Differential Privacy

Tetsuya Sato Yasuhiko Minamide

March 17, 2025

Abstract

This entry contains formalization of differential privacy in a general setting, based on the seminal textbook of Dwork and Roth [3] and the papers of statistical divergence for differential privacy [2, 5].

This entry includes the following formalization:

- Measurable space of lists and measurability of list operations,
- Laplace distribution,
- The statisatical divergence for differential privacy,
- Differential privacy and its basic properties,
- Randomized Response,
- Laplace mechanism,
- The report noisy max mechanism.

Contents

theory *List-Space*
imports
HOL−Analysis.Finite-Product-Measure
Source-and-Sink-Algebras-Constructions
Measurable-Isomorphism
begin

1 Measurable space of finite lists

This entry introduces a measurable space of finite lists over a measurable space. The construction is inspired from Hirata's idea on list quasi-Borel spaces [Hirata et al., ITP 2023]. First, we construct countable coproduct space of product of n spaces. Then, we trasnsfer it to list space via bijections.

1.1 miscellaneous lemmas

lemma *measurable-pair-assoc*[*measurable*]:
shows $(\lambda((a, b), x).\ (a, b, x)) \in (A \bigotimes_M B) \bigotimes_M C \rightarrow_M A \bigotimes_M B \bigotimes_M C$
⟨*proof*⟩

lemma *measurable-pair-assoc'*[*measurable*]:

shows $(\lambda((a,\ b),\ x,\ l).\ (a,\ b,\ x,\ l)) \in (A \bigotimes_M B) \bigotimes_M C \bigotimes_M D \rightarrow_M A \bigotimes_M B \bigotimes_M C \bigotimes_M D$
⟨*proof*⟩

1.2 construction of list space

fun *pair-to-list* :: $(nat \times (nat \Rightarrow 'a)) \Rightarrow 'a\ list$ **where**
Zero: *pair-to-list* $(0,-) = []$
| *Suc*: *pair-to-list* $(Suc\ n,f) = (f\ 0)\ \#$ *pair-to-list* $(n,\ \lambda\ n.\ f\ (Suc\ n))$

fun *list-to-pair* :: $'a\ list \Rightarrow (nat \times (nat \Rightarrow 'a))$ **where**
Nil: *list-to-pair* $[] = (0,\lambda\text{-}.undefined)$
| *Cons*: *list-to-pair* $(x\#xs) = (Suc\ (fst($*list-to-pair* $xs)),\ \lambda n.\ if\ n = 0\ then\ x\ else\ (snd($*list-to-pair* $xs))(n - 1))$

definition *listM* :: $'a\ measure \Rightarrow 'a\ list\ measure$ **where**
listM M =
initial-source (*lists* (*space M*)) $\{($*list-to-pair*$,\ \Pi_M\ n{\in}UNIV.\ \Pi_S\ i{\in}\{..<n\}.\ M)\}$

lemma *space-listM*:
shows *space* (*listM M*) = (*lists* (*space M*))
⟨*proof*⟩

lemma *lists-listM*[*measurable*]:
shows *lists* (*space M*) ∈ *sets* (*listM M*)
⟨*proof*⟩

lemma *comp-list-to-list*:
shows *pair-to-list* (*list-to-pair xs*) = *xs*
⟨*proof*⟩

lemma *list-to-pair-function*:
shows *list-to-pair* ∈ *lists* (*space M*) → *space* $(\Pi_M\ n{\in}UNIV.\ \Pi_S\ i{\in}\{..<n\}.\ M)$
⟨*proof*⟩

lemma *list-to-pair-measurable*:
shows *list-to-pair* ∈ (*listM M*) $\rightarrow_M (\Pi_M\ n{\in}UNIV.\ \Pi_S\ i{\in}\{..<n\}.\ M)$
⟨*proof*⟩

lemma *comp-pair-to-pair*:
shows $(n,f) \in (SIGMA\ n{:}UNIV.\ \{..<n\} \rightarrow_E (space\ M)) \Longrightarrow$ (*list-to-pair* (*pair-to-list* $(n,f)) = (n,f))$
⟨*proof*⟩

lemma *pair-to-list-function*:
shows *pair-to-list* ∈ *space* $(\Pi_M\ n{\in}UNIV.\ \Pi_S\ i{\in}\{..<n\}.\ M) \rightarrow$ *lists*

(*space M*)
⟨*proof*⟩

proposition *pair-to-list-measurable*:
shows $\textit{pair-to-list} \in (\Pi_M\ n{\in}\textit{UNIV}.\ \Pi_S\ i{\in}\{..{<}n\}.\ M) \to_M (\textit{listM}\ M)$
⟨*proof*⟩

proposition *measurable-iff-list-and-pair*:
shows *measurable-iff-pair-to-list*: $\bigwedge f.\ (f \in (\textit{listM}\ M) \to_M N) \longleftrightarrow (f\ o\ \textit{pair-to-list} \in (\Pi_M\ n{\in}\textit{UNIV}.\ \Pi_S\ i{\in}\{..{<}n\}.\ M) \to_M N)$
and *measurable-iff-list-to-pair*: $\bigwedge f.\ (f\ o\ \textit{list-to-pair} \in (\textit{listM}\ M) \to_M N) \longleftrightarrow (f \in (\Pi_M\ n{\in}\textit{UNIV}.\ \Pi_S\ i{\in}\{..{<}n\}.\ M) \to_M N)$
and *measurable-iff-pair-to-list2*: $\bigwedge f.\ (f \in N \to_M (\Pi_M\ n{\in}\textit{UNIV}.\ \Pi_S\ i{\in}\{..{<}n\}.\ M)) \longleftrightarrow (\textit{pair-to-list} \circ f{\in}\ N \to_M \textit{listM}\ M \wedge f \in \textit{space}\ N \to \textit{space}\ (\Pi_M\ n{\in}\textit{UNIV}.\ \Pi_S\ i{\in}\{..{<}n\}.\ M))$
and *measurable-iff-list-to-pair2*: $\bigwedge f.\ (f \in N \to_M \textit{listM}\ M) \longleftrightarrow (\textit{list-to-pair} \circ f \in N \to_M (\Pi_M\ n{\in}\textit{UNIV}.\ \Pi_S\ i{\in}\{..{<}n\}.\ M) \wedge f \in \textit{space}\ N \to \textit{space}\ (\textit{listM}\ M))$
⟨*proof*⟩

proposition *measurable-iff-list-and-pair-plus*:
shows *measurable-iff-pair-to-list-plus*: $\bigwedge f.\ (f \in K \bigotimes_M (\textit{listM}\ M) \to_M N) \longleftrightarrow (f\ o\ (\lambda\ q.\ ((\textit{fst}\ q),\textit{pair-to-list}\ (\textit{snd}\ q))) \in K \bigotimes_M (\Pi_M\ n{\in}\textit{UNIV}.\ \Pi_S\ i{\in}\{..{<}n\}.\ M) \to_M N)$
and *measurable-iff-list-to-pair-plus*: $\bigwedge f.\ (f \in K \bigotimes_M (\Pi_M\ n{\in}\textit{UNIV}.\ \Pi_S\ i{\in}\{..{<}n\}.\ M) \to_M N) \longleftrightarrow (f\ o\ (\lambda\ p.\ ((\textit{fst}\ p),\ \textit{list-to-pair}\ (\textit{snd}\ p))) \in K \bigotimes_M (\textit{listM}\ M) \to_M N)$
⟨*proof*⟩

lemma *measurable-iff-on-list*:
shows $\bigwedge f.\ (f \in (\textit{listM}\ M) \to_M N) \longleftrightarrow (\forall\ n{\in}\textit{UNIV}.\ (f\ o\ \textit{pair-to-list}\ o\ \textit{coProj}\ n) \in (\Pi_S\ i{\in}\{..{<}n\}.\ M) \to_M N)$
⟨*proof*⟩

1.3 measurability of functions defined inductively on lists

1.3.1 Measurability of (#)

definition *shift-prod* :: $'a \Rightarrow (\textit{nat} \Rightarrow 'a) \Rightarrow (\textit{nat} \Rightarrow 'a)$ **where**
$\textit{shift-prod}\ x\ f = (\lambda\ m.\ \textit{if}\ m = 0\ \textit{then}\ x\ \textit{else}\ f(m - 1))$

lemma *shift-prod-function*:
shows $(\lambda\ (x,xs).(\textit{shift-prod}\ x\ xs)) \in \textit{space}\ M \times (\Pi_E\ i{\in}\{..{<}n\}.\ \textit{space}\ M) \to (\Pi_E\ i{\in}\{..{<}(\textit{Suc}\ n)\}.\ \textit{space}\ M)$
⟨*proof*⟩

lemma *shift-prod-measurable*:
shows $(\lambda\ (x,xs).(\textit{shift-prod}\ x\ xs)) \in M \bigotimes_M (\textit{prod-initial-source}$

$\{..<n\}$ $(\lambda$-$.M))$ $\rightarrow_M$ (*prod-initial-source* $\{..<(Suc\ n)\}$ $(\lambda$-$.M))$
⟨*proof*⟩

lemma *shift-prod-measurable'*:
assumes $x \in (space\ M)$
shows (*shift-prod* x) $\in$ (*prod-initial-source* $\{..<n\}$ $(\lambda$-$.M))$ $\rightarrow_M$ (*prod-initial-source* $\{..<(Suc\ n)\}$ $(\lambda$-$.M))$
⟨*proof*⟩

definition *shift-list* :: $'a$ *measure* $\Rightarrow$ $'a \Rightarrow nat \times (nat \Rightarrow 'a) \Rightarrow nat \times (nat \Rightarrow 'a)$ **where**
shift-list $M\ x$ =
coTuple (*UNIV* :: *nat set*)
$(\lambda\ n :: nat.$ (*space* (*prod-initial-source* $\{..<n\}$ $(\lambda$-$.M))))$
$(\lambda\ n :: nat.$ *coProj* $(Suc\ n)$ o (*shift-prod* x))

lemma *shift-list-def2*:
shows *shift-list* $M\ x\ (n, f) = (Suc\ n,$ *shift-prod* $x\ f)$
⟨*proof*⟩

lemma *shift-list-measurable*:
shows $(\lambda\ (x,(n,f)).$ (*shift-list* $M\ x\ (n,f))) \in (M \bigotimes_M (\Pi_M\ n{\in}UNIV.\ \Pi_S\ i{\in}\{..<n\}.\ M)) \rightarrow_M (\Pi_M\ n{\in}UNIV.\ \Pi_S\ i{\in}\{..<n\}.\ M)$
⟨*proof*⟩

lemma *shift-list-measurable'*:
assumes $x \in (space\ M)$
shows (*shift-list* $M\ x$) $\in (\Pi_M\ n{\in}UNIV.\ \Pi_S\ i{\in}\{..<n\}.\ M) \rightarrow_M (\Pi_M\ n{\in}UNIV.\ \Pi_S\ i{\in}\{..<n\}.\ M)$
⟨*proof*⟩

lemma *measurable-Cons*[*measurable*]:
shows $(\lambda\ (x,xs).\ x\ \#\ xs) \in M \bigotimes_M$ (*listM* M) $\rightarrow_M$ (*listM* M)
⟨*proof*⟩

lemma *listM-Nil*[*measurable*]:
shows $Nil \in space$ (*listM* M)
⟨*proof*⟩

lemma *measurable-Cons'*[*measurable*]:
shows $x \in space\ M \Longrightarrow Cons\ x \in$ (*listM* M) $\rightarrow_M$ (*listM* M)
⟨*proof*⟩

1.3.2 Measurability of $\lambda p.\ [p]$, the unit of list monad.

lemma *measurable-unit-ListM*[*measurable*]:
shows $(\lambda p.\ [p]) \in M \rightarrow_M$ *listM* M
⟨*proof*⟩

1.3.3 Measurability of *rec-list*

Since the notion of measurable functions does not support higher-order functions in general, the following theorems for measurability of*rec-list* are restricted.

lemma *measurable-rec-list-func2'*:
  fixes $T :: 'a \Rightarrow ('b \Rightarrow 'd)$
    and $F :: 'a \Rightarrow 'c \Rightarrow 'c\ list \Rightarrow ('b \Rightarrow 'd) \Rightarrow ('b \Rightarrow 'd)$
  assumes $(\lambda(a,b).\ T\ a\ b) \in K \bigotimes_M L \rightarrow_M N$
    and $\bigwedge i\ g.\ (\lambda\ ((a,\ q),\ l).\ g\ a\ q\ l) \in (K \bigotimes_M L) \bigotimes_M (\Pi_S\ i{\in}\{..{<}i\}.\ M) \rightarrow_M N$
$\Longrightarrow (\lambda((a,b),x,l).\ (F\ a\ x\ ((pair\text{-}to\text{-}list\ o\ (coProj\ i))\ l))\ (\lambda q.\ (g\ a\ q\ l))\ b) \in (K \bigotimes_M L) \bigotimes_M M \bigotimes_M (\Pi_S\ i{\in}\{..{<}i\}.\ M) \rightarrow_M N$

  shows $(\lambda((a,b),xs).\ (rec\text{-}list\ (T\ a)\ (F\ a))\ xs\ b) \in (K \bigotimes_M L) \bigotimes_M (listM\ M) \rightarrow_M N$
⟨*proof*⟩

lemma *measurable-rec-list-func2*:
  fixes $T :: 'a \Rightarrow ('b \Rightarrow 'd)$
    and $F :: 'a \Rightarrow 'c \Rightarrow 'c\ list \Rightarrow ('b \Rightarrow 'd) \Rightarrow ('b \Rightarrow 'd)$
  assumes $(\lambda(a,b).\ T\ a\ b) \in K \bigotimes_M L \rightarrow_M N$
    and $\bigwedge i\ g.\ (\lambda\ (a,\ q,\ l).\ g\ a\ q\ l) \in K \bigotimes_M L \bigotimes_M (\Pi_S\ i{\in}\{..{<}i\}.\ M) \rightarrow_M N$
$\Longrightarrow (\lambda(a,b,x,l).\ (F\ a\ x\ ((pair\text{-}to\text{-}list\ o\ (coProj\ i))\ l))\ (\lambda q.\ (g\ a\ q\ l))\ b) \in K \bigotimes_M L \bigotimes_M M \bigotimes_M (\Pi_S\ i{\in}\{..{<}i\}.\ M) \rightarrow_M N$

  shows $(\lambda(a,b,xs).\ (rec\text{-}list\ (T\ a)\ (F\ a))\ xs\ b) \in K \bigotimes_M L \bigotimes_M (listM\ M) \rightarrow_M N$
⟨*proof*⟩

lemma *measurable-rec-list'-func*:
  fixes $T :: ('c \Rightarrow 'd)$
    and $F :: 'b \Rightarrow 'b\ list \Rightarrow ('c \Rightarrow 'd) \Rightarrow ('c \Rightarrow 'd)$
  assumes $T \in K \rightarrow_M N$
    and $\bigwedge i\ g.\ g \in K \bigotimes_M (\Pi_S\ i{\in}\{..{<}i\}.\ M) \rightarrow_M N$
$\Longrightarrow (\lambda p.\ (F\ (fst\ (snd\ p))\ (pair\text{-}to\text{-}list\ (coProj\ i(snd\ (snd\ (p)))))\ (\lambda q.\ (g\ (q,snd\ (snd\ (p))))))\ (fst\ p)) \in K \bigotimes_M M \bigotimes_M (\Pi_S\ i{\in}\{..{<}i\}.\ M) \rightarrow_M N$

  shows $(\lambda p.\ (rec\text{-}list\ T\ F)\ (snd\ p)\ (fst\ p)) \in K \bigotimes_M (listM\ M) \rightarrow_M N$
⟨*proof*⟩

lemma *measurable-rec-list''-func*:
  fixes $T :: ('c \Rightarrow 'd)$
    and $F :: 'b \Rightarrow 'b\ list \Rightarrow ('c \Rightarrow 'd) \Rightarrow ('c \Rightarrow 'd)$
  assumes $T \in K \rightarrow_M N$
    and $\bigwedge i\ g.\ (\lambda(q,v).\ g\ q\ v) \in K \bigotimes_M (\Pi_S\ i{\in}\{..{<}i\}.\ M) \rightarrow_M N$

$\Longrightarrow (\lambda(x,y,xs).\ (F\ y\ (\textit{pair-to-list}\ (\textit{coProj}\ i\ xs))\ (\lambda q.\ (g\ q\ xs)))\ x) \in K \bigotimes_M M \bigotimes_M (\Pi_S\ i{\in}\{..{<}i\}.\ M) \rightarrow_M N$
shows $(\lambda(x,xs).\ (\textit{rec-list}\ T\ F)\ xs\ x) \in K \bigotimes_M (\textit{listM}\ M) \rightarrow_M N$
⟨*proof*⟩

lemma *measurable-rec-list'*:
assumes $F \in (N \bigotimes_M M \bigotimes_M (\textit{listM}\ M) \rightarrow_M N)$
shows $(\lambda p.\ (\textit{rec-list}\ (\textit{fst}\ p)\ (\lambda\ y\ xs\ x.\ F(x,y,xs))\ (\textit{snd}\ p))) \in N \bigotimes_M (\textit{listM}\ M) \rightarrow_M N$
⟨*proof*⟩

lemma *measurable-rec-list'2*:
assumes $(\lambda p.\ (F\ (\textit{fst}\ (\textit{snd}\ p))\ (\textit{snd}\ (\textit{snd}\ p))\ (\textit{fst}\ p))) \in (N \bigotimes_M M \bigotimes_M (\textit{listM}\ M) \rightarrow_M N)$
shows $(\lambda p.\ (\textit{rec-list}\ (\textit{fst}\ p)\ F\ (\textit{snd}\ p))) \in N \bigotimes_M (\textit{listM}\ M) \rightarrow_M N$
⟨*proof*⟩

lemma *measurable-rec-list*:
assumes $F \in (N \bigotimes_M M \bigotimes_M (\textit{listM}\ M) \rightarrow_M N)$
and $T \in \textit{space}\ N$
shows $\textit{rec-list}\ T\ (\lambda\ y\ xs\ x.\ F(x,y,xs)) \in (\textit{listM}\ M) \rightarrow_M N$
⟨*proof*⟩

lemma *measurable-rec-list2*:
assumes $(\lambda p.\ (F\ (\textit{fst}\ (\textit{snd}\ p))\ (\textit{snd}\ (\textit{snd}\ p))\ (\textit{fst}\ p))) \in (N \bigotimes_M M \bigotimes_M (\textit{listM}\ M) \rightarrow_M N)$
and $T \in \textit{space}\ N$
shows $\textit{rec-list}\ T\ F \in (\textit{listM}\ M) \rightarrow_M N$
⟨*proof*⟩

lemma *measurable-rec-list'''*:
assumes $(\lambda(x,y,xs).\ F\ x\ y\ xs) \in N \bigotimes_M M \bigotimes_M (\textit{listM}\ M) \rightarrow_M N$
and $T \in \textit{space}\ N$
shows $\textit{rec-list}\ T\ (\lambda\ y\ xs\ x.\ F\ x\ y\ xs) \in (\textit{listM}\ M) \rightarrow_M N$
⟨*proof*⟩

1.3.4 Measurability of *case-list*

lemma *measurable-case-list*[*measurable*(*raw*)]:
assumes [*measurable*]:$(\lambda p.\ (F\ (\textit{fst}\ p)\ (\textit{snd}\ p))) \in M \bigotimes_M (\textit{listM}\ M) \rightarrow_M N$
and $T \in \textit{space}\ N$
shows $\textit{case-list}\ T\ F \in (\textit{listM}\ M) \rightarrow_M N$
⟨*proof*⟩

lemma *measurable-case-list2*[*measurable*]:
assumes [*measurable*]:$(\lambda(x,b,bl).\ g\ x\ b\ bl) \in N \bigotimes_M L \bigotimes_M (\textit{listM}\ L) \rightarrow_M M$
and [*measurable*]:$a \in N \rightarrow_M M$

shows $(\lambda(x,xs).\ \mathit{case\text{-}list}\ (a\ x)\ (g\ x)\ xs) \in N \bigotimes_M (\mathit{listM}\ L) \to_M M$
⟨*proof*⟩

lemma *measurable-case-list′*:
assumes $(\lambda(x,b,bl).\ g\ x\ b\ bl) \in N \bigotimes_M L \bigotimes_M (\mathit{listM}\ L) \to_M M$
and $a \in N \to_M M$
and $l \in N \to_M \mathit{listM}\ L$
shows $(\lambda x.\ \mathit{case\text{-}list}\ (a\ x)\ (g\ x)\ (l\ x)) \in N \to_M M$
⟨*proof*⟩

1.3.5 Measurability of *foldr*

lemma *measurable-foldr*:
assumes $(\lambda\ (x,y).\ F\ x\ y) \in (N \bigotimes_M M \to_M N)$
shows $(\lambda\ (T,xs).\ \mathit{foldr}\ (\lambda\ x\ y.\ F\ y\ x)\ xs\ T) \in N \bigotimes_M (\mathit{listM}\ M) \to_M N$
⟨*proof*⟩

1.3.6 Measurability of (@)

lemma *measurable-append*[*measurable*]:
shows $(\lambda\ (xs,ys).\ xs\ @\ ys) \in (\mathit{listM}\ M) \bigotimes_M (\mathit{listM}\ M) \to_M (\mathit{listM}\ M)$
⟨*proof*⟩

1.3.7 Measurability of *rev*

lemma *measurable-rev*[*measurable*]:
shows $\mathit{rev} \in (\mathit{listM}\ M) \to_M (\mathit{listM}\ M)$
⟨*proof*⟩

1.3.8 Measurability of *concat*, that is, the bind of the list monad

lemma *measurable-concat*[*measurable*]:
shows $\mathit{concat} \in \mathit{listM}\ (\mathit{listM}\ M) \to_M (\mathit{listM}\ M)$
⟨*proof*⟩

1.3.9 Measurability of *map*, the functor part of the list monad

lemma *measurable-map*[*measurable*]:
assumes [*measurable*]: $f \in M \to_M N$
shows $(\mathit{map}\ f) \in (\mathit{listM}\ M) \to_M (\mathit{listM}\ N)$
⟨*proof*⟩

1.3.10 Measurability of *length*

lemma *measurable-length*[*measurable*]:
shows $\mathit{length} \in \mathit{listM}\ M \to_M \mathit{count\text{-}space}\ (\mathit{UNIV} :: \mathit{nat\ set})$

⟨*proof*⟩

lemma *sets-listM-length*[*measurable*]:
shows $\{xs.\ xs \in space\ (listM\ M) \wedge length\ xs = n\} \in sets\ (listM\ M)$
⟨*proof*⟩

1.3.11 Measurability of *foldl*

lemma *measurable-foldl* [*measurable*]:
assumes [*measurable*]:$(\lambda(x,y).\ F\ y\ x) \in (M \bigotimes_M N \rightarrow_M N)$
shows $(\lambda\ (T,xs).\ foldl\ F\ T\ xs) \in N \bigotimes_M (listM\ M) \rightarrow_M N$
⟨*proof*⟩

1.3.12 Measurability of *fold*

lemma *measurable-fold* [*measurable*]:
assumes [*measurable*]: $(\lambda(x,y).\ F\ x\ y) \in (M \bigotimes_M N \rightarrow_M N)$
shows $(\lambda\ (T,xs).\ fold\ F\ xs\ T) \in N \bigotimes_M (listM\ M) \rightarrow_M N$
⟨*proof*⟩

1.3.13 Measurability of *drop*

lemma *measurable-drop*[*measurable*]:
shows $drop\ n \in listM\ M \rightarrow_M listM\ M$
⟨*proof*⟩

lemma *measurable-drop'*[*measurable*]:
shows $(\lambda\ (n,xs).\ drop\ n\ xs) \in count\text{-}space\ (UNIV :: nat\ set) \bigotimes_M listM\ M \rightarrow_M listM\ M$
⟨*proof*⟩

1.3.14 Measurability of *take*

lemma *measurable-take*[*measurable*]:
shows $take\ n \in listM\ M \rightarrow_M listM\ M$
⟨*proof*⟩

lemma *measurable-take'*[*measurable*]:
shows $(\lambda\ (n,xs).\ take\ n\ xs) \in count\text{-}space\ (UNIV :: nat\ set) \bigotimes_M listM\ M \rightarrow_M listM\ M$
⟨*proof*⟩

1.3.15 Measurability of (!) plus a default value

Since the @term undefined is harmful to prove measurability, we introduce the total function version of (!) that returns a fixed default value when the nth element is not found.

primrec *nth-total* :: $'a \Rightarrow 'a\ list \Rightarrow nat \Rightarrow 'a$ **where**
$nth\text{-}total\ d\ []\ n = d\ |$

nth-total d (*x* # *xs*) *n* = (*case n of 0* ⇒ *x* | *Suc k* ⇒ *nth-total d xs k*)

value*nth-total* (*15* :: *nat*) [] (*0* :: *nat*) :: *nat*
value*nth-total* (*15* :: *nat*) [*1*] (*0* :: *nat*) :: *nat*
value*nth-total* (*15* :: *nat*) [*1*,*2*,*3*] (*0* :: *nat*) :: *nat*
value*nth-total* (*15* :: *nat*) [*1*,*2*,*3*] (*1* :: *nat*) :: *nat*
value*nth-total* (*15* :: *nat*) [*1*,*2*,*3*] (*2* :: *nat*) :: *nat*
value*nth-total* (*15* :: *nat*) [*1*,*2*,*3*] (*3* :: *nat*) :: *nat*

value*nth* [*1*] (*0* :: *nat*) :: *nat*
value*nth* [*1*,*2*,*3*] (*0* :: *nat*) :: *nat*
value*nth* [*1*,*2*,*3*] (*1* :: *nat*) :: *nat*
value*nth* [*1*,*2*,*3*] (*2* :: *nat*) :: *nat*

The total version *nth-total* is the same as (!) if the nth element exists and otherwise it returns the default value.

lemma *cong-nth-total-nth*:
shows ((*n* :: *nat*) < *length xs* ∧ *0* < *length xs*) ⟹ *nth-total d xs n* = *nth xs n*
⟨*proof*⟩

lemma *cong-nth-total-default*:
shows ¬((*n* :: *nat*) < *length xs* ∧ *0* < *length xs*) ⟹ *nth-total d xs n* = *d*
⟨*proof*⟩

lemma *nth-total-def2*:
shows *nth-total d xs n* = (*if* (*n* < *length xs* ∧ *0* < *length xs*) *then xs* ! *n else d*)
⟨*proof*⟩

context
fixes *d*
begin

primrec *nth-total′* :: *′a list* ⇒ *nat* ⇒ *′a* **where**
nth-total′ [] *n* = *d* |
nth-total′ (*x* # *xs*) *n* = (*case n of 0* ⇒ *x* | *Suc k* ⇒ *nth-total′ xs k*)

lemma *measurable-nth-total′*:
assumes *d*: *d* ∈ *space M*
shows (λ (*n*,*xs*). *nth-total′ xs n*) ∈ (*count-space UNIV*) $\bigotimes_M$ *listM M* $\rightarrow_M$ *M*
⟨*proof*⟩

lemma *nth-total-nth-total′*:
shows *nth-total d xs n* = *nth-total′ xs n*
⟨*proof*⟩

end

lemma *measurable-nth-total*[*measurable*]:
  assumes $d \in$ *space M*
  shows $(\lambda\ (n,xs).\ nth\text{-}total\ d\ xs\ n) \in (count\text{-}space\ UNIV) \bigotimes_M listM\ M \rightarrow_M M$
  ⟨*proof*⟩

1.3.16 Definition and Measurability of list insert.

definition *list-insert* :: $'a \Rightarrow 'a\ list \Rightarrow nat \Rightarrow 'a\ list$ **where**
  list-insert x xs i = (*take i xs*) @ [*x*] @ (*drop i xs*)

value*list-insert* (*15* :: *nat*) [] (*0* :: *nat*) :: *nat list*
value*list-insert* (*15* :: *nat*) [] (*1* :: *nat*) :: *nat list*
value*list-insert* (*15* :: *nat*) [*1*,*2*,*3*] (*0* :: *nat*) :: *nat list*
value*list-insert* (*15* :: *nat*) [*1*,*2*,*3*] (*1* :: *nat*) :: *nat list*
value*list-insert* (*15* :: *nat*) [*1*,*2*,*3*] (*2* :: *nat*) :: *nat list*
value*list-insert* (*15* :: *nat*) [*1*,*2*,*3*] (*3* :: *nat*) :: *nat list*
value*list-insert* (*15* :: *nat*) [*1*,*2*,*3*] (*4* :: *nat*) :: *nat list*

lemma *measurable-list-insert*[*measurable*]:
  fixes *n* :: *nat*
  shows $(\lambda\ (d,xs).\ list\text{-}insert\ d\ xs\ n) \in M \bigotimes_M listM\ M \rightarrow_M listM\ M$
  ⟨*proof*⟩

primrec *list-insert'* :: $'a \Rightarrow 'a\ list \Rightarrow nat \Rightarrow 'a\ list$ **where**
  list-insert' d [] *n* = [*d*]
| *list-insert' d* (*x* # *xs*) *n* = (*case n of 0* ⇒ *d* # (*x* # *xs*) | *Suc k* ⇒ *x* # (*list-insert' d xs k*))

lemma *list-insert'-is-list-insert*:
  shows *list-insert' x xs i* = *list-insert x xs i*
⟨*proof*⟩

definition *list-exclude* :: *nat*⇒ *'a list* ⇒ *'a list* **where**
  list-exclude n xs = ((*take n xs*) @ (*drop* (*Suc n*) *xs*))

value*list-exclude* (*0* :: *nat*) [] :: *nat list*
value*list-exclude* (*0* :: *nat*) [*1*,*2*] :: *nat list*
value*list-exclude* (*1* :: *nat*) [*1*,*2*] :: *nat list*
value*list-exclude* (*2* :: *nat*) [*1*,*2*] :: *nat list*

lemma *measurable-list-exclude*[*measurable*]:
  shows $list\text{-}exclude\ n \in listM\ M \rightarrow_M listM\ M$
  ⟨*proof*⟩

lemma *insert-exclude*:
shows $n < length\ xs \Longrightarrow xs \neq [] \Longrightarrow list\text{-}insert'\ (nth\text{-}total\ d\ xs\ n)\ (list\text{-}exclude\ n\ xs)\ n = xs$
⟨*proof*⟩

1.3.17 Measurability of *list-update*

lemma *update-insert-exclude*:
shows $n < length\ xs \Longrightarrow xs \neq [] \Longrightarrow list\text{-}update\ xs\ n\ x = list\text{-}insert'\ x\ (list\text{-}exclude\ n\ xs)\ n$
⟨*proof*⟩

lemma *list-update-def2*:
shows $list\text{-}update\ xs\ n\ x = If\ (n < length\ xs \wedge xs \neq [])\ (list\text{-}insert'\ x\ (list\text{-}exclude\ n\ xs)\ n)\ xs$
⟨*proof*⟩

lemma *measurable-list-update*[*measurable*]:
shows $(\lambda\ (x,xs).\ list\text{-}update\ xs\ (n :: nat)\ x) \in M \bigotimes_M (listM\ M) \rightarrow_M (listM\ M)$
⟨*proof*⟩

1.3.18 Measurability of *zip*

lemma *measurable-zip*[*measurable*]:
shows $(\lambda(xs,ys).\ (zip\ xs\ ys)) \in (listM\ M) \bigotimes_M (listM\ N) \rightarrow_M (listM\ (M \bigotimes_M N))$
⟨*proof*⟩

1.3.19 Measurability of *map2*

lemma *measurable-map2*[*measurable*]:
assumes [*measurable*]: $(\lambda(x,y).\ f\ x\ y) \in M \bigotimes_M M' \rightarrow_M N$
shows $(\lambda(xs,ys).\ map2\ f\ xs\ ys) \in (listM\ M) \bigotimes_M (listM\ M') \rightarrow_M (listM\ N)$
⟨*proof*⟩

end

theory *L1-norm-list*
imports *HOL−Analysis.Abstract-Metric-Spaces*
HOL−Library.Extended-Real
begin

2 L1-norm on Lists with fixed length

lemma *list-sum-dist-Cons*:
fixes *a b xs ys n* **assumes** *xn*: $length\ xs = n$ **and** *yn*: $length\ ys = n$

shows $(\sum i = 1..Suc\ n.\ d\ ((a \# xs)\ !\ (i - 1))\ ((b \# ys)\ !\ (i - 1)))$
$= d\ a\ b + (\sum i = 1..n.\ d\ (xs\ !\ (i - 1))\ (ys\ !\ (i - 1)))$
$\langle proof \rangle$

2.1 A locale for L1-norm of Lists

locale *L1-norm-list* = *Metric-space* +
fixes *n*::*nat*
begin

definition *space-L1-norm-list* ::('*a list*) *set* **where**
space-L1-norm-list = {*xs*. *xs*∈ *lists M* ∧ *length xs* = *n*}

lemma *in-space-L1-norm-list-iff*:
shows *x* ∈ *space-L1-norm-list* ⟷(*x*∈ *lists M* ∧ *length x* = *n*)
⟨*proof*⟩

definition *dist-L1-norm-list* ::('*a list*) ⇒ ('*a list*) ⇒ *real* **where**
dist-L1-norm-list xs ys = $(\sum\ i\in\{1..n\}.\ d\ (nth\ xs\ (i-1))\ (nth\ ys\ (i-1)))$

lemma *dist-L1-norm-list-nonneg*:
shows $\bigwedge$ *x y*. *0* ≤ *dist-L1-norm-list x y*
⟨*proof*⟩

lemma *dist-L1-norm-list-commute*:
shows $\bigwedge$ *x y*. *dist-L1-norm-list x y* = *dist-L1-norm-list y x*
⟨*proof*⟩

lemma *dist-L1-norm-list-zero*:
shows $\bigwedge$ *x y*. *length x* = *n* ⟹ *length y* = *n* ⟹ *x* ∈ *lists M* ⟹ *y* ∈ *lists M* ⟹ (*dist-L1-norm-list x y* = *0* = (*x* = *y*))
⟨*proof*⟩

lemma *dist-L1-norm-list-trans*:
shows$\bigwedge$ *x y z*. *length x* = *n* ⟹ *length y* = *n* ⟹ *length z* = *n*⟹ *x* ∈ *lists M* ⟹ *y* ∈ *lists M* ⟹ *z* ∈ *lists M* ⟹ (*dist-L1-norm-list x z* ≤ *dist-L1-norm-list x y* + *dist-L1-norm-list y z*)
⟨*proof*⟩

lemma *Metric-L1-norm-list* : *Metric-space space-L1-norm-list dist-L1-norm-list*
⟨*proof*⟩

end

sublocale *L1-norm-list* ⊆ *MetL1*: *Metric-space space-L1-norm-list dist-L1-norm-list*
⟨*proof*⟩

2.2 Lemmas on L1-norm on lists of natural numbers

lemma *L1-dist-k*:
  fixes *x y k*::*nat*
  assumes *real-of-int* $|int\ x - int\ y| \le k$
  shows $(x,y) \in \{(x,y) \mid x\ y.\ |int\ x - int\ y| \le 1\}$ ^^ *k*
  ⟨*proof*⟩

context
  fixes *n*::*nat*
begin

interpretation *L11nat*: *L1-norm-list* (*UNIV*::*nat set*) ($\lambda x\ y.\ |int\ x - int\ y|$) *n*
  ⟨*proof*⟩
interpretation *L11nat2*: *L1-norm-list* (*UNIV*::*nat set*) ($\lambda x\ y.\ |int\ x - int\ y|$) *Suc n*
  ⟨*proof*⟩

abbreviation *adj-L11nat* ≡ {(*xs*, *ys*) |*xs ys*. *xs* ∈ *L11nat.space-L1-norm-list* ∧ *ys* ∈ *L11nat.space-L1-norm-list* ∧ *L11nat.dist-L1-norm-list xs ys* ≤ *1*}
abbreviation *adj-L11nat2* ≡ {(*xs*, *ys*) |*xs ys*. *xs* ∈ *L11nat2.space-L1-norm-list* ∧ *ys* ∈ *L11nat2.space-L1-norm-list* ∧ *L11nat2.dist-L1-norm-list xs ys* ≤ *1*}

lemma *L1-adj-iterate-Cons1*:
  assumes *xs* ∈ *L11nat.space-L1-norm-list*
    and *ys* ∈ *L11nat.space-L1-norm-list*
    and (*xs*, *ys*) ∈ *adj-L11nat* ^^ *k*
  shows (*x#xs*, *x#ys*) ∈ *adj-L11nat2* ^^ *k*
  ⟨*proof*⟩

lemma *L1-adj-Cons1*:
  assumes *xs* ∈ *L11nat.space-L1-norm-list*
    and *real-of-int* $|int\ x - int\ y| \le 1$
  shows (*x* # *xs*, *y* # *xs*) ∈ *adj-L11nat2*
  ⟨*proof*⟩

lemma *L1-adj-iterate-Cons2*:
  assumes *xs* ∈ *L11nat.space-L1-norm-list* **and** *real-of-int* $|int\ x - int\ y| \le k$
  shows (*x* # *xs*, *y* # *xs*) ∈ *adj-L11nat2* ^^ *k*
⟨*proof*⟩

lemma *L1-adj-iterate*:
  fixes *k*::*nat*
  assumes *xs* ∈ *L11nat.space-L1-norm-list*
    and *ys* ∈ *L11nat.space-L1-norm-list*

and $L11nat.dist\text{-}L1\text{-}norm\text{-}list\ xs\ ys \leq k$
shows $(xs,ys) \in adj\text{-}L11nat\ \widehat{}\widehat{}\ k$
$\langle proof \rangle$

end

hide-fact (**open**) *list-sum-dist-Cons*

end

theory *Laplace-Distribution*
imports *HOL−Probability.Probability*
Additional-Lemmas-for-Integrals
Additional-Lemmas-for-Calculation
begin

3 Laplace Distribution

3.1 Desity Function and Cumulative Distribution Function

We refer `HOL/Probability/Distributions.thy` in the standard library.

definition *laplace-density* :: $real \Rightarrow real \Rightarrow real \Rightarrow real$ **where**
$laplace\text{-}density\ l\ m\ x = (if\ l > 0\ then\ exp(-|x - m| / l) / (2 * l)\ else\ 0)$

definition *laplace-CDF* :: $real \Rightarrow real \Rightarrow real \Rightarrow real$ **where**
$laplace\text{-}CDF\ l\ m\ x =$
$(if\ l > 0\ then$
$if\ x < m\ then\ exp((x - m) / l) / 2\ else\ 1 - exp(-(x - m) / l) / 2$
$else\ 0)$

lemma *laplace-density-nonneg*[*simp*]:
shows $0 \leq laplace\text{-}density\ l\ m\ x$
$\langle proof \rangle$

lemma *laplace-CDF-nonneg*[*simp*]:
shows $0 \leq laplace\text{-}CDF\ l\ m\ x$
$\langle proof \rangle$

lemma *borel-measurable-laplace-density*[*measurable*]:
shows $laplace\text{-}density\ l\ m \in borel \rightarrow_M borel$
$\langle proof \rangle$

lemma *borel-measurable-laplace-density2*[*measurable*]:
shows $(\lambda\ m :: real.\ (laplace\text{-}density\ l\ m\ x)) \in borel \rightarrow_M borel$

⟨*proof*⟩

lemma *laplace-density-mirror*:
shows *laplace-density l m* (*2* $*$ *m* $-$ *x*) = *laplace-density l m x*
⟨*proof*⟩

lemma *laplace-density-shift*:
shows *laplace-density l* (*m* + *c*) (*x* + *c*) = *laplace-density l m x*
⟨*proof*⟩

lemma *borel-measurable-laplace-CDF*[*measurable*]:
shows *laplace-CDF l m* $\in$ *borel* $\rightarrow_M$ *borel*
⟨*proof*⟩

lemma *borel-measurable-laplace-CDF2*[*measurable*]:
shows (λ *m* :: *real*. *laplace-CDF l m x*)$\in$ *borel* $\rightarrow_M$ *borel*
⟨*proof*⟩

lemma *laplace-CDF-mirror*:
assumes *l* > *0*
shows *laplace-CDF l m* (*2* $*$ *m* $-$ *x*) = *1* $-$ *laplace-CDF l m x*
⟨*proof*⟩

lemma *laplace-CDF-shift*:
shows *laplace-CDF l* (*m* + *c*) (*x* + *c*) = *laplace-CDF l m x*
⟨*proof*⟩

lemma *nn-integral-laplace-density-pos*:
assumes *pos*[*arith*]: *0* < *l* **and** *1*: *a* $\geq$ *m*
shows ($\int^+$ *x* $\in$ {*a*..}. *ennreal* (*laplace-density l m x*) ∂*lborel*) = *1* $-$ *laplace-CDF l m a*
⟨*proof*⟩

lemma *nn-integral-laplace-density-pos-center*:
assumes *pos*[*arith*]: *0* < *l*
shows ($\int^+$ *x* $\in$ {*m*..}. *ennreal* (*laplace-density l m x*) ∂*lborel*) = *1/2*
⟨*proof*⟩

lemma *nn-integral-laplace-density-neg*:
assumes *pos*[*arith*]: *0* < *l* **and** *1*: *a* $\leq$ *m*
shows ($\int^+$ *x* $\in$ {..*a*}. *ennreal* (*laplace-density l m x*) ∂*lborel*) = *laplace-CDF l m a*
⟨*proof*⟩

lemma *nn-integral-laplace-density-neg-center*:
assumes *pos*[*arith*]: *0* < *l*
shows ($\int^+$ *x* $\in$ {..*m*}. *ennreal* (*laplace-density l m x*) ∂*lborel*) = *1/2*

⟨*proof*⟩

proposition *nn-integral-laplace-density-mass-1*:
  assumes *pos*[*arith*]: $0 < l$
  shows $(\int^+ x.\ ennreal\ (laplace\text{-}density\ l\ m\ x)\ \partial lborel) = 1$
⟨*proof*⟩

lemma *emeasure-laplace-density-mass-1*:
  $0 < l \Longrightarrow emeasure\ (density\ lborel\ (laplace\text{-}density\ l\ m))\ UNIV = 1$
  ⟨*proof*⟩

proposition *nn-integral-laplace-density*:
  assumes *pos*[*arith*]: $0 < l$
  shows $(\int^+ x \in \{..a\}.\ ennreal\ (laplace\text{-}density\ l\ m\ x)\ \partial lborel) = laplace\text{-}CDF\ l\ m\ a$
⟨*proof*⟩

lemma *emeasure-laplace-density*:
  assumes $0 < l$
  shows $emeasure\ (density\ lborel\ (laplace\text{-}density\ l\ m))\ \{..\ a\} = laplace\text{-}CDF\ l\ m\ a$
  ⟨*proof*⟩

3.2 Moments

lemma *laplace-moment-0-a*:
  assumes *pos*[*arith*]: $0 < l$
    and *posa*[*arith*]: $0 \leq a$
  shows $has\text{-}bochner\text{-}integral\ lborel\ (\lambda\ x.\ (indicator\ \{0..a\}\ x *_R (laplace\text{-}density\ l\ 0\ x)))(1/2 - exp(-a\ /\ l)\ /2)$
⟨*proof*⟩

lemma *laplace-moment-0-pos*:
  assumes *pos*[*arith*]: $0 < l$
  shows $has\text{-}bochner\text{-}integral\ lborel\ (\lambda\ x.\ (indicator\ \{0..\}\ x *_R (laplace\text{-}density\ l\ 0\ x)))(1/2 :: real)$
⟨*proof*⟩

lemma *laplace-moment-0*:
  fixes $k :: nat$
  assumes *pos*[*arith*]: $0 < l$
  shows $has\text{-}bochner\text{-}integral\ lborel\ (\lambda\ x.\ (indicator\ \{0..\}\ x *_R ((laplace\text{-}density\ l\ 0\ x) * x\hat{\ }k)))(fact\ k * l\hat{\ }k/2)$
    and $(\lambda a.\ LBINT\ x.\ indicator\ \{0..a\}\ x *_R ((laplace\text{-}density\ l\ 0\ x) * x\hat{\ }k)) \longlonglongrightarrow (LBINT\ y.\ (indicator\ \{0..\}\ y *_R ((laplace\text{-}density\ l\ 0\ y) * y\hat{\ }k)))$
⟨*proof*⟩

lemma *laplace-moment-even*:

fixes k :: *nat*
assumes *pos*[*arith*]: $0 < l$
shows *has-bochner-integral lborel* (λ *x*. ((*laplace-density l m x*) $*$ ($x - m$)^(2 $*$ k)))(*fact* (2 $*$ k) $*$ l^(2 $*$ k))
⟨*proof*⟩

lemma *laplace-moment-odd*:
fixes k :: *nat*
assumes *pos*[*arith*]: $0 < l$
shows *has-bochner-integral lborel* (λ *x*. ((*laplace-density l m x*) $*$ ($x - m$)^(2 $*$ k + 1)))(0)
⟨*proof*⟩

lemma *laplace-moment-abs-odd*:
fixes k :: *nat*
assumes *pos*[*arith*]: $0 < l$
shows *has-bochner-integral lborel* (λ *x*. ((*laplace-density l m x*) $*$ $|x - m|$^(2 $*$ k + 1)))(*fact* (2 $*$ k + 1) $*$ l ^ (2 $*$ k + 1))
⟨*proof*⟩

lemma *integral-laplace-moment-even*:
assumes *pos*[*arith*]: $0 < l$
shows $integral^L$ *lborel* (λx.(*laplace-density l m x*) $*$ ($x - m$)^(2 $*$ k)) = *fact* (2 $*$ k) $*$ l^(2 $*$ k)
⟨*proof*⟩

lemma *integral-laplace-moment-odd*:
assumes *pos*[*arith*]: $0 < l$
shows $integral^L$ *lborel* (λx.(*laplace-density l m x*) $*$ ($x - m$)^(2 $*$ k + 1)) = 0
⟨*proof*⟩

lemma *integral-laplace-moment-abs-odd*:
assumes *pos*[*arith*]: $0 < l$
shows $integral^L$ *lborel* (λx.(*laplace-density l m x*) $*$ $|x - m|$^(2 $*$ k + 1)) =*fact* (2 $*$ k + 1) $*$ l^(2 $*$ k + 1)
⟨*proof*⟩

lemma *integrable-laplace-moment*:
fixes k :: *nat*
assumes *pos*[*arith*]: $0 < l$
shows *integrable lborel* (λ *x*. ((*laplace-density l m x*) $*$ ($x - m$)^(k)))
⟨*proof*⟩

lemma *integrable-laplace-moment-abs*:
fixes k :: *nat*
assumes *pos*[*arith*]: $0 < l$
shows *integrable lborel* (λ *x*. ((*laplace-density l m x*) $*$ $|\ x - m\ |$^(k)))

⟨*proof*⟩

lemma *integrable-laplace-density*[*simp*, *intro*]:
  assumes *pos*[*arith*]: *0* < *l*
  shows *integrable lborel* (*laplace-density l m*)
  ⟨*proof*⟩

lemma *integral-laplace-density*[*simp*, *intro*]:
  assumes *pos*[*arith*]: *0* < *l*
  shows ($\int$ *x*. *laplace-density l m x* ∂*lborel*) = *1*
  ⟨*proof*⟩

3.3 The Probability Space Distributed by Laplace Distribution

lemma *prob-space-laplacian-density*:
  assumes *pos*[*arith*]: *0* < *l*
  shows *prob-space* (*density lborel* (*laplace-density l m*))
⟨*proof*⟩

lemma (**in** *prob-space*) *laplace-distributed-le*:
  assumes *D*: *distributed M lborel X* (*laplace-density l m*)
    and [*simp*, *arith*]: *0* < *l*
  shows $\mathcal{P}$(*x in M*. *X x* ≤ *a*) = *laplace-CDF l m a*
⟨*proof*⟩

lemma (**in** *prob-space*) *laplace-distributed-gt*:
  assumes *D*: *distributed M lborel X* (*laplace-density l m*)
    and [*simp*, *arith*]: *0* < *l*
  shows $\mathcal{P}$(*x in M*. *X x* > *a*) = *1* − *laplace-CDF l m a*
⟨*proof*⟩

end

theory *Comparable-Probability-Measures*
  imports*HOL−Probability.Probability*
    Additional-Lemmas-for-Integrals
begin

4 Comparable Pairs of Probability Measures

To compare two probability measures M and N on the same space, we introduce their Radon-Nikodym derivatives (i.e. density functions) with respect to the sum measure M + N. We could consider various base measures, but we choose the sum measure, because it is finite; it is easy to check the absolute continuity M, N « M + N; the Radon-Nikodym derivatives dM and dN are bounded and are essentially partition of unity(i.e. dM + dN = 1 a.e.).

4.1 Sum of two measures

definition *sum-measure* :: *'a measure* ⇒ *'a measure* ⇒ *'a measure*
where
sum-measure M N = *measure-of* (*space M*) (*sets M*) (λ *A*. *emeasure M A* + *emeasure N A*)

lemma *sum-measure-space*[*simp*, *measurable*]:
shows *space* (*sum-measure M N*) = *space M*
and *sets* (*sum-measure M N*) = *sets M*
⟨*proof*⟩

lemma *sum-measure-commutativitiy*:
assumes *finite-measure M*
and *finite-measure N*
and *space M* = *space N*
and *sets M* = *sets N*
shows *sum-measure N M* = *sum-measure M N*
⟨*proof*⟩

lemma *sum-measure-space2*:
assumes *space M* = *space N*
and *sets M* = *sets N*
shows *space* (*sum-measure M N*) = *space N*
and *sets* (*sum-measure M N*) = *sets N*
⟨*proof*⟩

This lemma is inspired from ⟦*finite-measure ?M*; *finite-measure ?N*; *sets ?M* = *sets ?N*; $\bigwedge$*A*. *A* ∈ *sets ?M* ⟹ *emeasure ?N A* ≤ *emeasure ?M A*; *?A* ∈ *sets ?M*⟧ ⟹ *emeasure* (*diff-measure ?M ?N*) *?A* = *emeasure ?M ?A* − *emeasure ?N ?A* in the standard library.

lemma *emeasure-sum-measure* [*simp*]:
assumes *fin*: *finite-measure M*
and *finite-measure N*
and *sets-eq*: *sets M* = *sets N*
and *A*: *A* ∈ *sets M*
shows *emeasure* (*sum-measure M N*) *A* = *emeasure M A* + *emeasure N A*
(**is**- = *?μ A*)
⟨*proof*⟩

lemma *sum-measure-finiteness* [*simp*]:

assumes *finite-measure M*
and *finite-measure N*
and *space M = space N*
and *sets M = sets N*
shows *finite-measure (sum-measure M N)*
⟨*proof*⟩

lemma *absolute-continuity-sum-measure* [*simp*]:
assumes *finite-measure M*
and *finite-measure N*
and *space M = space N*
and *sets M = sets N*
shows *absolutely-continuous (sum-measure M N) M*
⟨*proof*⟩

lemma *absolute-continuity-sum-measure2* [*simp*]:
assumes *finite-measure M*
and *finite-measure N*
and *space M = space N*
and *sets M = sets N*
shows *absolutely-continuous (sum-measure N M) M*
⟨*proof*⟩

lemma *density-sum-measure*:
assumes *finite-measure M*
and *finite-measure N*
and *space M = space N*
and *sets M = sets N*
shows *density (sum-measure M N) (RN-deriv (sum-measure M N) M) = M*
⟨*proof*⟩

lemma *density-sum-measure2*:
assumes *finite-measure M*
and *finite-measure N*
and *space M = space N*
and *sets M = sets N*
shows *density (sum-measure N M) (RN-deriv (sum-measure N M) M) = M*
⟨*proof*⟩

lemma *absolutely-continuous-emeasure-less* [*simp*]:
assumes *sets M = sets N*
and $\forall$ *A* $\in$ *sets M. emeasure M A* $\leq$ *emeasure N A*
shows *absolutely-continuous N M*
⟨*proof*⟩

lemma *emeasure-less-RN-deriv-bound-1* [*simp*]:
assumes *finite-measure M*

and *finite-measure N*
and *space M* = *space N*
and *sets M* = *sets N*
and ∀ *A* ∈ *sets M*. *emeasure M A* ≤ *emeasure N A*
shows *AE x in N* . *RN-deriv N M x* ≤ *1*
⟨*proof*⟩

lemma *functions-less-upto-AE*[*simp*]:
assumes (*f* :: -⇒ ′*b* :: {*linorder-topology*, *second-countable-topology*}) ∈ *borel-measurable M*
and (*g* :: -⇒′*b*) ∈ *borel-measurable M*
and (*h* :: -⇒ ′*b*) ∈ *borel-measurable M*
and *fgequal*: *AE x in M*. *f x* = *g x*
and *AE x in M*. *g x* ≤ *h x*
shows *AE x in M*. *f x* ≤ *h x*
⟨*proof*⟩

lemma *density-sum-measure-bounded*[*simp*]:
assumes *finite-measure M*
and *finite-measure N*
and *space M* = *space N*
and *sets M* = *sets N*
shows *AE x in* (*sum-measure M N*) .(*RN-deriv* (*sum-measure M N*) *M*) *x* ≤ *1*
⟨*proof*⟩

lemma *density-sum-measure-bounded2*[*simp*]:
assumes *finite-measure M*
and *finite-measure N*
and *space M* = *space N*
and *sets M* = *sets N*
shows *AE x in* (*sum-measure M N*) .(*RN-deriv* (*sum-measure M N*) *N*) *x* ≤ *1*
⟨*proof*⟩

4.2 Miscellaneous additional lemmas on probability measures

lemma *measurable-bind-prob-space-simple*:
assumes[*measurable*]: $f \in L \rightarrow_M$ (*prob-algebra K*)
shows (λ*M*. (*M* ⋙= *f*)) ∈ (*prob-algebra L*)$\rightarrow_M$ (*prob-algebra K*)
⟨*proof*⟩

lemma
assumes *M* ∈ *space* (*prob-algebra L*)
shows *actual-prob-space*: *prob-space M*
and *space-prob-algebra-sets*: *sets M* = *sets L*
and *space-prob-algebra-space*: *space M* = *space L*
⟨*proof*⟩

lemma *evaluations-probabilistic-process* [*simp*,*intro*]:
assumes *f*: $f \in$ *measurable L* (*prob-algebra K*)
and $A \in$ *sets K*
shows (λ *x*. *measure* (*f x*) *A*) $\in$ *borel-measurable L*
and (λ *x*. *emeasure* (*f x*) *A*) $\in$ *borel-measurable L*
and $\forall$ $x \in$ *space L*. *measure* (*f x*) *A* $\leq$ *1*
and $\forall$ $x \in$ *space L*. *measure* (*f x*) *A* $\geq$ *0*
and $\forall$ $x \in$ *space L*. *norm*(*measure* (*f x*) *A*) $\leq$ *1*
and $\forall$ $x \in$ *space L*. *emeasure* (*f x*) *A* $\leq$ *1*
and (*sets N* = *sets L*) $\Longrightarrow$ (*finite-measure N*)$\Longrightarrow$ *integrable N* (λ *x*. *measure* (*f x*) *A*)
⟨*proof*⟩

lemma *Giry-strength-bind-return*:
assumes $N \in$ *space* (*prob-algebra L*)
and $x \in$ *space K*
shows *return K x* $\bigotimes_M$ *N* = *N* $\gg=$ (λy. *return* (*return K x* $\bigotimes_M$ *N*) (*x*, *y*))
⟨*proof*⟩

4.3 intgrability for pointwise multiplication of funcitons

definition *ess-bounded* :: *'a measure* $\Rightarrow$ (*'a* $\Rightarrow$ *'b* :: {*banach*, *second-countable-topology*}) $\Rightarrow$ *bool* **where**
ess-bounded M f $\equiv$ ((*f* $\in$ *borel-measurable M*)$\wedge$($\exists$ *r* :: *real*. *AE x in M*. *norm*(*f x*) $\leq$ *r*))

lemma *integrable-mult-ess-bounded-integrable*[*simp*]:
fixes *f* :: *-*$\Rightarrow$ *'b* :: {*banach*, *second-countable-topology*,*real-normed-algebra*}
assumes *ess-bounded M f*
and *integrable M g*
shows *integrable M* (λ *x*. *g x* $*$ *f x*)
⟨*proof*⟩

lemma *integrable-mult-integrable-ess-bounded*[*simp*]:
fixes *f* :: *-*$\Rightarrow$ *'b* :: {*banach*, *second-countable-topology*,*real-normed-algebra*}
assumes *ess-bounded M f* **and** *integrable M g*
shows *integrable M* (λ *x*. *f x* $*$ *g x*)
⟨*proof*⟩

lemma *ess-bounded-const-real*[*simp*,*intro*]:
shows *ess-bounded M* (λ *x*. *r* :: *real*)
⟨*proof*⟩

lemma
fixes *f g* :: *-* $\Rightarrow$ *'b* :: {*banach*, *second-countable-topology*, *linorder-topology*}

assumes *ess-bounded M f* **and** *ess-bounded M g*
shows *ess-bounded-max-real*[*simp*,*intro*]: *ess-bounded M* (λ *x*. (*max* (*f x*) (*g x*)))
and *ess-bounded-min-real*[*simp*,*intro*]: *ess-bounded M* (λ *x*. (*min* (*f x*) (*g x*)))
⟨*proof*⟩

lemma
fixes *f g* :: -⇒ ′*b* :: {*banach*, *second-countable-topology*,*real-normed-algebra*}
assumes *ess-bounded M f*
and *ess-bounded M g*
shows *ess-bounded-add*[*simp*,*intro*]: *ess-bounded M* (λ *x*. *f x* + *g x*)
and *ess-bounded-diff*[*simp*,*intro*]: *ess-bounded M* (λ *x*. *f x* − *g x*)
and *ess-bounded-mult*[*simp*,*intro*]: *ess-bounded M* (λ *x*. *f x* ∗ *g x*)
⟨*proof*⟩

lemma *integrable-ess-bounded-finite-measure*[*simp*]:
assumes *finite-measure M*
and *ess-bounded M f*
shows *integrable M f*
⟨*proof*⟩

lemma *indicat-real-ess-bounded*[*simp*,*intro*]:
assumes *A* ∈ *sets M*
shows *ess-bounded M* (*indicat-real A*)
⟨*proof*⟩

lemma *indicat-real-integrable-finite-measure*[*simp*,*intro*]:
assumes *finite-measure M* **and** *A* ∈ *sets M*
shows *integrable M* (*indicat-real A*)
⟨*proof*⟩

lemma *probability-kernel-evaluation-ess-bounded*:
assumes *f* ∈ *measurable L* (*prob-algebra M*) **and** *A* ∈ *sets M*
shows *ess-bounded L* (λ *x*. *measure* (*f x*) *A*)
⟨*proof*⟩

lemma *probability-kernel-evaluation-integrable*[*simp*,*intro*]:
assumes *finite-measure L*
and *f* ∈ *measurable L* (*prob-algebra M*)
and *A* ∈ *sets M*
shows *integrable L* (λ *x*. *measure* (*f x*) *A*)
⟨*proof*⟩

4.4 real-valued bounded density functions of finite measures

definition *real-RN-deriv* :: ′*a measure* ⇒ ′*a measure* ⇒ ′*a* ⇒ *real*
where

real-RN-deriv L K =
  (*if* $\exists$ *k*. (*k* $\in$ *borel-measurable L*)
   $\wedge$ (*density L* (*ennreal o k*) = *K*)
   $\wedge$ (*AE x in K*. *0* < *k x*) $\wedge$ ($\forall$ *x*. *0* $\leq$ *k x*)
  then SOME k. ((*k* $\in$ *borel-measurable L*)
  $\wedge$ (*density L* (*ennreal o k*) = *K*)
  $\wedge$ (*AE x in K*. *0* < *k x*) $\wedge$ ($\forall$ *x*. *0* $\leq$ *k x*))
  else (λ-. *0*))

lemma *real-RN-deriv-I*[*intro*]:
assumes *k* $\in$ *borel-measurable L* $\wedge$ *density L* (*ennreal* $\circ$ *k*) = *K* $\wedge$ (*AE x in K*. *0* < *k x*) $\wedge$ ($\forall$ *x*. *0* $\leq$ *k x*)
shows *density L* (*ennreal o* (*real-RN-deriv L K*)) = *K*
and (*AE x in K*. *0* < *real-RN-deriv L K x*)
⟨*proof*⟩

lemma *real-RN-deriv-density*[*simp*]:
assumes *sigma-finite-measure L*
and *absolutely-continuous L K*
and *finite-measure K*
and *sets K* = *sets L*
shows *density L* (*ennreal o real-RN-deriv L K*) = *K*
⟨*proof*⟩

lemma *real-RN-deriv-positive-AE*[*simp*,*intro*]:
assumes *sigma-finite-measure L*
and *absolutely-continuous L K*
and *finite-measure K*
and *sets K* = *sets L*
shows *AE x in K*. *0* < *real-RN-deriv L K x*
⟨*proof*⟩

lemma *borel-measurable-real-RN-deriv*[*measurable*]:
shows *real-RN-deriv L K* $\in$ *borel-measurable L*
⟨*proof*⟩

lemma *real-RN-deriv-nonegative*[*simp*,*intro*]:
shows $\forall$ *x*. *0* $\leq$ *real-RN-deriv L K x*
⟨*proof*⟩

lemma *real-RN-deriv-equals-RN-deriv-AE*:
assumes *sigma-finite-measure L*
and *absolutely-continuous L K*
and *finite-measure K*
and *sets K* = *sets L*
shows *AE x in L*. (*ennreal o real-RN-deriv L K*) *x* = (*RN-deriv L K*) *x*
⟨*proof*⟩

lemma *real-RN-deriv-equals-RN-deriv-AE2*:
assumes *sigma-finite-measure L*
and *absolutely-continuous L K*
and *finite-measure K*
and *sets K = sets L*
shows *AE x in L. real-RN-deriv L K x = enn2real((RN-deriv L K) x)*
⟨*proof*⟩

lemma *real-RN-deriv-bind*[*simp*]:
assumes *sigma-finite-measure L*
and *absolutely-continuous L K*
and $K \in$ *space* (*prob-algebra L*)
and $f \in$ *measurable L* (*prob-algebra M*)
and $A \in$ (*sets M*)
shows *measure* ($K \gg= f$) $A = \int$ *x*. (*real-RN-deriv L K x*) $*$ (*measure* (*f x*) *A*) ∂L
⟨*proof*⟩

4.5 Locale for pairs of probability measures to compare.

locale *comparable-probability-measures* =
fixes *L M N* :: *'a measure*
assumes *M*: $M \in$ *space* (*prob-algebra L*)
and *N*: $N \in$ *space* (*prob-algebra L*)
begin

lemma *prob-spaceM*[*simp*,*intro*]: *prob-space M*
⟨*proof*⟩
lemma *prob-spaceN*[*simp*,*intro*]: *prob-space N*
⟨*proof*⟩
lemma *spaceM*[*simp*]: *sets M = sets L*
⟨*proof*⟩
lemma *spaceN*[*simp*]: *sets N = sets L*
⟨*proof*⟩
lemma *finM*[*simp*,*intro*]: *finite-measure M*
⟨*proof*⟩
lemma *finN*[*simp*,*intro*]: *finite-measure N*
⟨*proof*⟩
lemma *spaceML*[*simp*]: *space M = space L*
⟨*proof*⟩
lemma *spaceNL*[*simp*]: *space N = space L*
⟨*proof*⟩
lemma *McontMN*[*simp*,*intro*]: *absolutely-continuous* (*sum-measure M N*) *M*
⟨*proof*⟩

lemma *NcontMN*[*simp*,*intro*]: *absolutely-continuous* (*sum-measure M N*) *N*
⟨*proof*⟩
lemma *MNfinite*[*simp*,*intro*]: *finite-measure* (*sum-measure M N*)
⟨*proof*⟩
lemma *MNsfinite*[*simp*,*intro*]: *sigma-finite-measure* (*sum-measure M N*)
⟨*proof*⟩
lemma *setMN-L*[*simp*]: *sets* (*sum-measure M N*) = *sets L*
⟨*proof*⟩
lemma *spaceMN-L*[*simp*]: *space* (*sum-measure M N*) = *space L*
⟨*proof*⟩
lemma *spaceMN-M*: *space* (*sum-measure M N*) = *space M*
⟨*proof*⟩
lemma *setMN-M*: *sets* (*sum-measure M N*) = *sets M*
⟨*proof*⟩
lemma *spaceMN-N*: *space* (*sum-measure M N*) = *space N*
⟨*proof*⟩
lemma *setMN-N*: *sets* (*sum-measure M N*) = *sets N*
⟨*proof*⟩
lemma *space-sumM*[*simp*]: *M* ∈ *space* (*prob-algebra* (*sum-measure M N*))
⟨*proof*⟩
lemma *space-sumN*[*simp*]: *N* ∈ *space* (*prob-algebra* (*sum-measure M N*))
⟨*proof*⟩

definition *dM* = *real-RN-deriv* (*sum-measure M N*) *M*

lemma
shows *borel-measurable-dM*[*measurable*]: *dM* ∈ *borel-measurable* (*sum-measure M N*)
and *dM-positive-AE*[*simp*]: *AE x in M. 0* < *dM x*
and *dM-density* [*simp*]: *density* (*sum-measure M N*) (*ennreal o dM*) = *M*
and *dM-nonnegative*[*simp*]: (∀ *x. 0* ≤ *dM x*)
⟨*proof*⟩

lemma *dM-RN-deriv-AE*:
shows *AE x in sum-measure M N. dM x* = *enn2real* (*RN-deriv* (*sum-measure M N*) *M x*)
⟨*proof*⟩

lemma *dM-less-1-AE*:
shows *AE x in* (*sum-measure M N*). *dM x* ≤ *1*
⟨*proof*⟩

lemma *dM-bounded*:
shows *AE x in* (*sum-measure M N*). $|dM\ x| \leq 1$
⟨*proof*⟩

lemma *dM-boundes2*[*simp*,*intro*]:
shows *ess-bounded* (*sum-measure M N*) *dM*
⟨*proof*⟩

lemma *dM-integrable*[*simp*,*intro*]:
shows *integrable*(*sum-measure M N*) *dM*
⟨*proof*⟩

lemma *dM-bind-integral*[*simp*]:
assumes $f \in$ *measurable* (*sum-measure M N*) (*prob-algebra K*) $A \in$ (*sets K*)
shows *measure* ($M \gg= f$) $A = \int x.$ (*dM x*) $*$ (*measure* (*f x*) *A*) ∂ (*sum-measure M N*)
⟨*proof*⟩

definition *dN* = *real-RN-deriv* (*sum-measure M N*) *N*

lemma
shows *borel-measurable-dN*[*measurable*]: $dN \in$ *borel-measurable* (*sum-measure M N*)
and *dN-positive-AE*[*simp*]: *AE x in N*. $0 < dN\ x$
and *dN-density*[*simp*]: *density* (*sum-measure M N*) (*ennreal o dN*) = *N*
and *dN-nonnegative*[*simp*]: ($\forall\ x.\ 0 \leq dN\ x$)
⟨*proof*⟩

lemma *dN-less-1-AE*:
shows *AE x in* (*sum-measure M N*). $dN\ x \leq 1$
⟨*proof*⟩

lemma *dN-bounded*:
shows *AE x in* (*sum-measure M N*). $|dN\ x| \leq 1$
⟨*proof*⟩

lemma *dN-boundes2*[*simp*,*intro*]:
shows *ess-bounded* (*sum-measure M N*) *dN*
⟨*proof*⟩

lemma *dN-integrable*[*simp*,*intro*]:
shows *integrable*(*sum-measure M N*) *dN*
⟨*proof*⟩

lemma *dN-bind-integral*[*simp*]:
assumes $f \in$ *measurable* (*sum-measure M N*) (*prob-algebra K*) **and** $A \in$ (*sets K*)

shows *measure* $(N \gg= f)$ $A = \int$ $x.$ $(dN\ x) * (measure\ (f\ x)\ A)\ \partial$ $(sum\text{-}measure\ M\ N)$
⟨*proof*⟩

lemma *dM-dN-partition-1-AE*:
shows *AE x in sum-measure M N*. $dM\ x + dN\ x = 1$
⟨*proof*⟩

end

end

theory *Differential-Privacy-Divergence*
imports *Comparable-Probability-Measures*
begin

5 Divergence for Differential Privacy

First, we introduce the divergence for differential privacy.

definition *DP-divergence* :: $'a$ *measure* $\Rightarrow$ $'a$ *measure* $\Rightarrow$ *real* $\Rightarrow$ *ereal*
where
DP-divergence $M\ N\ \varepsilon = Sup\ \{ereal(measure\ M\ A - (exp\ \varepsilon) * measure\ N\ A) \mid A::'a\ set.\ A \in (sets\ M)\}$

lemma *DP-divergence-SUP*:
shows *DP-divergence* $M\ N\ \varepsilon = (\bigsqcup\ (A::'a\ set) \in (sets\ M).\ ereal(measure\ M\ A - (exp\ \varepsilon) * measure\ N\ A))$
⟨*proof*⟩

5.1 Basic Properties

5.1.1 Non-negativity

lemma *DP-divergence-nonnegativity*:
shows $0 \leq$ *DP-divergence* $M\ N\ \varepsilon$
⟨*proof*⟩

5.1.2 Graded predicate

lemma *DP-divergence-forall*:
shows $(\forall\ A \in (sets\ M).\ (measure\ M\ A - (exp\ \varepsilon) * measure\ N\ A \leq (\delta :: real)))$
$\longleftrightarrow$ *DP-divergence* $M\ N\ \varepsilon \leq (\delta :: real)$
⟨*proof*⟩

lemma *DP-divergence-leE*:
assumes *DP-divergence* $M\ N\ \varepsilon \leq (\delta :: real)$
shows *measure* $M\ A \leq (exp\ \varepsilon) * measure\ N\ A + (\delta :: real)$

⟨*proof*⟩

lemma *DP-divergence-leI*:
 assumes $\bigwedge A.\ A \in (sets\ M) \Longrightarrow measure\ M\ A \leq (exp\ \varepsilon) * measure\ N\ A + (\delta :: real)$
 shows $DP\text{-}divergence\ M\ N\ \varepsilon \leq (\delta :: real)$
 ⟨*proof*⟩

5.1.3 (*0*,*0*)-DP means the equality of distributions

lemma *prob-measure-eqI-le*:
 assumes M: $M \in space\ (prob\text{-}algebra\ L)$
 and N: $N \in space\ (prob\text{-}algebra\ L)$
 and le:$\forall\ A \in sets\ M.\ emeasure\ M\ A \leq emeasure\ N\ A$
 shows $M = N$
⟨*proof*⟩

lemma *DP-divergence-zero*:
 assumes M: $M \in space\ (prob\text{-}algebra\ L)$
 and N: $N \in space\ (prob\text{-}algebra\ L)$
 and $DP0$:$DP\text{-}divergence\ M\ N\ (0::real) \leq (0::real)$
 shows $M = N$
⟨*proof*⟩

5.1.4 Conversion from pointwise DP [4]

lemma *DP-divergence-pointwise*:
 assumes M: $M \in space\ (prob\text{-}algebra\ L)$
 and N: $N \in space\ (prob\text{-}algebra\ L)$
 and C: $C \in sets\ M$
 and $\forall\ A{\in}sets\ M.\ A \subseteq C \longrightarrow measure\ M\ A \leq exp\ \varepsilon * measure\ N\ A$
 and $1 - \delta \leq measure\ M\ C$
 shows $DP\text{-}divergence\ M\ N\ (\varepsilon::real) \leq \delta$
⟨*proof*⟩

5.1.5 (Reverse-) Monotonicity for ε

lemma *DP-divergence-monotonicity*:
 assumes $\varepsilon 1 \leq \varepsilon 2$
 shows $DP\text{-}divergence\ M\ N\ \varepsilon 2 \leq DP\text{-}divergence\ M\ N\ \varepsilon 1$
⟨*proof*⟩

corollary *DP-divergence-monotonicity′*:
 assumes M: $M \in space\ (prob\text{-}algebra\ L)$
 and N: $N \in space\ (prob\text{-}algebra\ L)$
 and $\varepsilon 1 \leq \varepsilon 2$ **and** $\delta 1 \leq \delta 2$
 shows $DP\text{-}divergence\ M\ N\ \varepsilon 1 \leq \delta 1 \Longrightarrow DP\text{-}divergence\ M\ N\ \varepsilon 2 \leq \delta 2$
 ⟨*proof*⟩

5.1.6 Reflexivity

lemma *DP-divergence-reflexivity*:
shows *DP-divergence M M 0 = 0*
⟨*proof*⟩

corollary *DP-divergence-reflexivity′*:
assumes $0 \leq \varepsilon$
shows *DP-divergence M M* $\varepsilon = 0$
⟨*proof*⟩

5.1.7 Transitivity

lemma *DP-divergence-transitivity*:
assumes *DP1*: *DP-divergence M1 M2* $\varepsilon 1 \leq 0$
and *DP2*: *DP-divergence M2 M3* $\varepsilon 2 \leq 0$
shows *DP-divergence M1 M3* $(\varepsilon 1+\varepsilon 2) \leq 0$
⟨*proof*⟩

5.1.8 Composability

proposition *DP-divergence-composability*:
assumes *M*: $M \in$ *space* (*prob-algebra L*)
and *N*: $N \in$ *space* (*prob-algebra L*)
and *f*: $f \in L \rightarrow_M$ *prob-algebra K*
and *g*: $g \in L \rightarrow_M$ *prob-algebra K*
and *div1*: *DP-divergence M N* $\varepsilon 1 \leq (\delta 1$ :: *real*)
and *div2*: $\forall x \in$ (*space L*). *DP-divergence* (*f x*) (*g x*) $\varepsilon 2 \leq (\delta 2$ :: *real*)
and $0 \leq \varepsilon 1$ **and** $0 \leq \varepsilon 2$
shows *DP-divergence* ($M \gg= f$) ($N \gg= g$) $(\varepsilon 1 + \varepsilon 2) \leq \delta 1 + \delta 2$
⟨*proof*⟩

5.1.9 Post-processing inequality

corollary *DP-divergence-postprocessing*:
assumes *M*: $M \in$ *space* (*prob-algebra L*)
and *N*: $N \in$ *space* (*prob-algebra L*)
and *f*: $f \in L \rightarrow_M$ (*prob-algebra K*)
and *div1*: *DP-divergence M N* $\varepsilon 1 \leq (\delta 1$ :: *real*)
and $0 \leq \varepsilon 1$
shows *DP-divergence* (*bind M f*) (*bind N f*) $\varepsilon 1 \leq \delta 1$
⟨*proof*⟩

5.1.10 Law for the strength of the Giry monad

lemma *DP-divergence-strength*:
assumes $M \in$ *space* (*prob-algebra L*)
and $N \in$ *space* (*prob-algebra L*)
and *DP-divergence M N* $\varepsilon 1 \leq (\delta 1$ :: *real*)

and $0 \leq \varepsilon 1$
and $x{:}x \in$ *space K*
shows *DP-divergence* ((*return K x*) $\bigotimes_M$ *M*) ((*return K x*) $\bigotimes_M$ *N*) $\varepsilon 1 \leq \delta 1$
⟨*proof*⟩

5.1.11 Additivity: law for the double-strength of the Giry monad

lemma *DP-divergence-additivity*:
assumes *M*: $M \in$ *space* (*prob-algebra L*)
and *N*: $N \in$ *space* (*prob-algebra L*)
and *M2*: $M2 \in$ *space* (*prob-algebra K*)
and *N2*: $N2 \in$ *space* (*prob-algebra K*)
and *div1*: *DP-divergence M N* $\varepsilon 1 \leq (\delta 1 :: real)$
and *div2'*: *DP-divergence M2 N2* $\varepsilon 2 \leq (\delta 2 :: real)$
and $0 \leq \varepsilon 1$ **and** $0 \leq \varepsilon 2$
shows *DP-divergence* ($M \bigotimes_M M2$) ($N \bigotimes_M N2$) ($\varepsilon 1 + \varepsilon 2$) $\leq \delta 1 + \delta 2$
⟨*proof*⟩

5.2 Hypotehsis testing interpretation

5.2.1 Privacy region

definition *DP-region-one-side* :: *real* ⇒ *real* ⇒ (*real* × *real*) *set* **where**
DP-region-one-side ε δ = $\{(x{::}real, y{::}real).\ (x - exp(\varepsilon) * y \leq \delta) \wedge 0 \leq x \wedge x \leq 1 \wedge 0 \leq y \wedge y \leq 1\}$

lemma *DP-divergence-region*:
assumes *M*: $M \in$ *space* (*prob-algebra L*)
and *N*: $N \in$ *space* (*prob-algebra L*)
shows (∀ $A \in$ (*sets M*). ((*measure M A*, *measure N A*) ∈ *DP-region-one-side* ε δ)) ⟷ *DP-divergence M N* $\varepsilon \leq (\delta :: real)$
⟨*proof*⟩

5.2.2 2-generatedness of *DP-divergence* [1]

lemma *DP-divergence-2-generated-deterministic*:
assumes *M*: $M \in$ *space* (*prob-algebra L*)
and *N*: $N \in$ *space* (*prob-algebra L*)
and $0 \leq \varepsilon$
shows *DP-divergence M N* ε = ($\bigsqcup$ $f \in L \rightarrow_M$ (*count-space* (*UNIV* :: *bool set*)). *DP-divergence* (*distr M* (*count-space UNIV*) *f*) (*distr N* (*count-space UNIV*) *f*) ε)
⟨*proof*⟩

lemma *DP-divergence-2-generated*:
assumes *M*: $M \in$ *space* (*prob-algebra L*)
and *N*: $N \in$ *space* (*prob-algebra L*)

assumes $0 \leq \varepsilon$
shows *DP-divergence* $M\ N\ \varepsilon = (\bigsqcup f \in L \rightarrow_M (\mathit{prob\text{-}algebra}\ (\mathit{count\text{-}space}\ (\mathit{UNIV} :: \mathit{bool\ set}))).\ \mathit{DP\text{-}divergence}\ (M \gg= f)\ (N \gg= f)\ \varepsilon)$
⟨*proof*⟩

5.3 real version of *DP-divergence*

5.3.1 finiteness

definition *DP-divergence-real* :: $'a\ \mathit{measure} \Rightarrow\ 'a\ \mathit{measure} \Rightarrow \mathit{real} \Rightarrow \mathit{real}$ **where**
$\mathit{DP\text{-}divergence\text{-}real}\ M\ N\ \varepsilon = \mathit{Sup}\ \{(\mathit{measure}\ M\ A - (\mathit{exp}\ \varepsilon) * \mathit{measure}\ N\ A) \mid A{::}'a\ \mathit{set}.\ A \in (\mathit{sets}\ M)\}$

lemma *DP-divergence-real-SUP*:
shows $\mathit{DP\text{-}divergence\text{-}real}\ M\ N\ \varepsilon = (\bigsqcup (A{::}'a\ \mathit{set}) \in (\mathit{sets}\ M).\ (\mathit{measure}\ M\ A - (\mathit{exp}\ \varepsilon) * \mathit{measure}\ N\ A))$
⟨*proof*⟩

lemma *DP-divergence-real-leI*:
assumes $\bigwedge A.\ A \in (\mathit{sets}\ M) \Longrightarrow \mathit{measure}\ M\ A \leq (\mathit{exp}\ \varepsilon) * \mathit{measure}\ N\ A + (\delta :: \mathit{real})$
shows $\mathit{DP\text{-}divergence\text{-}real}\ M\ N\ \varepsilon \leq (\delta :: \mathit{real})$
⟨*proof*⟩

5.3.2 real version of *DP-divergence*

If M and N are finite then *DP-divergence* and the following version *DP-divergence-real* are the same. However, if M and N are not finite, *DP-divergence-real* $M\ N\ \varepsilon$ is not well defined. Hence, the latter needs extra assumptions for the finiteness of measures M and N. For example the following lemmas need a kind of finiteness of$M$ and N.

lemma *DP-divergence-is-real*:
assumes M: $M \in \mathit{space}\ (\mathit{prob\text{-}algebra}\ L)$
and N: $N \in \mathit{space}\ (\mathit{prob\text{-}algebra}\ L)$
shows $\mathit{DP\text{-}divergence}\ M\ N\ \varepsilon = \mathit{DP\text{-}divergence\text{-}real}\ M\ N\ \varepsilon$
⟨*proof*⟩

corollary *DP-divergence-real-forall*:
assumes M: $M \in \mathit{space}\ (\mathit{prob\text{-}algebra}\ L)$
and N: $N \in \mathit{space}\ (\mathit{prob\text{-}algebra}\ L)$
shows $(\forall\ A \in (\mathit{sets}\ M).\ (\mathit{measure}\ M\ A - (\mathit{exp}\ \varepsilon) * \mathit{measure}\ N\ A \leq (\delta :: \mathit{real}))) \longleftrightarrow \mathit{DP\text{-}divergence\text{-}real}\ M\ N\ \varepsilon \leq (\delta :: \mathit{real})$
⟨*proof*⟩

corollary *DP-divergence-real-inequality1*:
assumes M: $M \in \mathit{space}\ (\mathit{prob\text{-}algebra}\ L)$
and N: $N \in \mathit{space}\ (\mathit{prob\text{-}algebra}\ L)$

and $A \in (sets\ M)$ **and** $DP\text{-}divergence\text{-}real\ M\ N\ \varepsilon \leq (\delta :: real)$
shows $measure\ M\ A \leq (exp\ \varepsilon) * measure\ N\ A + (\delta :: real)$
⟨*proof*⟩

end

theory *Differential-Privacy-Randomized-Response*
imports*Differential-Privacy-Divergence*
begin

6 Randomized Response Mechanism

definition *RR-mechanism* :: *real* ⇒ *bool* ⇒ *bool pmf*
where *RR-mechanism* ε x =
(*if* x *then* (*bernoulli-pmf* ((*exp* ε) / (*1* + *exp* ε)))
else (*bernoulli-pmf* (*1* / (*1* + *exp* ε))))

lemma *meausrable-RR-mechanism*[*measurable*]:
shows *measure-pmf o* (*RR-mechanism* ε) ∈ *measurable* (*count-space UNIV*) (*prob-algebra* (*count-space UNIV*))
⟨*proof*⟩

lemma *RR-probability-flip1*:
fixes ε :: *real*
assumes $\varepsilon \geq 0$
shows $1 - 1 / (1 + exp\ \varepsilon) = exp\ \varepsilon / (1 + exp\ \varepsilon)$
⟨*proof*⟩

lemma *RR-probability-flip2*:
fixes ε :: *real*
assumes $\varepsilon \geq 0$
shows $1 - exp\ \varepsilon / (1 + exp\ \varepsilon) = 1 / (1 + exp\ \varepsilon)$
⟨*proof*⟩

lemma *RR-mechanism-flip*:
assumes $\varepsilon \geq 0$
shows *bind-pmf* (*RR-mechanism* ε x) (λ b :: *bool*. *return-pmf* (¬ b)) = (*RR-mechanism* ε (¬ x))
⟨*proof*⟩

proposition *DP-RR-mechanism*:
fixes ε :: *real* **and** x y :: *bool*
assumes $\varepsilon > 0$
shows *DP-divergence* (*RR-mechanism* ε x) (*RR-mechanism* ε y) ε ≤ (*0* :: *real*)
⟨*proof*⟩

end

theory *Differential-Privacy-Laplace-Mechanism*
imports*Differential-Privacy-Divergence*
Laplace-Distribution
begin

7 Laplace mechanism

7.1 Laplace mechanism as a noise-adding mechanism

7.1.1 Laplace distribution with scale b and average μ

definition *Lap-dist* :: *real* ⇒ *real* ⇒ *real measure* **where**
Lap-dist b μ = (*if* $b \leq 0$ *then return borel* μ *else density lborel* (*laplace-density* b μ))

lemma shows *prob-space-Lap-dist*[*simp*,*intro*]: *prob-space* (*Lap-dist* b x)
and *subprob-space-Lap-dist*[*simp*,*intro*]: *subprob-space* (*Lap-dist* b x)
and *sets-Lap-dist*[*measurable-cong*]: *sets* (*Lap-dist* b x) = *sets borel*
and *space-Lap-dist*: *space* (*Lap-dist* ε x) = *UNIV*
⟨*proof*⟩

lemma *measurable-Lap-dist*[*measurable*]:
shows *Lap-dist* $b \in$ *borel* $\rightarrow_M$ *prob-algebra borel*
⟨*proof*⟩

lemma *Lap-dist-space-prob-algebra*[*simp*,*measurable*]:
shows (*Lap-dist* b x) $\in$ *space* (*prob-algebra borel*)
⟨*proof*⟩

lemma *Lap-dist-space-subprob-algebra*[*simp*,*measurable*]:
shows (*Lap-dist* b x) $\in$ *space* (*subprob-algebra borel*)
⟨*proof*⟩

7.1.2 The Laplace distribution with scale b and average *0*

definition *Lap-dist0* $b \equiv$ *Lap-dist* b *0*

Actually *Lap-dist* b is a noise-addition of Laplace distribution with scale b and average *0*.

lemma shows *prob-space-Lap-dist0*[*simp*,*intro*,*measurable*]: *prob-space* (*Lap-dist0* b)

and *subprob-space-Lap-dist0*[*simp*,*intro*,*measurable*]: *subprob-space* (*Lap-dist0 b*)
and *sets-Lap-dist0*[*measurable-cong*]: *sets* (*Lap-dist0 b*) = *sets borel*
and *space-Lap-dist0*: *space* (*Lap-dist0 b*) = *UNIV*
and *Lap-dist0-space-prob-algebra*[*simp*,*measurable*]: (*Lap-dist0 b*) ∈ *space* (*prob-algebra borel*)
and *Lap-dist0-space-subprob-algebra*[*simp*,*measurable*]: (*Lap-dist0 b*) ∈ *space* (*subprob-algebra borel*)
⟨*proof*⟩

lemma *Lap-dist-def2*:
shows (*Lap-dist b x*) = *do*{*r* ← *Lap-dist0 b*; *return borel* (*x* + *r*)}
⟨*proof*⟩

corollary *Lap-dist-shift*:
shows (*Lap-dist b* (*x* + *y*)) = *do*{*r* ← *Lap-dist b x*; *return borel* (*y* + *r*)}
⟨*proof*⟩

7.1.3 Differential Privacy of Laplace noise addition

proposition *DP-divergence-Lap-dist′*:
assumes *b* > *0*
and | *x* − *y* | ≤ *r*
shows *DP-divergence* (*Lap-dist b x*) (*Lap-dist b y*) (*r* / *b*) ≤ (*0* :: *real*)
⟨*proof*⟩

corollary *DP-divergence-Lap-dist′-eps*:
assumes ε > *0*
and | *x* − *y* | ≤ *r*
shows *DP-divergence* (*Lap-dist* (*r* / ε) *x*) (*Lap-dist* (*r* / ε) *y*) ε ≤ (*0* :: *real*)
⟨*proof*⟩

corollary *DP-divergence-Lap-dist-eps*:
assumes ε > *0*
and | *x* − *y* | ≤ *1*
shows *DP-divergence* (*Lap-dist* (*1* / ε) *x*) (*Lap-dist* (*1* / ε) *y*) ε ≤ (*0* :: *real*)
⟨*proof*⟩

end

theory *Differential-Privacy-Laplace-Mechanism-Multi*
imports *List-Space/List-Space*
Differential-Privacy-Laplace-Mechanism

begin

7.2 Bundled Laplace Noise

lemma *space-listM-borel-UNIV*[*measurable*,*simp*]:
  shows *space* (*listM borel*) = *UNIV*
  ⟨*proof*⟩

context
  fixes *b*::*real*
begin

The list of Laplace distribution with scale b **and average** *0* **primrec** *Lap-dist0-list* :: *nat* ⇒ (*real list*) *measure* **where**
  Lap-dist0-list 0 = *return* (*listM borel*) [] |
  Lap-dist0-list (*Suc n*) = *do*{*x1* ← (*Lap-dist0 b*); *x2* ← (*Lap-dist0-list n*); *return* (*listM borel*) (*x1* # *x2*)}

A function adding Laplace noise to each element of a real list **primrec** *Lap-dist-list* :: *real list* ⇒ (*real list*) *measure* **where**
  Lap-dist-list [] = *return* (*listM borel*) []|
  Lap-dist-list (*x* # *xs*) = *do*{*x1* ← (*Lap-dist b x*); *x2* ← (*Lap-dist-list xs*); *return* (*listM borel*) (*x1* # *x2*)}

lemma *measurable-Lap-dist-list*[*measurable*]:
  shows *Lap-dist-list* ∈ *listM borel* $\rightarrow_M$ *prob-algebra* (*listM borel*)
⟨*proof*⟩

lemma *prob-space-Lap-dist-list*[*measurable*,*simp*]:
  shows *prob-space* (*Lap-dist-list xs*)
  ⟨*proof*⟩

lemma *Laprepsp'*[*measurable*,*simp*]:
  shows *Lap-dist-list xs* ∈ *space* (*prob-algebra* (*listM borel*))
  ⟨*proof*⟩

lemma *Laprepsp*[*measurable*,*simp*]:
  shows *Lap-dist-list xs* ∈ *space* (*subprob-algebra* (*listM borel*))
  ⟨*proof*⟩

lemma *sets-Lap-dist-list*[*measurable-cong*]:
  shows ⋀*zs*. *sets*(*Lap-dist-list zs*) = *sets*(*listM borel*)
  ⟨*proof*⟩

lemma *space-Lap-dist-list*:
  shows ⋀*zs*. *space*(*Lap-dist-list zs*) = *space* (*listM borel*)
  ⟨*proof*⟩

lemma *emeasure-Lap-dist-list-length*:

shows *emeasure* (*Lap-dist-list ys*) {*xs. length xs* = *length ys*} = *1*
⟨*proof*⟩

lemma *Lap-dist0-list-Lap-dist-list*:
shows *Lap-dist-list* (*replicate n 0*) = *Lap-dist0-list n*
⟨*proof*⟩

corollary
shows *prob-space-Lap-dist0-list*[*measurable*,*simp*]: *prob-space* (*Lap-dist0-list n*)
and *prob-algebra-Lap-dist0-list*[*measurable*,*simp*]: *Lap-dist0-list n* ∈ *space* (*prob-algebra* (*listM borel*))
and *subprob-algebra-Lap-dist0-list*[*measurable*,*simp*]: *Lap-dist0-list n* ∈ *space* (*subprob-algebra* (*listM borel*))
and *sets-Lap-dist0-list*[*measurable-cong*]:*sets*(*Lap-dist0-list n*) = *sets*(*listM borel*)
⟨*proof*⟩

corollary *emeasure-Lap-dist0-list-length*:
shows *emeasure* (*Lap-dist0-list n*) {*xs. length xs* = *n*} = *1*
⟨*proof*⟩

lemma *Lap-dist-list-def2*:
shows *Lap-dist-list xs* = *do*{*ys* ← (*Lap-dist0-list* (*length xs*)); *return* (*listM borel*) (*map2* (+) *xs ys*)}
⟨*proof*⟩

end

lemma *sum-list-cons*:
fixes *xs ys* :: ′*a list* **and** *n* :: *nat*
assumes *length xs* = *n* **and** *length ys* = *n*
shows (∑ *i*∈{*1..Suc n*}. *d* (*nth* (*x* # *xs*) (*i*−*1*)) (*nth* (*y* # *ys*) (*i*−*1*))) = *d x y* + (∑ *i*∈{*1..n*}. *d* (*nth xs* (*i*−*1*)) (*nth ys* (*i*−*1*)))
⟨*proof*⟩

lemma *adj-Cons-partition*:
fixes *xs ys* :: *real list* **and** *n* :: *nat* **and** *r* :: *real*
assumes *length xs* = *n* **and** *length ys* = *n*
and *adj*: (∑ *i*∈{*1..Suc n*}. | *nth* (*x* # *xs*) (*i*−*1*) − *nth* (*y* # *ys*) (*i*−*1*) |) ≤ *r*
and *posr*: *r* ≥ *0*
obtains *r1 r2*::*real* **where** *0* ≤ *r1* **and** *0* ≤ *r2* **and** |*x* − *y*| ≤ *r1* **and** (∑ *i* = *1..n*. |*xs* ! (*i* − *1*) − *ys* ! (*i* − *1*)|) ≤ *r2* **and** *r1* + *r2* ≤ *r*
⟨*proof*⟩

lemma *adj-list-0*:
fixes *xs ys* :: *real list* **and** *n* :: *nat*

shows *length xs* $=$ *n* $\Longrightarrow$ *length ys* $=$ *n* $\Longrightarrow$ ($\sum$ *i*$\in${*1..n*}. | *nth xs* (*i*$-$*1*) $-$ *nth ys* (*i*$-$*1*) |) $\leq$ *0* $\Longrightarrow$ *xs* $=$ *ys*
⟨*proof*⟩

lemma *DP-Lap-dist-list*:
fixes *xs ys* :: *real list* **and** *n* :: *nat* **and** *r* :: *real* **and** *b*::*real*
assumes *posb*: *b* $>$ (*0* :: *real*)
and *length xs* $=$ *n* **and** *length ys* $=$ *n*
and *adj*: ($\sum$ *i*$\in${*1..n*}. | *nth xs* (*i*$-$*1*) $-$ *nth ys* (*i*$-$*1*) |) $\leq$ *r*
and *posr*: *r* $\geq$ *0*
shows *DP-divergence* (*Lap-dist-list b xs*) (*Lap-dist-list b ys*) (*r* / *b*) $\leq$ *0*
⟨*proof*⟩

corollary *DP-Lap-dist-list-eps*:
fixes *xs ys* :: *real list* **and** *n* :: *nat* **and** *r* :: *real*
assumes *pose*: ε $>$ (*0* :: *real*)
and *length xs* $=$ *n* **and** *length ys* $=$ *n*
and *adj*: ($\sum$ *i*$\in${*1..n*}. | *nth xs* (*i*$-$*1*) $-$ *nth ys* (*i*$-$*1*) |) $\leq$ *r*
and *posr*: *r* $\geq$ *0*
shows *DP-divergence* (*Lap-dist-list* (*r* / ε) *xs*) (*Lap-dist-list* (*r* / ε) *ys*) ε $\leq$ *0*
⟨*proof*⟩

hide-fact(**open**) *adj-Cons-partition sum-list-cons adj-list-0*

end

theory *Differential-Privacy-Standard*
imports
Differential-Privacy-Laplace-Mechanism-Multi
L1-norm-list
HOL.Transitive-Closure
begin

8 Formalization of Differential privacy

AFDP in this section means the textbook "The Algorithmic Foundations of Differential Privacy"written by Cynthia Dwork and Aaron Roth.

8.1 Predicate of Differential privacy

The inequality for DP (cf. [Def 2.4, AFDP]) **definition**
DP-inequality:: *'a measure* $\Rightarrow$ *'a measure* $\Rightarrow$ *real* $\Rightarrow$ *real* $\Rightarrow$ *bool***where**
DP-inequality M N ε δ $\equiv$ ($\forall$ *A* $\in$ *sets M*. *measure M A* $\leq$ (*exp* ε) $*$ *measure N A* $+$ δ)

The divergence for DP (cf. [Barthe & Olmedo, ICALP 2013]) **proposition** *DP-inequality-cong-DP-divergence*:
shows *DP-inequality M N* ε δ $\longleftrightarrow$ *DP-divergence M N* $\varepsilon \leq \delta$
$\langle proof \rangle$

corollary *DP-inequality-zero*:
assumes $M \in$ *space* (*prob-algebra L*)
and $N \in$ *space* (*prob-algebra L*)
and *DP-inequality M N 0 0*
shows $M = N$
$\langle proof \rangle$

Definition of the standard differential privacy with adjacency, and its basic properties We first we abstract the domain of database and the adjacency relations. later we instantiate them according to the textbook.

definition *differential-privacy* :: ($'a \Rightarrow 'b$ *measure*) $\Rightarrow$ ($'a$ *rel*) $\Rightarrow$ *real* $\Rightarrow$ *real* $\Rightarrow$ *bool* **where**
differential-privacy M adj ε δ $\equiv$ $\forall$(*d1*,*d2*)$\in$*adj*. *DP-inequality* (*M d1*) (*M d2*) ε δ $\wedge$ *DP-inequality* (*M d2*) (*M d1*) ε δ

lemma *differential-privacy-adj-sym*:
assumes *sym adj*
shows *differential-privacy M adj* ε δ $\longleftrightarrow$ ($\forall$ (*d1*,*d2*) $\in$ *adj*. *DP-inequality* (*M d1*) (*M d2*) ε δ)
$\langle proof \rangle$

lemma *differential-privacy-symmetrize*:
assumes *differential-privacy M adj* ε δ
shows *differential-privacy M* (*adj* $\cup$ adj^{-1}) ε δ
$\langle proof \rangle$

lemma *differential-privacy-restrict*:
assumes *differential-privacy M adj* ε δ
and $adj' \subseteq adj$
shows *differential-privacy M* adj' ε δ
$\langle proof \rangle$

Lemmas for group privacy (cf. [Theorem 2.2, AFDP])
lemma *pure-differential-privacy-comp*:
assumes $adj1 \subseteq$ (*space X*) $\times$ (*space X*)
and $adj2 \subseteq$ (*space X*) $\times$ (*space X*)
and *differential-privacy M adj1* $\varepsilon 1$ *0*
and *differential-privacy M adj2* $\varepsilon 2$ *0*
and $M{:}M \in X \rightarrow_M$ (*prob-algebra R*)
shows *differential-privacy M* (*adj1 O adj2*) ($\varepsilon 1 + \varepsilon 2$) *0*

⟨*proof*⟩

lemma *adj-trans-k*:
assumes $adj \subseteq A \times A$
and $0 < k$
shows $(adj \frown\frown k) \subseteq A \times A$
⟨*proof*⟩

lemma *pure-differential-privacy-trans-k*:
assumes $adj \subseteq (space\ X) \times (space\ X)$
and *differential-privacy* $M\ adj\ \varepsilon\ 0$
and $M[measurable]$:$M \in X \rightarrow_M (prob\text{-}algebra\ R)$
shows *differential-privacy* $M\ (adj \frown\frown k)\ (k * \varepsilon)\ 0$
⟨*proof*⟩

The relaxation of parameters (obvious in the pencil and paper manner). **lemma** *DP-inequality-relax*:
assumes *1*: $\varepsilon \leq \varepsilon'$ **and** *2*: $\delta \leq \delta'$
and *DP*: *DP-inequality* $M\ N\ \varepsilon\ \delta$
shows *DP-inequality* $M\ N\ \varepsilon'\ \delta'$
⟨*proof*⟩

proposition *differential-privacy-relax*:
assumes *DP*:*differential-privacy* $M\ adj\ \varepsilon\ \delta$
and *1*: $\varepsilon \leq \varepsilon'$**and** *2*: $\delta \leq \delta'$
shows *differential-privacy* $M\ adj\ \varepsilon'\ \delta'$
⟨*proof*⟩

Stability for post-processing (cf. [Prop 2.1, AFDP])
proposition *differential-privacy-postprocessing*:
assumes $\varepsilon \geq 0$
and *differential-privacy* $M\ adj\ \varepsilon\ \delta$
and M: $M \in X \rightarrow_M (prob\text{-}algebra\ R)$
and f: $f \in R \rightarrow_M (prob\text{-}algebra\ R')$
and $adj \subseteq (space\ X) \times (space\ X)$
shows *differential-privacy* $(\lambda x.\ do\{y \leftarrow M\ x;\ f\ y\})\ adj\ \varepsilon\ \delta$
⟨*proof*⟩

corollary *differential-privacy-postprocessing-deterministic*:
assumes $\varepsilon \geq 0$
and *differential-privacy* $M\ adj\ \varepsilon\ \delta$
and $M[measurable]$: $M \in X \rightarrow_M (prob\text{-}algebra\ R)$
and $f[measurable]$: $f \in R \rightarrow_M R'$
and $adj \subseteq (space\ X) \times (space\ X)$
shows *differential-privacy* $(\lambda x.\ do\{y \leftarrow M\ x;\ return\ R'\ (f\ y)\ \}\ )\ adj$
$\varepsilon\ \delta$
⟨*proof*⟩

To handle the sensitivity, we prepare the conversions of adjacency relations by pre-processing.

lemma *differential-privacy-preprocessing*:
assumes $\varepsilon \geq 0$
and *differential-privacy M adj* ε δ
and *f*: $f \in X' \to_M X$
and *ftr*: $\forall (x,y) \in adj'. (f\ x, f\ y) \in adj$
and $adj \subseteq (space\ X) \times (space\ X)$
and $adj' \subseteq (space\ X') \times (space\ X')$
shows *differential-privacy* $(M\ o\ f)$ adj' ε δ
⟨*proof*⟩

"Adaptive"composition (cf. [Theorem B.1, AFDP]) proposition *differential-privacy-composition-adaptive*:
assumes $\varepsilon \geq 0$
and $\varepsilon' \geq 0$
and *M*: $M \in X \to_M (prob\text{-}algebra\ Y)$
and *DPM*: *differential-privacy M adj* ε δ
and *N*: $N \in (X \bigotimes_M Y) \to_M (prob\text{-}algebra\ Z)$
and *DPN*: $\forall\ y \in space\ Y.$ *differential-privacy* $(\lambda\ x.\ N\ (x,y))$ *adj* ε' δ'
and $adj \subseteq (space\ X) \times (space\ X)$
shows *differential-privacy* $(\lambda x.\ do\{y \leftarrow M\ x;\ N\ (x,\ y)\})$ *adj* $(\varepsilon + \varepsilon')$ $(\delta + \delta')$
⟨*proof*⟩

"Sequential"composition [Theorem 3.14, AFDP, generalized] proposition *differential-privacy-composition-pair*:
assumes $\varepsilon \geq 0$
and $\varepsilon' \geq 0$
and *DPM*: *differential-privacy M adj* ε δ
and *M*[*measurable*]: $M \in X \to_M (prob\text{-}algebra\ Y)$
and *DPN*: *differential-privacy N adj* ε' δ'
and *N*[*measurable*]: $N \in X \to_M (prob\text{-}algebra\ Z)$
and $adj \subseteq (space\ X) \times (space\ X)$
shows *differential-privacy* $(\lambda x.\ do\{y \leftarrow M\ x;\ z \leftarrow N\ x;\ return\ (Y \bigotimes_M Z)\ (y,z)\}\ )$ *adj* $(\varepsilon + \varepsilon')$ $(\delta + \delta')$
⟨*proof*⟩

Laplace mechanism (1-dimensional version) (cf. [Def. 3.3 and Thm 3.6, AFDP]) locale *Lap-Mechanism-1dim* =
fixes X::'*a measure*
and f::'$a \Rightarrow real$
and *adj*::('*a rel*)
assumes [*measurable*]: $f \in X \to_M borel$
and *adj*: $adj \subseteq (space\ X) \times (space\ X)$
begin

definition *sensitivity*:: *ereal* **where**

sensitivity = *Sup*{ *ereal* |(*f x*) − (*f y*)| | *x y*::*'a*. *x* ∈ *space X* ∧ *y* ∈ *space X* ∧ (*x*,*y*) ∈ *adj* }

definition *LapMech-1dim*::*real* ⇒ *'a* ⇒ *real measure***where**
LapMech-1dim ε x = *Lap-dist* ((*real-of-ereal sensitivity*) / *ε*) (*f x*)

lemma *measurable-LapMech-1dim*[*measurable*]:
shows *LapMech-1dim ε* ∈ *X* $\rightarrow_M$ *prob-algebra borel*
⟨*proof*⟩

lemma *LapMech-1dim-def2*:
shows *LapMech-1dim ε x* = *do*{*y* ← *Lap-dist0* ((*real-of-ereal sensitivity*) / *ε*); *return borel* (*f x* + *y*) }
⟨*proof*⟩

proposition *differential-privacy-LapMech-1dim*:
assumes *pose*: *0* < *ε* **and** *sensitivity* > *0***and** *sensitivity* < ∞
shows *differential-privacy* (*LapMech-1dim ε*) *adj ε 0*
⟨*proof*⟩

end

Laplace mechanism (generalized version) (cf. [Def. 3.3 and Thm 3.6, AFDP])

locale *Lap-Mechanism-list* =
fixes *X*::*'a measure*
and *f*::*'a* ⇒ *real list*
and *adj*::(*'a rel*)
and *m*::*nat*
assumes [*measurable*]: *f* ∈ *X* $\rightarrow_M$ (*listM borel*)
and *len*: ⋀ *x*. *x* ∈ *space X* ⟹ *length* (*f x*) = *m*
and *adj*: *adj* ⊆ (*space X*) × (*space X*)
begin

definition *sensitivity*:: *ereal* **where**
sensitivity = *Sup*{ *ereal* (∑ *i*∈{*1*..*m*}. | *nth* (*f x*) (*i*−*1*) − *nth* (*f y*) (*i*−*1*) |) | *x y*::*'a*. *x* ∈ *space X* ∧ *y* ∈ *space X* ∧ (*x*,*y*) ∈ *adj*}

definition *LapMech-list*::*real* ⇒ *'a* ⇒ (*real list*) *measure***where**
LapMech-list ε x = *Lap-dist-list* ((*real-of-ereal sensitivity*) / *ε*) (*f x*)

lemma *LapMech-list-def2*:
assumes *x* ∈ *space X*
shows *LapMech-list ε x* = *do*{ *xs* ← *Lap-dist0-list* (*real-of-ereal sensitivity* / *ε*) *m*; *return* (*listM borel*) (*map2* (+) (*f x*) *xs*)}
⟨*proof*⟩

proposition *differential-privacy-LapMech-list*:
  assumes *pose*: $\varepsilon > 0$
    and *sensitivity* > 0
    and *sensitivity* $< \infty$
  shows *differential-privacy* (*LapMech-list* ε) *adj* ε *0*
⟨*proof*⟩

end

8.2 Formalization of Results in [AFDP]

We finally instantiate X and *adj* according to the textbook [AFDP]

locale *results-AFDP* =
  fixes *n* ::*nat*
begin

interpretation *L1-norm-list* (*UNIV*::*nat set*) (λ *x y*. |*int x* − *int y*|) *n*
  ⟨*proof*⟩

definition *sp-Dataset* :: *nat list measure***where**
  sp-Dataset ≡ *restrict-space* (*listM* (*count-space UNIV*)) *space-L1-norm-list*

definition *adj-L1-norm* :: (*nat list* × *nat list*) *set***where**
  adj-L1-norm ≡ {(*xs*,*ys*)| *xs ys*. *xs* ∈ *space sp-Dataset* ∧ *ys* ∈ *space sp-Dataset* ∧ *dist-L1-norm-list xs ys* ≤ *1*}

definition *dist-L1-norm* :: *nat* ⇒ (*nat list* × *nat list*) *set***where**
  dist-L1-norm k ≡ {(*xs*,*ys*)| *xs ys*. *xs* ∈ *space sp-Dataset* ∧ *ys* ∈ *space sp-Dataset* ∧ *dist-L1-norm-list xs ys* ≤ *k*}

abbreviation
  differential-privacy-AFDP M ε δ ≡ *differential-privacy M adj-L1-norm* ε δ

lemma *adj-sub*: *adj-L1-norm* ⊆ *space sp-Dataset* × *space sp-Dataset*
  ⟨*proof*⟩

lemmas *differential-privacy-relax-AFDP′* =
  differential-privacy-relax[*of* - *adj-L1-norm*]

lemmas *differential-privacy-postprocessing-AFDP* =
  differential-privacy-postprocessing[*of* - - *adj-L1-norm*, *OF* - - - - *adj-sub*]

lemmas *differential-privacy-composition-adaptive-AFDP* =
  differential-privacy-composition-adaptive[*of* - - - - - *adj-L1-norm*, *OF* - - - - - - *adj-sub*]

lemmas *differential-privacy-composition-pair-AFDP* =
differential-privacy-composition-pair[*of* - - - *adj-L1-norm*, *OF* - - - -
- - *adj-sub*]

Group privacy [Theorem 2.2, AFDP].

lemma *group-privacy-AFDP*:
assumes *M*: $M \in$ *sp-Dataset* $\rightarrow_M$ *prob-algebra Y*
and *DP*: *differential-privacy-AFDP M* ε *0*
shows *differential-privacy M* (*dist-L1-norm k*) (*real k* $*$ ε) *0*
⟨*proof*⟩

context
fixes *f*::*nat list* ⇒ *real*
assumes [*measurable*]: $f \in$ *sp-Dataset* $\rightarrow_M$ *borel*
begin

interpretation *Lap-Mechanism-1dim sp-Datasetf adj-L1-norm*
⟨*proof*⟩

thm *LapMech-1dim-def*

definition *LapMech-1dim-AFDP* :: *real* ⇒ *nat list* ⇒ *real measure***where**
LapMech-1dim-AFDP ε *x* = *do*{*y* ← *Lap-dist0* ((*real-of-ereal sensitivity*) / ε); *return borel* (*f x* + *y*) }

lemma *LapMech-1dim-AFDP′*:
shows *LapMech-1dim-AFDP* = *LapMech-1dim*
⟨*proof*⟩

lemmas *differential-privacy-Lap-Mechanism-1dim-AFDP* =
differential-privacy-LapMech-1dim[*of* - , *simplified LapMech-1dim-AFDP′*[*symmetric*]]

end

context
fixes *f*::*nat list* ⇒ *real list*
and *m*::*nat*
assumes [*measurable*]: $f \in$ *sp-Dataset* $\rightarrow_M$ (*listM borel*)
and *len*: $\bigwedge x.\ x \in$ *space X* $\Longrightarrow$ *length* (*f x*) = *m*
begin

interpretation *Lap-Mechanism-list sp-Datasetf adj-L1-norm m*
⟨*proof*⟩

definition *LapMech-list-AFDP* :: *real* ⇒ *nat list* ⇒ *real list mea-*

*sure***where**
LapMech-list-AFDP ε x = do{ ys ← (Lap-dist0-list(real-of-ereal sensitivity / ε) m); return (listM borel) (map2 (+) (f x) ys) }

thm *LapMech-list-def*

lemma *LapMech-list-AFDP′*:
assumes *x ∈ space sp-Dataset*
shows *LapMech-list-AFDP ε x = LapMech-list ε x*
⟨*proof*⟩

lemma *differential-privacy-Lap-Mechanism-list-AFDP*:
assumes *0 < ε*
and *0 < sensitivity*
and *sensitivity < ∞*
shows *differential-privacy-AFDP (LapMech-list-AFDP ε) ε 0*
⟨*proof*⟩

end

end

end

theory *Differential-Privacy-Example-Report-Noisy-Max*
imports *Differential-Privacy-Standard*
begin

9 Report Noisy Max mechanism for counting query with Laplace noise

lemma *measurable-let*[*measurable*]:
assumes [*measurable*]: $g \in M \rightarrow_M L$
and [*measurable*]: $(\lambda(x,y).\ f\ x\ y) \in M \bigotimes_M L \rightarrow_M N$
shows $(\lambda x.\ (Let\ (g\ x)\ (f\ x))) \in M \rightarrow_M N$
⟨*proof*⟩

9.1 Formalization of argmax procedure

primrec *max-argmax* :: *real list ⇒ (ereal × nat)* **where**
max-argmax [] = (−∞,0)|
max-argmax (x#xs) = (let (m, i) = max-argmax xs in if x > m then (x,0) else (m, Suc i))

value *max-argmax* []
value *max-argmax* [*1*]
value *max-argmax* [*2*,*1*]
value *max-argmax* [*1*,*2*,*3*,*4*,*5*]
value *max-argmax* [*1*,*5*,*2*,*3*,*4*]
value ([*1*,*2*,*3*,*4*,*5*]::*int list*) ! *4*

primrec *max-argmax′* :: *real list* ⇒ (*ereal* × *nat*) **where**
max-argmax′ [] = (−∞,*0*)|
max-argmax′ (*x*#*xs*) = (*if x* > *fst* (*max-argmax′ xs*) *then* (*x*,*0*) *else* (*fst* (*max-argmax′ xs*),*Suc* (*snd* (*max-argmax′ xs*))))

lemma *max-argmax-is-max-argmax′*:
shows *max-argmax xs* = *max-argmax′ xs*
⟨*proof*⟩

lemma *measurable-max-argmax*[*measurable*]:
shows *max-argmax* ∈ (*listM* (*borel* :: *real measure*)) $\rightarrow_M$ (*borel* :: (*ereal measure*)) $\bigotimes_M$ *count-space UNIV*
⟨*proof*⟩

definition *argmax-list* :: *real list* ⇒ *nat* **where**
argmax-list = *snd o max-argmax*

lemma *measurable-argmax-list*[*measurable*]:
shows *argmax-list* ∈ (*listM* (*borel* :: *real measure*)) $\rightarrow_M$ *count-space UNIV*
⟨*proof*⟩

lemma *argmax-list-le-length*:
shows *length xs* = *Suc k* ⟹ (*argmax-list xs*) ≤ *k*
⟨*proof*⟩

lemma *fst-max-argmax-append*:
shows (*fst* (*max-argmax* (*xs* @ *ys*))) = *max* (*fst* (*max-argmax xs*)) (*fst* (*max-argmax ys*))
⟨*proof*⟩

lemma *fst-max-argmax-is-max*:
shows *fst* (*max-argmax xs*) = *Max* (*set xs* ∪ {−∞})
⟨*proof*⟩

lemma *argmax-list-max-argmax*:
assumes *xs* ≠ []
shows (*argmax-list xs*) = *m* ⟷ *max-argmax xs* = (*ereal* (*xs* ! *m*), *m*)
⟨*proof*⟩

lemma *argmax-list-gives-max*:
 assumes $xs \neq []$
 shows $(argmax\text{-}list\ xs) = m \Longrightarrow \forall\, i \in \{0..<length\ xs\}.\ xs\ !\ i \leq xs\ !\ m$
⟨*proof*⟩

lemma *argmax-list-gives-max2*:
 assumes $xs \neq []$
 shows $(x \leq (xs\ !\ m) \wedge argmax\text{-}list\ xs = m) = (max\text{-}argmax\ (x\#xs) = ((xs\ !\ m), (Suc\ m)))$
⟨*proof*⟩

lemma *fst-max-argmax-adj*:
 fixes *xs ys rs* :: *real list* **and** *n* :: *nat*
 assumes $length\ xs = n$
  and $length\ ys = n$
  and $length\ rs = n$
  and *adj*: $list\text{-}all2\ (\lambda\ x\ y.\ x \geq y \wedge x \leq y + 1)\ xs\ ys$
 shows $(fst\ (max\text{-}argmax\ (map2\ (+)\ xs\ rs))) \geq (fst\ (max\text{-}argmax\ (map2\ (+)\ ys\ rs))) \wedge (fst\ (max\text{-}argmax\ (map2\ (+)\ xs\ rs))) \leq (fst\ (max\text{-}argmax\ (map2\ (+)\ ys\ rs))) + 1$
 ⟨*proof*⟩

9.2 An auxiliary function to calculate the argmax of a list where an element has been inserted.

definition *argmax-insert* :: *real* ⇒ *real list* ⇒ *nat* ⇒ *nat* **where**
 $argmax\text{-}insert\ k\ ks\ i = argmax\text{-}list\ (list\text{-}insert\ k\ ks\ i)$

lemma *argmax-insert-i-i-0*:
 shows $(argmax\text{-}insert\ k\ xs\ 0 = 0) \longleftrightarrow (k > (fst\ (max\text{-}argmax\ xs)))$
 ⟨*proof*⟩

lemma *argmax-insert-i-i-expand*:
 assumes $xs \neq []$
  and $m \leq n$
  and $length\ xs = n$
 shows $(argmax\text{-}insert\ k\ xs\ m = m) \longleftrightarrow (max\text{-}argmax\ ((take\ m\ xs)\ @\ [k]\ @\ (drop\ m\ xs))) = (k,m)$
 ⟨*proof*⟩

lemma *argmax-insert-i-i-iterate*:
 assumes $m \leq n$
  and $length\ xs = n$
 shows $(argmax\text{-}insert\ k\ (x\ \#\ xs)\ (Suc\ m) = (Suc\ m)) \longleftrightarrow ((x \leq k) \wedge (argmax\text{-}insert\ k\ xs\ m = m))$
 ⟨*proof*⟩

lemma *argmax-insert-i-i*:

assumes $m \leq n$
and *length xs* $= n$
shows (*argmax-insert k xs m* $= m$) $\longleftrightarrow$ (*ereal k* $>$ (*fst* (*max-argmax* (*drop m xs*))) $\wedge$ (*ereal k* $\geq$ (*fst* (*max-argmax* (*take m xs*)))))
⟨*proof*⟩

lemma *argmax-insert-i-i′*:
assumes $m \leq n$
and *length xs* $= n$
shows (*argmax-insert k xs m* $= m$) $\longleftrightarrow$ (*ereal k* $\geq$ (*fst* (*max-argmax xs*)) $\wedge$ (*ereal k* $\neq$ (*fst* (*max-argmax* (*drop m xs*)))))
⟨*proof*⟩

lemma *argmax-insert-i-i′-region*:
assumes *length xs* $= n$ **and** *length ys* $= n$ **and** $i \leq n$
shows $\{$*r* | *r* :: *real*. *argmax-insert* (*r* + *d*) ((*map2* (+) *xs ys*)) *i* = *i*$\}$ = $\{$*r* | (*r* :: *real*). (*ereal* (*r* + *d*) $\geq$ (*fst* (*max-argmax* (*map2* (+) *xs ys*))) $\wedge$ (*ereal* (*r* + *d*) $\neq$ (*fst* (*max-argmax* (*drop i* (*map2* (+) *xs ys*))))))$\}$
⟨*proof*⟩

10 Formal proof of DP of RNM

10.1 A formal proof of the main part of differential privacy of report noisy max [Claim 3.9, AFDP]

locale *Lap-Mechanism-RNM-mainpart* =
fixes *M*::′*a measure*
and *adj*::′*a rel*
and *c*::′*a* $\Rightarrow$ *real list*
assumes *c*: $c \in M \rightarrow_M$ (*listM borel*)
and *cond*: $\forall$ (*x*,*y*) $\in$ *adj*. *list-all2* ($\lambda x\ y.\ y \leq x \wedge x \leq y + 1$) (*c x*) (*c y*) $\vee$ *list-all2* ($\lambda x\ y.\ y \leq x \wedge x \leq y + 1$) (*c y*) (*c x*)
and *adj*: *adj* $\subseteq$ (*space M*) $\times$ (*space M*)
begin

We define an abstracted version of report noisy max mechanism. We later instantiate *c* with the counting query.

definition *LapMech-RNM* :: *real* $\Rightarrow$ ′*a* $\Rightarrow$ *nat measure* **where**
LapMech-RNM ε *x* = *do* $\{$*y* $\leftarrow$ *Lap-dist-list* (*1* / ε) (*c x*); *return* (*count-space UNIV*) (*argmax-list y*)$\}$

lemma *measurable-LapMech-RNM*[*measurable*]:
shows *LapMech-RNM* $\varepsilon \in M \rightarrow_M$ *prob-algebra*(*count-space UNIV*)
⟨*proof*⟩

Calculating the density of the probability distributions sampled from the main part of *LapMech-RNM* **context**

fixes ε::*real*
assumes *pose*:$0 < \varepsilon$
begin

The main part of *LapMech-RNM* **definition** *RNM′* :: *real list* ⇒ *nat measure* **where**
RNM′ zs = *do* {*y* ← *Lap-dist-list* (*1* / ε) (*zs*); *return* (*count-space UNIV*) (*argmax-list y*)}

lemma *RNM′-def2*:
shows *RNM′ zs* = *do* {*rs* ← *Lap-dist0-list* (*1* / ε) (*length zs*); *y* ← *return* (*listM borel*) (*map2* (+) *zs rs*); *return* (*count-space UNIV*) (*argmax-list y*)}
⟨*proof*⟩

lemma *measurable-RNM′*[*measurable*]:
shows *RNM′* ∈ *listM borel* $\rightarrow_M$ *prob-algebra*(*count-space UNIV*)
⟨*proof*⟩

lemma *LapMech-RNM-RNM′-c*:
shows *LapMech-RNM* ε = *RNM′ o c*
⟨*proof*⟩

lemma *RNM′-Nil*:
shows *RNM′* [] = *return* (*count-space* (*UNIV* :: *nat set*)) *0*
⟨*proof*⟩

lemma *RNM′-singleton*:
shows *RNM′* [*x*] = *return* (*count-space* (*UNIV* :: *nat set*)) *0*
⟨*proof*⟩

lemma *RNM′-support*:
assumes *i* ≥ *length xs*
and *xs* ≠ []
shows *emeasure* (*RNM′ xs*){*i*} = *0*
⟨*proof*⟩

The conditional distribution of *RNM′* **when** *n* − *1* **values of noise are fixed.** **definition** *RNM-M* :: *real list* ⇒ *real list* ⇒ *real* ⇒ *nat* ⇒ *nat measure* **where**
RNM-M cs rs d i = *do*{*r* ← (*Lap-dist0* (*1*/ε)); *return* (*count-space UNIV*) (*argmax-insert* (*r*+*d*) ((λ (*xs*,*ys*). (*map2* (+) *xs ys*))(*cs*, *rs*)) *i*)}

lemma *space-RNM-M*:
shows *space* (*RNM-M cs rs d i*) = (*UNIV* :: *nat set*)
⟨*proof*⟩

lemma *sets-RNM-M*[*measurable-cong*]:

shows *sets* (*RNM-M cs rs d i*) = *sets* (*count-space* (*UNIV* :: *nat set*))
⟨*proof*⟩

lemma *RNM-M-Nil*:
shows *emeasure* (*RNM-M* [] [] *d 0*) $\{j \in UNIV.\ j = 0\} = 1$
and $k \neq 0 \Longrightarrow$ *emeasure* (*RNM-M* [] [] *d 0*) $\{j \in UNIV.\ j = k\} = 0$
⟨*proof*⟩

lemma *RNM-M-Nil-indicator*:
shows *emeasure* (*RNM-M* [] [] *d 0*) = *indicator* $\{A :: nat\ set.\ A \in UNIV \wedge 0 \in A\}$
⟨*proof*⟩

lemma *RNM-M-Nil-is-return-0*:
shows (*RNM-M* [] [] *d 0*) = *return* (*count-space UNIV*) *0*
⟨*proof*⟩

lemma *RNM-M-probability*:
fixes *cs rs* :: *real list*
assumes *length cs* = *n* **and** *length rs* = *n*
and $i \leq n$
and *pose*: $\varepsilon > 0$
shows $\mathcal{P}$(*j in* (*RNM-M cs rs d i*). *j* = *i*) = $\mathcal{P}$(*r in* (*Lap-dist0* ($1/\varepsilon$)). *ereal*(*r* + *d*) ≥ (*fst* (*max-argmax* (*map2* (+) *cs rs*))))
⟨*proof*⟩

differential privacy inequality on *RNM-M* **lemma** *DP-RNM-M-i*:
fixes *xs ys rs* :: *real list* **and** *x y* :: *real* **and** *i n* :: *nat*
assumes *length xs* = *n* **and** *length ys* = *n* **and** *length rs* = *n*
and *adj'*: $x \geq y \wedge x \leq y + 1$
and *adj*: *list-all2* (λ *x y*. $x \geq y \wedge x \leq y + 1$) *xs ys*
and $i \leq n$
shows $\mathcal{P}$(*j in* (*RNM-M xs rs x i*). *j* = *i*) ≤ (*exp* ε) ∗ $\mathcal{P}$(*j in* (*RNM-M ys rs y i*). *j* = *i*) ∧ $\mathcal{P}$(*j in* (*RNM-M ys rs y i*). *j* = *i*) ≤ (*exp* ε) ∗ $\mathcal{P}$(*j in* (*RNM-M xs rs x i*). *j* = *i*)
⟨*proof*⟩

Aggregating the differential privacy inequalities on *RNM-M* **to those of** *RNM'* **lemma** *RNM'-expand*:
fixes *n* :: *nat*
assumes *length xs* = *n* **and** $i \leq n$
shows (*RNM'* (*list-insert x xs i*)) = *do*{*rs* ← (*Lap-dist0-list* (*1* / ε) (*length xs*)); (*RNM-M xs rs x i*)}
⟨*proof*⟩

lemma *DP-RNM'-M-i*:
fixes *xs ys* :: *real list* **and** *i n* :: *nat*

assumes *lxs*: *length xs* = *n*
and *lys*: *length ys* = *n*
and *adj*: *list-all2* (λ *x y*. $x \geq y \wedge x \leq y + 1$) *xs ys*
shows $\mathcal{P}$(*j in* (*RNM′ xs*). *j* = *i*) ≤ (*exp* ε) ∗ $\mathcal{P}$(*j in* (*RNM′ ys*). *j* = *i*) ∧ $\mathcal{P}$(*j in* (*RNM′ ys*). *j* = *i*) ≤ (*exp* ε) ∗ $\mathcal{P}$(*j in* (*RNM′ xs*). *j* = *i*)
⟨*proof*⟩

lemma *DP-divergence-RNM′*:
fixes *xs ys* :: *real list*
assumes *adj*: *list-all2* (λ *x y*. $x \geq y \wedge x \leq y + 1$) *xs ys*
shows *DP-divergence* (*RNM′ xs*) (*RNM′ ys*) ε ≤ *0* ∧ *DP-divergence* (*RNM′ ys*) (*RNM′ xs*) ε ≤ *0*
⟨*proof*⟩

lemma *differential-privacy-LapMech-RNM′*:
shows *differential-privacy RNM′* {(*xs*, *ys*) | *xs ys*. *list-all2* (λ *x y*. $x \geq y \wedge x \leq y + 1$) *xs ys* } ε *0*
⟨*proof*⟩

corollary *differential-privacy-LapMech-RNM′-sym*:
shows *differential-privacy RNM′* {(*xs*, *ys*) | *xs ys*. *list-all2* (λ *x y*. $x \geq y \wedge x \leq y + 1$) *xs ys* ∨ *list-all2* (λ *x y*. $x \geq y \wedge x \leq y + 1$) *ys xs* } ε *0*
⟨*proof*⟩

theorem *differential-privacy-LapMech-RNM*:
shows *differential-privacy* (*LapMech-RNM* ε) *adj* ε *0*
⟨*proof*⟩

end

end

10.2 Formal Proof of Exact [Claim 3.9,AFDP]

In the above theorem ⟦*Lap-Mechanism-RNM-mainpart ?M ?adj ?c*; *0* < *?ε*⟧ ⟹ *differential-privacy* (*Lap-Mechanism-RNM-mainpart.LapMech-RNM ?c ?ε*) *?adj ?ε 0*, the query *c* to add noise is abstracted. We here instantiate it with the counting query. Thus we here formalize and show the monotonicity and Lipschitz property of it.

locale *Lap-Mechanism-RNM-counting* =
fixes *n*::*nat*
and *m*::*nat*
and *Q* :: *nat* ⇒ *nat* ⇒ *bool*
assumes ⋀*i*. *i* ∈ {*0*..<*m*} ⟹ (*Q i*) ∈ *UNIV* → *UNIV*
begin

interpretation *results-AFDP n*

$\langle proof \rangle$

interpretation *L1-norm-list* (*UNIV*::*nat set*) (λ *x y*. |*int x* − *int y*|) *n*
$\langle proof \rangle$

lemma *adjacency-1-int-list*:
assumes (*xs*,*ys*) ∈ *adj-L1-norm*
shows (*xs* = *ys*) ∨ (∃ *as a b bs*. *xs* = *as* @ (*a* # *bs*) ∧ *ys* = *as* @ (*b* # *bs*) ∧ |*int a* − *int b*| = *1*)
$\langle proof \rangle$

formalization of a list of counting query **primrec** *counting′*::*nat* ⇒ *nat* ⇒ *nat list* ⇒ *nat* **where**
counting′ i 0 - = 0 |
counting′ i (*Suc k*) *xs* = (*if Q i k then* (*nth-total 0 xs k*) *else 0*) + *counting′ i k xs*

lemma *measurable-counting′*[*measurable*]:
shows (λ *xs*. *counting′ i k xs*) ∈ (*listM* (*count-space* (*UNIV*::*nat set*))) $\rightarrow_M$ (*count-space* (*UNIV*::*nat set*))
$\langle proof \rangle$

lemma *counting′-sum*:
assumes *k* ≤ *length xs*
shows *counting′ i k xs* = (∑*j*∈{*0*..<*k*}. (*if Q i j then* (*xs* ! *j*) *else 0*))
$\langle proof \rangle$

lemma *sensitvity-counting′*:
assumes (*xs*,*ys*) ∈ *adj-L1-norm*
and *len*: *k* ≤ *n*
shows |*int* (*counting′ i k xs*) − *int* (*counting′ i k ys*)| ≤ *1*
$\langle proof \rangle$

A component of a tuple of counting queries **definition** *counting*::*nat* ⇒ *nat list* ⇒ *nat* **where**
counting i xs = *counting′ i* (*length xs*) *xs*

lemma *measurable-counting*[*measurable*]:
shows (*counting i*) ∈ (*listM* (*count-space* (*UNIV*::*nat set*))) $\rightarrow_M$ (*count-space* (*UNIV*::*nat set*))
$\langle proof \rangle$

lemma *counting-sum*:
shows *counting i xs* = (∑*j*∈{*0*..<*length xs*}. (*if Q i j then* (*xs* ! *j*) *else 0*))
$\langle proof \rangle$

lemma *sensitvity-counting*:
assumes $(xs,ys) \in$ *adj-L1-norm*
shows $|int\ (counting\ k\ xs) - int\ (counting\ k\ ys)| \leq 1$
⟨*proof*⟩

A list(tuple) of counting queries **definition** *counting-query*::*nat list* ⇒ *nat list* **where**
counting-query xs = *map* (λ *k*. *counting k xs*) [*0*..<*m*]

lemma *measurable-counting-query*[*measurable*]:
shows *counting-query* ∈ *listM* (*count-space UNIV*) $\rightarrow_M$ *listM* (*count-space UNIV*)
⟨*proof*⟩

lemma *length-counting-query*:
shows *length*(*counting-query xs*) = *m*
⟨*proof*⟩

lemma *counting-query-nth*:
fixes *k*::*nat* **assumes** *k* < *m*
shows *counting-query xs* ! *k* = *counting k xs*
⟨*proof*⟩

corollary *counting-query-nth′*:
fixes *k*::*nat* **assumes** *k* < *m*
shows *map real* (*counting-query xs*) ! *k* = *real* (*counting k xs*)
⟨*proof*⟩

end

A formalization of the report noisy max of noisy counting **context**

Lap-Mechanism-RNM-counting
begin

interpretation *L1-norm-list* (*UNIV*::*nat set*) (λ *x y*. |*int x* − *int y*|) *n*
⟨*proof*⟩

interpretation *results-AFDP n*
⟨*proof*⟩

definition *RNM-counting* :: *real* ⇒ *nat list* ⇒ *nat measure* **where**
RNM-counting ε *x* = *do* {
y ← *Lap-dist-list* (*1* / ε) (*counting-query x*);
return (*count-space UNIV*) (*argmax-list y*)
}

Naive evaluation of differential privacy of *RNM-counting* The naive proof is as follows: We first show that the counting query has the sensitivity m. *RNM-counting* adds the Laplace noise with scale $1 / \varepsilon$ to each element given by *counting-query*. Thanks to the postprocessing inequality, the final process *argmax-list* does not change the differential privacy. Therefore the differential privacy of *RNM-counting* is $m * \varepsilon$

interpretation *Lap-Mechanism-list listM* (*count-space UNIV*) *counting-query adj-L1-norm m*
⟨*proof*⟩

lemma *sensitvity-counting-query-part*:
fixes k::*nat* **assumes** $k < m$
and (*xs*,*ys*) ∈ *adj-L1-norm*
shows |*map real* (*counting-query xs*) ! k − *map real* (*counting-query ys*) ! k | ≤ *1*
⟨*proof*⟩

lemma *sensitvity-counting-query*:
shows *sensitivity* ≤ m
⟨*proof*⟩

corollary *sensitvity-counting-query-finite*:
shows *sensitivity* < ∞
⟨*proof*⟩

theorem *Naive-differential-privacy-LapMech-RNM-AFDP*:
assumes *pose*: (ε::*real*) > *0*
shows *differential-privacy-AFDP* (*RNM-counting* ε) (*real* ($m * \varepsilon$)) *0*
⟨*proof*⟩

True evaluation of differential privacy of *RNM-counting* in contrast to the naive proof, *counting-query* and *argmax-list* are essential.

lemma *finer-sensitivity-counting-query*:
assumes (*xs*,*ys*) ∈ *adj-L1-norm*
shows *list-all2* (λ *x y*. $x \geq y \wedge x \leq y + 1$) (*counting-query xs*) (*counting-query ys*) ∨ *list-all2* (λ *x y*. $x \geq y \wedge x \leq y + 1$) (*counting-query ys*) (*counting-query xs*)
⟨*proof*⟩

lemma *list-all2-map-real-adj*:
assumes *list-all2* (λ*x y*. $y \leq x \wedge x \leq y + 1$) *xs ys*
shows *list-all2* (λ*x y*. $y \leq x \wedge x \leq y + 1$) (*map real xs*) (*map real ys*)
⟨*proof*⟩

lemma *finer-sensitivity-counting-query'*:
assumes $(xs,ys) \in$ *adj-L1-norm*
shows *list-all2* $(\lambda\ x\ y.\ x \geq y \wedge x \leq y + 1)$ (*map real* (*counting-query xs*))(*map real* (*counting-query ys*)) $\vee$ *list-all2* $(\lambda\ x\ y.\ x \geq y \wedge x \leq y + 1)$ (*map real* (*counting-query ys*))(*map real* (*counting-query xs*))
⟨*proof*⟩

interpretation *Lap-Mechanism-RNM-mainpart* (*listM* (*count-space UNIV*)) *adj-L1-norm counting-query*
⟨*proof*⟩

theorem *differential-privacy-LapMech-RNM-AFDP*:
assumes *pose*: $(\varepsilon::real) > 0$
shows *differential-privacy-AFDP* (*RNM-counting* ε) ε *0*
⟨*proof*⟩

end

end

References

[1] B. Balle, G. Barthe, M. Gaboardi, J. Hsu, and T. Sato. Hypothesis testing interpretations and renyi differential privacy. In S. Chiappa and R. Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics (AISTATS 2020)*, volume 108 of *Proceedings of Machine Learning Research*, pages 2496–2506, Online, 26–28 Aug 2020. PMLR.

[2] G. Barthe and F. Olmedo. Beyond differential privacy: Composition theorems and relational logic for f-divergences between probabilistic programs. In *Automata, Languages, and Programming: 40th International Colloquium, ICALP 2013, Proceedings, Part II*, pages 49–60. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[3] C. Dwork and A. Roth. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3-4):211–407, 2013.

[4] S. P. Kasiviswanathan and A. Smith. A note on differential privacy: Defining resistance to arbitrary side information. *Journal of Privacy and Confidentiality*, 6(1), 2014.

[5] T. Sato and S. Katsumata. Divergences on monads for relational program logics. *Mathematical Structures in Computer Science*, 33(45):427485, 2023.