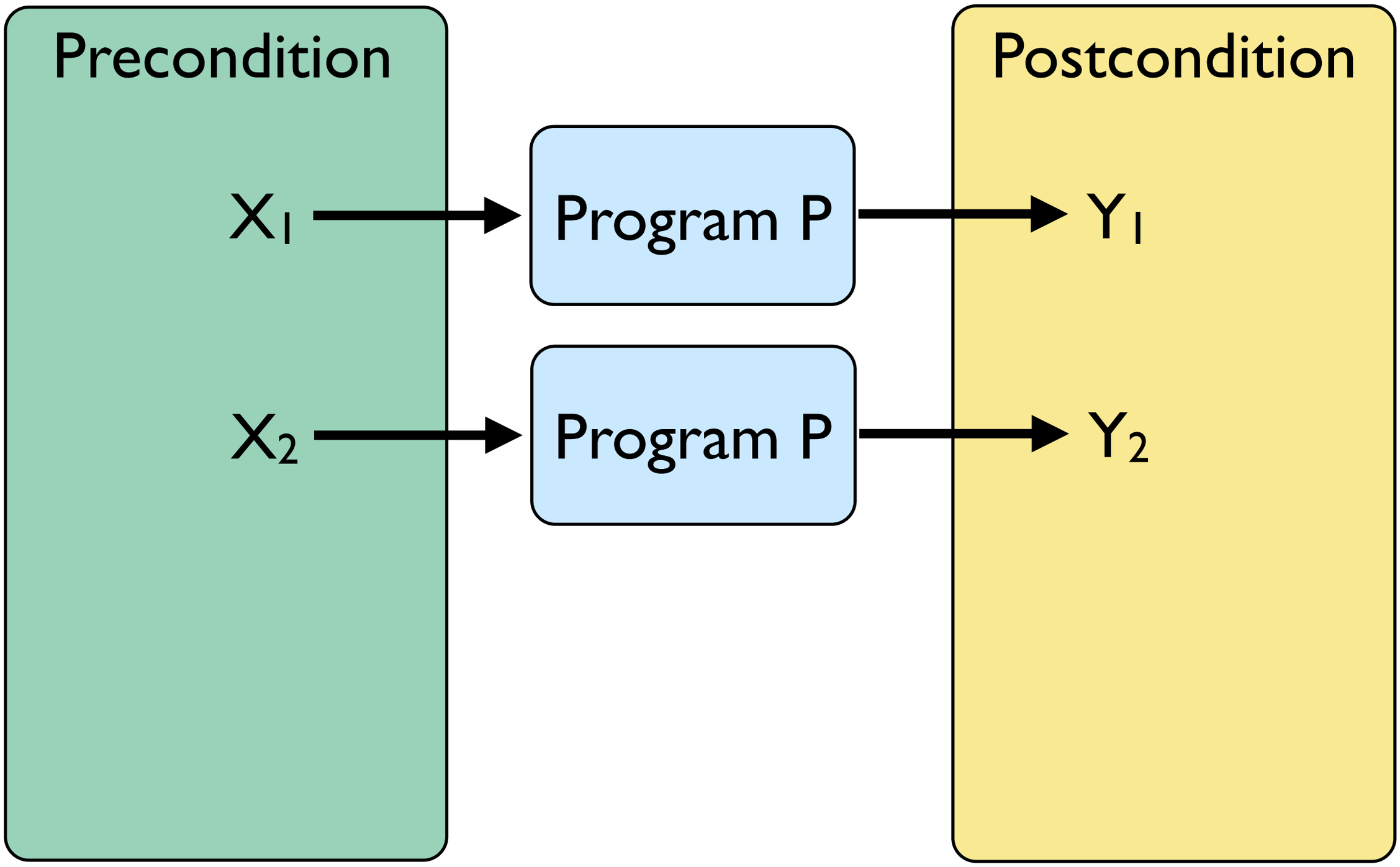


$$P : \{x \mid Pre(x)\} \rightarrow \{y \mid Post(y)\}$$

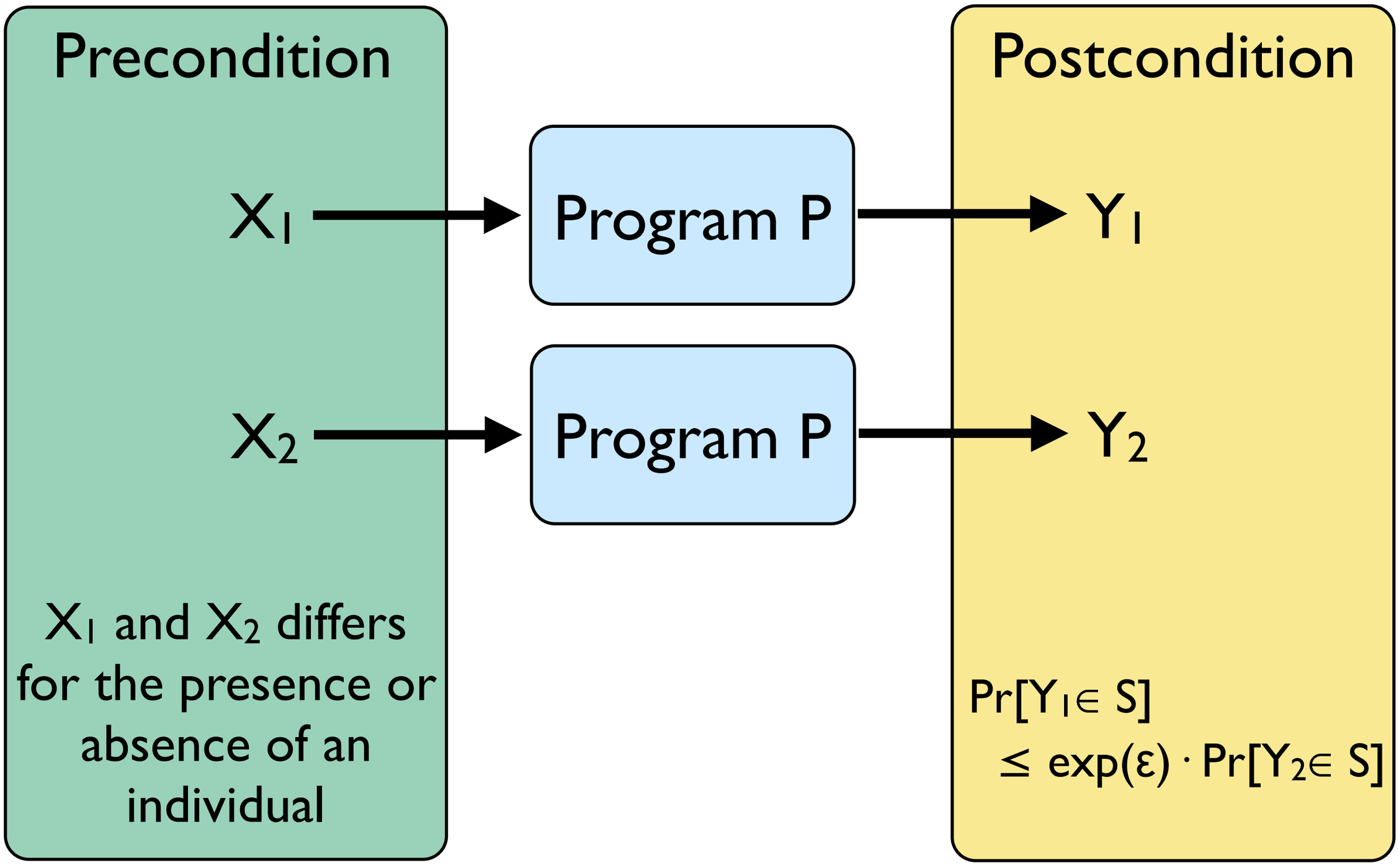
Example

$$exp : \{x \mid x \geq 0\} \rightarrow \{y \mid y \geq 1\}$$

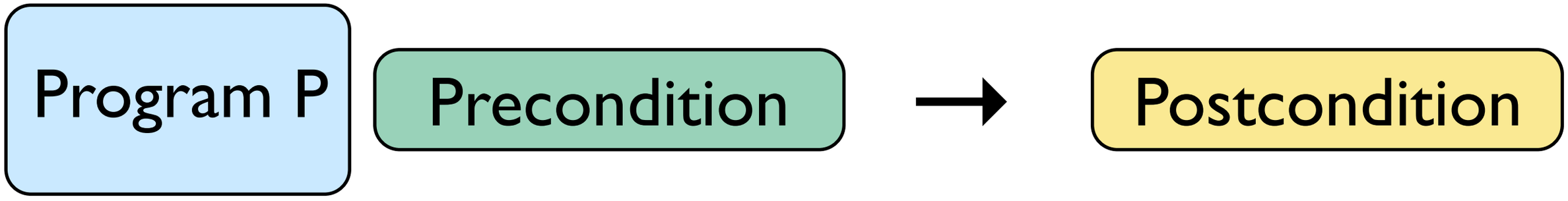
Differential privacy as a Relational Property



Differential privacy as a Relational Property



HOARe²: Relational Refinement Types



$$P : \{x \mid Pre(x_1, x_2)\} \rightarrow \{y \mid Post(y_1, y_2)\}$$

HOARe²: Relational Refinement Types

Program P

Precondition → Postcondition

$P : \{x \mid Pre(x_1, x_2)\} \rightarrow \{y \mid Post(y_1, y_2)\}$

Logical Relations

HOARe²: Relational Refinement Types

Program P | Precondition → Postcondition

$$P : \{x \mid \mathrm{Pre}(x_1, x_2)\} \rightarrow \{y \mid \mathrm{Post}(y_1, y_2)\}$$

Example: Monotonicity of exponential

$$\exp : \{x \mid x_1 \leq x_2\} \rightarrow \{y \mid y_1 \leq y_2\}$$

HOARe^2: Higher Order Approximate Relational Refinement Types for DP

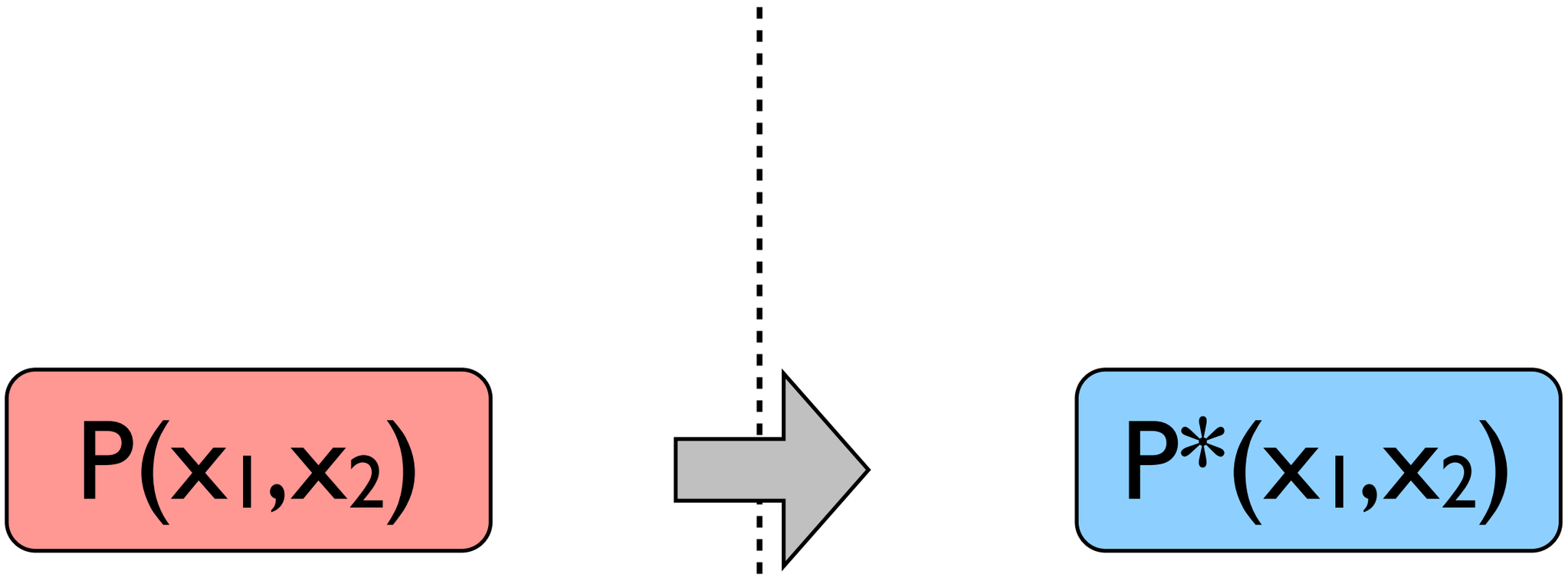
Components

- Relational Refinement Types
- Higher Order Refinements
- Partiality Monad
- Semantic Subtyping
- Approximate Equivalence for Distributions

reasoning about DP

Barthe et al, POPL'15

Lifting of P



Relation

Relation over distributions

Lifting of P

- Given dist. μ_1 over A and μ_2 over B:

$$\mu_1 \; P^* \; \mu_2$$

iff it exists a dist. μ over AxB s.t.

- $\mu(x) > 0$ implies $x \in R$
- $\boldsymbol{\pi}_1\mu \leqq \mu_1$ and $\boldsymbol{\pi}_2\mu \leqq \mu_2$

(ε,δ)-Lifting of P

- Given dist. μ_1 over A and μ_2 over B:

$$\mu_1 \; P^*_{\varepsilon\delta} \; \mu_2$$

iff it exists a dist. μ over AxB s.t.

- $\mu(x) > 0$ implies $x \in R$
- $\boldsymbol{\pi}_1\mu \leqq \mu_1$ and $\boldsymbol{\pi}_2\mu \leqq \mu_2$
- $\max_A\left(\boldsymbol{\pi}_i\mu(A) - e^{\varepsilon}\,\mu_i(A),\; \mu_i(A) - e^{\varepsilon}\,\boldsymbol{\pi}_i\mu(A)\right) \leqq \delta$

Verifying Differential Privacy

If we can conclude

$$C : \{x : db \mid d(y_1, y_2) \leqq 1\} \to \{y : O \mid y_1 =^{*}_{\varepsilon\delta} y_2\}$$

then C is (ε,δ)-differentially private.

Programming Languages for Differentially Private Probabilistic Inference

Differential Privacy

Probabilistic Inference

Programming Languages for Differentially Private Probabilistic Inference

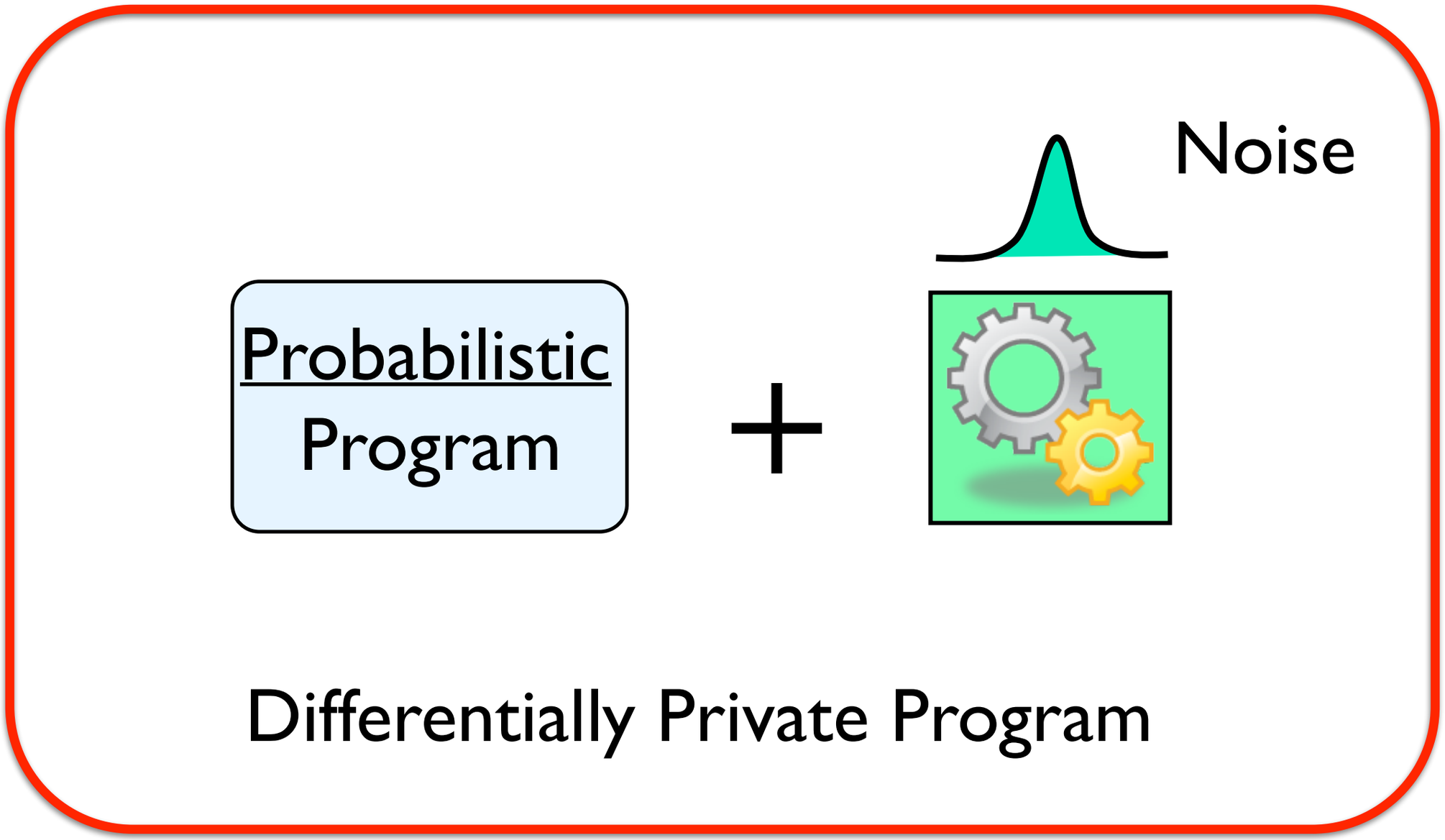
Differential Privacy

Probabilistic Inference

Programming Language Tools

Adding noise

Adding noise

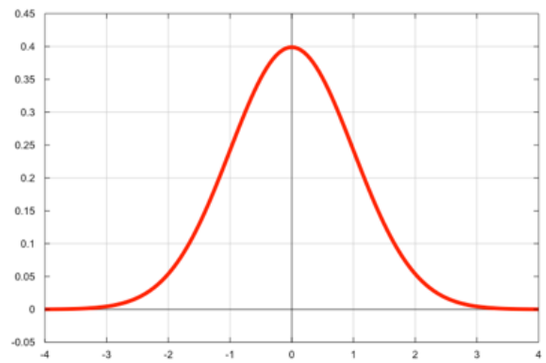


Adding noise

Adding noise

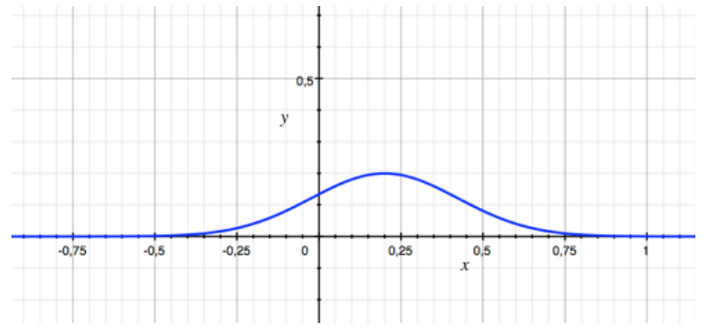
Adding noise

Prior Distribution



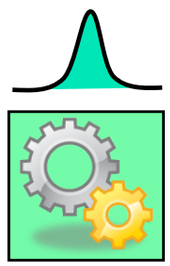
Posterior Distribution

Probabilistic Program



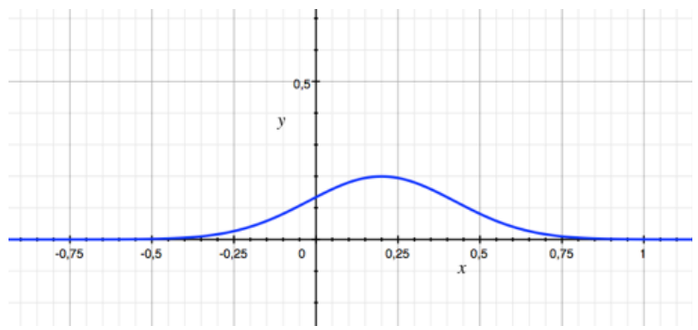
Adding noise

Prior Distribution



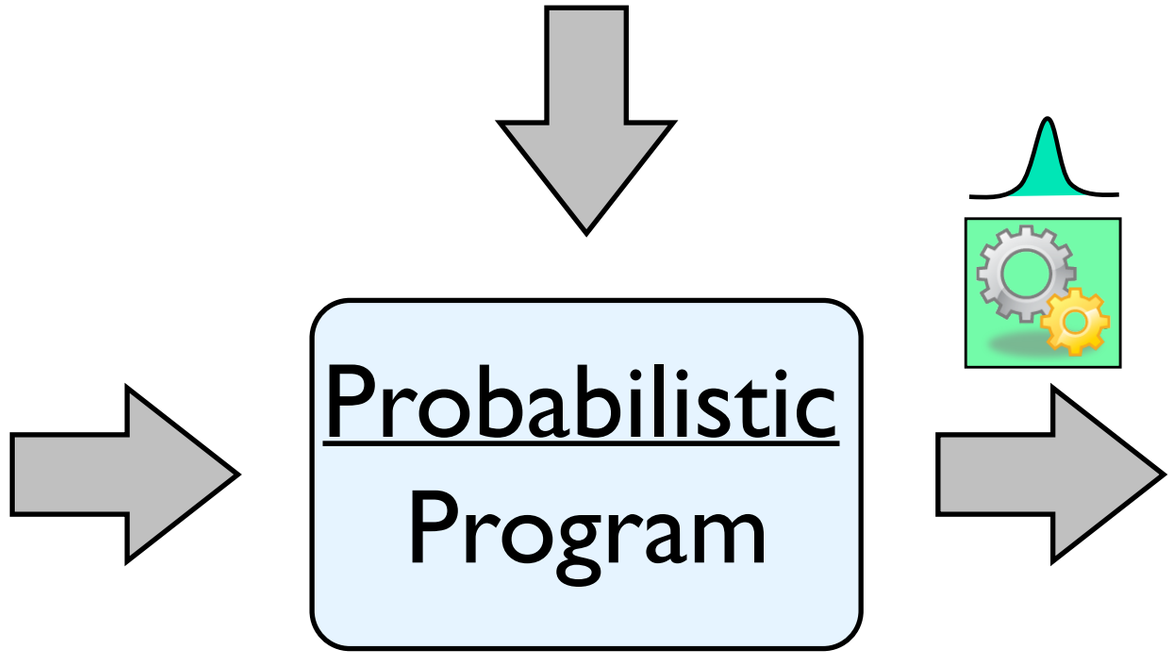
Probabilistic Program

Posterior Distribution

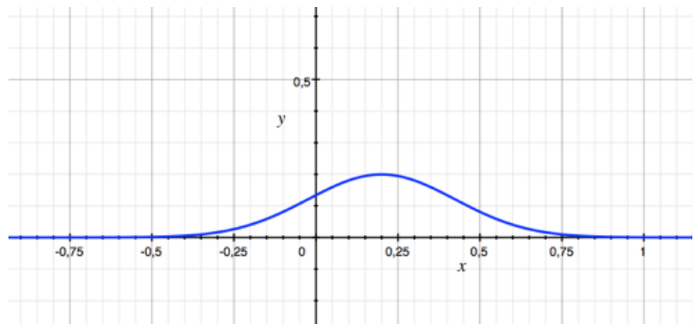


Adding noise

Prior Distribution

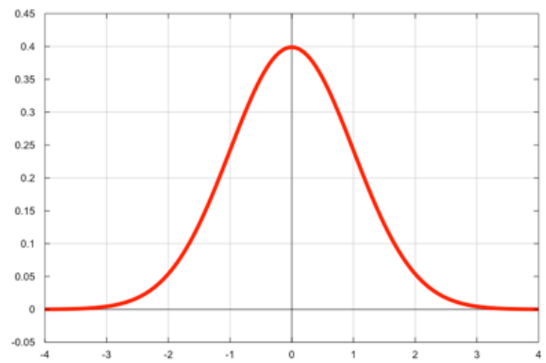


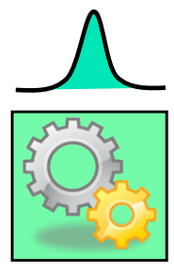
Posterior Distribution



Adding noise

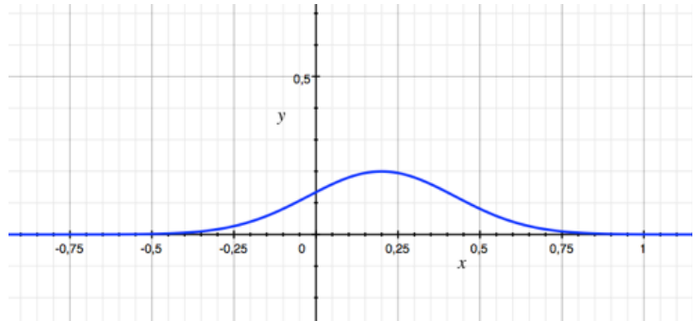
Prior Distribution





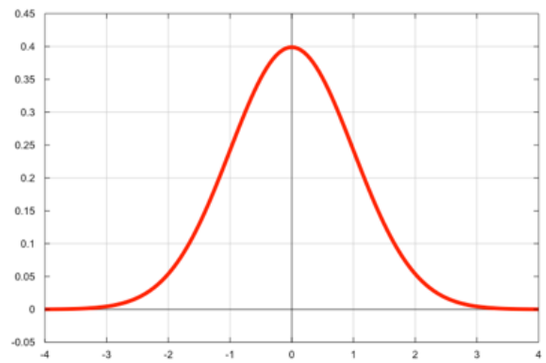
Posterior Distribution

Probabilistic Program



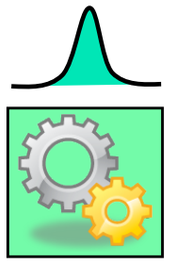
Adding noise

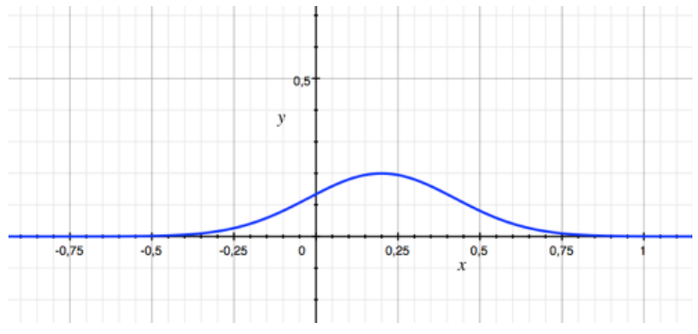
Prior Distribution



Posterior Distribution

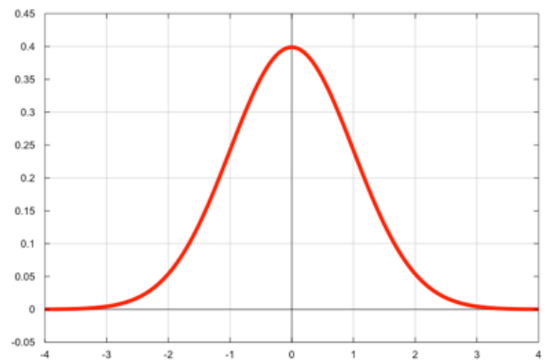
Probabilistic Program





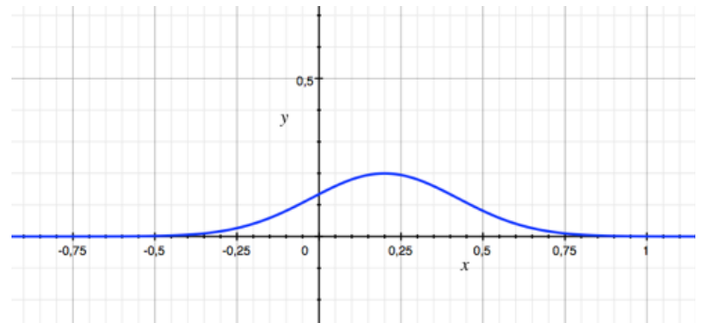
Adding noise

Prior Distribution



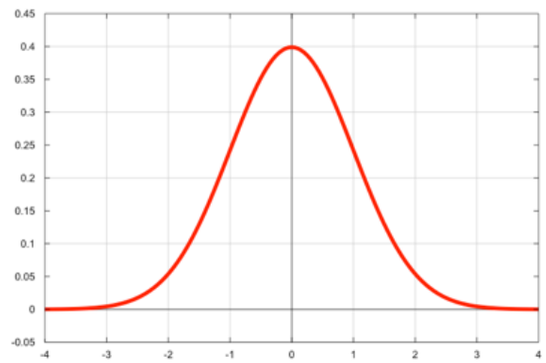
Posterior Distribution

Probabilistic Program



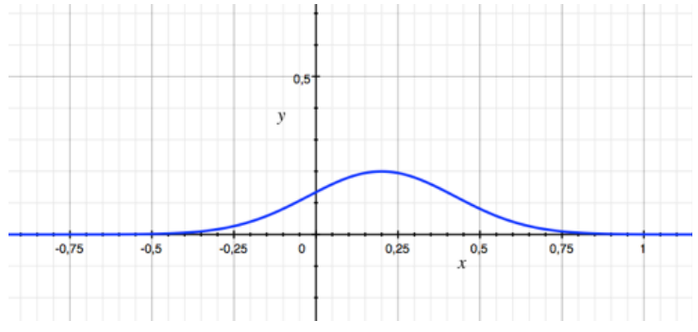
Adding noise

Prior Distribution



Posterior Distribution

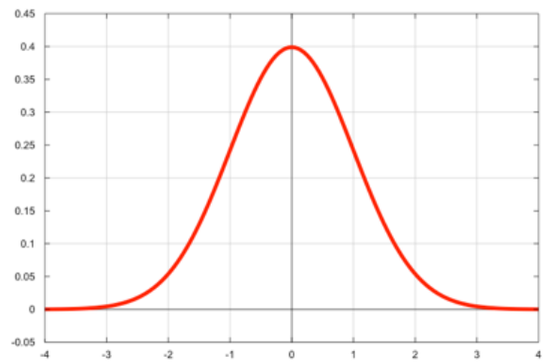
Probabilistic Program



Probabilistic Inference

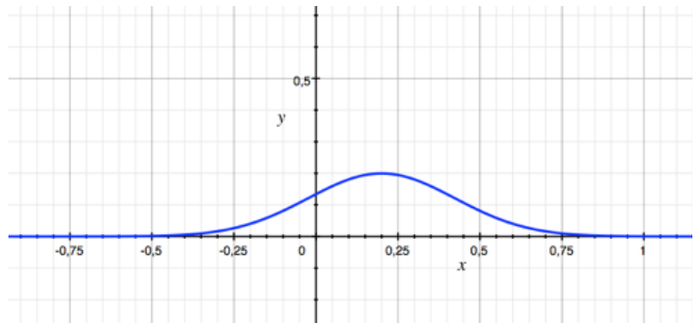
Adding noise

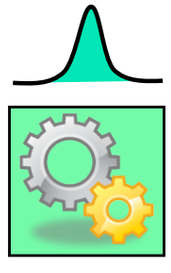
Prior Distribution



Posterior Distribution

Probabilistic Program

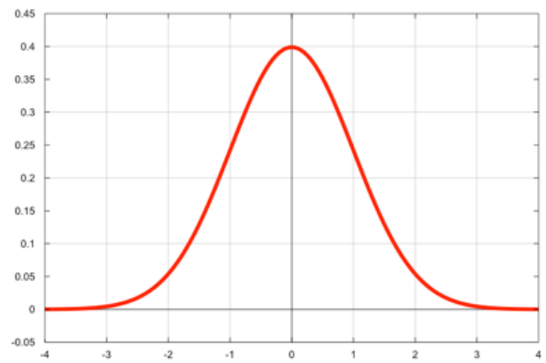




Probabilistic Inference

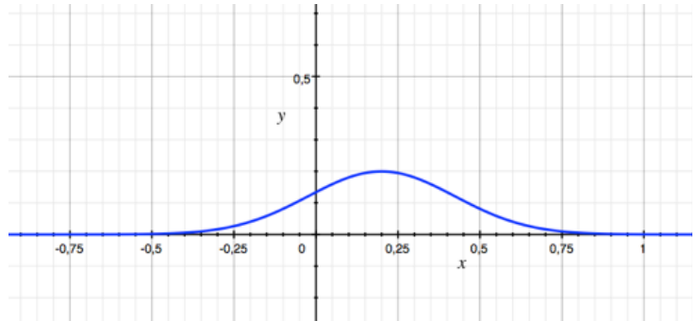
Adding noise

Prior Distribution



Posterior Distribution

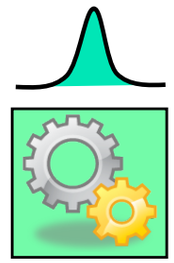
Probabilistic Program

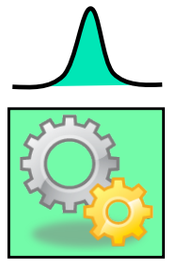


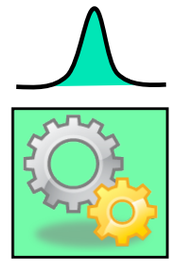
Probabilistic Inference

Adding noise

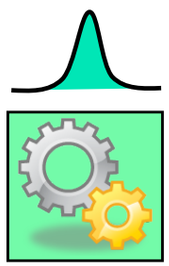
Prior Distribution



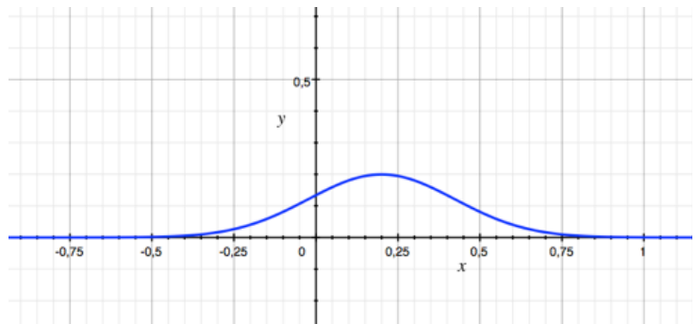


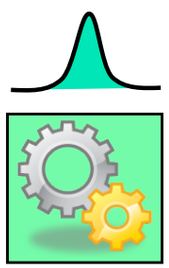


Probabilistic Program



Posterior Distribution

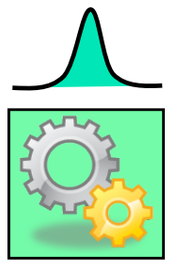




Probabilistic Inference

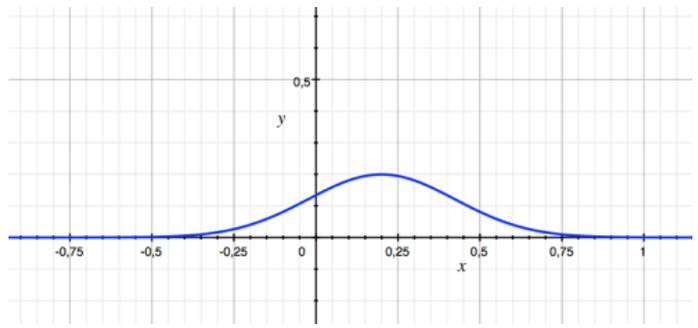
Adding noise on the data

Prior Distribution



Probabilistic Program

Posterior Distribution



Probabilistic Inference

An example

```
function privBerInput (l: B list) (p1: R) (p2: R): M[(0,1)]{
  let function vExp (l: B list) : M[B list]{
    match l with
      |nil -> mreturn nil
      |x::xs -> coercion
                (exp eps((0,0)->1,(0,1)->0,(1,1)->1,
                         (1,0)->0) x) :: (vExp l)
  } in
  mlet nl = (vExp l) in
  let prior = mreturn(beta(p1,p2)) in
  let function Ber (l: B list) (p:M[(0,1)]): M[(0,1)]{
    match l with
      |nil -> ran(p)
      |x::xs -> observe y => y = x in (Ber xs p)
  }
  in
  mreturn(infer (Ber nl prior))
}
```

An example

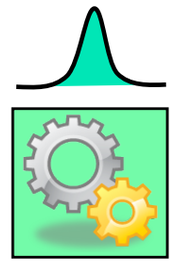
```
function privBerInput (l: B list) (p1: R) (p2: R): M[(0,1)]{
  let function vExp (l: B list) : M[B list]{
    match l with
      |nil -> mreturn nil
      |x::xs -> coercion
                (exp eps((0,0)->1,(0,1)->0,(1,1)->1,
                         (1,0)->0) x) :: (vExp l)
  } in
  mlet nl = (vExp l) in
  let prior = mreturn(beta(p1,p2)) in
  let function Ber (l: B list) (p:M[(0,1)]): M[(0,1)]{
    match l with
      |nil -> ran(p)
      |x::xs -> observe y => y = x in (Ber xs p)
  }
  in
  mreturn(infer (Ber nl prior))
}
```

Noise

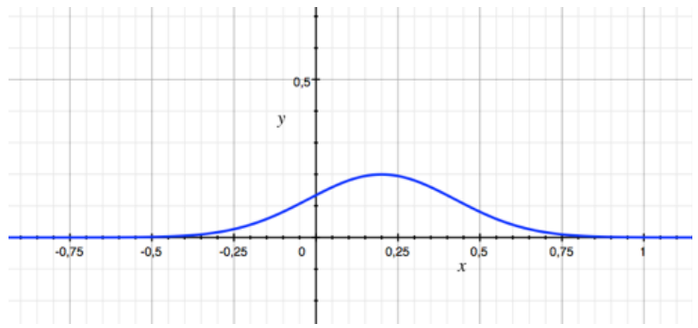
Adding noise on the output

Prior Distribution

Probabilistic Program



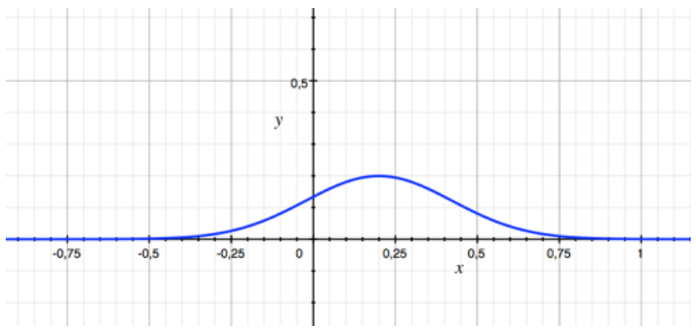
Posterior Distribution



Probabilistic Inference

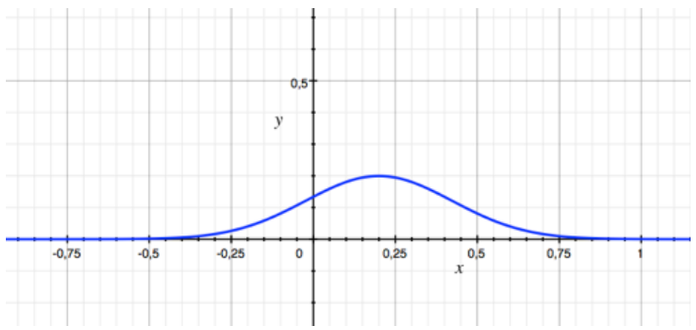
DP for Probabilistic Programs

Posterior Distribution



DP for Probabilistic Programs

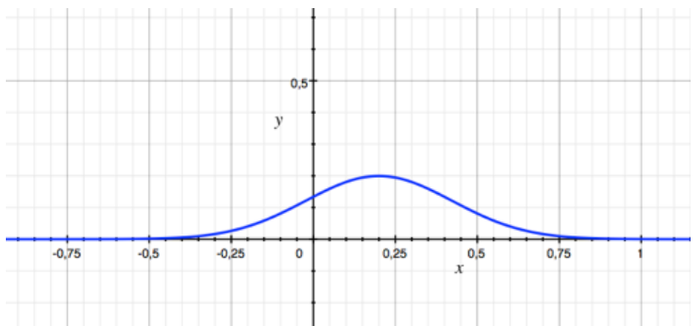
Posterior Distribution



Releasing the Parameters

DP for Probabilistic Programs

Posterior Distribution

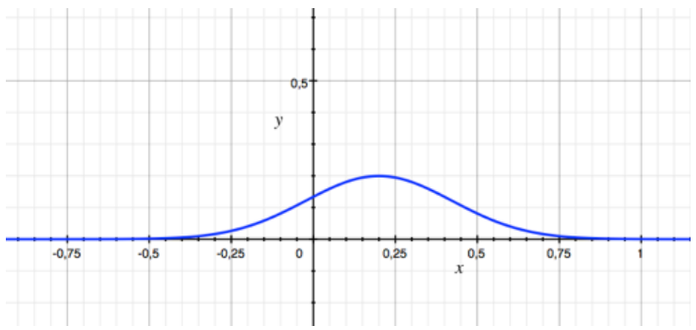


Releasing the Parameters

Sampling from the Distribution

DP for Probabilistic Programs

Posterior Distribution



Releasing the Parameters

Sampling from the Distribution

An example

A distance over distributions

```
function privBerInput (l: B list) (p1: R) (p2: R): M[(0,1)]{
  let function hellingerDistance (a0:R) (b0:R) (a1:R) (b1:R) : R {
        let gamma (r:R) = (r-1)! in
        let betaf (a:R) (b:R) = gamma(a)*gamma(b))/gamma(a+b) in
        let num=betaf ((a0+a1)/2.0) ((b0+b1)/2.0) in
        let denum=Math.Sqrt((betaf a0 b0)*(betaf a1 b1)) in
        Math.Sqrt(1.0-(num/denum))
  }
  in
  let function score (input:M[(0,1)]) (output:M[(0,1)]) : R {
    let beta(a0,b0) = input in
    let beta(a1,b1) = output in
    (-1.0) * (hellingerDistance a0 b0 a1 b1)
  } in
  let prior = mreturn(beta(p1,p2)) in
  let function Ber (l: B list) (p:M[(0,1)]): M[(0,1)]{
    match l with
      |nil -> ran(p)
      |x::xs -> observe y => y = x in (Ber xs p)
  }
  in
  exp eps score (infer (Ber l prior))
}
```

An example

A distance over distributions

```
function privBerInput (l: B list) (p1: R) (p2: R): M[(0,1)]{
  let function hellingerDistance (a0:R) (b0:R) (a1:R) (b1:R) : R {
        let gamma (r:R) = (r-1)! in
        let betaf (a:R) (b:R) = gamma(a)*gamma(b))/gamma(a+b) in
        let num=betaf ((a0+a1)/2.0) ((b0+b1)/2.0) in
        let denum=Math.Sqrt((betaf a0 b0)*(betaf a1 b1)) in
        Math.Sqrt(1.0-(num/denum))
  }
  in
  let function score (input:M[(0,1)]) (output:M[(0,1)]) : R {
    let beta(a0,b0) = input in
    let beta(a1,b1) = output in
    (-1.0) * (hellingerDistance a0 b0 a1 b1)
  } in
  let prior = mreturn(beta(p1,p2)) in
  let function Ber (l: B list) (p:M[(0,1)]): M[(0,1)]{
    match l with
      |nil -> ran(p)
      |x::xs -> observe y => y = x in (Ber xs p)
  }
  in
  exp eps score (infer (Ber l prior))
}
```

Noise

An example

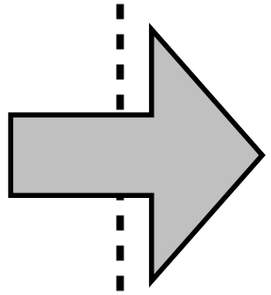
A distance over distributions

```
function privBerInput (l: B list) (p1: R) (p2: R): M[(0,1)]{
  let function hellingerDistance (a0:R) (b0:R) (a1:R) (b1:R) : R {
        let gamma (r:R) = (r-1)! in
        let betaf (a:R) (b:R) = gamma(a)*gamma(b))/gamma(a+b) in
        let num=betaf ((a0+a1)/2.0) ((b0+b1)/2.0) in
        let denum=Math.Sqrt((betaf a0 b0)*(betaf a1 b1)) in
        Math.Sqrt(1.0-(num/denum))
  }
  in
  let function score (input:M[(0,1)]) (output:M[(0,1)]) : R {
    let beta(a0,b0) = input in
    let beta(a1,b1) = output in
    (-1.0) * (hellingerDistance a0 b0 a1 b1)
  } in
  let prior = mreturn(beta(p1,p2)) in
  let function Ber (l: B list) (p:M[(0,1)]): M[(0,1)]{
    match l with
      |nil -> ran(p)
      |x::xs -> observe y => y = x in (Ber xs p)
  }
  in
  exp eps score (infer (Ber l prior))
}
```

Noise

More general Lifting of P

$P(x_1,x_2)$



$P^*(x_1,x_2)$

Relation

Relation over distributions

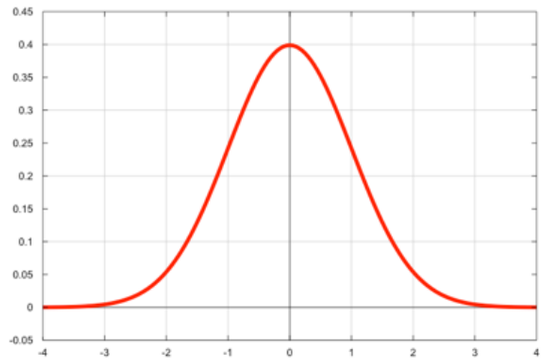
Accuracy

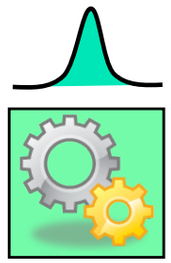
Different ways of adding noise can have different accuracy.

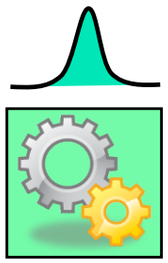
- we have theoretical accuracy (how to integrate it in a framework for reasoning about DP?)
- we have experimental accuracy (we need a framework for for test our programs)

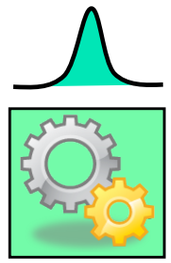
A Nice Playground!

Prior Distribution



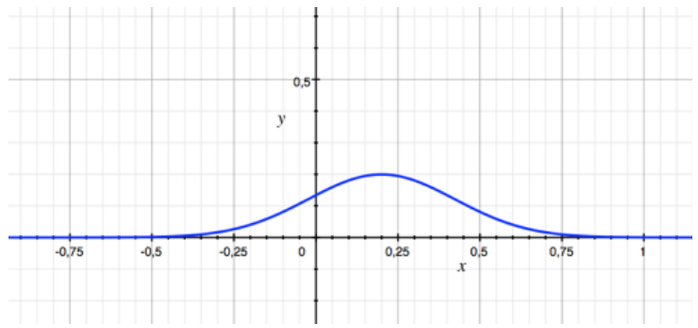


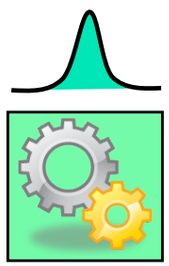


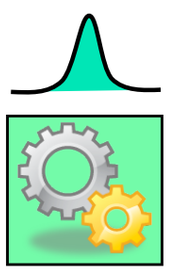


Probabilistic Program

Posterior Distribution







Probabilistic Inference

Tänan teid väga!