

# Differential Privacy using Quasi-Borel Spaces

Michikazu Hirata

February 6, 2026

## Abstract

This entry formalizes differential privacy using quasi-Borel spaces. In general, differential privacy is discussed using measurable spaces. Sato and Katsumata showed that quasi-Borel spaces are also applied to formulate differential privacy [1]. We formalize basic definitions and properties of differential privacy using quasi-Borel spaces, and show two examples: randomized response and the naive report noisy max algorithm.

## Contents

<b>1</b>	<b>Definitions</b>	<b>1</b>
1.1	Divergence for Differential Privacy using QBS . . . . .	2
1.2	Differential Privacy using QBS . . . . .	5
<b>2</b>	<b>Examples</b>	<b>7</b>
2.1	Randomized Response . . . . .	7
2.2	Laplace Distribution in QBS . . . . .	8
2.3	Naive Report Noisy Max Mechanism . . . . .	9

**theory** *DP-QBS*

**imports** *Differential-Privacy.Differential-Privacy-Divergence*  
*Differential-Privacy.Differential-Privacy-Standard*  
*S-Finite-Measure-Monad.Monad-QuasiBorel*

**begin**

**declare** *qbs-morphism-imp-measurable[measurable-dest]*

## 1 Definitions

Details of differential privacy using quasi-Borel spaces are found at [1]

## 1.1 Divergence for Differential Privacy using QBS

**definition** *DP-qbs-divergence* :: 'a qbs-measure  $\Rightarrow$  'a qbs-measure  $\Rightarrow$  real  $\Rightarrow$  ereal  
 (*DP'-divergence<sub>Q</sub>*) **where**  
*DP-qbs-divergence-qbs-l*: *DP-divergence<sub>Q</sub>* p q e  $\equiv$  *DP-divergence* (qbs-l p) (qbs-l q)  
 e

**abbreviation** *DP-qbs-inequality* (*DP'-inequality<sub>Q</sub>*) **where**  
*DP-qbs-inequality* p q  $\varepsilon$   $\delta$   $\equiv$  *DP-divergence<sub>Q</sub>* p q  $\varepsilon$   $\leq$  ereal  $\delta$

**lemmas** *DP-qbs-divergence-def* = *DP-qbs-divergence-qbs-l*[*simplified DP-divergence-SUP*]

**lemma** *DP-qbs-divergence-nonneg[simp]*:  $0 \leq$  *DP-divergence<sub>Q</sub>* p q e  
**by**(*auto simp: le-SUP-iff zero-ereal-def DP-qbs-divergence-def intro!*: *beXI*[**where**  
*x={}*])

**lemma** *DP-qbs-divergence-le-ereal-iff*:  
*DP-divergence<sub>Q</sub>* p q  $\varepsilon$   $\leq$  ereal  $\delta$   $\longleftrightarrow$  ( $\forall A \in$  sets (qbs-l p). *measure* (qbs-l p) A  $-$   
*exp*  $\varepsilon$  \* *measure* (qbs-l q) A  $\leq$   $\delta$ )  
**by** (*auto simp: DP-divergence-forall DP-qbs-divergence-qbs-l*)

**corollary** *DP-qbs-divergence-le-ereal-dest*:  
**assumes** *DP-divergence<sub>Q</sub>* p q  $\varepsilon$   $\leq$  ereal  $\delta$   
**shows** *measure* (qbs-l p) A  $\leq$  *exp*  $\varepsilon$  \* *measure* (qbs-l q) A +  $\delta$   
**using** *assms order.trans[OF DP-qbs-divergence-nonneg assms]*  
**by**(*cases A  $\in$  sets (qbs-l p)*) (*auto simp: DP-qbs-divergence-le-ereal-iff measure-notin-sets*)

**corollary** *DP-qbs-divergence-le-erealI*:  
**assumes**  $\bigwedge A. A \in$  sets (qbs-l p)  $\implies$  *measure* (qbs-l p) A  $\leq$  *exp*  $\varepsilon$  \* *measure*  
 (qbs-l q) A +  $\delta$   
**shows** *DP-divergence<sub>Q</sub>* p q  $\varepsilon$   $\leq$  ereal  $\delta$   
**using** *assms by(fastforce simp: DP-qbs-divergence-le-ereal-iff)*

**lemma** *DP-qbs-divergence-zero*:  
**assumes** p  $\in$  monadP-qbs X  
**and** q  $\in$  monadP-qbs X  
**and** *DP-inequality<sub>Q</sub>* p q 0 0  
**shows** p = q  
**by**(*auto intro!: inj-onD[OF qbs-l-inj-P] DP-divergence-zero[where L=qbs-to-measure*  
 X]  
*assms[simplified DP-qbs-divergence-qbs-l] measurable-space[OF*  
*qbs-l-measurable-prob]*  
*simp: space-L*)

**lemma** *DP-qbs-divergence-antimono*: a  $\leq$  b  $\implies$  *DP-divergence<sub>Q</sub>* p q b  $\leq$  *DP-divergence<sub>Q</sub>*  
 p q a  
**by**(*auto simp: DP-qbs-divergence-def intro!: SUP-mono' mult-right-mono*)

**lemma** *DP-qbs-divergence-refl[simp]*:  $DP\text{-divergence}_Q p p 0 = 0$   
**unfolding** *DP-qbs-divergence-qbs-l* **by**(rule *DP-divergence-reflexivity*)

**lemma** *DP-qbs-divergence-refl'[simp]*:  $0 \leq e \implies DP\text{-divergence}_Q p p e = 0$   
**by**(intro *antisym DP-qbs-divergence-nonneg*) (auto simp: *DP-qbs-divergence-def SUP-le-iff mult-le-cancel-right1*)

**lemma** *DP-qbs-divergence-trans'*:  
**assumes** *DP-inequality\_Q*  $p q \varepsilon \delta$   
**and** *DP-inequality\_Q*  $q l \varepsilon' 0$   
**shows** *DP-inequality\_Q*  $p l (\varepsilon + \varepsilon') \delta$   
**unfolding** *DP-qbs-divergence-le-ereal-iff diff-le-eq*

**proof** *safe*

**fix**  $A$

**assume** [*measurable*]:  $A \in \text{sets } (qbs\text{-l } p)$

**show**  $\text{measure } (qbs\text{-l } p) A \leq \delta + \exp (\varepsilon + \varepsilon') * \text{measure } (qbs\text{-l } l) A$

**proof** –

**have**  $\text{measure } (qbs\text{-l } p) A \leq \delta + \exp \varepsilon * \text{measure } (qbs\text{-l } q) A$

**using** *assms(1)* **by**(auto simp: *DP-qbs-divergence-le-ereal-iff diff-le-eq*)

**also have**  $\dots \leq \delta + \exp \varepsilon * (\exp \varepsilon' * \text{measure } (qbs\text{-l } l) A)$

**using** *DP-qbs-divergence-le-ereal-dest assms(2)* **by** *fastforce*

**finally show** *?thesis*

**by** (*simp add: exp-add*)

**qed**

**qed**

**lemmas** *DP-qbs-divergence-trans = DP-qbs-divergence-trans'[where  $\delta=0$ ]*

**proposition** *DP-qbs-divergence-compose*:

**assumes** [*qbs,measurable*]:  $p \in \text{monadP-qbs } X q \in \text{monadP-qbs } X f \in X \rightarrow_Q$   
 $\text{monadP-qbs } Y g \in X \rightarrow_Q \text{monadP-qbs } Y$

**and** *dp1*:  $DP\text{-divergence}_Q p q \varepsilon \leq \text{ereal } \delta$

**and** *dp2*:  $\bigwedge x. x \in \text{qbs-space } X \implies DP\text{-divergence}_Q (f x) (g x) \varepsilon' \leq \text{ereal } \delta'$

**and** [*arith*]:  $0 \leq \varepsilon 0 \leq \varepsilon'$

**shows**  $DP\text{-divergence}_Q (p \ggg f) (q \ggg g) (\varepsilon + \varepsilon') \leq \text{ereal } (\delta + \delta')$

**proof** –

**interpret** *comparable-probability-measures qbs-to-measure*  $X$  *qbs-l*  $p$  *qbs-l*  $q$

**by**(auto simp: *comparable-probability-measures-def space-L intro!*: *qbs-l-measurable-prob[THEN measurable-space]*)

**note** [*measurable,simp*] = *qbs-l-measurable-prob*

**show** *?thesis*

**by**(auto simp: *qbs-l-bind-qbsP[of - X - Y] space-L M N DP-qbs-divergence-qbs-l*

*intro!*: *DP-divergence-composability[where  $K=qbs\text{-to-measure } Y$  and  $L=qbs\text{-to-measure$*

$X$ ]

*dp1 [simplified DP-qbs-divergence-qbs-l] dp2 [simplified DP-qbs-divergence-qbs-l]*)

**qed**

**corollary** *DP-qbs-divergence-dataprocessing:*

**assumes**  $[qbs]: p \in \text{monadP-qbs } X \ q \in \text{monadP-qbs } X \ f \in X \rightarrow_Q \text{monadP-qbs } Y$   
**and**  $dp: \text{DP-divergence}_Q \ p \ q \ \varepsilon \leq \text{ereal } \delta$   
**and**  $[arith]: 0 \leq \varepsilon$   
**shows**  $\text{DP-divergence}_Q \ (p \ggg f) \ (q \ggg f) \ \varepsilon \leq \text{ereal } \delta$

**proof** –

**interpret** *comparable-probability-measures qbs-to-measure*  $X \ \text{qbs-l } p \ \text{qbs-l } q$   
**by**(*auto simp: comparable-probability-measures-def space-L intro!: qbs-l-measurable-prob[THEN measurable-space]*)  
**note**  $[\text{measurable}] = \text{qbs-l-measurable-prob qbs-morphism-imp-measurable[OF assms(3)]}$   
**show** *?thesis*  
**by**(*auto simp: qbs-l-bind-qbsP[of - X - Y] space-L M N DP-qbs-divergence-qbs-l intro!: DP-divergence-postprocessing[where L= qbs-to-measure X and K=qbs-to-measure Y]*  
 $dp[\text{simplified DP-qbs-divergence-qbs-l}]$ )

**qed**

**lemma** *DP-qbs-divergence-additive:*

**assumes**  $[qbs]: p \in \text{monadP-qbs } X \ q \in \text{monadP-qbs } X \ p' \in \text{monadP-qbs } Y \ q' \in \text{monadP-qbs } Y$   
**and**  $div1: \text{DP-divergence}_Q \ p \ q \ \varepsilon \leq \text{ereal } \delta$   
**and**  $div2: \text{DP-divergence}_Q \ p' \ q' \ \varepsilon' \leq \text{ereal } \delta'$   
**and**  $[arith]: 0 \leq \varepsilon \ 0 \leq \varepsilon'$   
**shows**  $\text{DP-divergence}_Q \ (p \otimes_{Qmes} p') \ (q \otimes_{Qmes} q') \ (\varepsilon + \varepsilon') \leq \text{ereal } (\delta + \delta')$

**proof** –

**note**  $[qbs] = \text{return-qbs-morphismP bind-qbs-morphismP qbs-space-monadPM}$   
**have**  $\text{DP-divergence}_Q \ (p \otimes_{Qmes} p') \ (q \otimes_{Qmes} q') \ (\varepsilon + \varepsilon')$   
 $= \text{DP-divergence}_Q \ (p \ggg (\lambda x. p' \ggg (\lambda y. \text{return-qbs } (X \otimes_Q Y) \ (x, y))))$   
 $(q \ggg (\lambda x. q' \ggg (\lambda y. \text{return-qbs } (X \otimes_Q Y) \ (x, y)))) \ (\varepsilon + \varepsilon')$   
**by**(*simp add: qbs-pair-measure-def[of - X - Y]*)  
**also have**  $\dots \leq \text{ereal } (\delta + \delta')$   
**by**(*auto intro!: DP-qbs-divergence-compose[of - X - X  $\otimes_Q$  Y] div1 div2 DP-qbs-divergence-dataprocessing[of - Y - X  $\otimes_Q$  Y]*)

**finally show** *?thesis* .

**qed**

**corollary** *DP-qbs-divergence-strength:*

**assumes**  $[qbs]: p \in \text{monadP-qbs } X \ q \in \text{monadP-qbs } X \ x \in \text{qbs-space } Y$   
**and**  $dp: \text{DP-divergence}_Q \ p \ q \ \varepsilon \leq \text{ereal } \delta$   
**and**  $[simp]: 0 \leq \varepsilon$   
**shows**  $\text{DP-divergence}_Q \ (\text{return-qbs } Y \ x \ \otimes_{Qmes} p) \ (\text{return-qbs } Y \ x \ \otimes_{Qmes} q)$   
 $\varepsilon \leq \text{ereal } \delta$

**proof** –

**note**  $[qbs] = \text{return-qbs-morphismP}$   
**show** *?thesis*

by(auto intro!: DP-qbs-divergence-additive[of - Y - - X - 0 0,simplified] dp)  
qed

## 1.2 Differential Privacy using QBS

**definition** *DP-qbs (differential'-privacy)<sub>Q</sub>* **where**

*DP-qbs-qbs-L:differential-privacy<sub>Q</sub>*  $M \equiv \text{differential-privacy } (\lambda x. \text{qbs-l } (M x))$

**lemma** *DP-qbs-def:*

*differential-privacy<sub>Q</sub>*  $M \text{ adj } \varepsilon \delta \longleftrightarrow$   
 $(\forall (d1, d2) \in \text{adj}. \text{DP-inequality}_Q (M d1) (M d2) \varepsilon \delta \wedge \text{DP-inequality}_Q (M d2)$   
 $(M d1) \varepsilon \delta)$   
 by(*simp add: DP-inequality-cong-DP-divergence differential-privacy-def DP-qbs-qbs-L*  
*DP-qbs-divergence-qbs-l*)

**lemma** *DP-qbs-adj-sym:*

**assumes** *sym adj*  
**shows** *differential-privacy<sub>Q</sub>*  $M \text{ adj } \varepsilon \delta \longleftrightarrow (\forall (d1, d2) \in \text{adj}. \text{DP-inequality}_Q$   
 $(M d1) (M d2) \varepsilon \delta)$   
 by(auto *simp: differential-privacy-adj-sym[OF assms] DP-inequality-cong-DP-divergence*  
*DP-qbs-qbs-L DP-qbs-divergence-qbs-l*)

**lemma** *pure-DP-qbs-comp:*

**assumes**  $\text{adj} \subseteq \text{qbs-space } X \times \text{qbs-space } X$   
**and**  $\text{adj}' \subseteq \text{qbs-space } X \times \text{qbs-space } X$   
**and** *differential-privacy<sub>Q</sub>*  $M \text{ adj } \varepsilon 0$   
**and** *differential-privacy<sub>Q</sub>*  $M \text{ adj}' \varepsilon' 0$   
**and**  $M \in X \rightarrow_Q \text{monadP-qbs } Y$   
**shows** *differential-privacy<sub>Q</sub>*  $M (\text{adj } O \text{adj}') (\varepsilon + \varepsilon') 0$   
**using** *assms*  
 by(auto intro!: *pure-differential-privacy-comp[where X=qbs-to-measure X and*  
*R=qbs-to-measure Y]*  
*simp: space-L DP-qbs-qbs-L*)

**lemma** *pure-DP-qbs-trans-k:*

**assumes**  $\text{adj} \subseteq \text{qbs-space } X \times \text{qbs-space } X$   
**and** *differential-privacy<sub>Q</sub>*  $M \text{ adj } \varepsilon 0$   
**and**  $M \in X \rightarrow_Q \text{monadP-qbs } Y$   
**shows** *differential-privacy<sub>Q</sub>*  $M (\text{adj } \overset{\sim}{\sim} k) (k * \varepsilon) 0$   
**using** *assms*  
 by(auto intro!: *pure-differential-privacy-trans-k[where X=qbs-to-measure X and*  
*R=qbs-to-measure Y]*  
*simp: space-L DP-qbs-qbs-L*)

**proposition** *DP-qbs-postprocessing:*

**assumes**  $\varepsilon \geq 0$   
**and** *differential-privacy<sub>Q</sub>*  $M \text{ adj } \varepsilon \delta$   
**and** [*qbs,measurable*]:  $M \in X \rightarrow_Q \text{monadP-qbs } Y$   
**and** [*qbs,measurable*]:  $N \in Y \rightarrow_Q \text{monadP-qbs } Z$

**and**  $adj \subseteq \text{qbs-space } X \times \text{qbs-space } X$   
**shows**  $\text{differential-privacy}_Q (\lambda x. M x \gg N) adj \varepsilon \delta$   
**using**  $\text{assms by}(\text{auto simp: DP-qbs-def intro!: DP-qbs-divergence-dataprocessing[of } - Y - - Z])$

**corollary** *DP-qbs-postprocessing-return:*

**assumes**  $\varepsilon \geq 0$   
**and**  $\text{differential-privacy}_Q M adj \varepsilon \delta$   
**and**  $M \in X \rightarrow_Q \text{monadP-qbs } Y$   
**and**  $N \in Y \rightarrow_Q Z$   
**and**  $adj \subseteq \text{qbs-space } X \times \text{qbs-space } X$   
**shows**  $\text{differential-privacy}_Q (\lambda x. M x \gg (\lambda y. \text{return-qbs } Z (N y))) adj \varepsilon \delta$   
**by**( $\text{intro DP-qbs-postprocessing[where } X=X \text{ and } Y=Y \text{ and } Z=Z]$ )  
*(use return-qbs-morphismP[of Z] assms in auto)*

**lemma** *DP-qbs-preprocessing:*

**assumes**  $\varepsilon \geq 0$   
**and**  $\text{differential-privacy}_Q M adj \varepsilon \delta$   
**and**  $[\text{measurable}]: f \in X' \rightarrow_Q X$   
**and**  $\forall (x,y) \in adj'. ((f x), (f y)) \in adj$   
**and**  $adj \subseteq \text{qbs-space } X \times \text{qbs-space } X$   
**and**  $adj' \subseteq \text{qbs-space } X' \times \text{qbs-space } X'$   
**shows**  $\text{differential-privacy}_Q (M \circ f) adj' \varepsilon \delta$   
**using**  $\text{assms by}(\text{auto simp: DP-qbs-def})$

**proposition** *DP-qbs-bind-adaptive:*

**assumes**  $\varepsilon \geq 0$  **and**  $\varepsilon' \geq 0$   
**and**  $[\text{qbs}]: M \in X \rightarrow_Q \text{monadP-qbs } Y$   
**and**  $\text{differential-privacy}_Q M adj \varepsilon \delta$   
**and**  $[\text{qbs}]: N \in X \Rightarrow_Q Y \Rightarrow_Q \text{monadP-qbs } Z$   
**and**  $\bigwedge y. y \in \text{qbs-space } Y \implies \text{differential-privacy}_Q (\lambda x. N x y) adj \varepsilon' \delta'$   
**and**  $adj \subseteq \text{qbs-space } X \times \text{qbs-space } X$   
**shows**  $\text{differential-privacy}_Q (\lambda x. M x \gg N x) adj (\varepsilon + \varepsilon') (\delta + \delta')$   
**using**  $\text{assms by}(\text{fastforce simp add: DP-qbs-def intro!: DP-qbs-divergence-compose[of } - Y - - Z])$

**proposition** *DP-qbs-bind-pair:*

**assumes**  $\varepsilon \geq 0$   $\varepsilon' \geq 0$   
**and**  $[\text{qbs}]: M \in X \rightarrow_Q \text{monadP-qbs } Y$   
**and**  $\text{differential-privacy}_Q M adj \varepsilon \delta$   
**and**  $[\text{qbs}]: N \in X \rightarrow_Q \text{monadP-qbs } Z$   
**and**  $\text{differential-privacy}_Q N adj \varepsilon' \delta'$   
**and**  $adj \subseteq \text{qbs-space } X \times \text{qbs-space } X$   
**shows**  $\text{differential-privacy}_Q (\lambda x. M x \gg (\lambda y. N x \gg (\lambda z. \text{return-qbs } (Y \otimes_Q Z) (y,z)))) adj (\varepsilon + \varepsilon') (\delta + \delta')$   
**proof** –  
**note**  $[\text{qbs}] = \text{return-qbs-morphismP bind-qbs-morphismP}$   
**show** *?thesis*  
**using**  $\text{assms by}(\text{auto intro!: DP-qbs-bind-adaptive[where } X=X \text{ and } Y=Y$

```

and  $Z=Y \otimes_Q Z]$ 
      DP-qbs-postprocessing[where  $X=X$  and  $Y=Z$  and  $Z=Y$ 
 $\otimes_Q Z]$ )
qed

end

```

## 2 Examples

```

theory DP-QBS-Examples
  imports DP-QBS
      Differential-Privacy.Differential-Privacy-Randomized-Response
begin

```

```

lemma qbs-space-list-qbs-borel[qbs]:  $\bigwedge r. r \in \text{qbs-space } (\text{list-qbs borel}_Q)$ 
  and qbs-space-list-qbs-count-space[qbs]:  $\bigwedge i. r \in \text{qbs-space } (\text{list-qbs } (\text{count-space}_Q$ 
  (UNIV :: - :: countable)))
  by(auto simp: list-qbs-space)

```

### 2.1 Randomized Response

```

lemma qbs-morphism-RR-mechanism[qbs]: qbs-pmf  $\circ$  RR-mechanism  $e \in \text{count-space}_Q$ 
  UNIV  $\rightarrow_Q$  monadP-qbs (count-space}_Q UNIV)
  by(auto intro!: qbs-morphism-count-space' qbs-pmf-qbsP)

```

```

lemma qbs-DP-RR-mechanism:

```

```

  assumes [arith]:  $\varepsilon \geq 0$ 
  shows DP-divergence}_Q (RR-mechanism  $\varepsilon$   $x$ ) (RR-mechanism  $\varepsilon$   $y$ )  $\varepsilon = 0$ 
proof(intro antisym DP-qbs-divergence-nonneg)
  have [arith]:  $1 + \exp \varepsilon > 0$ 
    by(auto simp: add-pos-nonneg intro!:divide-le-eq-1-pos[THEN iffD2])
  have [arith]:  $1 / (1 + \exp \varepsilon) \leq \exp \varepsilon * \exp \varepsilon / (1 + \exp \varepsilon)$ 
    by(rule order.trans[where b=exp \varepsilon * (1 / (1 + exp \varepsilon))])
    (auto simp: divide-right-mono mult-le-cancel-right1)
  show DP-divergence}_Q (RR-mechanism  $\varepsilon$   $x$ ) (RR-mechanism  $\varepsilon$   $y$ )  $\varepsilon \leq 0$ 
    unfolding zero-ereal-def
proof(rule DP-qbs-divergence-le-erealI)
  fix  $A :: \text{bool set}$ 
  have ineq1: measure-pmf.prob (bernoulli-pmf ( $\exp \varepsilon / (1 + \exp \varepsilon)$ ))  $A$ 
     $\leq \exp \varepsilon * \text{measure-pmf.prob}$  (bernoulli-pmf ( $\exp \varepsilon / (1 + \exp \varepsilon)$ ))  $A$ 
  and ineq2: measure-pmf.prob (bernoulli-pmf ( $1 / (1 + \exp \varepsilon)$ ))  $A$ 
     $\leq \exp \varepsilon * \text{measure-pmf.prob}$  (bernoulli-pmf ( $1 / (1 + \exp \varepsilon)$ ))  $A$ 
  by(auto simp: mult-le-cancel-right1)
  have ineq3: measure-pmf.prob (bernoulli-pmf ( $\exp \varepsilon / (1 + \exp \varepsilon)$ ))  $A$ 
     $\leq \exp \varepsilon * \text{measure-pmf.prob}$  (bernoulli-pmf ( $1 / (1 + \exp \varepsilon)$ ))  $A$ 
proof –
  consider  $A = \{\}$  |  $A = \{\text{True}\}$  |  $A = \{\text{False}\}$  |  $A = \text{UNIV}$ 
  by (metis (full-types) UNIV-eq-I is-singletonI' is-singleton-the-lem)

```

```

    then show ?thesis
  by cases (simp-all add: measure-pmf-single RR-probability-flip1 RR-probability-flip2)
qed
have ineq4: measure-pmf.prob (bernoulli-pmf (1 / (1 + exp ε))) A
  ≤ exp ε * measure-pmf.prob (bernoulli-pmf (exp ε / (1 + exp ε))) A
proof -
  consider A = {} | A = {True} | A = {False} | A = UNIV
  by (metis (full-types) UNIV-eq-I is-singletonI' is-singleton-the-lem)
  then show ?thesis
  by cases (simp-all add: measure-pmf-single RR-probability-flip1 RR-probability-flip2)
qed
show measure (qbs-l (qbs-pmf (RR-mechanism ε x))) A ≤ exp ε * measure
(qbs-l (qbs-pmf (RR-mechanism ε y))) A + 0
  using ineq1 ineq2 ineq3 ineq4 by(auto simp: RR-mechanism-def)
qed
qed

```

## 2.2 Laplace Distribution in QBS

**lemma** *qbs-morphism-laplace-density*[qbs]:  $\text{laplace-density} \in \text{borel}_Q \Rightarrow_Q \text{borel}_Q \Rightarrow_Q \text{borel}_Q \Rightarrow_Q \text{borel}_Q$

```

proof -
  have [simp]: laplace-density = (λ l m x. if l > 0 then exp(-|x - m| / l) / (2 * l)
  else 0)
  by standard+ (simp add: laplace-density-def)
  show ?thesis
  by simp
qed

```

**definition** *qbs-Lap-mechanism* ( $\text{Lap}'\text{-mechanism}_Q$ ) **where**

$\text{Lap-mechanism}_Q \equiv \lambda e x. \text{if } e \leq 0 \text{ then return-qbs borel}_Q x \text{ else density-qbs l borel}_Q (\text{laplace-density } e x)$

**lemma** *qbs-morphism-Lap-mechanism*[qbs]:  $\text{Lap-mechanism}_Q \in \text{borel}_Q \rightarrow_Q \text{borel}_Q \Rightarrow_Q \text{monadP-qbs borel}_Q$

```

by(intro curry-preserves-morphisms qbs-morphism-monadPI)
(auto intro!: prob-space-return
  simp: qbs-Lap-mechanism-def qbs-l-return-qbs space-L qbs-l-density-qbs[of -
  borel] prob-space-laplacian-density)

```

**lemma** *qbs-l-Lap-mechanism*:  $\text{qbs-l} (\text{Lap-mechanism}_Q e r) = \text{Lap-dist } e r$

```

by(auto simp: qbs-Lap-mechanism-def qbs-l-return-qbs space-L qbs-l-density-qbs[of -
  borel] Lap-dist-def cong: return-cong)

```

**lemma** *qbs-Lap-mechanism-qbs-l-inverse*:  $\text{Lap-mechanism}_Q e x = \text{qbs-l-inverse} (\text{Lap-dist } e x)$

```

by(auto intro!: inj-onD[OF qbs-l-inj-P[of borel]] standard-borel-ne.qbs-l-qbs-l-inverse[OF
  - sets-Lap-dist,symmetric]
  standard-borel-ne.qbs-l-inverse-in-space-monadP[OF - sets-Lap-dist] simp: qbs-l-Lap-mechanism)

```

**proposition** *qbs-DP-Lap-mechanism*:  
**assumes**  $\varepsilon > 0$  **and**  $|x - y| \leq r$   
**shows**  $DP\text{-divergence}_Q (Lap\text{-mechanism}_Q (1 / \varepsilon) x) (Lap\text{-mechanism}_Q (1 / \varepsilon) y) (r * \varepsilon) = 0$   
**using**  $DP\text{-divergence-Lap-dist}'$  [**where**  $b=1 / \varepsilon$ ]  
**by** (*intro antisym DP-qbs-divergence-nonneg*)  
*(auto simp: DP-qbs-qbs-L DP-qbs-divergence-qbs-l qbs-l-Lap-mechanism zero-ereal-def intro!: assms)*

## 2.3 Naive Report Noisy Max Mechanism

**primrec** *qbs-NaiveRNM* :: *real*  $\Rightarrow$  *real list*  $\Rightarrow$  *real qbs-measure* **where**  
*qbs-NaiveRNM*  $\varepsilon [] = \text{return-qbs borel } 0$  |  
*qbs-NaiveRNM*  $\varepsilon (x \# xs) =$   
*(case xs of*  
*Nil*  $\Rightarrow Lap\text{-mechanism}_Q (1 / \varepsilon) x$  |  
*y#ys*  $\Rightarrow \text{do } \{x1 \leftarrow Lap\text{-mechanism}_Q (1 / \varepsilon) x; x2 \leftarrow \text{qbs-NaiveRNM } \varepsilon xs;$   
*return-qbs borel (max x1 x2)\})*

**lemma** *qbs-morphism-NaiveRNM*[*qbs*]: *qbs-NaiveRNM*  $\in \text{borel}_Q \Rightarrow_Q \text{list-qbs borel}$   
 $\Rightarrow_Q \text{monadP-qbs borel}_Q$

**proof** –  
**note** [*qbs*] = *return-qbs-morphismP bind-qbs-morphismP*  
**show** *?thesis*  
**by** (*simp add: qbs-NaiveRNM-def*)  
**qed**

**theorem** *qbs-DP-NaiveRNM'*:  
**assumes** *pos[arith,simp]:*  $\varepsilon > 0$   
**and** *length xs = n and length ys = n*  
**and** *adj: (* $\sum i < n. |nth\ xs\ i - nth\ ys\ i|$ *)*  $\leq r$   
**shows**  $DP\text{-divergence}_Q (qbs\text{-NaiveRNM } \varepsilon\ xs) (qbs\text{-NaiveRNM } \varepsilon\ ys) (r * \varepsilon) = 0$   
**using** *assms(2,3,4)*  
**proof** (*induct ys arbitrary: xs n r*)  
**case Nil**  
**then show** *?case*  
**by** *simp*  
**next**  
**case** *ih:(Cons y ys')*  
**show** *?case (is ?lhs = -)*  
**proof** (*cases ys'*)  
**case Nil**  
**then have**  $\exists a. xs = [a]$   
**using** *ih*  
**by** (*metis length-Suc-conv length-greater-0-conv*)  
**then show** *?thesis*  
**using** *ih by(auto intro!: qbs-DP-Lap-mechanism)*  
**next**

```

case  $h:(Cons\ y'\ ys')$ 
note  $[qbs] = bind-qbs-morphismP\ return-qbs-morphismP$ 
obtain  $x\ x'\ xs''$  where  $xs:xs = x \# x' \# xs''$ 
  by  $(metis\ h\ ih(2)\ ih(3)\ length-Suc-conv)$ 
define  $xs'$  where  $xs' = x' \# xs''$ 
obtain  $n'$  where  $n:n = Suc\ n'$ 
  using  $ih(3)$  by force
have  $xs-x's':xs = x \# xs'$ 
  by $(auto\ simp: xs'-def\ h\ xs)$ 
have  $[simp]:length\ xs' = n'\ length\ ys' = n'$ 
  using  $ih(2)\ ih(3)$  by $(auto\ simp: xs-x's'\ n)$ 
define  $r1$  where  $r1 = |x - y|$ 
define  $r2$  where  $r2 = (\sum j < n'. |xs' ! j - ys' ! j|)$ 
have  $(\sum i < n. |xs ! i - (y \# ys') ! i|) = |x - y| + (\sum i < n'. |xs' ! i - ys' ! i|)$ 
proof -
  have  $(\sum i < n. |xs ! i - (y \# ys') ! i|) = (\sum i \in \{0\} \cup \{Suc\ 0..<n\}. |xs ! i -$ 
 $(y \# ys') ! i|)$ 
  using  $atLeast1-lessThan-eq-remove0\ lessThan-Suc-eq-insert-0\ n$  by fastforce
  also have  $\dots = (\sum i \in \{0\}. |xs ! i - (y \# ys') ! i|) + (\sum i \in \{Suc\ 0..<n\}. |xs$ 
 $! i - (y \# ys') ! i|)$ 
  by $(subst\ sum-Un)\ auto$ 
  also have  $\dots = |x - y| + (\sum i < n'. |xs ! Suc\ i - (y \# ys') ! Suc\ i|)$ 
  unfolding  $n$  by $(subst\ sum.atLeast-Suc-lessThan-Suc-shift)\ (simp\ add: xs-x's'$ 
 $n\ lessThan-atLeast0)$ 
  finally show  $?thesis$  by $(simp\ add: xs-x's')$ 
qed
hence  $r12[arith,simp]: r1 + r2 \leq r\ 0 \leq r1\ 0 \leq r2\ |x - y| \leq r1\ (\sum j < n'. |xs'$ 
 $! j - ys' ! j|) \leq r2$ 
  using  $ih(4)$  by $(auto\ simp: r1-def\ r2-def)$ 

have  $?lhs =$ 
   $DP-divergence_Q$ 
   $(Lap-mechanism_Q\ (1 / \epsilon)\ x \gg (\lambda x1. qbs-NaiveRNM\ \epsilon\ xs' \gg (\lambda x2.$ 
 $return-qbs\ borel_Q\ (max\ x1\ x2))))$ 
   $(Lap-mechanism_Q\ (1 / \epsilon)\ y \gg (\lambda x1. qbs-NaiveRNM\ \epsilon\ ys' \gg (\lambda x2.$ 
 $return-qbs\ borel_Q\ (max\ x1\ x2))))$ 
   $(r * \epsilon)$ 
  by $(auto\ simp: h\ xs\ xs'-def)$ 
also have  $\dots \leq$ 
   $DP-divergence_Q$ 
   $(Lap-mechanism_Q\ (1 / \epsilon)\ x \gg (\lambda x1. qbs-NaiveRNM\ \epsilon\ xs' \gg (\lambda x2.$ 
 $return-qbs\ borel_Q\ (max\ x1\ x2))))$ 
   $(Lap-mechanism_Q\ (1 / \epsilon)\ y \gg (\lambda x1. qbs-NaiveRNM\ \epsilon\ ys' \gg (\lambda x2.$ 
 $return-qbs\ borel_Q\ (max\ x1\ x2))))$ 
   $(r1 * \epsilon + r2 * \epsilon)$ 
  by $(auto\ intro!: DP-qbs-divergence-antimono\ simp: distrib-right[symmetric])$ 
also have  $\dots \leq ereal\ (0 + 0)$ 
  by $(intro\ DP-qbs-divergence-compose[of - qbs-borel - - qbs-borel]$ 
 $DP-qbs-divergence-dataprocessing[of - qbs-borel - - qbs-borel])$ 

```

```

      (auto simp: qbs-DP-Lap-mechanism ih(1))
    finally show ?thesis
      using antisym zero-ereal-def by fastforce
  qed
qed

```

**definition** *adj-naive-RNM* :: real  $\Rightarrow$  (real list  $\times$  real list) set **where**  
*adj-naive-RNM*  $r \equiv \{(xs, ys). \text{length } xs = \text{length } ys \wedge (\sum_{i < \text{length } xs}. |\text{nth } xs \ i - \text{nth } ys \ i|) \leq r\}$

**theorem** *qbs-DP-NaiveRNM*:

```

  assumes pos:  $\varepsilon > 0$ 
  shows differential-privacyQ (qbs-NaiveRNM  $\varepsilon$ ) (adj-naive-RNM  $r$ ) ( $r * \varepsilon$ ) 0
  proof (safe intro!: DP-qbs-adj-sym[THEN iffD2])
    fix  $xs \ ys$ 
    assume *:  $(xs, ys) \in \text{adj-naive-RNM } r$ 
    let ?n = length  $xs$ 
    have length  $xs = ?n$  and length  $ys = ?n$  and  $(\sum_{i < ?n}. |\text{nth } xs \ i - \text{nth } ys \ i|) \leq$ 
 $r$ 
      using * by (auto simp: adj-naive-RNM-def)
    from qbs-DP-NaiveRNM'[OF pos this]
    show DP-inequalityQ (qbs-NaiveRNM  $\varepsilon$   $xs$ ) (qbs-NaiveRNM  $\varepsilon$   $ys$ ) ( $r * \varepsilon$ ) 0
      by simp
  qed (auto intro!: symI simp: adj-naive-RNM-def abs-minus-commute)
end

```

## References

- [1] T. Sato and S. Katsumata. Divergences on monads for relational program logics. *Mathematical Structures in Computer Science*, 33(45):427–485, 2023.