Differential Privacy using Quasi-Borel Spaces

Michikazu Hirata

March 17, 2025

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

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* ⇒ *'a qbs-measure* ⇒ *real* ⇒ *ereal*
(*DP'-divergence*$_Q$) **where**
DP-qbs-divergence-qbs-l: *DP-divergence*$_Q$ *p q e* ≡ *DP-divergence* (*qbs-l p*) (*qbs-l q*)
e

abbreviation *DP-qbs-inequality* (*DP'-inequality*$_Q$) **where**
DP-qbs-inequality p q ε δ ≡ *DP-divergence*$_Q$ *p q* ε ≤ *ereal* δ

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

lemma *DP-qbs-divergence-nonneg*[*simp*]: *0* ≤ *DP-divergence*$_Q$ *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$_Q$ *p q* ε ≤ *ereal* δ ⟷ (∀ *A* ∈ *sets* (*qbs-l p*). *measure* (*qbs-l p*) *A* −
exp ε ∗ *measure* (*qbs-l q*) *A* ≤ δ)
  by (*auto simp*: *DP-divergence-forall DP-qbs-divergence-qbs-l*)

corollary *DP-qbs-divergence-le-ereal-dest*:
  assumes *DP-divergence*$_Q$ *p q* ε ≤ *ereal* δ
  shows *measure* (*qbs-l p*) *A* ≤ *exp* ε ∗ *measure* (*qbs-l q*) *A* + δ
  using *assms order.trans*[*OF DP-qbs-divergence-nonneg assms*]
  by(*cases A* ∈ *sets* (*qbs-l p*)) (*auto simp*: *DP-qbs-divergence-le-ereal-iff measure-notin-sets*)

corollary *DP-qbs-divergence-le-erealI*:
  assumes ⋀ *A*. *A* ∈ *sets* (*qbs-l p*) ⟹ *measure* (*qbs-l p*) *A* ≤ *exp* ε ∗ *measure*
(*qbs-l q*) *A* + δ
  shows *DP-divergence*$_Q$ *p q* ε ≤ *ereal* δ
  using *assms* **by**(*fastforce simp*: *DP-qbs-divergence-le-ereal-iff*)

lemma *DP-qbs-divergence-zero*:
  assumes *p* ∈ *monadP-qbs X*
    and *q* ∈ *monadP-qbs X*
    and *DP-inequality*$_Q$ *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* ≤ *b* ⟹ *DP-divergence*$_Q$ *p q b* ≤ *DP-divergence*$_Q$
p q a
  by(*auto simp*: *DP-qbs-divergence-def intro*!: *SUP-mono'* *mult-right-mono*)

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

lemma *DP-qbs-divergence-refl′*[*simp*]: *0* ≤ *e* ⟹ *DP-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* ε δ
and *DP-inequality*$_Q$ *q l* ε′ *0*
shows *DP-inequality*$_Q$ *p l* (ε + ε′) δ
unfolding *DP-qbs-divergence-le-ereal-iff diff-le-eq*
proof *safe*
fix *A*
assume [*measurable*]:*A* ∈ *sets* (*qbs-l p*)
show *measure* (*qbs-l p*) *A* ≤ δ + *exp* (ε + ε′) ∗ *measure* (*qbs-l l*) *A*
proof −
have *measure* (*qbs-l p*) *A* ≤ δ + *exp* ε ∗ *measure* (*qbs-l q*) *A*
using *assms*(*1*) **by**(*auto simp*: *DP-qbs-divergence-le-ereal-iff diff-le-eq*)
also have ... ≤ δ + *exp* ε ∗ (*exp* ε′ ∗ *measure* (*qbs-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** δ=*0*]

proposition *DP-qbs-divergence-compose*:
assumes [*qbs*,*measurable*]:*p* ∈ *monadP-qbs X q* ∈ *monadP-qbs X f* ∈ *X* →$_Q$ *monadP-qbs Y g* ∈ *X* →$_Q$ *monadP-qbs Y*
and *dp1*:*DP-divergence*$_Q$ *p q* ε ≤ *ereal* δ
and *dp2*:⋀*x*. *x* ∈ *qbs-space X* ⟹ *DP-divergence*$_Q$ (*f x*) (*g x*) ε′ ≤ *ereal* δ′
and [*arith*]: *0* ≤ ε *0* ≤ ε′
shows *DP-divergence*$_Q$ (*p* ⪢= *f*) (*q* ⪢= *g*) (ε + ε′) ≤ *ereal* (δ + δ′)
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-to-measure Y* **and** *L*=*qbs-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*]:$p \in$ *monadP-qbs X* $q \in$ *monadP-qbs X* $f \in X \rightarrow_Q$ *monadP-qbs Y*
and *dp*: *DP-divergence*$_Q$ $p\ q\ \varepsilon \leq$ *ereal* δ
and [*arith*]:$0 \leq \varepsilon$
shows *DP-divergence*$_Q$ $(p \gg= f)\ (q \gg= f)\ \varepsilon \leq$ *ereal* δ
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*] = *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[*simplified DP-qbs-divergence-qbs-l*])
qed

lemma *DP-qbs-divergence-additive*:
assumes [*qbs*]:$p \in$ *monadP-qbs X* $q \in$ *monadP-qbs X* $p' \in$ *monadP-qbs Y* $q' \in$ *monadP-qbs Y*
and *div1*: *DP-divergence*$_Q$ $p\ q\ \varepsilon \leq$ *ereal* δ
and *div2*: *DP-divergence*$_Q$ $p'\ q'\ \varepsilon' \leq$ *ereal* δ'
and [*arith*]:$0 \leq \varepsilon\ 0 \leq \varepsilon'$
shows *DP-divergence*$_Q$ $(p \bigotimes_{Qmes} p')\ (q \bigotimes_{Qmes} q')\ (\varepsilon + \varepsilon') \leq$ *ereal* $(\delta + \delta')$
proof −
note [*qbs*] = *return-qbs-morphismP bind-qbs-morphismP qbs-space-monadPM*
have *DP-divergence*$_Q$ $(p \bigotimes_{Qmes} p')\ (q \bigotimes_{Qmes} q')\ (\varepsilon + \varepsilon')$
= *DP-divergence*$_Q$ $(p \gg= (\lambda x.\ p' \gg= (\lambda y.$ *return-qbs* $(X \bigotimes_Q Y)\ (x, y))))$
$(q \gg= (\lambda x.\ q' \gg= (\lambda y.$ *return-qbs* $(X \bigotimes_Q Y)\ (x, y))))\ (\varepsilon + \varepsilon')$
by(*simp add*: *qbs-pair-measure-def*[*of - X - Y*])
also have ... $\leq$ *ereal* $(\delta + \delta')$
by(*auto intro*!: *DP-qbs-divergence-compose*[*of - X - - X* $\bigotimes_Q$ *Y*] *div1 div2*
DP-qbs-divergence-dataprocessing[*of - Y - - X* $\bigotimes_Q$ *Y*])
finally show *?thesis* **.**
qed

corollary *DP-qbs-divergence-strength*:
assumes [*qbs*]:$p \in$ *monadP-qbs X* $q \in$ *monadP-qbs X* $x \in$ *qbs-space Y*
and *dp*: *DP-divergence*$_Q$ $p\ q\ \varepsilon \leq$ *ereal* δ
and [*simp*]:$0 \leq \varepsilon$
shows *DP-divergence*$_Q$ (*return-qbs Y x* $\bigotimes_{Qmes}$ *p*) (*return-qbs Y x* $\bigotimes_{Qmes}$ *q*) $\varepsilon \leq$ *ereal* δ
proof −
note [*qbs*] = *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*$_Q$) **where**
DP-qbs-qbs-L:*differential-privacy*$_Q$ *M* ≡ *differential-privacy* (λ*x*. *qbs-l* (*M x*))

lemma *DP-qbs-def*:
differential-privacy$_Q$ *M adj* ε δ ⟷
(∀ (*d1*, *d2*)∈*adj*. *DP-inequality*$_Q$ (*M d1*) (*M d2*) ε δ ∧ *DP-inequality*$_Q$ (*M d2*) (*M d1*) ε δ)
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*$_Q$ *M adj* ε δ ⟷ (∀ (*d1*,*d2*) ∈ *adj*. *DP-inequality*$_Q$ (*M d1*) (*M d2*) ε δ)
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 *adj* ⊆ *qbs-space X* × *qbs-space X*
and *adj′* ⊆ *qbs-space X* × *qbs-space X*
and *differential-privacy*$_Q$ *M adj* ε *0*
and *differential-privacy*$_Q$ *M adj′* ε′ *0*
and *M* ∈ *X* →$_Q$ *monadP-qbs Y*
shows *differential-privacy*$_Q$ *M* (*adj O adj′*) (ε + ε′) *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 *adj* ⊆ *qbs-space X* × *qbs-space X*
and *differential-privacy*$_Q$ *M adj* ε *0*
and *M* ∈ *X* →$_Q$ *monadP-qbs Y*
shows *differential-privacy*$_Q$ *M* (*adj* ^^ *k*) (*k* ∗ ε) *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 ε ≥ *0*
and *differential-privacy*$_Q$ *M adj* ε δ
and [*qbs*,*measurable*]:*M* ∈ *X* →$_Q$ *monadP-qbs Y*
and [*qbs*,*measurable*]:*N* ∈ *Y* →$_Q$ *monadP-qbs Z*

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

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

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

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

and $Z{=}Y \bigotimes_Q Z$]
DP-qbs-postprocessing[**where** $X{=}X$ **and** $Y{=}Z$ **and** $Z{=}Y$
$\bigotimes_Q Z$])
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*]: $\bigwedge r.\ r \in$ *qbs-space* (*list-qbs* $borel_Q$)
and *qbs-space-list-qbs-count-space*[*qbs*]: $\bigwedge i.\ r \in$ *qbs-space* (*list-qbs* ($count\text{-}space_Q$
(*UNIV* :: - :: *countable*)))
by(*auto simp*: *list-qbs-space*)

2.1 Randomized Response

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

lemma *qbs-DP-RR-mechanism*:
assumes [*arith*]:$\varepsilon \geq 0$
shows $DP\text{-}divergence_Q$ (*RR-mechanism* ε *x*) (*RR-mechanism* ε *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 $DP\text{-}divergence_Q$ (*RR-mechanism* ε *x*) (*RR-mechanism* ε *y*) $\varepsilon \leq 0$
unfolding *zero-ereal-def*
proof(*rule DP-qbs-divergence-le-erealI*)
fix *A* :: *bool set*
have *ineq1*: *measure-pmf.prob* (*bernoulli-pmf* ($exp\ \varepsilon\ /\ (1 + exp\ \varepsilon)$)) *A*
$\leq exp\ \varepsilon *$ *measure-pmf.prob* (*bernoulli-pmf* ($exp\ \varepsilon\ /\ (1 + exp\ \varepsilon)$)) *A*
and *ineq2*: *measure-pmf.prob* (*bernoulli-pmf* ($1\ /\ (1 + exp\ \varepsilon)$)) *A*
$\leq exp\ \varepsilon *$ *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*
$\leq exp\ \varepsilon *$ *measure-pmf.prob* (*bernoulli-pmf* ($1\ /\ (1 + exp\ \varepsilon)$)) *A*
proof −
consider $A = \{\} \mid A = \{True\} \mid A = \{False\} \mid A = UNIV$
by (*metis* (*full-types*) *UNIV-eq-I is-singletonI′ is-singleton-the-elem*)

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*
$\leq$ *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-elem*)
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* $\leq$ *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*]:*laplace-density* $\in$ $borel_Q \Rightarrow_Q borel_Q \Rightarrow_Q borel_Q \Rightarrow_Q borel_Q$
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* (Lap'-$mechanism_Q$) **where**
Lap-$mechanism_Q$ $\equiv$ λ*e x*. *if e* $\leq$ *0 then return-qbs* $borel_Q$ *x else density-qbs* $lborel_Q$ (*laplace-density e x*)

lemma *qbs-morphism-Lap-mechanism*[*qbs*]:Lap-$mechanism_Q$ $\in$ $borel_Q \rightarrow_Q borel_Q \Rightarrow_Q$ *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*: *qbs-l* (Lap-$mechanism_Q$ *e r*) = *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*:Lap-$mechanism_Q$ *e x*= *qbs-l-inverse* (*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-divergence*$_Q$ (*Lap-mechanism*$_Q$ (*1* / ε) *x*) (*Lap-mechanism*$_Q$ (*1* / ε) *y*) ($r * \varepsilon$) = *0*
  using *DP-divergence-Lap-dist′*[**where** *b=1* / ε]
  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* ⇒ *real list* ⇒ *real qbs-measure* **where**
  qbs-NaiveRNM ε [] = *return-qbs borel 0* |
  qbs-NaiveRNM ε (*x* # *xs*) =
  (*case xs of*
    Nil ⇒ *Lap-mechanism*$_Q$ (*1* / ε) *x* |
    y#ys ⇒ *do* {*x1* ← *Lap-mechanism*$_Q$ (*1* / ε) *x*; *x2* ← *qbs-NaiveRNM* ε *xs*;
          return-qbs borel (*max x1 x2*)})

lemma *qbs-morhpism-NaiveRNM*[*qbs*]: *qbs-NaiveRNM* ∈ *borel*$_Q$ ⇒$_Q$ *list-qbs borel* ⇒$_Q$ *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*|) ≤ *r*
  shows *DP-divergence*$_Q$ (*qbs-NaiveRNM* ε *xs*) (*qbs-NaiveRNM* ε *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 ∃ *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-xs'*:$xs = x \# xs'$
  by(*auto simp*: *xs'-def h xs*)
have [*simp*]:*length* $xs' = n'$ *length* $ys' = n'$
  using *ih*(*2*) *ih*(*3*) **by**(*auto simp*: *xs-xs' 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 ... $= (\sum i\in\{0\}.\ |xs\ !\ i - (y \# ys')\ !\ i|) + (\sum i\in\{Suc\ 0..<n\}.\ |xs\ !\ i - (y \# ys')\ !\ i|)$
    by(*subst sum-Un*) *auto*
  also have ... $= |x - y| + (\sum i<n'.\ |xs\ !\ Suc\ i - (y \# ys')\ !\ Suc\ i|)$
    unfolding *n* **by**(*subst sum.atLeast-Suc-lessThan-Suc-shift*) (*simp add*: *xs-xs' n lessThan-atLeast0*)
  finally show *?thesis* **by**(*simp add*: *xs-xs'*)
qed
hence *r12*[*arith*,*simp*]: $r1 + r2 \le r$ $0 \le r1$ $0 \le r2$ $|x - y| \le r1$ $(\sum j<n'.\ |xs'\ !\ j - ys'\ !\ j|) \le r2$
  using *ih*(*4*) **by**(*auto simp*: *r1-def r2-def*)

have *?lhs* =
    $DP\text{-}divergence_Q$
      ($Lap\text{-}mechanism_Q$ (*1* / ε) x $\ggg$ ($\lambda x1.$ *qbs-NaiveRNM* ε xs' $\ggg$ ($\lambda x2.$ *return-qbs* $borel_Q$ (*max x1 x2*))))
      ($Lap\text{-}mechanism_Q$ (*1* / ε) y $\ggg$ ($\lambda x1.$ *qbs-NaiveRNM* ε ys' $\ggg$ ($\lambda x2.$ *return-qbs* $borel_Q$ (*max x1 x2*))))
      $(r * \varepsilon)$
  by(*auto simp*: *h xs xs'-def*)
also have ... $\le$
    $DP\text{-}divergence_Q$
      ($Lap\text{-}mechanism_Q$ (*1* / ε) x $\ggg$ ($\lambda x1.$ *qbs-NaiveRNM* ε xs' $\ggg$ ($\lambda x2.$ *return-qbs* $borel_Q$ (*max x1 x2*))))
      ($Lap\text{-}mechanism_Q$ (*1* / ε) y $\ggg$ ($\lambda x1.$ *qbs-NaiveRNM* ε ys' $\ggg$ ($\lambda x2.$ *return-qbs* $borel_Q$ (*max x1 x2*))))
      $(r1 * \varepsilon + r2 * \varepsilon)$
  by(*auto intro*!: *DP-qbs-divergence-antimono simp*: *distrib-right*[*symmetric*])
also have ... $\le$ *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* ⇒ (*real list* × *real list*) *set* **where**
adj-naive-RNM r ≡ {(*xs*,*ys*). *length xs* = *length ys* ∧ (∑ *i*<*length xs*. |*nth xs i* − *nth ys i*|) ≤ *r*}

theorem *qbs-DP-NaiveRNM*:
assumes *pos*: *ε* > *0*
shows *differential-privacy*$_Q$ (*qbs-NaiveRNM ε*) (*adj-naive-RNM r*) (*r* ∗ *ε*) *0*
proof(*safe intro*!: *DP-qbs-adj-sym*[*THEN iffD2*])
fix *xs ys*
assume ∗:(*xs*, *ys*) ∈ *adj-naive-RNM r*
let *?n* = *length xs*
have *length xs* = *?n* **and** *length ys* = *?n* **and** (∑ *i*<*?n*. |*nth xs i* − *nth ys i*|) ≤ *r*
using ∗ **by**(*auto simp*: *adj-naive-RNM-def*)
from *qbs-DP-NaiveRNM'*[*OF pos this*]
show *DP-inequality*$_Q$ (*qbs-NaiveRNM ε xs*) (*qbs-NaiveRNM ε ys*) (*r* ∗ *ε*) *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.