

Formalization of Randomized Approximation Algorithms for Frequency Moments

Emin Karayel

January 16, 2022

Abstract

In 1999 Alon et. al. introduced the still active research topic of approximating the frequency moments of a data stream using randomized algorithms with minimal space usage. This includes the problem of estimating the cardinality of the stream elements—the zeroth frequency moment. But, also higher order frequency moments that provide information about the skew of the data stream, which is for example critical information for parallel processing. The frequency moment of a data stream $a_1, \dots, a_m \in U$ can be defined as $F_k := \sum_{u \in U} C(u, a)^k$ where $C(u, a)$ is the count of occurrences of u in the stream a . They introduce both lower bounds and upper bounds, which were later improved by newer publications. The algorithms have guaranteed success probability and accuracy, without making any assumptions on the input distribution. They are an interesting use-case for formal verification, because they rely on deep results from both algebra and analysis, require a large body of existing results. This work contains the formal verification of three algorithms for the approximation of F_0 , F_2 and F_k for $k \geq 3$. To achieve it, the formalization also includes reusable components common to all algorithms, such as universal hash families, the median method, formal modelling of one-pass data stream algorithms and a generic flexible encoding library for the verification of space complexities.

Contents

1	Encoding	2
2	Field	8
3	Float	9
4	Lists	11
5	Frequency Moments	11

6 Primes	12
7 Multisets	13
8 Probability Spaces	14
9 Median	18
10 Order Statistics	22
11 Counting Polynomials	24
11.1 Interpolation Polynomials	25
12 Indexed Products of Probability Mass Functions	28
13 Universal Hash Families	31
14 Universal Hash Family for $\{0.. < p\}$	34
14.1 Encoding	35
15 Landau Symbols	36
16 Frequency Moment 0	38
17 Partitions	42
18 Frequency Moment 2	44
19 Frequency Moment k	47
A Informal proof of correctness for the F_0 algorithm	51
A.1 Case $F_0 \geq t$	52
A.2 Case $F_0 < t$	55

1 Encoding

```

theory Encoding
  imports Main HOL–Library.Sublist HOL–Library.Extended-Real HOL–Library.FuncSet

  HOL.Transcendental
begin

```

This section contains a flexible library for encoding high level data structures into bit strings. The library defines encoding functions for primitive types, as well as combinators to build encodings for more complex types. It is used to measure the size of the data structures.

```

fun is-prefix where

```

is-prefix (Some x) (Some y) = *prefix* x y |
is-prefix - - = *False*

type-synonym 'a *encoding* = 'a \rightarrow *bool list*

definition *is-encoding* :: 'a *encoding* \Rightarrow *bool*
where *is-encoding* $f = (\forall x\ y. \text{is-prefix } (f\ x) (f\ y) \longrightarrow x = y)$

lemma *encoding-imp-inj*:
assumes *is-encoding* f
shows *inj-on* f (*dom* f)
 $\langle \text{proof} \rangle$

definition *decode* **where**
 $\text{decode } f\ t =$
 if ($\exists! z. \text{is-prefix } (f\ z) (\text{Some } t)$) *then*
 (*let* $z = (\text{THE } z. \text{is-prefix } (f\ z) (\text{Some } t))$ *in* ($z, \text{drop } (\text{length } (\text{the } (f\ z)))\ t$))
 else
 (*undefined*, t)
)

lemma *decode-elim*:
assumes *is-encoding* f
assumes $f\ x = \text{Some } r$
shows $\text{decode } f\ (r@r1) = (x, r1)$
 $\langle \text{proof} \rangle$

lemma *decode-elim-2*:
assumes *is-encoding* f
assumes $x \in \text{dom } f$
shows $\text{decode } f\ (\text{the } (f\ x)@r1) = (x, r1)$
 $\langle \text{proof} \rangle$

lemma *snd-decode-suffix*:
 $\text{suffix } (\text{snd } (\text{decode } f\ t))\ t$
 $\langle \text{proof} \rangle$

lemma *snd-decode-len*:
assumes $\text{decode } f\ t = (u, v)$
shows $\text{length } v \leq \text{length } t$
 $\langle \text{proof} \rangle$

lemma *encoding-by-witness*:
assumes $\bigwedge x\ y. x \in \text{dom } f \implies g\ (\text{the } (f\ x)@y) = (x, y)$
shows *is-encoding* f
 $\langle \text{proof} \rangle$

fun *bit-count* :: *bool list option* \Rightarrow *ereal* **where**
bit-count *None* = ∞ |

$bit_count\ (Some\ x) = ereal\ (length\ x)$

fun *append-encoding* :: *bool list option* \Rightarrow *bool list option* \Rightarrow *bool list option* (**infixr** $@_S\ 65$)
where
append-encoding (*Some* *x*) (*Some* *y*) = *Some* (*x* $@_S$ *y*) |
append-encoding - - = *None*

lemma *bit-count-append*: $bit_count\ (x1@_Sx2) = bit_count\ x1 + bit_count\ x2$
 $\langle proof \rangle$

Encodings for lists

fun *list_S* **where**
list_S *f* [] = *Some* [*False*] |
list_S *f* (*x* $\#$ *xs*) = *Some* [*True*] $@_Sf\ x@_Slist_S\ f\ xs$

function *decode-list* :: (*'a* \Rightarrow *bool list option*) \Rightarrow *bool list*
 \Rightarrow *'a list* \times *bool list*

where
decode-list *e* (*True* $\#x0$) = (
let (*r1*,*x1*) = *decode* *e* *x0* *in* (
let (*r2*,*x2*) = *decode-list* *e* *x1* *in* (*r1* $\#r2$,*x2*))) |
decode-list *e* (*False* $\#x0$) = ([], *x0*) |
decode-list *e* [] = *undefined*
 $\langle proof \rangle$

termination
 $\langle proof \rangle$

lemma *list-encoding-dom*:
assumes $set\ l \subseteq dom\ f$
shows $l \in dom\ (list_S\ f)$
 $\langle proof \rangle$

lemma *list-bit-count*:
 $bit_count\ (list_S\ f\ xs) = (\sum x \leftarrow xs. bit_count\ (f\ x) + 1) + 1$
 $\langle proof \rangle$

lemma *list-bit-count-est*:
assumes $\bigwedge x. x \in set\ xs \Rightarrow bit_count\ (f\ x) \leq a$
shows $bit_count\ (list_S\ f\ xs) \leq ereal\ (length\ xs) * (a+1) + 1$
 $\langle proof \rangle$

lemma *list-bit-count-estI*:
assumes $\bigwedge x. x \in set\ xs \Rightarrow bit_count\ (f\ x) \leq a$
assumes $ereal\ (real\ (length\ xs)) * (a+1) + 1 \leq h$
shows $bit_count\ (list_S\ f\ xs) \leq h$
 $\langle proof \rangle$

lemma *list-encoding-aux*:

assumes *is-encoding* *f*
shows $x \in \text{dom } (\text{list}_S f) \implies \text{decode-list } f \text{ (the } (\text{list}_S f x) @ y) = (x, y)$
 $\langle \text{proof} \rangle$

lemma *list-encoding*:
assumes *is-encoding* *f*
shows *is-encoding* $(\text{list}_S f)$
 $\langle \text{proof} \rangle$

Encoding for natural numbers

fun *nat-encoding-aux* :: $\text{nat} \Rightarrow \text{bool list}$
where
 $\text{nat-encoding-aux } 0 = [\text{False}] \mid$
 $\text{nat-encoding-aux } (\text{Suc } n) = \text{True} \# (\text{odd } n) \# \text{nat-encoding-aux } (n \text{ div } 2)$

fun N_S **where** $N_S n = \text{Some } (\text{nat-encoding-aux } n)$

fun *decode-nat* :: $\text{bool list} \Rightarrow \text{nat} \times \text{bool list}$
where
 $\text{decode-nat } (\text{False} \# y) = (0, y) \mid$
 $\text{decode-nat } (\text{True} \# x \# xs) =$
 $(\text{let } (n, rs) = \text{decode-nat } xs \text{ in } (n * 2 + 1 + (\text{if } x \text{ then } 1 \text{ else } 0), rs)) \mid$
 $\text{decode-nat } - = \text{undefined}$

lemma *nat-encoding-aux*:
 $\text{decode-nat } (\text{nat-encoding-aux } x @ y) = (x, y)$
 $\langle \text{proof} \rangle$

lemma *nat-encoding*:
shows *is-encoding* N_S
 $\langle \text{proof} \rangle$

lemma *nat-bit-count*:
 $\text{bit-count } (N_S n) \leq 2 * \log 2 (\text{real } n + 1) + 1$
 $\langle \text{proof} \rangle$

lemma *nat-bit-count-est*:
assumes $n \leq m$
shows $\text{bit-count } (N_S n) \leq 2 * \log 2 (1 + \text{real } m) + 1$
 $\langle \text{proof} \rangle$

Encoding for integers

fun I_S :: $\text{int} \Rightarrow \text{bool list option}$
where
 $I_S n = (\text{if } n \geq 0 \text{ then } \text{Some } [\text{True}] @_S N_S (\text{nat } n) \text{ else } \text{Some } [\text{False}] @_S (N_S (\text{nat } (-n-1))))$

fun *decode-int* :: $\text{bool list} \Rightarrow (\text{int} \times \text{bool list})$
where

$\text{decode-int } (\text{True}\#xs) = (\lambda(x::\text{nat},y). (\text{int } x, y)) (\text{decode-nat } xs) \mid$
 $\text{decode-int } (\text{False}\#xs) = (\lambda(x::\text{nat},y). (-(\text{int } x)-1, y)) (\text{decode-nat } xs) \mid$
 $\text{decode-int } [] = \text{undefined}$

lemma *int-encoding: is-encoding* I_S
 $\langle \text{proof} \rangle$

lemma *int-bit-count:*
 $\text{bit-count } (I_S \ x) \leq 2 * \log 2 \ (|x|+1) + 2$
 $\langle \text{proof} \rangle$

lemma *int-bit-count-est:*
assumes $\text{abs } n \leq m$
shows $\text{bit-count } (I_S \ n) \leq 2 * \log 2 \ (m+1) + 2$
 $\langle \text{proof} \rangle$

Encoding for Cartesian products

fun *encode-prod* :: $'a \text{ encoding} \Rightarrow 'b \text{ encoding} \Rightarrow ('a \times 'b) \text{ encoding}$ (**infixr** \times_S 65)
where
 $\text{encode-prod } e1 \ e2 \ x = e1 \ (\text{fst } x) @_S \ e2 \ (\text{snd } x)$

fun *decode-prod* :: $'a \text{ encoding} \Rightarrow 'b \text{ encoding} \Rightarrow \text{bool list} \Rightarrow ('a \times 'b) \times \text{bool list}$
where
 $\text{decode-prod } e1 \ e2 \ x0 =$
 $\text{let } (r1, x1) = \text{decode } e1 \ x0 \text{ in } ($
 $\text{let } (r2, x2) = \text{decode } e2 \ x1 \text{ in } ((r1, r2), x2))$

lemma *prod-encoding-dom:*
 $x \in \text{dom } (e1 \times_S e2) = (\text{fst } x \in \text{dom } e1 \wedge \text{snd } x \in \text{dom } e2)$
 $\langle \text{proof} \rangle$

lemma *prod-encoding:*
assumes *is-encoding* $e1$
assumes *is-encoding* $e2$
shows *is-encoding* $(\text{encode-prod } e1 \ e2)$
 $\langle \text{proof} \rangle$

lemma *prod-bit-count:*
 $\text{bit-count } ((e1 \times_S e2) \ (x_1, x_2)) = \text{bit-count } (e1 \ x_1) + \text{bit-count } (e2 \ x_2)$
 $\langle \text{proof} \rangle$

lemma *prod-bit-count-2:*
 $\text{bit-count } ((e1 \times_S e2) \ x) = \text{bit-count } (e1 \ (\text{fst } x)) + \text{bit-count } (e2 \ (\text{snd } x))$
 $\langle \text{proof} \rangle$

Encoding for dependent sums

fun *encode-dependent-sum* :: $'a \text{ encoding} \Rightarrow ('a \Rightarrow 'b \text{ encoding}) \Rightarrow ('a \times 'b) \text{ encoding}$ (**infixr** \times_D 65)
where

$$\text{encode-dependent-sum } e1 \ e2 \ x = e1 \ (\text{fst } x) @_S \ e2 \ (\text{fst } x) \ (\text{snd } x)$$

lemma *dependent-encoding*:

assumes *is-encoding* $e1$

assumes $\bigwedge x. \text{is-encoding } (e2 \ x)$

shows *is-encoding* $(\text{encode-dependent-sum } e1 \ e2)$

<proof>

lemma *dependent-bit-count*:

$\text{bit-count } ((e1 \times_D \ e2) \ (x_1, x_2)) = \text{bit-count } (e1 \ x_1) + \text{bit-count } (e2 \ x_1 \ x_2)$

<proof>

This lemma helps derive an encoding on the domain of an injective function using an existing encoding on its image.

lemma *encoding-compose*:

assumes *is-encoding* f

assumes *inj-on* $g \ \{x. P \ x\}$

shows *is-encoding* $(\lambda x. \text{if } P \ x \text{ then } f \ (g \ x) \text{ else } \text{None})$

<proof>

Encoding for extensional maps defined on an enumerable set.

definition $\text{fun}_S :: 'a \text{ list} \Rightarrow 'b \text{ encoding} \Rightarrow ('a \Rightarrow 'b) \text{ encoding}$ (**infixr** \rightarrow_S 65)

where

$\text{fun}_S \ xs \ e \ f = ($
 $\quad \text{if } f \in \text{extensional } (\text{set } xs) \text{ then}$
 $\quad \quad \text{list}_S \ e \ (\text{map } f \ xs)$
 $\quad \text{else}$
 $\quad \quad \text{None})$

lemma *encode-extensional*:

assumes *is-encoding* e

shows *is-encoding* $(\lambda x. (xs \rightarrow_S \ e) \ x)$

<proof>

lemma *extensional-bit-count*:

assumes $f \in \text{extensional } (\text{set } xs)$

shows $\text{bit-count } ((xs \rightarrow_S \ e) \ f) = (\sum x \leftarrow xs. \text{bit-count } (e \ (f \ x)) + 1) + 1$

<proof>

Encoding for ordered sets.

fun set_S **where** $\text{set}_S \ e \ S = (\text{if finite } S \text{ then } \text{list}_S \ e \ (\text{sorted-list-of-set } S) \text{ else } \text{None})$

lemma *encode-set*:

assumes *is-encoding* e

shows *is-encoding* $(\lambda S. \text{set}_S \ e \ S)$

<proof>

lemma *set-bit-count*:

assumes *finite* S

shows $\text{bit-count } (\text{set}_S \ e \ S) = (\sum x \in S. \text{bit-count } (e \ x) + 1) + 1$
 $\langle \text{proof} \rangle$

lemma *set-bit-count-est*:

assumes *finite* S

assumes $\text{card } S \leq m$

assumes $0 \leq a$

assumes $\bigwedge x. x \in S \implies \text{bit-count } (f \ x) \leq a$

shows $\text{bit-count } (\text{set}_S \ f \ S) \leq \text{ereal } (\text{real } m) * (a + 1) + 1$

$\langle \text{proof} \rangle$

end

2 Field

theory *Field*

imports *Main HOL-Algebra.Ring-Divisibility HOL-Algebra.IntRing*

begin

This section contains a proof that the factor ring $ZFact \ p$ for *prime* p is a field. Note that the bulk of the work has already been done in *HOL-Algebra*, in particular it is established that $ZFact \ p$ is a domain.

However, any domain with a finite carrier is already a field. This can be seen by establishing that multiplication by a non-zero element is an injective map between the elements of the carrier of the domain. But an injective map between sets of the same non-finite cardinality is also surjective. Hence we can find the unit element in the image of such a map.

Additionally the canonical bijection between $ZFact \ p$ and $\{0..<p\}$ is introduced, which is useful for hashing natural numbers.

definition *zfact-embed* :: $\text{nat} \Rightarrow \text{nat} \Rightarrow \text{int set}$ **where**

zfact-embed $p \ k = \text{Idl}_{\mathbb{Z}} \ \{ \text{int } p \} \ +>_{\mathbb{Z}} \ (\text{int } k)$

lemma *zfact-embed-ran*:

assumes $p > 0$

shows $\text{zfact-embed } p \ \{0..<p\} = \text{carrier } (ZFact \ p)$

$\langle \text{proof} \rangle$

lemma *zfact-embed-inj*:

assumes $p > 0$

shows $\text{inj-on } (\text{zfact-embed } p) \ \{0..<p\}$

$\langle \text{proof} \rangle$

lemma *zfact-embed-bij*:

assumes $p > 0$

shows $\text{bij-betw } (\text{zfact-embed } p) \ \{0..<p\} \ (\text{carrier } (ZFact \ p))$

$\langle \text{proof} \rangle$

lemma *zfact-card*:
assumes $(p :: \text{nat}) > 0$
shows $\text{card } (\text{carrier } (\text{ZFact } (\text{int } p))) = p$
 $\langle \text{proof} \rangle$

lemma *zfact-finite*:
assumes $(p :: \text{nat}) > 0$
shows $\text{finite } (\text{carrier } (\text{ZFact } (\text{int } p)))$
 $\langle \text{proof} \rangle$

lemma *finite-domains-are-fields*:
assumes $\text{domain } R$
assumes $\text{finite } (\text{carrier } R)$
shows $\text{field } R$
 $\langle \text{proof} \rangle$

lemma *zfact-prime-is-field*:
assumes $\text{prime } (p :: \text{nat})$
shows $\text{field } (\text{ZFact } (\text{int } p))$
 $\langle \text{proof} \rangle$

end

3 Float

This section contains results about floating point numbers in addition to "HOL-Library.Float"

theory *Float-Ext*
imports *HOL-Library.Float Encoding*
begin

lemma *round-down-ge*:
 $x \leq \text{round-down } \text{prec } x + 2^{\text{powr } (-\text{prec})}$
 $\langle \text{proof} \rangle$

lemma *truncate-down-ge*:
 $x \leq \text{truncate-down } \text{prec } x + \text{abs } x * 2^{\text{powr } (-\text{prec})}$
 $\langle \text{proof} \rangle$

lemma *truncate-down-pos*:
assumes $x \geq 0$
shows $x * (1 - 2^{\text{powr } (-\text{prec})}) \leq \text{truncate-down } \text{prec } x$
 $\langle \text{proof} \rangle$

lemma *truncate-down-eq*:
assumes $\text{truncate-down } r \ x = \text{truncate-down } r \ y$
shows $\text{abs } (x - y) \leq \max (\text{abs } x) (\text{abs } y) * 2^{\text{powr } (-\text{real } r)}$
 $\langle \text{proof} \rangle$

definition *rat-of-float* :: *float* \Rightarrow *rat* **where**
rat-of-float *f* = *of-int* (*mantissa* *f*) *
 (if *exponent* *f* \geq 0 then $2^{\text{nat } (\text{exponent } f)}$ else $1 / 2^{\text{nat } (-\text{exponent } f)}$))

lemma *real-of-rat-of-float*: *real-of-rat* (*rat-of-float* *x*) = *real-of-float* *x*
 <proof>

Definition of an encoding for floating point numbers.

definition *F_S* **where** *F_S* *f* = (*I_S* \times_S *I_S*) (*mantissa* *f*, *exponent* *f*)

lemma *encode-float*:
is-encoding *F_S*
 <proof>

lemma *truncate-mantissa-bound*:
 $\text{abs } (\lfloor x * 2^{\text{powr } (\text{real } r - \text{real-of-int } \lfloor \log 2 |x| \rfloor)} \rfloor) \leq 2^{r+1}$ (is ?lhs \leq -)
 <proof>

lemma *suc-n-le-2-pow-n*:
fixes *n* :: *nat*
shows $n + 1 \leq 2^n$
 <proof>

lemma *float-bit-count*:
fixes *m* :: *int*
fixes *e* :: *int*
defines *f* \equiv *float-of* (*m* * $2^{\text{powr } e}$)
shows *bit-count* (*F_S* *f*) $\leq 4 + 2 * (\log 2 (|m| + 2) + \log 2 (|e| + 1))$
 <proof>

lemma *float-bit-count-zero*:
bit-count (*F_S* (*float-of* 0)) = 4
 <proof>

lemma *log-est*: $\log 2 (\text{real } n + 1) \leq n$
 <proof>

lemma *truncate-float-bit-count*:
 $\text{bit-count } (F_S (\text{float-of } (\text{truncate-down } r \ x))) \leq 8 + 4 * \text{real } r + 2 * \log 2 (2 + \text{abs } (\log 2 (\text{abs } x)))$
 (is ?lhs \leq ?rhs)
 <proof>

end

4 Lists

```
theory List-Ext
imports Main HOL.List
begin
```

This section contains results about lists in addition to "HOL.List"

```
lemma count-list-gr-1:
   $(x \in \text{set } xs) = (\text{count-list } xs \ x \geq 1)$ 
   $\langle \text{proof} \rangle$ 
```

```
lemma count-list-append:  $\text{count-list } (xs@ys) \ v = \text{count-list } xs \ v + \text{count-list } ys \ v$ 
   $\langle \text{proof} \rangle$ 
```

```
lemma count-list-card:  $\text{count-list } xs \ x = \text{card } \{k. k < \text{length } xs \wedge xs[k] = x\}$ 
   $\langle \text{proof} \rangle$ 
```

```
lemma card-gr-1-iff:
  assumes finite S
  assumes  $x \in S$ 
  assumes  $y \in S$ 
  assumes  $x \neq y$ 
  shows  $\text{card } S > 1$ 
   $\langle \text{proof} \rangle$ 
```

```
lemma count-list-ge-2-iff:
  assumes  $y < z$ 
  assumes  $z < \text{length } xs$ 
  assumes  $xs[y] = xs[z]$ 
  shows  $\text{count-list } xs \ (xs[y]) > 1$ 
   $\langle \text{proof} \rangle$ 
```

end

5 Frequency Moments

```
theory Frequency-Moments
imports Main HOL.List HOL.Rat List-Ext
begin
```

This section contains a definition of the frequency moments of a stream.

```
definition F where
   $F \ k \ xs = (\sum x \in \text{set } xs. (\text{rat-of-nat } (\text{count-list } xs \ x) \ k))$ 
```

```
lemma F-gr-0:
  assumes  $as \neq []$ 
  shows  $F \ k \ as > 0$ 
   $\langle \text{proof} \rangle$ 
```

end

6 Primes

This section introduces a function that finds the smallest primes above a given threshold.

theory *Primes-Ext*

imports *Main HOL-Computational-Algebra.Primes Bertrands-Postulate.Bertrand*

begin

lemma *inf-primes*: $wf ((\lambda n. (Suc\ n, n)) \cdot \{n. \neg (prime\ n)\})$ (**is** $wf\ ?S$)
 $\langle proof \rangle$

function *find-prime-above* :: $nat \Rightarrow nat$ **where**

find-prime-above $n = (if\ prime\ n\ then\ n\ else\ find-prime-above\ (Suc\ n))$

$\langle proof \rangle$

termination

$\langle proof \rangle$

declare *find-prime-above.simps* [*simp del*]

lemma *find-prime-above-is-prime*:

prime (*find-prime-above* n)

$\langle proof \rangle$

lemma *find-prime-above-min*:

find-prime-above $n \geq 2$

$\langle proof \rangle$

lemma *find-prime-above-lower-bound*:

find-prime-above $n \geq n$

$\langle proof \rangle$

lemma *find-prime-above-upper-boundI*:

assumes *prime* m

shows $n \leq m \implies find-prime-above\ n \leq m$

$\langle proof \rangle$

lemma *find-prime-above-upper-bound*:

find-prime-above $n \leq 2*n+2$

$\langle proof \rangle$

end

7 Multisets

```
theory Multiset-Ext
imports Main HOL.Real HOL-Library.Multiset
begin
```

This section contains results about multisets in addition to "HOL.Multiset"

This is a induction scheme over the distinct elements of a multisets: We can represent each multiset as a sum like: *replicate-mset* n_1 x_1 + *replicate-mset* n_2 x_2 + ... + *replicate-mset* n_k x_k where the x_i are distinct.

```
lemma disj-induct-mset:
  assumes  $P \{ \# \}$ 
  assumes  $\bigwedge n M x. P M \implies \neg(x \in \# M) \implies n > 0 \implies P (M + \text{replicate-mset } n \ x)$ 
  shows  $P M$ 
  <proof>
```

```
lemma prod-mset-conv:
  fixes  $f :: 'a \Rightarrow 'b :: \{ \text{comm-monoid-mult} \}$ 
  shows  $\text{prod-mset } (\text{image-mset } f \ A) = \text{prod } (\lambda x. f \ x \wedge (\text{count } A \ x)) (\text{set-mset } A)$ 
  <proof>
```

```
lemma sum-collapse:
  fixes  $f :: 'a \Rightarrow 'b :: \{ \text{comm-monoid-add} \}$ 
  assumes finite  $A$ 
  assumes  $z \in A$ 
  assumes  $\bigwedge y. y \in A \implies y \neq z \implies f \ y = 0$ 
  shows  $\text{sum } f \ A = f \ z$ 
  <proof>
```

There is a version *sum-list-map-eq-sum-count* but it doesn't work if the function maps into the reals.

```
lemma sum-list-eval:
  fixes  $f :: 'a \Rightarrow 'b :: \{ \text{ring, semiring-1} \}$ 
  shows  $\text{sum-list } (\text{map } f \ xs) = (\sum x \in \text{set } xs. \text{of-nat } (\text{count-list } xs \ x) * f \ x)$ 
  <proof>
```

```
lemma prod-list-eval:
  fixes  $f :: 'a \Rightarrow 'b :: \{ \text{ring, semiring-1, comm-monoid-mult} \}$ 
  shows  $\text{prod-list } (\text{map } f \ xs) = (\prod x \in \text{set } xs. (f \ x) \wedge (\text{count-list } xs \ x))$ 
  <proof>
```

```
lemma sorted-sorted-list-of-multiset:  $\text{sorted } (\text{sorted-list-of-multiset } M)$ 
  <proof>
```

```
lemma count-mset:  $\text{count } (\text{mset } xs) \ a = \text{count-list } xs \ a$ 
  <proof>
```

lemma *swap-filter-image*: $\text{filter-mset } g \ (\text{image-mset } f \ A) = \text{image-mset } f \ (\text{filter-mset } (g \circ f) \ A)$
 ⟨proof⟩

lemma *list-eq-iff*:
 assumes $\text{mset } xs = \text{mset } ys$
 assumes $\text{sorted } xs$
 assumes $\text{sorted } ys$
 shows $xs = ys$
 ⟨proof⟩

lemma *sorted-list-of-multiset-image-commute*:
 assumes $\text{mono } f$
 shows $\text{sorted-list-of-multiset } (\text{image-mset } f \ M) = \text{map } f \ (\text{sorted-list-of-multiset } M)$ (is $?A = ?B$)
 ⟨proof⟩

end

8 Probability Spaces

Some additional results about probability spaces in addition to "HOL-Probability".

theory *Probability-Ext*
 imports *Main HOL-Probability.Independent-Family Multiset-Ext HOL-Probability.Stream-Space*
HOL-Probability.Probability-Mass-Function
begin

lemma *measure-inters*: $\text{measure } M \ (E \cap \text{space } M) = \mathcal{P}(x \text{ in } M. \ x \in E)$
 ⟨proof⟩

lemma *set-comp-subsetI*: $(\bigwedge x. \ P \ x \implies f \ x \in B) \implies \{f \ x \mid x. \ P \ x\} \subseteq B$
 ⟨proof⟩

lemma *set-comp-cong*:
 assumes $\bigwedge x. \ P \ x \implies f \ x = h \ (g \ x)$
 shows $\{f \ x \mid x. \ P \ x\} = h \ ` \ \{g \ x \mid x. \ P \ x\}$
 ⟨proof⟩

lemma *indep-sets-distr*:
 assumes $f \in \text{measurable } M \ N$
 assumes $\text{prob-space } M$
 assumes $\text{prob-space.indep-sets } M \ (\lambda i. \ (\lambda a. \ f \ - \ ` \ a \cap \text{space } M) \ ` \ A \ i) \ I$
 assumes $\bigwedge i. \ i \in I \implies A \ i \subseteq \text{sets } N$
 shows $\text{prob-space.indep-sets } (\text{distr } M \ N \ f) \ A \ I$
 ⟨proof⟩

lemma *indep-vars-distr*:
 assumes $f \in \text{measurable } M \ N$

assumes $\bigwedge i. i \in I \implies X' i \in \text{measurable } N (M' i)$
assumes $\text{prob-space.indep-vars } M M' (\lambda i. (X' i) \circ f) I$
assumes $\text{prob-space } M$
shows $\text{prob-space.indep-vars } (\text{distr } M N f) M' X' I$
 $\langle \text{proof} \rangle$

Random variables that depend on disjoint sets of the components of a product space are independent.

lemma *make-ext*:
assumes $\bigwedge x. P x = P (\text{restrict } x I)$
shows $(\forall x \in \text{Pi } I A. P x) = (\forall x \in \text{PiE } I A. P x)$
 $\langle \text{proof} \rangle$

lemma *PiE-reindex*:
assumes $\text{inj-on } f I$
shows $\text{PiE } I (A \circ f) = (\lambda a. \text{restrict } (a \circ f) I) \text{ ` PiE } (f \text{ ` } I) A \text{ (is ?lhs = ?f ` ?rhs)}$
 $\langle \text{proof} \rangle$

lemma *(in prob-space) indep-sets-reindex*:
assumes $\text{inj-on } f I$
shows $\text{indep-sets } A (f \text{ ` } I) = \text{indep-sets } (\lambda i. A (f i)) I$
 $\langle \text{proof} \rangle$

lemma *(in prob-space) indep-vars-reindex*:
assumes $\text{inj-on } f I$
assumes $\text{indep-vars } M' X' (f \text{ ` } I)$
shows $\text{indep-vars } (M' \circ f) (\lambda k \omega. X' (f k) \omega) I$
 $\langle \text{proof} \rangle$

lemma *(in prob-space) variance-divide*:
fixes $f :: 'a \Rightarrow \text{real}$
assumes $\text{integrable } M f$
shows $\text{variance } (\lambda \omega. f \omega / r) = \text{variance } f / r^2$
 $\langle \text{proof} \rangle$

lemma *pmf-eq*:
assumes $\bigwedge x. x \in \text{set-pmf } \Omega \implies (x \in P) = (x \in Q)$
shows $\text{measure } (\text{measure-pmf } \Omega) P = \text{measure } (\text{measure-pmf } \Omega) Q$
 $\langle \text{proof} \rangle$

lemma *pmf-mono-1*:
assumes $\bigwedge x. x \in P \implies x \in \text{set-pmf } \Omega \implies x \in Q$
shows $\text{measure } (\text{measure-pmf } \Omega) P \leq \text{measure } (\text{measure-pmf } \Omega) Q$
 $\langle \text{proof} \rangle$

definition *(in prob-space) covariance where*
 $\text{covariance } f g = \text{expectation } (\lambda \omega. (f \omega - \text{expectation } f) * (g \omega - \text{expectation } g))$

lemma (in *prob-space*) *real-prod-integrable*:

fixes $f\ g :: 'a \Rightarrow \text{real}$

assumes [*measurable*]: $f \in \text{borel-measurable } M\ g \in \text{borel-measurable } M$

assumes *sq-int*: $\text{integrable } M\ (\lambda\omega. f\ \omega^{\wedge 2})\ \text{integrable } M\ (\lambda\omega. g\ \omega^{\wedge 2})$

shows $\text{integrable } M\ (\lambda\omega. f\ \omega * g\ \omega)$

<proof>

lemma (in *prob-space*) *covariance-eq*:

fixes $f :: 'a \Rightarrow \text{real}$

assumes $f \in \text{borel-measurable } M\ g \in \text{borel-measurable } M$

assumes $\text{integrable } M\ (\lambda\omega. f\ \omega^{\wedge 2})\ \text{integrable } M\ (\lambda\omega. g\ \omega^{\wedge 2})$

shows $\text{covariance } f\ g = \text{expectation } (\lambda\omega. f\ \omega * g\ \omega) - \text{expectation } f * \text{expectation } g$

g

<proof>

lemma (in *prob-space*) *covar-integrable*:

fixes $f\ g :: 'a \Rightarrow \text{real}$

assumes $f \in \text{borel-measurable } M\ g \in \text{borel-measurable } M$

assumes $\text{integrable } M\ (\lambda\omega. f\ \omega^{\wedge 2})\ \text{integrable } M\ (\lambda\omega. g\ \omega^{\wedge 2})$

shows $\text{integrable } M\ (\lambda\omega. (f\ \omega - \text{expectation } f) * (g\ \omega - \text{expectation } g))$

<proof>

lemma (in *prob-space*) *sum-square-int*:

fixes $f :: 'b \Rightarrow 'a \Rightarrow \text{real}$

assumes *finite I*

assumes $\bigwedge i. i \in I \implies f\ i \in \text{borel-measurable } M$

assumes $\bigwedge i. i \in I \implies \text{integrable } M\ (\lambda\omega. f\ i\ \omega^{\wedge 2})$

shows $\text{integrable } M\ (\lambda\omega. (\sum i \in I. f\ i\ \omega)^2)$

<proof>

lemma (in *prob-space*) *var-sum-1*:

fixes $f :: 'b \Rightarrow 'a \Rightarrow \text{real}$

assumes *finite I*

assumes $\bigwedge i. i \in I \implies f\ i \in \text{borel-measurable } M$

assumes $\bigwedge i. i \in I \implies \text{integrable } M\ (\lambda\omega. f\ i\ \omega^{\wedge 2})$

shows

$\text{variance } (\lambda\omega. (\sum i \in I. f\ i\ \omega)) = (\sum i \in I. (\sum j \in I. \text{covariance } (f\ i) (f\ j)))$

(*is ?lhs = ?rhs*)

<proof>

lemma (in *prob-space*) *covar-self-eq*:

fixes $f :: 'a \Rightarrow \text{real}$

shows $\text{covariance } f\ f = \text{variance } f$

<proof>

lemma (in *prob-space*) *covar-indep-eq-zero*:

fixes $f\ g :: 'a \Rightarrow \text{real}$

assumes $\text{integrable } M\ f$

assumes $\text{integrable } M\ g$

assumes *indep-var borel f borel g*
shows *covariance f g = 0*
 <proof>

lemma (*in prob-space*) *var-sum-2*:
fixes *f :: 'b \Rightarrow 'a \Rightarrow real*
assumes *finite I*
assumes $\bigwedge i. i \in I \implies f\ i \in \text{borel-measurable } M$
assumes $\bigwedge i. i \in I \implies \text{integrable } M\ (\lambda\omega. f\ i\ \omega^{\wedge 2})$
shows *variance* $(\lambda\omega. (\sum i \in I. f\ i\ \omega)) =$
 $(\sum i \in I. \text{variance } (f\ i)) + (\sum i \in I. \sum j \in I - \{i\}. \text{covariance } (f\ i) (f\ j))$
 <proof>

lemma (*in prob-space*) *var-sum-pairwise-indep*:
fixes *f :: 'b \Rightarrow 'a \Rightarrow real*
assumes *finite I*
assumes $\bigwedge i. i \in I \implies f\ i \in \text{borel-measurable } M$
assumes $\bigwedge i. i \in I \implies \text{integrable } M\ (\lambda\omega. f\ i\ \omega^{\wedge 2})$
assumes $\bigwedge i\ j. i \in I \implies j \in I \implies i \neq j \implies \text{indep-var borel } (f\ i) \text{ borel } (f\ j)$
shows *variance* $(\lambda\omega. (\sum i \in I. f\ i\ \omega)) = (\sum i \in I. \text{variance } (f\ i))$
 <proof>

lemma (*in prob-space*) *indep-var-from-indep-vars*:
assumes *i \neq j*
assumes *indep-vars* $(\lambda-. M')\ f\ \{i, j\}$
shows *indep-var* $M'\ (f\ i)\ M'\ (f\ j)$
 <proof>

lemma (*in prob-space*) *var-sum-pairwise-indep-2*:
fixes *f :: 'b \Rightarrow 'a \Rightarrow real*
assumes *finite I*
assumes $\bigwedge i. i \in I \implies f\ i \in \text{borel-measurable } M$
assumes $\bigwedge i. i \in I \implies \text{integrable } M\ (\lambda\omega. f\ i\ \omega^{\wedge 2})$
assumes $\bigwedge J. J \subseteq I \implies \text{card } J = 2 \implies \text{indep-vars } (\lambda -. \text{borel})\ f\ J$
shows *variance* $(\lambda\omega. (\sum i \in I. f\ i\ \omega)) = (\sum i \in I. \text{variance } (f\ i))$
 <proof>

lemma (*in prob-space*) *var-sum-all-indep*:
fixes *f :: 'b \Rightarrow 'a \Rightarrow real*
assumes *finite I*
assumes $\bigwedge i. i \in I \implies f\ i \in \text{borel-measurable } M$
assumes $\bigwedge i. i \in I \implies \text{integrable } M\ (\lambda\omega. f\ i\ \omega^{\wedge 2})$
assumes *indep-vars* $(\lambda -. \text{borel})\ f\ I$
shows *variance* $(\lambda\omega. (\sum i \in I. f\ i\ \omega)) = (\sum i \in I. \text{variance } (f\ i))$
 <proof>

end

9 Median

theory *Median*

imports *Main HOL-Probability.Hoeffding HOL-Library.Multiset Probability-Ext HOL.List*

begin

This section includes an amplification result for estimation algorithms using the median method.

fun *sort-primitive* **where**

sort-primitive $i\ j\ f\ k = (if\ k = i\ then\ min\ (f\ i)\ (f\ j)\ else\ (if\ k = j\ then\ max\ (f\ i)\ (f\ j)\ else\ f\ k))$

fun *sort-map* **where**

sort-map $f\ n = fold\ id\ [sort-primitive\ j\ i.\ i <- [0..<n], j <- [0..<i]]\ f$

lemma *sort-map-ind*:

sort-map $f\ (Suc\ n) = fold\ id\ [sort-primitive\ j\ n.\ j <- [0..<n]]\ (sort-map\ f\ n)$
 $\langle proof \rangle$

lemma *sort-map-strict-mono*:

fixes $f :: nat \Rightarrow 'b :: linorder$

shows $j < n \implies i < j \implies sort-map\ f\ n\ i \leq sort-map\ f\ n\ j$

$\langle proof \rangle$

lemma *sort-map-mono*:

fixes $f :: nat \Rightarrow 'b :: linorder$

shows $j < n \implies i \leq j \implies sort-map\ f\ n\ i \leq sort-map\ f\ n\ j$

$\langle proof \rangle$

lemma *sort-map-perm*:

fixes $f :: nat \Rightarrow 'b :: linorder$

shows $image-mset\ (sort-map\ f\ n)\ (mset\ [0..<n]) = image-mset\ f\ (mset\ [0..<n])$

$\langle proof \rangle$

lemma *sort-map-eq-sort*:

fixes $f :: nat \Rightarrow ('b :: linorder)$

shows $map\ (sort-map\ f\ n)\ [0..<n] = sort\ (map\ f\ [0..<n])\ (is\ ?A = ?B)$

$\langle proof \rangle$

definition *median* **where**

median $f\ n = sort\ (map\ f\ [0..<n])\ !\ (n\ div\ 2)$

lemma *median-alt-def*:

assumes $n > 0$

shows $median\ f\ n = (sort-map\ f\ n)\ (n\ div\ 2)$

$\langle proof \rangle$

definition *up-ray* $:: ('a :: linorder)\ set \Rightarrow bool$ **where**

$up\text{-ray } I = (\forall x y. x \in I \longrightarrow x \leq y \longrightarrow y \in I)$

lemma *up-ray-borel*:

assumes *up-ray* ($I :: ('a :: linorder\text{-topology}) set$)

shows $I \in borel$

<proof>

definition *down-ray* :: $('a :: linorder) set \Rightarrow bool$ **where**

$down\text{-ray } I = (\forall x y. y \in I \longrightarrow x \leq y \longrightarrow x \in I)$

lemma *down-ray-borel*:

assumes *down-ray* ($I :: ('a :: linorder\text{-topology}) set$)

shows $I \in borel$

<proof>

definition *interval* :: $('a :: linorder) set \Rightarrow bool$ **where**

$interval I = (\forall x y z. x \in I \longrightarrow z \in I \longrightarrow x \leq y \longrightarrow y \leq z \longrightarrow y \in I)$

lemma *interval-borel*:

assumes *interval* ($I :: ('a :: linorder\text{-topology}) set$)

shows $I \in borel$

<proof>

lemma *interval-rule*:

assumes *interval* I

assumes $a \leq x \ x \leq b$

assumes $a \in I$

assumes $b \in I$

shows $x \in I$

<proof>

lemma *sorted-int*:

assumes *interval* I

assumes *sorted* xs

assumes $k < length\ xs \ i \leq j \ j \leq k$

assumes $xs ! i \in I \ xs ! k \in I$

shows $xs ! j \in I$

<proof>

lemma *mid-in-interval*:

assumes $2 * length\ (filter\ (\lambda x. x \in I)\ xs) > length\ xs$

assumes *interval* I

assumes *sorted* xs

shows $xs ! (length\ xs \ div\ 2) \in I$

<proof>

lemma *median-est*:

assumes *interval* I

assumes $2 * card\ \{k. k < n \wedge f\ k \in I\} > n$

shows $\text{median } f \ n \in I$
 $\langle \text{proof} \rangle$

lemma *median-measurable*:

fixes $X :: \text{nat} \Rightarrow 'a \Rightarrow ('b :: \{\text{linorder}, \text{topological-space}, \text{linorder-topology}, \text{second-countable-topology}\})$
assumes $n \geq 1$
assumes $\bigwedge i. i < n \implies X \ i \in \text{measurable } M \ \text{borel}$
shows $(\lambda x. \text{median } (\lambda i. X \ i \ x) \ n) \in \text{measurable } M \ \text{borel}$
 $\langle \text{proof} \rangle$

lemma (*in prob-space*) *median-bound*:

fixes $n :: \text{nat}$
fixes $I :: ('b :: \{\text{linorder-topology}, \text{second-countable-topology}\}) \ \text{set}$
assumes *interval* I
assumes $\alpha > 0$
assumes $\varepsilon \in \{0 < .. < 1\}$
assumes *indep-vars* $(\lambda -. \text{borel}) \ X \ \{0 .. < n\}$
assumes $n \geq -\ln \varepsilon / (2 * \alpha^2)$
assumes $\bigwedge i. i < n \implies \mathcal{P}(\omega \text{ in } M. X \ i \ \omega \in I) \geq 1/2 + \alpha$
shows $\mathcal{P}(\omega \text{ in } M. \text{median } (\lambda i. X \ i \ \omega) \ n \in I) \geq 1 - \varepsilon$ (**is** $\mathcal{P}(\omega \text{ in } M. ?\text{lhs } \omega) \geq ?C$)
 $\langle \text{proof} \rangle$

lemma (*in prob-space*) *median-bound-1*:

fixes $a \ b :: \text{real}$
fixes $n :: \text{nat}$
assumes $\alpha > 0$
assumes $\varepsilon \in \{0 < .. < 1\}$
assumes *indep-vars* $(\lambda -. \text{borel}) \ X \ \{0 .. < n\}$
assumes $n \geq -\ln \varepsilon / (2 * \alpha^2)$
assumes $\bigwedge i. i < n \implies \mathcal{P}(\omega \text{ in } M. X \ i \ \omega \in \{a..b\}) \geq 1/2 + \alpha$
shows $\mathcal{P}(\omega \text{ in } M. \text{median } (\lambda i. X \ i \ \omega) \ n \in \{a..b\}) \geq 1 - \varepsilon$ (**is** $\mathcal{P}(\omega \text{ in } M. ?\text{lhs } \omega) \geq ?C$)
 $\langle \text{proof} \rangle$

lemma (*in prob-space*) *median-bound-2*:

fixes $\mu :: \text{real}$
fixes $\delta :: \text{real}$
assumes $\varepsilon \in \{0 < .. < 1\}$
assumes *indep-vars* $(\lambda -. \text{borel}) \ X \ \{0 .. < n\}$
assumes $n \geq -18 * \ln \varepsilon$
assumes $\bigwedge i. i < n \implies \mathcal{P}(\omega \text{ in } M. \text{abs } (X \ i \ \omega - \mu) > \delta) \leq 1/3$
shows $\mathcal{P}(\omega \text{ in } M. \text{abs } (\text{median } (\lambda i. X \ i \ \omega) \ n - \mu) \leq \delta) \geq 1 - \varepsilon$
 $\langle \text{proof} \rangle$

lemma *sorted-mono-map*:

assumes *sorted* xs
assumes *mono* f

```

shows sorted (map f xs)
  <proof>

lemma map-sort:
  assumes mono f
  shows sort (map f xs) = map f (sort xs)
  <proof>

lemma median-cong:
  assumes  $\bigwedge i. i < n \implies f\ i = g\ i$ 
  shows median f n = median g n
  <proof>

lemma median-restrict:
  assumes  $n > 0$ 
  shows median ( $\lambda i \in \{0..<n\}. f\ i$ ) n = median f n
  <proof>

lemma median-rat:
  assumes  $n > 0$ 
  shows real-of-rat (median f n) = median ( $\lambda i. \text{real-of-rat } (f\ i)$ ) n
  <proof>

lemma median-const:
  assumes  $k > 0$ 
  shows median ( $\lambda i \in \{0..<k\}. a$ ) k = a
  <proof>

end
theory Set-Ext
imports Main
begin

This is like card-vimage-inj but supports inj-on instead.

lemma card-vimage-inj-on:
  assumes inj-on f B
  assumes  $A \subseteq f^{-1} B$ 
  shows card ( $f^{-1} A \cap B$ ) = card A
  <proof>

lemma card-ordered-pairs:
  fixes  $M :: ('a :: \text{linorder}) \text{ set}$ 
  assumes finite M
  shows  $2 * \text{card } \{(x,y) \in M \times M. x < y\} = \text{card } M * (\text{card } M - 1)$ 
  <proof>

end

```

10 Order Statistics

theory *OrderStatistics*

imports *Main HOL-Library.Multiset List-Ext Multiset-Ext Set-Ext*
begin

This section contains definitions and results about order statistics.

definition *rank-of* :: 'a :: linorder \Rightarrow 'a set \Rightarrow nat **where** *rank-of* $x\ S = \text{card } \{y \in S. y < x\}$

The function *rank-of* returns the rank of an element within a set.

lemma *rank-mono*:

assumes *finite S*
shows $x \leq y \implies \text{rank-of } x\ S \leq \text{rank-of } y\ S$
 $\langle \text{proof} \rangle$

lemma *rank-mono-commute*:

assumes *finite S*
assumes $S \subseteq T$
assumes *strict-mono-on f T*
assumes $x \in T$
shows $\text{rank-of } x\ S = \text{rank-of } (f\ x)\ (f\ ` S)$
 $\langle \text{proof} \rangle$

definition *least* **where** $\text{least } k\ S = \{y \in S. \text{rank-of } y\ S < k\}$

The function *least* returns the k smallest elements of a finite set.

lemma *rank-strict-mono*:

assumes *finite S*
shows *strict-mono-on* $(\lambda x. \text{rank-of } x\ S)\ S$
 $\langle \text{proof} \rangle$

lemma *rank-of-image*:

assumes *finite S*
shows $(\lambda x. \text{rank-of } x\ S)\ ` S = \{0..<\text{card } S\}$
 $\langle \text{proof} \rangle$

lemma *card-least*:

assumes *finite S*
shows $\text{card } (\text{least } k\ S) = \min k\ (\text{card } S)$
 $\langle \text{proof} \rangle$

lemma *least-subset*: $\text{least } k\ S \subseteq S$

$\langle \text{proof} \rangle$

lemma *preserve-rank*:

assumes *finite S*

shows $\text{rank-of } x (\text{least } m \ S) = \min m (\text{rank-of } x \ S)$
 $\langle \text{proof} \rangle$

lemma *rank-insert*:
assumes *finite T*
shows $\text{rank-of } y (\text{insert } v \ T) = \text{of-bool } (v < y \wedge v \notin T) + \text{rank-of } y \ T$
 $\langle \text{proof} \rangle$

lemma *least-mono-commute*:
assumes *finite S*
assumes *strict-mono-on f S*
shows $f \circ \text{least } k \ S = \text{least } k \ (f \circ S)$
 $\langle \text{proof} \rangle$

lemma *least-insert*:
assumes *finite S*
shows $\text{least } k (\text{insert } x (\text{least } k \ S)) = \text{least } k (\text{insert } x \ S)$ (**is** *?lhs = ?rhs*)
 $\langle \text{proof} \rangle$

definition *count-le* **where** $\text{count-le } x \ M = \text{size } \{\#y \in \# \ M. \ y \leq x\# \}$
definition *count-less* **where** $\text{count-less } x \ M = \text{size } \{\#y \in \# \ M. \ y < x\# \}$

definition *nth-mset* $:: \text{nat} \Rightarrow ('a :: \text{linorder}) \text{multiset} \Rightarrow 'a$ **where**
 $\text{nth-mset } k \ M = \text{sorted-list-of-multiset } M \ ! \ k$

lemma *nth-mset-bound-left*:
assumes $k < \text{size } M$
assumes $\text{count-less } x \ M \leq k$
shows $x \leq \text{nth-mset } k \ M$
 $\langle \text{proof} \rangle$

lemma *nth-mset-bound-left-excl*:
assumes $k < \text{size } M$
assumes $\text{count-le } x \ M \leq k$
shows $x < \text{nth-mset } k \ M$
 $\langle \text{proof} \rangle$

lemma *nth-mset-bound-right*:
assumes $k < \text{size } M$
assumes $\text{count-le } x \ M > k$
shows $\text{nth-mset } k \ M \leq x$
 $\langle \text{proof} \rangle$

lemma *nth-mset-commute-mono*:
assumes *mono f*
assumes $k < \text{size } M$
shows $f (\text{nth-mset } k \ M) = \text{nth-mset } k \ (\text{image-mset } f \ M)$
 $\langle \text{proof} \rangle$

lemma *nth-mset-max*:
assumes *size* $A > k$
assumes $\bigwedge x. x \leq \text{nth-mset } k \ A \implies \text{count } A \ x \leq 1$
shows $\text{nth-mset } k \ A = \text{Max } (\text{least } (k+1) \ (\text{set-mset } A))$ **and** $\text{card } (\text{least } (k+1) \ (\text{set-mset } A)) = k+1$
 $\langle \text{proof} \rangle$

end

11 Counting Polynomials

theory *PolynomialCounting*
imports *MainHOL-Algebra.Polynomial-Divisibility* *HOL-Algebra.Polynomials*
HOL-Library.FuncSet
Set-Ext
begin

This section contains results about the count of polynomials with a given degree interpolating a certain number of points.

definition *bounded-degree-polynomials*
where *bounded-degree-polynomials* $F \ n = \{x. x \in \text{carrier } (\text{poly-ring } F) \wedge (\text{degree } x < n \vee x = [])\}$

lemma *bounded-degree-polynomials-length*:
 $\text{bounded-degree-polynomials } F \ n = \{x. x \in \text{carrier } (\text{poly-ring } F) \wedge \text{length } x \leq n\}$
 $\langle \text{proof} \rangle$

lemma *fin-degree-bounded*:
assumes *ring* F
assumes *finite* $(\text{carrier } F)$
shows *finite* $(\text{bounded-degree-polynomials } F \ n)$
 $\langle \text{proof} \rangle$

lemma *fin-fixed-degree*:
assumes *ring* F
assumes *finite* $(\text{carrier } F)$
shows *finite* $\{p. p \in \text{carrier } (\text{poly-ring } F) \wedge \text{length } p = n\}$
 $\langle \text{proof} \rangle$

lemma *nonzero-length-polynomials-count*:
assumes *ring* F
assumes *finite* $(\text{carrier } F)$
shows $\text{card } \{p. p \in \text{carrier } (\text{poly-ring } F) \wedge \text{length } p = \text{Suc } n\}$
 $= (\text{card } (\text{carrier } F) - 1) * \text{card } (\text{carrier } F) ^ n$
 $\langle \text{proof} \rangle$

lemma *fixed-degree-polynomials-count*:
assumes *ring* F
assumes *finite* $(\text{carrier } F)$

shows $\text{card } (\{p. p \in \text{carrier } (\text{poly-ring } F) \wedge \text{length } p = n\}) =$
 $(\text{if } n \geq 1 \text{ then } (\text{card } (\text{carrier } F) - 1) * (\text{card } (\text{carrier } F) ^{(n-1)}) \text{ else } 1)$
 $\langle \text{proof} \rangle$

lemma *bounded-degree-polynomials-count*:

assumes *ring* F

assumes *finite* $(\text{carrier } F)$

shows $\text{card } (\text{bounded-degree-polynomials } F \ n) = \text{card } (\text{carrier } F) ^n$

$\langle \text{proof} \rangle$

lemma *non-empty-bounded-degree-polynomials*:

assumes *ring* F

shows $\text{bounded-degree-polynomials } F \ k \neq \{\}$

$\langle \text{proof} \rangle$

11.1 Interpolation Polynomials

It is well known that over any field there is exactly one polynomial with degree at most $k - 1$ interpolating k points. That there is never more than one such polynomial follow from the fact that a polynomial of degree $k - 1$ cannot have more than $k - 1$ roots. This is already shown in HOL-Algebra in *field.size-roots-le-degree*. Existence is usually shown using Lagrange interpolation.

In the case of finite fields it is actually only necessary to show either that there is at most one such polynomial or at least one - because a function whose domain and co-domain has the same finite cardinality is injective if and only if it is surjective.

In the following a more generic result (over finite fields) is shown, counting the number of polynomials of degree $k + n - 1$ interpolating k points for non-negative n . As it turns out there are $(\text{card } (\text{carrier } F))^n$ such polynomials. The trick is to observe that, for a given fix on the coefficients of order k to $k + n - 1$ and the values at k points there is at most one fitting polynomial.

An alternative way of stating the above result is that there is bijection between the polynomials of degree $n + k - 1$ and the product space $F^k \times F^n$ where the first component is the evaluation of the polynomials at k distinct points and the second component are the coefficients of order at least k .

definition *split-poly* **where** *split-poly* $F \ K \ p =$

$(\text{restrict } (\text{ring.eval } F \ p) \ K, \ \lambda k. \text{ring.coeff } F \ p \ (k + \text{card } K))$

The bijection *split-poly* returns the evaluation of the polynomial at the points in K and the coefficients of order at least $\text{card } K$.

In the following it is shown that its image is a subset of the product space mentioned above, and that *split-poly* is injective and finally that its image is exactly that product space using cardinalities.

lemma *split-poly-image*:

assumes *field* F

assumes $K \subseteq \text{carrier } F$

shows $\text{split-poly } F \ K \text{ ' bounded-degree-polynomials } F \ (\text{card } K + n) \subseteq$
 $(K \rightarrow_E \text{carrier } F) \times \{f. \text{range } f \subseteq \text{carrier } F \wedge (\forall k \geq n. f \ k = \mathbf{0}_F)\}$

$\langle \text{proof} \rangle$

lemma *poly-neg-coeff*:

assumes *domain* F

assumes $x \in \text{carrier } (\text{poly-ring } F)$

shows $\text{ring.coeff } F \ (\ominus_{\text{poly-ring } F} x) \ k = \ominus_F \text{ring.coeff } F \ x \ k$

$\langle \text{proof} \rangle$

lemma *poly-subtract-coeff*:

assumes *domain* F

assumes $x \in \text{carrier } (\text{poly-ring } F)$

assumes $y \in \text{carrier } (\text{poly-ring } F)$

shows $\text{ring.coeff } F \ (x \ominus_{\text{poly-ring } F} y) \ k = \text{ring.coeff } F \ x \ k \ominus_F \text{ring.coeff } F \ y \ k$

$\langle \text{proof} \rangle$

lemma *poly-subtract-eval*:

assumes *domain* F

assumes $i \in \text{carrier } F$

assumes $x \in \text{carrier } (\text{poly-ring } F)$

assumes $y \in \text{carrier } (\text{poly-ring } F)$

shows $\text{ring.eval } F \ (x \ominus_{\text{poly-ring } F} y) \ i = \text{ring.eval } F \ x \ i \ominus_F \text{ring.eval } F \ y \ i$

$\langle \text{proof} \rangle$

lemma *poly-degree-bound-from-coeff*:

assumes *ring* F

assumes $x \in \text{carrier } (\text{poly-ring } F)$

assumes $\bigwedge k. k \geq n \implies \text{ring.coeff } F \ x \ k = \mathbf{0}_F$

shows $\text{degree } x < n \vee x = \mathbf{0}_{\text{poly-ring } F}$

$\langle \text{proof} \rangle$

lemma *max-roots*:

assumes *field* R

assumes $p \in \text{carrier } (\text{poly-ring } R)$

assumes $K \subseteq \text{carrier } R$

assumes *finite* K

assumes $\text{degree } p < \text{card } K$

assumes $\bigwedge x. x \in K \implies \text{ring.eval } R \ p \ x = \mathbf{0}_R$

shows $p = \mathbf{0}_{\text{poly-ring } R}$

$\langle \text{proof} \rangle$

lemma *split-poly-inj*:

assumes *field* F

assumes *finite* K

assumes $K \subseteq \text{carrier } F$

shows *inj-on* (*split-poly* F K) (*carrier* (*poly-ring* F))
 $\langle \text{proof} \rangle$

lemma

assumes *field* $F \wedge$ *finite* (*carrier* F)

shows

poly-count:card (*bounded-degree-polynomials* F n) = *card* (*carrier* F) n (**is** $?A$)

and

finite-poly-count: finite (*bounded-degree-polynomials* F n) (**is** $?B$)

$\langle \text{proof} \rangle$

lemma

assumes *finite* ($B :: 'b$ set)

assumes $y \in B$

shows

card-mostly-constant-maps:

card $\{f. \text{range } f \subseteq B \wedge (\forall x. x \geq n \longrightarrow f x = y)\} = \text{card } B \wedge^n$ (**is** *card* $?A = ?B$) **and**

finite-mostly-constant-maps:

finite $\{f. \text{range } f \subseteq B \wedge (\forall x. x \geq n \longrightarrow f x = y)\}$

$\langle \text{proof} \rangle$

lemma *split-poly-surj:*

assumes *field* F

assumes *finite* (*carrier* F)

assumes $K \subseteq \text{carrier } F$

shows *split-poly* F K ' *bounded-degree-polynomials* F (*card* $K + n$) =

$(K \rightarrow_E \text{carrier } F) \times \{f. \text{range } f \subseteq \text{carrier } F \wedge (\forall k \geq n. f k = \mathbf{0}_F)\}$

(**is** *split-poly* F K ' $?A = ?B$)

$\langle \text{proof} \rangle$

lemma *inv-subsetI:*

assumes $\bigwedge x. x \in A \Longrightarrow f x \in B \Longrightarrow x \in C$

shows $f - ' B \cap A \subseteq C$

$\langle \text{proof} \rangle$

lemma *interpolating-polynomials-count:*

assumes *field* F

assumes *finite* (*carrier* F)

assumes $K \subseteq \text{carrier } F$

assumes $f ' K \subseteq \text{carrier } F$

shows *card* $\{\omega \in \text{bounded-degree-polynomials } F (\text{card } K + n). (\forall k \in K. \text{ring.eval } F \omega k = f k)\} =$

card (*carrier* F) n

(**is** *card* $?A = ?B$)

$\langle \text{proof} \rangle$

end

12 Indexed Products of Probability Mass Functions

This section introduces a restricted version of *Pi-pmf* where the default value is undefined and contains some additional results about that case in addition to `HOL-Probability.Product_PMF`

theory *Product-PMF-Ext*

imports *Main Probability-Ext HOL-Probability.Product-PMF*

begin

definition *prod-pmf* **where** *prod-pmf I M = Pi-pmf I undefined M*

lemma *pmf-prod-pmf*:

assumes *finite I*

shows *pmf (prod-pmf I M) x = (if x ∈ extensional I then $\prod_{i \in I} (\text{pmf } (M \ i)) (x \ i)$ else 0)*

<proof>

lemma *set-prod-pmf*:

assumes *finite I*

shows *set-pmf (prod-pmf I M) = PiE I (set-pmf ∘ M)*

<proof>

lemma *set-pmf-iff'*: *x ∉ set-pmf M ⟷ pmf M x = 0*

<proof>

lemma *prob-prod-pmf*:

assumes *finite I*

shows *measure (measure-pmf (prod-pmf I M)) (Pi I A) = ($\prod_{i \in I} \text{measure } (M \ i) (A \ i)$)*

<proof>

lemma *prob-prod-pmf'*:

assumes *finite I*

assumes *J ⊆ I*

shows *measure (measure-pmf (prod-pmf I M)) (Pi J A) = ($\prod_{i \in J} \text{measure } (M \ i) (A \ i)$)*

<proof>

lemma *prob-prod-pmf-slice*:

assumes *finite I*

assumes *i ∈ I*

shows *measure (measure-pmf (prod-pmf I M)) {ω. P (ω i)} = measure (M i) {ω. P ω}*

<proof>

lemma *range-inter*: *range ((∩) F) = Pow F*

<proof>

On a finite set M the σ -Algebra generated by singletons and the empty set is already the power set of M .

lemma *sigma-sets-singletons-and-empty:*
assumes *countable* M
shows *sigma-sets* M (*insert* $\{\}$ $((\lambda k. \{k\}) \text{ ` } M)) = \text{Pow } M$
 $\langle \text{proof} \rangle$

lemma *indep-vars-pmf:*
assumes $\bigwedge a J. J \subseteq I \implies \text{finite } J \implies$
 $\mathcal{P}(\omega \text{ in measure-pmf } M. \forall i \in J. X \ i \ \omega = a \ i) = (\prod i \in J. \mathcal{P}(\omega \text{ in measure-pmf } M. X \ i \ \omega = a \ i))$
shows *prob-space.indep-vars* (*measure-pmf* M) $(\lambda i. \text{measure-pmf } (M' \ i)) \ X \ I$
 $\langle \text{proof} \rangle$

lemma *indep-vars-restrict:*
fixes $M :: 'a \Rightarrow 'b \text{ pmf}$
fixes $J :: 'c \text{ set}$
assumes *disjoint-family-on* $f \ J$
assumes $J \neq \{\}$
assumes $\bigwedge i. i \in J \implies f \ i \subseteq I$
assumes *finite* I
shows *prob-space.indep-vars* (*measure-pmf* (*prod-pmf* $I \ M$)) $(\lambda i. \text{measure-pmf } (\text{prod-pmf } (f \ i) \ M)) (\lambda i \ \omega. \text{restrict } \omega \ (f \ i)) \ J$
 $\langle \text{proof} \rangle$

lemma *indep-vars-restrict-intro:*
fixes $M :: 'a \Rightarrow 'b \text{ pmf}$
fixes $J :: 'c \text{ set}$
assumes $\bigwedge \omega \ i. i \in J \implies X \ i \ \omega = X \ i \ (\text{restrict } \omega \ (f \ i))$
assumes *disjoint-family-on* $f \ J$
assumes $J \neq \{\}$
assumes $\bigwedge i. i \in J \implies f \ i \subseteq I$
assumes *finite* I
assumes $\bigwedge \omega \ i. i \in J \implies X \ i \ \omega \in \text{space } (M' \ i)$
shows *prob-space.indep-vars* (*measure-pmf* (*prod-pmf* $I \ M$)) $M' \ (\lambda i \ \omega. X \ i \ \omega) \ J$
 $\langle \text{proof} \rangle$

lemma *has-bochner-integral-prod-pmfI:*
fixes $f :: 'a \Rightarrow 'b \Rightarrow ('c :: \{\text{second-countable-topology, banach, real-normed-field}\})$
assumes *finite* I
assumes $\bigwedge i. i \in I \implies \text{has-bochner-integral } (\text{measure-pmf } (M \ i)) \ (f \ i) \ (r \ i)$
shows *has-bochner-integral* (*prod-pmf* $I \ M$) $(\lambda x. (\prod i \in I. f \ i \ (x \ i))) (\prod i \in I. r \ i)$
 $\langle \text{proof} \rangle$

lemma
fixes $f :: 'a \Rightarrow 'b \Rightarrow ('c :: \{\text{second-countable-topology, banach, real-normed-field}\})$
assumes *finite* I
assumes $\bigwedge i. i \in I \implies \text{integrable } (\text{measure-pmf } (M \ i)) \ (f \ i)$

shows *prod-pmf-integrable*: *integrable* (*prod-pmf* *I* *M*) ($\lambda x. (\prod i \in I. f\ i\ (x\ i))$)
(is ?A) and
prod-pmf-integral: *integral*^L (*prod-pmf* *I* *M*) ($\lambda x. (\prod i \in I. f\ i\ (x\ i))$) =
 $(\prod i \in I. \text{integral}^L\ (M\ i)\ (f\ i))$ **(is ?B)**
<proof>

lemma *has-bochner-integral-prod-pmf-sliceI*:
fixes *f* :: 'a \Rightarrow ('b :: {*second-countable-topology*,*banach*,*real-normed-field*})
assumes *finite* *I*
assumes *i* \in *I*
assumes *has-bochner-integral* (*measure-pmf* (*M* *i*)) (*f*) *r*
shows *has-bochner-integral* (*prod-pmf* *I* *M*) ($\lambda x. (f\ (x\ i))$) *r*
<proof>

lemma
fixes *f* :: 'a \Rightarrow ('b :: {*second-countable-topology*,*banach*,*real-normed-field*})
assumes *finite* *I*
assumes *i* \in *I*
assumes *integrable* (*measure-pmf* (*M* *i*)) *f*
shows *integrable-prod-pmf-slice*: *integrable* (*prod-pmf* *I* *M*) ($\lambda x. (f\ (x\ i))$) **(is ?A)**
and
integral-prod-pmf-slice: *integral*^L (*prod-pmf* *I* *M*) ($\lambda x. (f\ (x\ i))$) = *integral*^L (*M* *i*) *f* **(is ?B)**
<proof>

lemma *variance-prod-pmf-slice*:
fixes *f* :: 'a \Rightarrow *real*
assumes *i* \in *I* *finite* *I*
assumes *integrable* (*measure-pmf* (*M* *i*)) ($\lambda \omega. f\ \omega^{\wedge} 2$)
shows *prob-space.variance* (*prod-pmf* *I* *M*) ($\lambda \omega. f\ (\omega\ i)$) = *prob-space.variance* (*M* *i*) *f*
<proof>

lemma *PiE-default-undefined-eq*: *PiE-dflt* *I* *undefined* *M* = *PiE* *I* *M*
<proof>

lemma *pmf-of-set-prod*:
assumes *finite* *I*
assumes $\bigwedge x. x \in I \implies \text{finite}\ (M\ x)$
assumes $\bigwedge x. x \in I \implies M\ x \neq \{\}$
shows *pmf-of-set* (*PiE* *I* *M*) = *prod-pmf* *I* ($\lambda i. \text{pmf-of-set}\ (M\ i)$)
<proof>

lemma *extensionality-iff*:
assumes *f* \in *extensional* *I*
shows $((\lambda i \in I. g\ i) = f) = (\forall i \in I. g\ i = f\ i)$

$\langle \text{proof} \rangle$

lemma *of-bool-prod*:

assumes *finite I*

shows *of-bool* $(\forall i \in I. P\ i) = (\prod i \in I. (\text{of-bool}\ (P\ i) :: 'a :: \text{field}))$

$\langle \text{proof} \rangle$

lemma *map-ptw*:

fixes *I* :: *'a set*

fixes *M* :: *'a \Rightarrow 'b pmf*

fixes *f* :: *'b \Rightarrow 'c*

assumes *finite I*

shows *prod-pmf* *I M* $\gg= (\lambda x. \text{return-pmf}\ (\lambda i \in I. f\ (x\ i))) = \text{prod-pmf}\ I\ (\lambda i. (M\ i\ \gg= (\lambda x. \text{return-pmf}\ (f\ x))))$

$\langle \text{proof} \rangle$

lemma *pair-pmfI*:

A $\gg= (\lambda a. B\ \gg= (\lambda b. \text{return-pmf}\ (f\ a\ b))) = \text{pair-pmf}\ A\ B\ \gg= (\lambda (a,b). \text{return-pmf}\ (f\ a\ b))$

$\langle \text{proof} \rangle$

lemma *pmf-pair'*:

pmf (*pair-pmf* *M N*) *x* = *pmf* *M* (*fst* *x*) * *pmf* *N* (*snd* *x*)

$\langle \text{proof} \rangle$

lemma *pair-pmf-ptw*:

assumes *finite I*

shows *pair-pmf* (*prod-pmf* *I A* :: $((i \Rightarrow 'a)\ \text{pmf})$) (*prod-pmf* *I B* :: $((i \Rightarrow 'b)\ \text{pmf})$) =

prod-pmf *I* $(\lambda i. \text{pair-pmf}\ (A\ i)\ (B\ i))\ \gg=$

$(\lambda f. \text{return-pmf}\ (\text{restrict}\ (\text{fst} \circ f)\ I, \text{restrict}\ (\text{snd} \circ f)\ I))$

(**is** *?lhs* = *?rhs*)

$\langle \text{proof} \rangle$

end

13 Universal Hash Families

theory *UniversalHashFamily*

imports *Main PolynomialCounting Product-PMF-Ext*

begin

A *k*-universal hash family \mathcal{H} is probability space, whose elements are hash functions with domain *U* and range *i.i* < *m* such that:

- For every fixed $x \in U$ and value $y < m$ exactly $\frac{1}{m}$ of the hash functions map x to y : $P_{h \in \mathcal{H}} (h(x) = y) = \frac{1}{m}$.
- For at most *k* universe elements: x_1, \dots, x_m the functions $h(x_1), \dots, h(x_m)$

are independent random variables.

In this section, we construct k -universal hash families following the approach outlined by Wegman and Carter using the polynomials of degree less than k over a finite field.

A hash function is just polynomial evaluation.

definition *hash* :: ('a, 'b) ring-scheme \Rightarrow 'a \Rightarrow 'a list \Rightarrow 'a
where *hash* F x ω = *ring.eval* F ω x

lemma *hash-range*:

assumes *ring* F
assumes $\omega \in$ *bounded-degree-polynomials* F n
assumes $x \in$ *carrier* F
shows *hash* F x $\omega \in$ *carrier* F
 \langle *proof* \rangle

lemma *hash-range-2*:

assumes *ring* F
assumes $\omega \in$ *bounded-degree-polynomials* F n
shows $(\lambda x. \text{hash } F \ x \ \omega) \text{ 'carrier } F \subseteq \text{carrier } F$
 \langle *proof* \rangle

lemma *poly-cards*:

assumes *field* $F \wedge$ *finite* (*carrier* F)
assumes $K \subseteq$ *carrier* F
assumes *card* $K \leq n$
assumes $y \text{ ' } K \subseteq$ (*carrier* F)
shows *card* $\{\omega \in$ *bounded-degree-polynomials* F $n. (\forall k \in K. \text{ring.eval } F \ \omega \ k =$
 $y \ k)\} =$
 $\text{card } (\text{carrier } F) \frown (n - \text{card } K)$
 \langle *proof* \rangle

lemma *poly-cards-single*:

assumes *field* $F \wedge$ *finite* (*carrier* F)
assumes $k \in$ *carrier* F
assumes $1 \leq n$
assumes $y \in$ *carrier* F
shows *card* $\{\omega \in$ *bounded-degree-polynomials* F $n. \text{ring.eval } F \ \omega \ k = y\} =$
 $\text{card } (\text{carrier } F) \frown (n - 1)$
 \langle *proof* \rangle

lemma *expand-subset-filter*: $\{x \in A. P \ x\} = A \cap \{x. P \ x\}$
 \langle *proof* \rangle

lemma *hash-prob*:

assumes *field* $F \wedge$ *finite* (*carrier* F)
assumes $K \subseteq$ *carrier* F
assumes *card* $K \leq n$

assumes $y \in K \subseteq \text{carrier } F$
shows $\mathcal{P}(\omega \text{ in pmf-of-set (bounded-degree-polynomials } F \ n). (\forall x \in K. \text{hash } F \ x \ \omega = y \ x)) = 1 / (\text{real } (\text{card } (\text{carrier } F)))^{\text{card } K}$
 <proof>

lemma *hash-prob-single*:

assumes $\text{field } F \wedge \text{finite } (\text{carrier } F)$
assumes $x \in \text{carrier } F$
assumes $1 \leq n$
assumes $y \in \text{carrier } F$
shows $\mathcal{P}(\omega \text{ in pmf-of-set (bounded-degree-polynomials } F \ n). \text{hash } F \ x \ \omega = y) = 1 / (\text{real } (\text{card } (\text{carrier } F)))$
 <proof>

lemma *hash-indep-pmf*:

assumes $\text{field } F \wedge \text{finite } (\text{carrier } F)$
assumes $J \subseteq \text{carrier } F$
assumes $\text{finite } J$
assumes $\text{card } J \leq n$
assumes $1 \leq n$
shows $\text{prob-space.indep-vars (pmf-of-set (bounded-degree-polynomials } F \ n)) (\lambda \cdot. \text{pmf-of-set } (\text{carrier } F)) (\text{hash } F) \ J$
 <proof>

We introduce k-wise independent random variables using the existing definition of independent random variables.

definition (in *prob-space*) *k-wise-indep-vars* ::

$\text{nat} \Rightarrow ('b \Rightarrow 'c \text{ measure}) \Rightarrow ('b \Rightarrow 'a \Rightarrow 'c) \Rightarrow 'b \text{ set} \Rightarrow \text{bool}$ **where**
 $k\text{-wise-indep-vars } k \ M' \ X' \ I = (\forall J \subseteq I. \text{card } J \leq k \longrightarrow \text{finite } J \longrightarrow \text{indep-vars } M' \ X' \ J)$

lemma *hash-k-wise-indep*:

assumes $\text{field } F \wedge \text{finite } (\text{carrier } F)$
assumes $1 \leq n$
shows $\text{prob-space.k-wise-indep-vars (pmf-of-set (bounded-degree-polynomials } F \ n)) \ n$
 $(\lambda \cdot. \text{pmf-of-set } (\text{carrier } F)) (\text{hash } F) (\text{carrier } F)$
 <proof>

lemma *hash-inj-if-degree-1*:

assumes $\text{field } F \wedge \text{finite } (\text{carrier } F)$
assumes $\omega \in \text{bounded-degree-polynomials } F \ n$
assumes $\text{degree } \omega = 1$
shows $\text{inj-on } (\lambda x. \text{hash } F \ x \ \omega) (\text{carrier } F)$
 <proof>

lemma (in *prob-space*) *k-wise-subset*:

assumes $k\text{-wise-indep-vars } k \ M' \ X' \ I$
assumes $J \subseteq I$

shows *k-wise-indep-vars* k M' X' J
 $\langle \text{proof} \rangle$

end

14 Universal Hash Family for $\{0.. < p\}$

Specialization of universal hash families from arbitrary finite fields to $\{0.. < p\}$.

theory *UniversalHashFamilyOfPrime*

imports *Field UniversalHashFamily Probability-Ext Encoding*
begin

lemma *fin-bounded-degree-polynomials*:

assumes $p > 0$

shows *finite* (*bounded-degree-polynomials* (*ZFact* (*int* p)) n)

$\langle \text{proof} \rangle$

lemma *ne-bounded-degree-polynomials*:

shows *bounded-degree-polynomials* (*ZFact* (*int* p)) $n \neq \{\}$

$\langle \text{proof} \rangle$

lemma *card-bounded-degree-polynomials*:

assumes $p > 0$

shows *card* (*bounded-degree-polynomials* (*ZFact* (*int* p)) n) = p^n

$\langle \text{proof} \rangle$

fun *hash* :: $\text{nat} \Rightarrow \text{nat} \Rightarrow \text{int set list} \Rightarrow \text{nat}$

where *hash* p x f = *the-inv-into* $\{0..<p\}$ (*zfact-embed* p) (*UniversalHashFamily.hash* (*ZFact* p) (*zfact-embed* p x) f)

declare *hash.simps* [*simp del*]

lemma *hash-range*:

assumes $p > 0$

assumes $\omega \in \text{bounded-degree-polynomials}$ (*ZFact* (*int* p)) n

assumes $x < p$

shows *hash* p x $\omega < p$

$\langle \text{proof} \rangle$

lemma *hash-inj-if-degree-1*:

assumes *prime* p

assumes $\omega \in \text{bounded-degree-polynomials}$ (*ZFact* (*int* p)) n

assumes *degree* $\omega = 1$

shows *inj-on* $(\lambda x. \text{hash } p \ x \ \omega)$ $\{0..<p\}$

$\langle \text{proof} \rangle$

lemma *hash-prob*:

```

assumes prime p
assumes  $K \subseteq \{0..<p\}$ 
assumes  $y \text{ ' } K \subseteq \{0..<p\}$ 
assumes  $\text{card } K \leq n$ 
shows  $\mathcal{P}(\omega \text{ in measure-pmf (pmf-of-set (bounded-degree-polynomials (ZFact (int$ 
 $p)) \text{ } n)).$ 
 $(\forall x \in K. \text{hash } p \text{ } x \text{ } \omega = (y \text{ } x))) = 1 / \text{real } p^{\text{card } K}$ 
<proof>

```

```

lemma hash-prob-2:
assumes prime p
assumes inj-on x K
assumes  $x \text{ ' } K \subseteq \{0..<p\}$ 
assumes  $y \text{ ' } K \subseteq \{0..<p\}$ 
assumes  $\text{card } K \leq n$ 
shows  $\mathcal{P}(\omega \text{ in measure-pmf (pmf-of-set (bounded-degree-polynomials (ZFact (int$ 
 $p)) \text{ } n)).$ 
 $(\forall k \in K. \text{hash } p \text{ } (x \text{ } k) \text{ } \omega = (y \text{ } k))) = 1 / \text{real } p^{\text{card } K}$  (is ?lhs = ?rhs)
<proof>

```

```

lemma hash-prob-range:
assumes prime p
assumes  $x < p$ 
assumes  $n > 0$ 
shows  $\mathcal{P}(\omega \text{ in measure-pmf (pmf-of-set (bounded-degree-polynomials (ZFact (int$ 
 $p)) \text{ } n)).$ 
 $\text{hash } p \text{ } x \text{ } \omega \in A = \text{card } (A \cap \{0..<p\}) / p$ 
<proof>

```

```

lemma hash-k-wise-indep:
assumes prime p
assumes  $1 \leq n$ 
shows prob-space.k-wise-indep-vars (measure-pmf (pmf-of-set (bounded-degree-polynomials
(ZFact (int p)) n)))
 $n (\lambda-. \text{pmf-of-set } \{0..<p\}) (\text{hash } p) \{0..<p\}$ 
<proof>

```

14.1 Encoding

```

fun zfactS where zfactS p x = (
  if  $x \in \text{zfact-embed } p \text{ ' } \{0..<p\}$  then
     $N_S (\text{the-inv-into } \{0..<p\} (\text{zfact-embed } p) \text{ } x)$ 
  else
    None
)

```

```

lemma zfact-encoding :
  is-encoding (zfactS p)
<proof>

```

```

lemma bounded-degree-polynomial-bit-count:
  assumes  $p > 0$ 
  assumes  $x \in \text{bounded-degree-polynomials } (\text{ZFact } p) \ n$ 
  shows  $\text{bit-count } (\text{list}_S (\text{zfact}_S \ p) \ x) \leq \text{ereal } (\text{real } n * (2 * \log 2 \ p + 2) + 1)$ 
   $\langle \text{proof} \rangle$ 

end

```

15 Landau Symbols

```

theory Landau-Ext
  imports HOL-Library.Landau-Symbols HOL.Topological-Spaces
begin

```

This section contains results about Landau Symbols in addition to "HOL-Library.Landau".

The following lemma is an intentional copy of *sum-in-bigo* with order of assumptions reversed *)

```

lemma sum-in-bigo-r:
  assumes  $f2 \in O[F'](g)$ 
  assumes  $f1 \in O[F'](g)$ 
  shows  $(\lambda x. f1 \ x + f2 \ x) \in O[F'](g)$ 
   $\langle \text{proof} \rangle$ 

```

```

lemma landau-sum:
  assumes eventually  $(\lambda x. g1 \ x \geq (0::\text{real})) \ F'$ 
  assumes eventually  $(\lambda x. g2 \ x \geq 0) \ F'$ 
  assumes  $f1 \in O[F'](g1)$ 
  assumes  $f2 \in O[F'](g2)$ 
  shows  $(\lambda x. f1 \ x + f2 \ x) \in O[F'](\lambda x. g1 \ x + g2 \ x)$ 
   $\langle \text{proof} \rangle$ 

```

```

lemma landau-sum-1:
  assumes eventually  $(\lambda x. g1 \ x \geq (0::\text{real})) \ F'$ 
  assumes eventually  $(\lambda x. g2 \ x \geq 0) \ F'$ 
  assumes  $f \in O[F'](g1)$ 
  shows  $f \in O[F'](\lambda x. g1 \ x + g2 \ x)$ 
   $\langle \text{proof} \rangle$ 

```

```

lemma landau-sum-2:
  assumes eventually  $(\lambda x. g1 \ x \geq (0::\text{real})) \ F'$ 
  assumes eventually  $(\lambda x. g2 \ x \geq 0) \ F'$ 
  assumes  $f \in O[F'](g2)$ 
  shows  $f \in O[F'](\lambda x. g1 \ x + g2 \ x)$ 
   $\langle \text{proof} \rangle$ 

```

```

lemma landau-ln-3:

```

assumes *eventually* $(\lambda x. (1::\text{real}) \leq f\ x) F'$
assumes $f \in O[F'](g)$
shows $(\lambda x. \ln (f\ x)) \in O[F'](g)$
 $\langle \text{proof} \rangle$

lemma *landau-ln-2*:
assumes $a > (1::\text{real})$
assumes *eventually* $(\lambda x. 1 \leq f\ x) F'$
assumes *eventually* $(\lambda x. a \leq g\ x) F'$
assumes $f \in O[F'](g)$
shows $(\lambda x. \ln (f\ x)) \in O[F'](\lambda x. \ln (g\ x))$
 $\langle \text{proof} \rangle$

lemma *landau-real-nat*:
fixes $f :: 'a \Rightarrow \text{int}$
assumes $(\lambda x. \text{of-int } (f\ x)) \in O[F'](g)$
shows $(\lambda x. \text{real } (\text{nat } (f\ x))) \in O[F'](g)$
 $\langle \text{proof} \rangle$

lemma *landau-ceil*:
assumes $(\lambda x. 1) \in O[F'](g)$
assumes $f \in O[F'](g)$
shows $(\lambda x. \text{real-of-int } \lceil f\ x \rceil) \in O[F'](g)$
 $\langle \text{proof} \rangle$

lemma *landau-nat-ceil*:
assumes $(\lambda x. 1) \in O[F'](g)$
assumes $f \in O[F'](g)$
shows $(\lambda x. \text{real } (\text{nat } \lceil f\ x \rceil)) \in O[F'](g)$
 $\langle \text{proof} \rangle$

lemma *landau-const-inv*:
assumes $c > (0::\text{real})$
assumes $(\lambda x. 1 / f\ x) \in O[F'](g)$
shows $(\lambda x. c / f\ x) \in O[F'](g)$
 $\langle \text{proof} \rangle$

lemma *eventually-nonneg-div*:
assumes *eventually* $(\lambda x. (0::\text{real}) \leq f\ x) F'$
assumes *eventually* $(\lambda x. 0 < g\ x) F'$
shows *eventually* $(\lambda x. 0 \leq f\ x / g\ x) F'$
 $\langle \text{proof} \rangle$

lemma *eventually-nonneg-add*:
assumes *eventually* $(\lambda x. (0::\text{real}) \leq f\ x) F'$
assumes *eventually* $(\lambda x. 0 \leq g\ x) F'$
shows *eventually* $(\lambda x. 0 \leq f\ x + g\ x) F'$
 $\langle \text{proof} \rangle$

```

lemma eventually-ln-ge-iff:
  assumes eventually ( $\lambda x. (\exp (c::real)) \leq f x$ )  $F'$ 
  shows eventually ( $\lambda x. c \leq \ln (f x)$ )  $F'$ 
   $\langle proof \rangle$ 

lemma div-commute:  $(a::real) / b = (1/b) * a$   $\langle proof \rangle$ 

lemma eventually-prod1':
  assumes  $B \neq bot$ 
  shows  $(\forall_F x \text{ in } A \times_F B. P (fst x)) \longleftrightarrow (\forall_F x \text{ in } A. P x)$ 
   $\langle proof \rangle$ 

lemma eventually-prod2':
  assumes  $A \neq bot$ 
  shows  $(\forall_F x \text{ in } A \times_F B. P (snd x)) \longleftrightarrow (\forall_F x \text{ in } B. P x)$ 
   $\langle proof \rangle$ 

instantiation rat :: linorder-topology
begin

definition open-rat :: rat set  $\Rightarrow$  bool
  where open-rat = generate-topology ( $range (\lambda a. \{..< a\}) \cup range (\lambda a. \{a <..\})$ )

instance
   $\langle proof \rangle$ 
end

lemma inv-at-right-0-inf:
   $\forall_F x \text{ in } at\_right\ 0. c \leq 1 / real\_of\_rat\ x$ 
   $\langle proof \rangle$ 

end

```

16 Frequency Moment 0

```

theory Frequency-Moment-0
  imports Main Primes-Ext Float-Ext Median OrderStatistics UniversalHashFamilyOfPrime Encoding
  Frequency-Moments Landau-Ext
begin

```

This section contains a formalization of the algorithm for the zero-th frequency moment. It is a KMV algorithm with a rounding method to match the space complexity of the best algorithm described in [2].

In addition of the Isabelle proof here, there is also an informal hand-writtend proof in Appendix A.

```

type-synonym f0-state = nat  $\times$  nat  $\times$  nat  $\times$  nat  $\times$  (nat  $\Rightarrow$  (int set list))  $\times$  (nat  $\Rightarrow$  float set)

```

```

fun f0-init :: rat  $\Rightarrow$  rat  $\Rightarrow$  nat  $\Rightarrow$  f0-state pmf where
  f0-init  $\delta$   $\varepsilon$   $n$  =
    do {
      let  $s$  = nat  $\lceil -18 * \ln (\text{real-of-rat } \varepsilon) \rceil$ ;
      let  $t$  = nat  $\lceil 80 / (\text{real-of-rat } \delta)^2 \rceil$ ;
      let  $p$  = find-prime-above (max  $n$  19);
      let  $r$  = nat  $(4 * \lceil \log 2 (1 / \text{real-of-rat } \delta) \rceil + 24)$ ;
       $h \leftarrow \text{prod-pmf } \{0..<s\} (\lambda-. \text{pmf-of-set } (\text{bounded-degree-polynomials } (\text{ZFact } (\text{int } p)) 2))$ ;
      return-pmf ( $s, t, p, r, h, (\lambda-. \{0..<s\}. \{\})$ )
    }

```

```

fun f0-update :: nat  $\Rightarrow$  f0-state  $\Rightarrow$  f0-state pmf where
  f0-update  $x (s, t, p, r, h, \text{sketch})$  =
    return-pmf ( $s, t, p, r, h, \lambda i \in \{0..<s\}. \text{least } t (\text{insert } (\text{float-of } (\text{truncate-down } r (\text{hash } p \ x \ (h \ i)))) (\text{sketch } i)))$ )

```

```

fun f0-result :: f0-state  $\Rightarrow$  rat pmf where
  f0-result ( $s, t, p, r, h, \text{sketch}$ ) = return-pmf (median ( $\lambda i \in \{0..<s\}. \text{if card } (\text{sketch } i) < t \text{ then of-nat } (\text{card } (\text{sketch } i)) \text{ else rat-of-nat } t * \text{rat-of-nat } p / \text{rat-of-float } (\text{Max } (\text{sketch } i)))$ )
    )  $s$ )

```

```

definition f0-sketch where
  f0-sketch  $p \ r \ t \ h \ x s$  = least  $t ((\lambda x. \text{float-of } (\text{truncate-down } r (\text{hash } p \ x \ h))) \text{ ' (set } xs))$ 

```

```

lemma f0-alg-sketch:
  fixes  $n :: \text{nat}$ 
  fixes  $as :: \text{nat list}$ 
  assumes  $\varepsilon \in \{0 < .. < 1\}$ 
  assumes  $\delta \in \{0 < .. < 1\}$ 
  defines  $\text{sketch} \equiv \text{fold } (\lambda a \ \text{state}. \text{state} \gg= \text{f0-update } a) \ \text{as } (\text{f0-init } \delta \ \varepsilon \ n)$ 
  defines  $t \equiv \text{nat } \lceil 80 / (\text{real-of-rat } \delta)^2 \rceil$ 
  defines  $s \equiv \text{nat } \lceil -(18 * \ln (\text{real-of-rat } \varepsilon)) \rceil$ 
  defines  $p \equiv \text{find-prime-above } (\text{max } n \ 19)$ 
  defines  $r \equiv \text{nat } (4 * \lceil \log 2 (1 / \text{real-of-rat } \delta) \rceil + 24)$ 
  shows  $\text{sketch} = \text{map-pmf } (\lambda x. (s, t, p, r, x, \lambda i \in \{0..<s\}. \text{f0-sketch } p \ r \ t \ (x \ i) \ as))$ 
    ( $\text{prod-pmf } \{0..<s\} (\lambda-. \text{pmf-of-set } (\text{bounded-degree-polynomials } (\text{ZFact } (\text{int } p)) 2)))$ )
  <proof>

```

```

lemma (in prob-space) prob-sub-additive:
  assumes Collect  $P \in \text{sets } M$ 
  assumes Collect  $Q \in \text{sets } M$ 
  shows  $\mathcal{P}(\omega \text{ in } M. P \ \omega \vee Q \ \omega) \leq \mathcal{P}(\omega \text{ in } M. P \ \omega) + \mathcal{P}(\omega \text{ in } M. Q \ \omega)$ 
  <proof>

```

lemma (in prob-space) prob-sub-additiveI:

assumes Collect $P \in \text{sets } M$

assumes Collect $Q \in \text{sets } M$

assumes $\mathcal{P}(\omega \text{ in } M. P \ \omega) \leq r1$

assumes $\mathcal{P}(\omega \text{ in } M. Q \ \omega) \leq r2$

shows $\mathcal{P}(\omega \text{ in } M. P \ \omega \vee Q \ \omega) \leq r1 + r2$

<proof>

lemma (in prob-space) prob-mono:

assumes Collect $Q \in \text{sets } M$

assumes $\bigwedge \omega. \omega \in \text{space } M \implies P \ \omega \implies Q \ \omega$

shows $\mathcal{P}(\omega \text{ in } M. P \ \omega) \leq \mathcal{P}(\omega \text{ in } M. Q \ \omega)$

<proof>

lemma in-events-pmf: $A \in \text{measure-pmf.events } \Omega$

<proof>

lemma pmf-add:

assumes $\bigwedge x. x \in P \implies x \in \text{set-pmf } \Omega \implies x \in Q \vee x \in R$

shows $\text{measure } (\text{measure-pmf } \Omega) \ P \leq \text{measure } (\text{measure-pmf } \Omega) \ Q + \text{measure } (\text{measure-pmf } \Omega) \ R$

<proof>

lemma pmf-mono:

assumes $\bigwedge x. x \in P \implies x \in Q$

shows $\text{measure } (\text{measure-pmf } \Omega) \ P \leq \text{measure } (\text{measure-pmf } \Omega) \ Q$

<proof>

lemma abs-ge-iff: $((x::\text{real}) \leq \text{abs } y) = (x \leq y \vee x \leq -y)$

<proof>

lemma two-powr-0: $2 \ \text{powr } (0::\text{real}) = 1$

<proof>

lemma count-nat-abs-diff-2:

fixes $x :: \text{nat}$

fixes $q :: \text{real}$

assumes $q \geq 0$

defines $A \equiv \{(k::\text{nat}). \text{abs } (\text{real } x - \text{real } k) \leq q \wedge k \neq x\}$

shows $\text{real } (\text{card } A) \leq 2 * q$ and finite A

<proof>

lemma f0-collision-prob:

fixes $p :: \text{nat}$

assumes Factorial-Ring.prime p

defines $\Omega \equiv \text{pmf-of-set } (\text{bounded-degree-polynomials } (\text{ZFact } (\text{int } p)) \ 2)$

assumes $M \subseteq \{0..<p\}$

assumes $c \geq 1$

assumes $r \geq 1$

shows $\mathcal{P}(\omega \text{ in measure-pmf } \Omega.$
 $\exists x \in M. \exists y \in M.$
 $x \neq y \wedge$
 $\text{truncate-down } r \text{ (hash } p \ x \ \omega) \leq c \wedge$
 $\text{truncate-down } r \text{ (hash } p \ x \ \omega) = \text{truncate-down } r \text{ (hash } p \ y \ \omega)) \leq$
 $6 * (\text{real (card } M))^2 * c^2 * 2^{\text{powr } -r} / (\text{real } p)^2 + 1 / \text{real } p \text{ (is } \mathcal{P}(\omega \text{ in } \cdot. ?l$
 $\omega) \leq ?r1 + ?r2)$
 $\langle \text{proof} \rangle$

lemma *inters-compr*: $A \cap \{x. P \ x\} = \{x \in A. P \ x\}$
 $\langle \text{proof} \rangle$

lemma *of-bool-square*: $(\text{of-bool } x)^2 = ((\text{of-bool } x)::\text{real})$
 $\langle \text{proof} \rangle$

theorem *f0-alg-correct*:
assumes $\varepsilon \in \{0 < .. < 1\}$
assumes $\delta \in \{0 < .. < 1\}$
assumes $\text{set as} \subseteq \{0 .. < n\}$
defines $M \equiv \text{fold } (\lambda a \text{ state. state } \ggg \text{f0-update } a) \text{ as } (\text{f0-init } \delta \ \varepsilon \ n) \ggg \text{f0-result}$
shows $\mathcal{P}(\omega \text{ in measure-pmf } M. |\omega - F \ 0 \ \text{as}| \leq \delta * F \ 0 \ \text{as}) \geq 1 - \text{of-rat } \varepsilon$
 $\langle \text{proof} \rangle$

fun *f0-space-usage* :: $(\text{nat} \times \text{rat} \times \text{rat}) \Rightarrow \text{real}$ **where**
 $\text{f0-space-usage } (n, \varepsilon, \delta) = ($
 $\text{let } s = \text{nat } \lceil -18 * \ln (\text{real-of-rat } \varepsilon) \rceil \text{ in}$
 $\text{let } r = \text{nat } (4 * \lceil \log 2 (1 / \text{real-of-rat } \delta) \rceil + 24) \text{ in}$
 $\text{let } t = \text{nat } \lceil 80 / (\text{real-of-rat } \delta)^2 \rceil \text{ in}$
 $8 +$
 $2 * \log 2 (\text{real } s + 1) +$
 $2 * \log 2 (\text{real } t + 1) +$
 $2 * \log 2 (\text{real } n + 10) +$
 $2 * \log 2 (\text{real } r + 1) +$
 $\text{real } s * (12 + 4 * \log 2 (10 + \text{real } n) +$
 $\text{real } t * (11 + 4 * r + 2 * \log 2 (\log 2 (\text{real } n + 9))))))$

definition *encode-state* **where**
 $\text{encode-state} =$
 $N_S \times_D (\lambda s.$
 $N_S \times_S ($
 $N_S \times_D (\lambda p.$
 $N_S \times_S ($
 $([0 .. < s] \rightarrow_S (\text{list}_S (\text{zfact}_S \ p)))) \times_S$
 $([0 .. < s] \rightarrow_S (\text{set}_S \ F_S))))))$

lemma *inj-on encode-state* $(\text{dom encode-state})$
 $\langle \text{proof} \rangle$

lemma *f-subset*:

```

assumes  $g \vdash A \subseteq h \vdash B$ 
shows  $(\lambda x. f (g x)) \vdash A \subseteq (\lambda x. f (h x)) \vdash B$ 
 $\langle proof \rangle$ 

theorem f0-exact-space-usage:
  assumes  $\varepsilon \in \{0 < .. < 1\}$ 
  assumes  $\delta \in \{0 < .. < 1\}$ 
  assumes  $set\ as \subseteq \{0 .. < n\}$ 
  defines  $M \equiv fold (\lambda a\ state. state \gg= f0\text{-}update\ a)\ as\ (f0\text{-}init\ \delta\ \varepsilon\ n)$ 
  shows  $AE\ \omega\ in\ M. bit\text{-}count\ (encode\text{-}state\ \omega) \leq f0\text{-}space\text{-}usage\ (n, \varepsilon, \delta)$ 
 $\langle proof \rangle$ 

lemma f0-asymptotic-space-complexity:
   $f0\text{-}space\text{-}usage \in O[at\text{-}top \times_F at\text{-}right\ 0 \times_F at\text{-}right\ 0](\lambda(n, \varepsilon, \delta). \ln (1 / of\text{-}rat$ 
 $\varepsilon) * (\ln (real\ n) + 1 / (of\text{-}rat\ \delta)^2 * (\ln (\ln (real\ n)) + \ln (1 / of\text{-}rat\ \delta))))$ 
  (is -  $\in O[?F](?rhs)$ )
 $\langle proof \rangle$ 

end

```

17 Partitions

```

theory Partitions
  imports Main HOL-Library.Multiset HOL.Real List-Ext
begin

```

This section introduces a function that enumerates all the partitions of $\{0 .. < n\}$. The partitions are represented as lists with n elements. If the element at index i and j have the same value, then i and j are in the same partition.

```

fun enum-partitions-aux ::  $nat \Rightarrow (nat \times nat\ list)\ list$ 
  where
    enum-partitions-aux 0 =  $[(0, [])]$  |
    enum-partitions-aux (Suc  $n$ ) =
       $[(c+1, c\#x). (c,x) \leftarrow enum\text{-}partitions\text{-}aux\ n]@$ 
       $[(c, y\#x). (c,x) \leftarrow enum\text{-}partitions\text{-}aux\ n, y \leftarrow [0 .. < c]]$ 

fun enum-partitions where enum-partitions  $n = map\ snd\ (enum\text{-}partitions\text{-}aux\ n)$ 

```

```

definition has-eq-relation ::  $nat\ list \Rightarrow 'a\ list \Rightarrow bool$  where
  has-eq-relation  $r\ xs = (length\ xs = length\ r \wedge (\forall i < length\ xs. \forall j < length\ xs. (xs\ !\ i = xs\ !\ j) = (r\ !\ i = r\ !\ j)))$ 

```

```

lemma filter-one-elim:
   $length\ (filter\ p\ xs) = 1 \implies (\exists u\ v\ w. xs = u@v\#w \wedge p\ v \wedge length\ (filter\ p\ u) = 0 \wedge length\ (filter\ p\ w) = 0)$ 

```

(**is** ?A xs \implies ?B xs)
 <proof>

lemma has-eq-elim:

has-eq-relation (r#rs) (x#xs) = (
 ($\forall i < \text{length } xs. (r = rs ! i) = (x = xs ! i)$) \wedge
 has-eq-relation rs xs)
 <proof>

lemma enum-partitions-aux-range:

$x \in \text{set } (\text{enum-partitions-aux } n) \implies \text{set } (\text{snd } x) = \{k. k < \text{fst } x\}$
 <proof>

lemma enum-partitions-aux-len:

$x \in \text{set } (\text{enum-partitions-aux } n) \implies \text{length } (\text{snd } x) = n$
 <proof>

lemma enum-partitions-complete-aux: $k < n \implies \text{length } (\text{filter } (\lambda x. x = k) [0..<n])$
 = Suc 0
 <proof>

lemma enum-partitions-complete:

$\text{length } (\text{filter } (\lambda p. \text{has-eq-relation } p \ x) (\text{enum-partitions } (\text{length } x))) = 1$
 <proof>

fun verify where

verify r x 0 = True |
 verify r x (Suc n) 0 = verify r x n n |
 verify r x (Suc n) (Suc m) = ((r ! n = r ! m) = (x ! n = x ! m)) \wedge (verify r x
 (Suc n) m))

lemma verify-elim-1:

verify r x (Suc n) m = (verify r x n n \wedge ($\forall i < m. (r ! n = r ! i) = (x ! n = x$
 ! i)))
 <proof>

lemma verify-elim:

verify r x m m = ($\forall i < m. \forall j < i. (r ! i = r ! j) = (x ! i = x ! j)$)
 <proof>

lemma has-eq-relation-elim:

has-eq-relation r xs = ($\text{length } r = \text{length } xs \wedge \text{verify } r \ xs \ (\text{length } xs) \ (\text{length } xs)$)
 <proof>

lemma sum-filter: $\text{sum-list } (\text{map } (\lambda p. \text{if } f \ p \text{ then } (r::\text{real}) \text{ else } 0) \ y) = r * (\text{length } (\text{filter } f \ y))$
 <proof>

lemma sum-partitions: $\text{sum-list } (\text{map } (\lambda p. \text{if } \text{has-eq-relation } p \ x \text{ then } (r::\text{real}) \text{ else } 0) \ y) = r * (\text{length } (\text{filter } f \ y))$

0) (enum-partitions (length x))) = r
 ⟨proof⟩

lemma *sum-partitions'*:
 assumes $n = \text{length } x$
 shows $\text{sum-list } (\text{map } (\lambda p. \text{of-bool } (\text{has-eq-relation } p \ x) * (r::\text{real})) (\text{enum-partitions } n)) = r$
 ⟨proof⟩

lemma *eq-rel-obtain-bij*:
 assumes $\text{has-eq-relation } u \ v$
 obtains f where $\text{bij-betw } f \ (\text{set } u) \ (\text{set } v) \ \wedge y. y \in \text{set } u \implies \text{count-list } u \ y = \text{count-list } v \ (f \ y)$
 ⟨proof⟩

end

18 Frequency Moment 2

theory *Frequency-Moment-2*
 imports *Main Median Partitions Primes-Ext Encoding List-Ext*
UniversalHashFamilyOfPrime Frequency-Moments Landau-Ext
begin

This section contains a formalization of the algorithm for the second frequency moment. It is based on the algorithm described in [1, §2.2]. The only difference is that the algorithm is adapted to work with prime field of odd order, which greatly reduces the implementation complexity.

fun *f2-hash* where
 $\text{f2-hash } p \ k \ h = (\text{if } \text{hash } p \ k \ h \in \{k. 2*k < p\} \text{ then } \text{int } p - 1 \text{ else } - \text{int } p - 1)$

type-synonym *f2-state* = $\text{nat} \times \text{nat} \times \text{nat} \times (\text{nat} \times \text{nat} \Rightarrow \text{int set list}) \times (\text{nat} \times \text{nat} \Rightarrow \text{int})$

fun *f2-init* :: $\text{rat} \Rightarrow \text{rat} \Rightarrow \text{nat} \Rightarrow \text{f2-state pmf}$ where
 $\text{f2-init } \delta \ \varepsilon \ n =$
 do {
 let $s_1 = \text{nat } \lceil 6 / \delta^2 \rceil$;
 let $s_2 = \text{nat } \lceil -(18 * \ln (\text{real-of-rat } \varepsilon)) \rceil$;
 let $p = \text{find-prime-above } (\text{max } n \ 3)$;
 $h \leftarrow \text{prod-pmf } (\{0..<s_1\} \times \{0..<s_2\}) (\lambda-. \text{pmf-of-set } (\text{bounded-degree-polynomials } (\text{ZFact } (\text{int } p)) \ 4))$;
 return-pmf $(s_1, s_2, p, h, (\lambda-. \in \{0..<s_1\} \times \{0..<s_2\}. (0 :: \text{int})))$
 }

fun *f2-update* :: $\text{nat} \Rightarrow \text{f2-state} \Rightarrow \text{f2-state pmf}$ where
 $\text{f2-update } x \ (s_1, s_2, p, h, \text{sketch}) =$
 return-pmf $(s_1, s_2, p, h, \lambda i \in \{0..<s_1\} \times \{0..<s_2\}. \text{f2-hash } p \ (h \ i) \ x + \text{sketch})$

$i)$

```

fun f2-result :: f2-state  $\Rightarrow$  rat pmf where
  f2-result (s1, s2, p, h, sketch) =
    return-pmf (median ( $\lambda i_2 \in \{0..<s_2\}$ .
      ( $\sum_{i_1 \in \{0..<s_1\}} \cdot (\text{rat-of-int } (\text{sketch } (i_1, i_2)))^2$ ) / (((rat-of-nat p)2 - 1) *
        rat-of-nat s1))) s2
    )

```

```

lemma f2-hash-exp:
  assumes Factorial-Ring.prime p
  assumes k < p
  assumes p > 2
  shows
    prob-space.expectation (pmf-of-set (bounded-degree-polynomials (ZFact (int p))
      4))
      ( $\lambda \omega. \text{real-of-int } (\text{f2-hash } p \ \omega \ k) \wedge^m$ ) =
      (((real p - 1)  $\wedge^m$  * (real p + 1) + (- real p - 1)  $\wedge^m$  * (real p - 1)) / (2
        * real p))
    <proof>

```

```

lemma
  assumes Factorial-Ring.prime p
  assumes p > 2
  assumes  $\bigwedge a. a \in \text{set } as \implies a < p$ 
  defines M  $\equiv$  measure-pmf (pmf-of-set (bounded-degree-polynomials (ZFact (int
    p)) 4))
  defines f  $\equiv$  ( $\lambda \omega. \text{real-of-int } (\text{sum-list } (\text{map } (\text{f2-hash } p \ \omega) \ as)) \wedge^2$ )
  shows var-f2:prob-space.variance M f  $\leq 2 * (\text{real-of-rat } (F \ 2 \ as) \wedge^2) * ((\text{real}
    p)^2 - 1)^2$  (is ?A)
  and exp-f2:prob-space.expectation M f = real-of-rat (F 2 as) * ((real p)2 - 1) (is
    ?B)
  <proof>

```

```

lemma f2-alg-sketch:
  fixes n :: nat
  fixes as :: nat list
  assumes  $\varepsilon \in \{0 < .. < 1\}$ 
  assumes  $\delta > 0$ 
  defines s1  $\equiv$  nat  $\lceil 6 / \delta^2 \rceil$ 
  defines s2  $\equiv$  nat  $\lceil -(18 * \ln (\text{real-of-rat } \varepsilon)) \rceil$ 
  defines p  $\equiv$  find-prime-above (max n 3)
  defines sketch  $\equiv$  fold ( $\lambda a \text{ state}. \text{state} \gg= \text{f2-update } a$ ) as (f2-init  $\delta \ \varepsilon \ n$ )
  defines  $\Omega \equiv \text{prod-pmf } (\{0..<s_1\} \times \{0..<s_2\}) (\lambda \cdot. \text{pmf-of-set } (\text{bounded-degree-polynomials}
    (\text{ZFact } (\text{int } p)) \ 4))$ 
  shows sketch =  $\Omega \gg= (\lambda h. \text{return-pmf } (s_1, s_2, p, h,$ 
     $\lambda i \in \{0..<s_1\} \times \{0..<s_2\}. \text{sum-list } (\text{map } (\text{f2-hash } p \ (h \ i)) \ as)))$ 
  <proof>

```

theorem *f2-alg-correct*:
assumes $\varepsilon \in \{0 < .. < 1\}$
assumes $\delta > 0$
assumes $set\ as \subseteq \{0 .. < n\}$
defines $M \equiv fold\ (\lambda a\ state.\ state \gg f2\text{-}update\ a)\ as\ (f2\text{-}init\ \delta\ \varepsilon\ n) \gg f2\text{-}result$
shows $\mathcal{P}(\omega\ in\ measure\text{-}pmf\ M.\ |\omega - F\ 2\ as| \leq \delta * F\ 2\ as) \geq 1 - of\text{-}rat\ \varepsilon$
 $\langle proof \rangle$

fun *f2-space-usage* :: $(nat \times nat \times rat \times rat) \Rightarrow real$ **where**
f2-space-usage $(n, m, \varepsilon, \delta) =$
 $let\ s_1 = nat\ \lceil 6 / \delta^2 \rceil\ in$
 $let\ s_2 = nat\ \lceil -(18 * ln\ (real\ of\text{-}rat\ \varepsilon)) \rceil\ in$
 $5 +$
 $2 * log\ 2\ (s_1 + 1) +$
 $2 * log\ 2\ (s_2 + 1) +$
 $2 * log\ 2\ (4 + 2 * real\ n) +$
 $s_1 * s_2 * (13 + 8 * log\ 2\ (4 + 2 * real\ n) + 2 * log\ 2\ (real\ m * (4 + 2 * real$
 $n) + 1\)))$

definition *encode-state* **where**
 $encode\text{-}state =$
 $N_S \times_D (\lambda s_1.$
 $N_S \times_D (\lambda s_2.$
 $N_S \times_D (\lambda p.$
 $(List.product\ [0 .. < s_1]\ [0 .. < s_2] \rightarrow_S (list_S\ (zfact_S\ p))) \times_S$
 $(List.product\ [0 .. < s_1]\ [0 .. < s_2] \rightarrow_S I_S))))$

lemma *inj-on encode-state* $(dom\ encode\text{-}state)$
 $\langle proof \rangle$

theorem *f2-exact-space-usage*:
assumes $\varepsilon \in \{0 < .. < 1\}$
assumes $\delta > 0$
assumes $set\ as \subseteq \{0 .. < n\}$
defines $M \equiv fold\ (\lambda a\ state.\ state \gg f2\text{-}update\ a)\ as\ (f2\text{-}init\ \delta\ \varepsilon\ n)$
shows $AE\ \omega\ in\ M.\ bit\text{-}count\ (encode\text{-}state\ \omega) \leq f2\text{-}space\text{-}usage\ (n,\ length\ as,\ \varepsilon,\ \delta)$
 $\langle proof \rangle$

theorem *f2-asymptotic-space-complexity*:
 $f2\text{-}space\text{-}usage \in O[at\text{-}top \times_F at\text{-}top \times_F at\text{-}right\ 0 \times_F at\text{-}right\ 0](\lambda\ (n,\ m,\ \varepsilon,\ \delta).$
 $(ln\ (1 / of\text{-}rat\ \varepsilon)) / (of\text{-}rat\ \delta)^2 * (ln\ (real\ n) + ln\ (real\ m)))$
 $(is\ - \in O[?F](?rhs))$
 $\langle proof \rangle$

end

19 Frequency Moment k

theory *Frequency-Moment-k*

imports *Main Median Product-PMF-Ext Lp.Lp List-Ext Encoding Frequency-Moments Landau-Ext*

begin

This section contains a formalization of the algorithm for the k -th frequency moment. It is based on the algorithm described in [1, §2.1].

type-synonym $fk\text{-state} = \text{nat} \times \text{nat} \times \text{nat} \times \text{nat} \times (\text{nat} \times \text{nat} \Rightarrow (\text{nat} \times \text{nat}))$

fun $fk\text{-init} :: \text{nat} \Rightarrow \text{rat} \Rightarrow \text{rat} \Rightarrow \text{nat} \Rightarrow fk\text{-state} \text{ pmf}$ **where**

$fk\text{-init } k \delta \varepsilon n =$
 do {
 let $s_1 = \text{nat } \lceil 3 * \text{real } k * (\text{real } n) \text{ powr } (1 - 1 / \text{real } k) / (\text{real-of-rat } \delta)^2 \rceil$;
 let $s_2 = \text{nat } \lceil -18 * \ln (\text{real-of-rat } \varepsilon) \rceil$;
 return-pmf ($s_1, s_2, k, 0, (\lambda \cdot. \text{undefined})$)
 }

fun $fk\text{-update} :: \text{nat} \Rightarrow fk\text{-state} \Rightarrow fk\text{-state} \text{ pmf}$ **where**

$fk\text{-update } a (s_1, s_2, k, m, r) =$
 do {
 $\text{coins} \leftarrow \text{prod-pmf } (\{0..<s_1\} \times \{0..<s_2\}) (\lambda \cdot. \text{bernoulli-pmf } (1 / (\text{real } m + 1)))$;
 return-pmf ($s_1, s_2, k, m + 1, \lambda i \in \{0..<s_1\} \times \{0..<s_2\}.$
 if coins i then
 $(a, 0)$
 else (
 let $(x, l) = r \ i$ in $(x, l + \text{of-bool } (x = a))$
)
 })

fun $fk\text{-result} :: fk\text{-state} \Rightarrow \text{rat} \text{ pmf}$ **where**

$fk\text{-result } (s_1, s_2, k, m, r) =$
 return-pmf ($\text{median } (\lambda i_2 \in \{0..<s_2\}.$
 $(\sum i_1 \in \{0..<s_1\} . \text{rat-of-nat } (\text{let } t = \text{snd } (r (i_1, i_2)) + 1 \text{ in } m * (t \sim k - (t - 1) \sim k))) / (\text{rat-of-nat } s_1)) \ s_2$
)

fun $fk\text{-update}' :: 'a \Rightarrow \text{nat} \Rightarrow \text{nat} \Rightarrow \text{nat} \Rightarrow (\text{nat} \times \text{nat} \Rightarrow ('a \times \text{nat})) \Rightarrow (\text{nat} \times \text{nat} \Rightarrow ('a \times \text{nat})) \text{ pmf}$ **where**

$fk\text{-update}' a s_1 s_2 m r =$
 do {
 $\text{coins} \leftarrow \text{prod-pmf } (\{0..<s_1\} \times \{0..<s_2\}) (\lambda \cdot. \text{bernoulli-pmf } (1 / (\text{real } m + 1)))$;
 return-pmf ($\lambda i \in \{0..<s_1\} \times \{0..<s_2\}.$
 if coins i then
 $(a, 0)$
 else (
 let $(x, l) = r \ i$ in $(x, l + \text{of-bool } (x = a))$
)
 })

```

    )
  )
}

```

```

fun fk-update'' :: 'a ⇒ nat ⇒ ('a × nat) ⇒ (('a × nat)) pmf where
  fk-update'' a m (x,l) =
    do {
      coin ← bernoulli-pmf (1/(real m+1));
      return-pmf (
        if coin then
          (a,0)
        else (
          (x, l + of-bool (x=a))
        )
      )
    }

```

lemma *bernoulli-pmf-1*: *bernoulli-pmf 1 = return-pmf True*
 ⟨proof⟩

lemma *split-space*:
 $(\sum a \in \{(u, v). v < \text{count-list as } u\}. (f (\text{snd } a))) =$
 $(\sum u \in \text{set as}. (\sum v \in \{0..<\text{count-list as } u\}. (f v)))$ (**is** ?lhs = ?rhs)
 ⟨proof⟩

lemma
assumes *as* ≠ []
shows *fin-space*: *finite* $\{(u, v). v < \text{count-list as } u\}$ **and**
non-empty-space: $\{(u, v). v < \text{count-list as } u\} \neq \{\}$ **and**
card-space: *card* $\{(u, v). v < \text{count-list as } u\} = \text{length as}$
 ⟨proof⟩

lemma *fk-alg-aux-5*:
assumes *as* ≠ []
shows *pmf-of-set* $\{k. k < \text{length as}\} \gg (\lambda k. \text{return-pmf } (as ! k, \text{count-list } (\text{drop } (k+1) as) (as ! k)))$
 $= \text{pmf-of-set } \{(u,v). v < \text{count-list as } u\}$
 ⟨proof⟩

lemma *fk-alg-aux-4*:
assumes *as* ≠ []
shows *fold* $(\lambda x (c, \text{state}). (c+1, \text{state} \gg \text{fk-update'' } x c)) as (0, \text{return-pmf undefined}) =$
 $(\text{length as}, \text{pmf-of-set } \{k. k < \text{length as}\} \gg (\lambda k. \text{return-pmf } (as ! k, \text{count-list } (\text{drop } (k+1) as) (as ! k))))$
 ⟨proof⟩

definition *if-then-else* **where** *if-then-else* *p q r* = (*if p then q else r*)

This definition is introduced to be able to temporarily substitute *if p then q*

else r with *if-then-else* $p \ q \ r$, which unblocks the simplifier to process q and r .

lemma *fk-alg-aux-2*:

```
fold (λx (c, state). (c+1, state ≫= fk-update' x s1 s2 c)) as (0, return-pmf (λ-.
undefined))
= (length as, prod-pmf ({0..s1} × {0..s2}) (λ-. (snd (fold (λx (c, state).
(c+1, state ≫= fk-update'' x c)) as (0, return-pmf undefined))))))
(is ?lhs = ?rhs)
⟨proof⟩
```

lemma *fk-alg-aux-1*:

```
fixes k :: nat
fixes ε :: rat
assumes δ > 0
assumes set as ⊆ {0..n}
assumes as ≠ []
defines sketch ≡ fold (λa state. state ≫= fk-update a) as (fk-init k δ ε n)
defines s1 ≡ nat ⌈3*real k*(real n) powr (1-1/real k)/ (real-of-rat δ)2⌋
defines s2 ≡ nat ⌈-(18 * ln (real-of-rat ε))⌋
shows sketch =
  map-pmf (λx. (s1, s2, k, length as, x))
  (snd (fold (λx (c, state). (c+1, state ≫= fk-update' x s1 s2 c)) as (0, return-pmf
(λ-. undefined))))
⟨proof⟩
```

lemma *power-diff-sum*:

```
assumes k > 0
shows (a :: 'a :: {comm-ring-1, power})^k - b^k = (a-b) * sum (λi. a^i *
b^(k-1-i)) {0..k} (is ?lhs = ?rhs)
⟨proof⟩
```

lemma *power-diff-est*:

```
assumes k > 0
assumes (a :: real) ≥ b
assumes b ≥ 0
shows a^k - b^k ≤ (a-b) * k * a^(k-1)
⟨proof⟩
```

Specialization of the Hoelder inequality for sums.

lemma *Holder-inequality-sum*:

```
assumes p > (0::real) q > 0 1/p + 1/q = 1
assumes finite A
shows |sum (λx. f x * g x) A| ≤ (sum (λx. |f x| powr p) A) powr (1/p) * (sum
(λx. |g x| powr q) A) powr (1/q)
⟨proof⟩
```

lemma *fk-estimate*:

```
assumes as ≠ []
assumes set as ⊆ {0..n}
```

assumes $k \geq 1$
shows $\text{real } (\text{length } as) * \text{real-of-rat } (F (2*k-1) as) \leq \text{real } n \text{ powr } (1 - 1 / \text{real } k) * (\text{real-of-rat } (F k as))^{\wedge 2}$
(is ?lhs ≤ ?rhs)
 <proof>

lemma *fk-alg-core-exp*:

assumes $as \neq []$
assumes $k \geq 1$
shows $\text{has-bochner-integral } (\text{measure-pmf } (\text{pmf-of-set } \{(u, v). v < \text{count-list } as\}))$
 $(\lambda a. \text{real } (\text{length } as) * \text{real } (\text{Suc } (\text{snd } a) ^ k - \text{snd } a ^ k)) (\text{real-of-rat } (F k as))$
 <proof>

lemma *fk-alg-core-var*:

assumes $as \neq []$
assumes $k \geq 1$
assumes $\text{set } as \subseteq \{0..<n\}$
shows $\text{prob-space.variance } (\text{measure-pmf } (\text{pmf-of-set } \{(u, v). v < \text{count-list } as\}))$
 $(\lambda a. \text{real } (\text{length } as) * \text{real } (\text{Suc } (\text{snd } a) ^ k - \text{snd } a ^ k))$
 $\leq (\text{real-of-rat } (F k as))^2 * \text{real } k * \text{real } n \text{ powr } (1 - 1 / \text{real } k)$
 <proof>

theorem *fk-alg-sketch*:

fixes $\varepsilon :: \text{rat}$
assumes $k \geq 1$
assumes $\delta > 0$
assumes $\text{set } as \subseteq \{0..<n\}$
assumes $as \neq []$
defines $\text{sketch} \equiv \text{fold } (\lambda a \text{ state}. \text{state} \gg \text{fk-update } a) as (\text{fk-init } k \delta \varepsilon n)$
defines $s_1 \equiv \text{nat } \lceil 3 * \text{real } k * (\text{real } n) \text{ powr } (1 - 1 / \text{real } k) / (\text{real-of-rat } \delta)^2 \rceil$
defines $s_2 \equiv \text{nat } \lceil -(18 * \ln (\text{real-of-rat } \varepsilon)) \rceil$
shows $\text{sketch} = \text{map-pmf } (\lambda x. (s_1, s_2, k, \text{length } as, x))$
 $(\text{prod-pmf } (\{0..<s_1\} \times \{0..<s_2\})) (\lambda-. \text{pmf-of-set } \{(u, v). v < \text{count-list } as\})$
 <proof>

lemma *fk-alg-correct*:

assumes $k \geq 1$
assumes $\varepsilon \in \{0 < .. < 1\}$
assumes $\delta > 0$
assumes $\text{set } as \subseteq \{0..<n\}$
defines $M \equiv \text{fold } (\lambda a \text{ state}. \text{state} \gg \text{fk-update } a) as (\text{fk-init } k \delta \varepsilon n) \gg \text{fk-result}$
shows $\mathcal{P}(\omega \text{ in measure-pmf } M. |\omega - F k as| \leq \delta * F k as) \geq 1 - \text{of-rat } \varepsilon$
 <proof>

fun *fk-space-usage* :: $(\text{nat} \times \text{nat} \times \text{nat} \times \text{rat} \times \text{rat}) \Rightarrow \text{real}$ **where**
fk-space-usage $(k, n, m, \varepsilon, \delta) = ($

```

let s1 = nat ⌈ 3*real k*(real n) powr (1-1/ real k) / (real-of-rat δ)2 ⌉ in
let s2 = nat ⌈ -(18 * ln (real-of-rat ε)) ⌉ in
5 +
2 * log 2 (s1 + 1) +
2 * log 2 (s2 + 1) +
2 * log 2 (real k + 1) +
2 * log 2 (real m + 1) +
s1 * s2 * (3 + 2 * log 2 (real n) + 2 * log 2 (real m)))

```

definition *encode-state* where

```

encode-state =
  NS ×D (λs1.
    NS ×D (λs2.
      NS ×S
      NS ×S
      (List.product [0..s1] [0..s2] →S (NS ×S NS))))

```

lemma *inj-on encode-state (dom encode-state)*
 ⟨*proof*⟩

theorem *fk-exact-space-usage:*

```

  assumes k ≥ 1
  assumes ε ∈ {0<..1}
  assumes δ > 0
  assumes set as ⊆ {0..n}
  assumes as ≠ []
  defines M ≡ fold (λa state. state ≫= fk-update a) as (fk-init k δ ε n)
  shows AE ω in M. bit-count (encode-state ω) ≤ fk-space-usage (k, n, length as,
    ε, δ) (is AE ω in M. (- ≤ ?rhs))
  ⟨proof⟩

```

lemma *fk-asymptotic-space-complexity:*

```

  fk-space-usage ∈
    O[at-top ×F at-top ×F at-top ×F at-right (0::rat) ×F at-right (0::rat)](λ (k, n,
    m, ε, δ).
      real k*(real n) powr (1-1/ real k) / (of-rat δ)2 * (ln (1 / of-rat ε)) * (ln (real
    n) + ln (real m)))
    (is - ∈ O[?F](?rhs))
  ⟨proof⟩

```

end

A Informal proof of correctness for the F_0 algorithm

This section contains a detailed informal proof for the correctness of the F_0 -algorithm. Because of the standard amplification result about medians

(see for example [1]) it is enough to show that each of the estimates the median is taken from is within the desired interval with success probability $\frac{2}{3}$.

To verify the latter, let a_1, \dots, a_m be the stream elements, where we assume that the elements are a subset of $\{0, \dots, n-1\}$ and $0 < \delta < 1$ be the desired relative accuracy. Let p be the smallest prime such that $p \geq \max(n, 19)$ and let h be a random polynomial over $GF(p)$ with degree strictly less than 2. The algorithm also introduces the internal parameters t, r defined by:

$$\begin{aligned} t &:= \lceil 80\delta^{-2} \rceil \\ r &:= 4\log_2 \lceil \delta^{-1} \rceil + 24 \end{aligned}$$

The estimate the algorithm obtains is:

$$\begin{aligned} A &:= \{a_1, \dots, a_m\} & H &:= \{\lfloor h(a) \rfloor_r \mid a \in A\} \\ R &:= \begin{cases} tp(\min_t(H))^{-1} & \text{if } |H| \geq t \\ |H| & \text{otherwise,} \end{cases} \end{aligned}$$

Here $\min_t(H)$ denotes the t -th smallest element of H . With these definitions, it is possible to state the goal as:

$$P(|R - F_0| \leq \delta |F_0|) \geq \frac{2}{3}.$$

which is shown by separately in the following two subsections for the cases $F_0 \geq t$ and $F_0 < t$.

A.1 Case $F_0 \geq t$

Let us introduce:

$$\begin{aligned} H^* &:= \{h(a) \mid a \in A\}^\# \\ R^* &:= tp(\text{rank}_t^\#(H^*))^{-1} \end{aligned}$$

These definitions correspond to the H, R but with a few minor modifications. The set H^* is a multiset, this means that each element also has a multiplicity, counting the number of *distinct* elements of A being mapped by h to the same value. Note that by definition: $|H^*| = |A|$. Similarly the operation $\min_t^\#$ obtains the t -th element of the multiset H (taking multiplicities into account). Note also that there is no rounding operation $\lfloor \cdot \rfloor_r$ in the definition of H^* . The key reason for the introduction of these alternative versions of H, R is that it is easier to show probabilistic bounds on the distances $|R^* - F_0|$ and $|R^* - R|$ as opposed to $|R - F_0|$ directly. In particular the plan is to

show:

$$\delta' := \frac{3}{4}\delta \quad (1)$$

$$P(|R^* - F_0| > \delta' F_0) \leq \frac{2}{9}, \text{ and} \quad (2)$$

$$P\left(|R^* - F_0| \leq \delta' F_0 \wedge |R - R^*| > \frac{\delta}{4} F_0\right) \leq \frac{1}{9} \quad (3)$$

I.e. the probability that R^* has not the relative accuracy of $\frac{3}{4}\delta$ is less than $\frac{2}{9}$ and the probability that assuming R^* has the relative accuracy of $\frac{3}{4}\delta$ but that R deviates by more than $\frac{1}{4}\delta F_0$ is at most $\frac{1}{9}$. Hence, the probability that neither of these events happen is at least $\frac{2}{3}$ but in that case:

$$|R - F_0| \leq |R - R^*| + |R^* - F_0| \leq \frac{\delta}{4} F_0 + \frac{3\delta}{4} F_0 = \delta F_0. \quad (4)$$

For the verification of [Equation 2](#) let us introduce:

$$Q(u) = |\{h(a) < u \mid a \in A\}|$$

and observe that $\min_t^\#(H^*) < u$ if $Q(u) \geq t$ and $\min_t^\#(H^*) \geq v$ if $Q(v) \leq t - 1$. To see why this is true note that, if at least t elements of A are mapped by h below a certain value, then the rank t element must also be within them, and thus also be below that value. And that the opposite direction of this conclusion is also true. Note that this relies on the fact that H^* is a multiset and that multiplicities are being taken into account, when computing the t -th smallest element.

Alternatively, it is also possible to write $Q(u) = \sum_{a \in A} 1_{\{h(a) < u\}}$ ¹, i.e., Q is a sum of pairwise independent $\{0, 1\}$ -valued random variables, with expectation $\frac{u}{p}$ and variance $\frac{u}{p} - \frac{u^2}{p^2}$.² Using linearity of expectation and Bienaymé's identity, it follows that $\text{Var } Q(u) \leq \mathbb{E} Q(u) = |A|up^{-1} = F_0up^{-1}$ for $u \in \{0, \dots, p\}$.

For $v = \left\lfloor \frac{tp}{(1-\delta')F_0} \right\rfloor$ it is possible to conclude:

$$\begin{aligned} t - 1 &\leq \frac{t}{(1-\delta')} - 3\sqrt{\frac{t}{(1-\delta')}} - 1 \\ &\leq \frac{F_0v}{p} - 3\sqrt{\frac{F_0v}{p}} \leq \mathbb{E}Q(v) - 3\sqrt{\text{Var}Q(v)} \end{aligned}$$

¹The notation 1_A is shorthand for the indicator function of A , i.e., $1_A(x) = 1$ if $x \in A$ and 0 otherwise.

²A consequence of h being chosen uniformly from a 2-independent hash family.

and thus using Tchebyshev's inequality:

$$\begin{aligned}
P(R^* < (1 - \delta') F_0) &= P\left(\text{rank}_t^\#(H^*) > \frac{tp}{(1 - \delta') F_0}\right) \\
&\leq P(\text{rank}_t^\#(H^*) \geq v) = P(Q(v) \leq t - 1) \\
&\leq P\left(Q(v) \leq \mathbb{E}Q(v) - 3\sqrt{\text{Var}Q(v)}\right) \leq \frac{1}{9}.
\end{aligned} \tag{5}$$

Similarly for $u = \left\lceil \frac{tp}{(1 + \delta') F_0} \right\rceil$ it is possible to conclude:

$$\begin{aligned}
t &\geq \frac{t}{(1 + \delta')} + 3\sqrt{\frac{t}{(1 + \delta')}} + 1 + 1 \\
&\geq \frac{F_0 u}{p} + 3\sqrt{\frac{F_0 u}{p}} \geq \mathbb{E}Q(u) + 3\sqrt{\text{Var}Q(v)}
\end{aligned}$$

and thus using Tchebyshev's inequality:

$$\begin{aligned}
P(R^* > (1 + \delta') F_0) &= P\left(\text{rank}_t^\#(H^*) < \frac{tp}{(1 + \delta') F_0}\right) \\
&\leq P(\text{rank}_t^\#(H^*) < u) = P(Q(u) \geq t) \\
&\leq P\left(Q(u) \geq \mathbb{E}Q(u) + 3\sqrt{\text{Var}Q(u)}\right) \leq \frac{1}{9}.
\end{aligned} \tag{6}$$

To verify Equation 3, note that

$$\min_t(H) = \lfloor \min_t^\#(H^*) \rfloor_r \tag{7}$$

if there are no collisions, induced by the application of $\lfloor h(\cdot) \rfloor_r$ on the elements of A . Even more carefully, note that the equation would remain true, as long as there are no collision within the smallest t elements of H^* . Because Equation 3 needs to be shown only in the case where $R^* \geq (1 - \delta') F_0$, i.e., when $\min_t^\#(H^*) \leq v$, it is enough to bound the probability of a collision in the range $[0; v]$. Moreover Equation 7 implies $|\min_t(H) - \min_t^\#(H^*)| \leq \max(\min_t^\#(H^*), \min_t(H)) 2^{-r}$ from which it is possible to derive $|R^* - R| \leq \frac{\delta}{4} F_0$. Another important fact is that h is injective with probability $1 - \frac{1}{p}$, this is because h is chosen uniformly from the polynomials of degree less than 2. If it is a degree 1 polynomial, it is a linear function on $GF(p)$ and thus injective. Because $p \geq 18$ the probability that h is not injective can be bounded by $1/18$. However, even if h is injective, there is still a possibility of collision, because of the application of the rounding operation $\lfloor \cdot \rfloor_r$. The

³The verification of this inequality is a lengthy but straightforward calculation using the definition of δ' and t .

plan is to bound that probability by $1/18$ as well to show [Equation 3](#).

$$\begin{aligned}
& P\left(|R^* - F_0| \leq \delta' F_0 \wedge |R - R^*| > \frac{\delta}{4} F_0\right) \\
& \leq P\left(R^* \geq (1 - \delta') F_0 \wedge \min_t^\#(H^*) \neq \min_t(H) \wedge h \text{ inj.}\right) + P(\neg h \text{ inj.}) \\
& \leq P(\exists a \neq b \in A. \lfloor h(a) \rfloor_r = \lfloor h(b) \rfloor_r \leq v \wedge h(a) \neq h(b)) + \frac{1}{18} \\
& \leq \frac{1}{18} + \sum_{a \neq b \in A} P(\lfloor h(a) \rfloor_r = \lfloor h(b) \rfloor_r \leq v \wedge h(a) \neq h(b)) \\
& \leq \frac{1}{18} + \sum_{a \neq b \in A} P(|h(a) - h(b)| \leq v 2^{-r} \wedge h(a) \leq v(1 + 2^{-r}) \wedge h(a) \neq h(b)) \\
& \leq \frac{1}{18} + \sum_{a \neq b \in A} \sum_{\substack{a', b' \in \{0, \dots, p-1\} \wedge a' \neq b' \\ |a' - b'| \leq v 2^{-r} \wedge a' \leq v(1 + 2^{-r})}} P(h(a) = a') P(h(b) = b') \\
& \leq \frac{1}{18} + 6 \frac{F_0^2 v^2}{p^2} 2^{-r} \leq \frac{1}{9}.
\end{aligned}$$

Which shows that [Equation 3](#) is true and [Equation 5](#) and [6](#) implies [Equation 2](#), which means the reasoning in [Equation 4](#) confirms:

$$P(|R - F_0| \leq \delta |F_0|) \geq \frac{2}{3} \quad (8)$$

The following subsection confirms that this is also true for the remaining case, if $F_0 < t$, concluding the proof.

A.2 Case $F_0 < t$

Note that in this case $|H| \leq F_0 < t$ and thus $R = |H|$, hence the goal is to show that: $P(|H| \neq F_0) \leq \frac{1}{3}$.

The latter can only happen, if there is a collision induced by the application

of $\lfloor h(\cdot) \rfloor_r$. As before h is not injective with probability at least $\frac{1}{18}$, hence:

$$\begin{aligned}
& P(|R - F_0| > \delta F_0) \\
& \leq P(R \neq F_0) \\
& \leq \frac{1}{18} + P(R \neq F_0 \wedge h \text{ injective}) \\
& \leq \frac{1}{18} + P(\exists a \neq b \in A. \lfloor h(a) \rfloor_r = \lfloor h(b) \rfloor_r) \\
& \leq \frac{1}{18} + \sum_{a \neq b \in A} P(\lfloor h(a) \rfloor_r = \lfloor h(b) \rfloor_r \wedge h(a) \neq h(b)) \\
& \leq \frac{1}{18} + \sum_{a \neq b \in A} P(|h(a) - h(b)| \leq p2^{-r} \wedge h(a) \neq h(b)) \\
& \leq \frac{1}{18} + \sum_{a \neq b \in A} \sum_{\substack{a', b' \in \{0, \dots, p-1\} \\ a' \neq b' \wedge |a' - b'| \leq p2^{-r}}} P(h(a) = a') P(h(b) = b') \\
& \leq \frac{1}{18} + F_0^2 2^{-r+1} \leq \frac{1}{9}.
\end{aligned}$$

Which concludes the proof. \square

References

- [1] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. *Journal of Computer and System Sciences*, 58(1):137–147, 1999.
- [2] Z. Bar-Yossef, T. S. Jayram, R. Kumar, D. Sivakumar, and L. Trevisan. Counting distinct elements in a data stream. In J. D. P. Rolim and S. Vadhan, editors, *Randomization and Approximation Techniques in Computer Science*, pages 1–10. Springer Berlin Heidelberg, 2002.