

Verification of the CVM algorithm with a New Recursive Analysis Technique

Emin Karayel, Derek Khu, Kuldeep S. Meel, Yong Kiam Tan,
and Seng Joe Watt

February 6, 2026

Abstract

In 2022, Chakraborty et al. [1] published a streaming algorithm (henceforth, the CVM algorithm) for the distinct elements problem, that deviated considerably from the state-of-the-art, due to its simplicity and avoidance of standard derandomization techniques, while still maintaining a close to optimal logarithmic space complexity.

In this entry, we verify the CVM algorithm's correctness using a new technique which simplifies the analysis considerably compared to the original proof by Chakraborty et al. The main idea is based on a probabilistic invariant that allows us to derive concentration bounds using the Cramér–Chernoff method.

This new technique opens up the possible algorithm design space, and we introduce a new variant of the CVM algorithm, that is total, and also has an additional property in addition to concentration: unbiasedness. This means the expected result of the algorithm is exactly equal to the desired result. The latter is also a new property, that neither the original CVM algorithm nor classic algorithms for the distinct elements problem possess.

Contents

1	Preliminary Definitions and Results	3
2	Abstract Algorithm	5
3	The Original CVM Algorithm	23
4	The New Unbiased Algorithm	30
A	Informal Proof	37
	A.1 Loop Invariant	37
	A.2 Concentration	40
	A.3 Unbiasedness	42

1 Preliminary Definitions and Results

```

theory CVM-Preliminary
  imports HOL-Probability.SPMF
begin

lemma bounded-finite:
  assumes ⟨finite S⟩
  shows ⟨bounded (f ' S)⟩
  using assms by (intro finite-imp-bounded) auto

lemma of-bool-power:
  assumes ⟨y > 0⟩
  shows ⟨(of-bool x::real) ^ (y::nat) = of-bool x⟩
  by (simp add: assms)

lemma card-filter-mono:
  assumes ⟨finite S⟩
  shows ⟨card (Set.filter p S) ≤ card S⟩
  by (intro card-mono assms) auto

fun foldM ::
  ⟨('a ⇒ ('b ⇒ 'c) ⇒ 'c) ⇒ ('b ⇒ 'c) ⇒ ('d ⇒ 'b ⇒ 'a) ⇒ 'd list ⇒ 'b ⇒ 'c⟩ where
  ⟨foldM - return' - [] val = return' val |
  ⟨foldM bind' return' f (x # xs) val =
    bind' (f x val) (foldM bind' return' f xs)⟩

abbreviation foldM-pmf ::
  ⟨('a ⇒ 'b ⇒ 'b pmf) ⇒ 'a list ⇒ 'b ⇒ 'b pmf⟩ where
  ⟨foldM-pmf ≡ foldM bind-pmf return-pmf⟩

lemma foldM-pmf-snoc: ⟨foldM-pmf f (xs@[y]) val = bind-pmf (foldM-pmf f xs val) (f y)⟩
  by (induction xs arbitrary:val)
  (simp-all add: bind-return-pmf bind-return-pmf' bind-assoc-pmf cong:bind-pmf-cong)

abbreviation foldM-spmf
  :: ⟨('a ⇒ 'b ⇒ 'b spmf) ⇒ 'a list ⇒ 'b ⇒ 'b spmf⟩ where
  ⟨foldM-spmf ≡ foldM bind-spmf return-spmf⟩

lemma foldM-spmf-snoc: ⟨foldM-spmf f (xs@[y]) val = bind-spmf (foldM-spmf f xs val) (f y)⟩
  by (induction xs arbitrary:val) (simp-all cong:bind-spmf-cong)

abbreviation ⟨prob-fail ≡ (λx. pmf x None)⟩

abbreviation ⟨fail-spmf ≡ return-pmf None⟩

abbreviation fails-or-satisfies :: ⟨('a ⇒ bool) ⇒ 'a option ⇒ bool⟩ where
  ⟨fails-or-satisfies ≡ case-option True⟩

lemma prob-fail-foldM-spmf-le :
  fixes
  p :: real and

```

```

    P :: ⟨'b ⇒ bool⟩ and
    f :: ⟨'a ⇒ 'b ⇒ 'b spmf⟩
  assumes
    ⟨∧ x y z. P y ⇒ z ∈ set-spmf (f x y) ⇒ P z⟩
    ⟨∧ x val. P val ⇒ prob-fail (f x val) ≤ p⟩
    ⟨P val⟩
  shows ⟨prob-fail (foldM-spmf f xs val) ≤ real (length xs) * p⟩
  using assms(3) proof (induction xs arbitrary: val)
    case Nil
      then show ?case by simp
    next
      case (Cons x xs)

        have p-ge-0: ⟨p ≥ 0⟩ using Cons(2) assms(2) order-trans[OF pmf-nonneg] by metis

        let ?val' = ⟨f x val⟩
        let ?μ' = ⟨measure-spmf ?val'⟩

        have
          ⟨prob-fail (foldM-spmf f (x # xs) val) =
            prob-fail ?val' + ∫ val'. prob-fail (foldM-spmf f xs val') ∂ ?μ'⟩
          by (simp add: pmf-bind-spmf-None)
        also have ⟨... ≤ p + ∫ -. length xs * p ∂ ?μ'⟩
          using assms(1)[OF Cons(2)]
          by (intro add-mono integral-mono-AE iffD2[OF AE-measure-spmf-iff] ballI assms(2))
        Cons
          measure-spmf.integrable-const measure-spmf.integrable-const-bound[where B=⟨1⟩]
          (simp-all add:pmf-le-1)
        also have ⟨... ≤ p + weight-spmf (f x val) * length xs * p⟩
          by simp
        also have ⟨... ≤ p + 1 * real (length xs) * p⟩
          by (intro add-mono mult-right-mono p-ge-0 weight-spmf-le-1) simp-all
        finally show ?case by (simp add: algebra-simps)
      qed

  lemma foldM-spmf-of-pmf-eq :
    shows ⟨foldM-spmf (λx y. spmf-of-pmf (f x y)) xs = spmf-of-pmf ∘ foldM-pmf f xs⟩
    (is ?thesis-0)
    and ⟨foldM-spmf (λx y. spmf-of-pmf (f x y)) xs val = spmf-of-pmf (foldM-pmf f xs
    val)⟩
    (is ?thesis-1)
  proof -
    show ?thesis-0
      by (induction xs) (simp-all add: spmf-of-pmf-bind)

    then show ?thesis-1 by simp
  qed

end

```

2 Abstract Algorithm

This section verifies an abstract version of the CVM algorithm, where the subsampling step can be an arbitrary randomized algorithm fulfilling an expectation invariant.

The abstract algorithm is presented in Algorithm 1.

Algorithm 1 Abstract CVM algorithm.

Input: Stream elements a_1, \dots, a_l , $0 < \varepsilon$, $0 < \delta < 1$, $\frac{1/2}{\varepsilon} f < 1$

Output: An estimate R , s.t., $\mathcal{P}(|R - |A|| > \varepsilon|A|) \leq \delta$ where $A := \{a_1, \dots, a_l\}$.

```

1:  $\chi \leftarrow \{\}, p \leftarrow 1, n \geq \lceil \frac{12}{\varepsilon^2} \ln(\frac{3l}{\delta}) \rceil$ 
2: for  $i \leftarrow 1$  to  $l$  do
3:    $b \stackrel{\$}{\leftarrow} \text{Ber}(p)$  ▷ insert  $a_i$  with probability  $p$  (and remove it otherwise)
4:   if  $b$  then
5:      $\chi \leftarrow \chi \cup \{a_i\}$ 
6:   else
7:      $\chi \leftarrow \chi - \{a_i\}$ 
8:   if  $|\chi| = n$  then
9:      $\chi \stackrel{\$}{\leftarrow} \text{subsample}(\chi)$  ▷ abstract subsampling step
10:     $p \leftarrow pf$ 
11: return  $\frac{|\chi|}{p}$  ▷ estimate cardinality of  $A$ 

```

For the subsampling step we assume that it fulfills the following inequality:

$$\int_{\text{subsample}(\chi)} \left(\prod_{i \in S} g(i \in \omega) \right) d\omega \leq \prod_{i \in S} \left(\int_{\text{Ber}(f)} g(\omega) d\omega \right) \quad (1)$$

for all non-negative functions g and $S \subseteq \chi$, where $\text{Ber}(p)$ denotes the Bernoulli-distribution.

The original CVM algorithm uses a subsampling step where each element of χ is retained independently with probability f . It is straightforward to see that this fulfills the above condition (with equality).

The new CVM algorithm variant proposed in this work uses a subsampling step where a random nf -sized subset of χ is kept. This also fulfills the above inequality, although this is harder to prove and will be explained in more detail in Section 4.

In this section, we will verify that the above abstract algorithm indeed fulfills the desired conditions on its estimate, as well as unbiasedness, i.e., that: $\mathbb{E}[R] = |A|$. The part that is not going to be verified in this section, is the fact that the algorithm keeps at most n elements in the state χ , because it is not unconditionally true, but will be ensured (by different means) for the concrete instantiations in the following sections.

An informal version of this proof is presented in Appendix A. For important lemmas and theorems, we include a reference to the corresponding statement in the appendix.

theory *CVM-Abstract-Algorithm*

```

imports
  HOL-Decision-Procs.Approximation
  CVM-Preliminary
  Finite-Fields.Finite-Fields-More-PMF
  Universal-Hash-Families.Universal-Hash-Families-More-Product-PMF
begin

unbundle no vec-syntax

datatype 'a state = State (state- $\chi$ : 'a set) (state-p: real)

datatype 'a run-state = FinalState 'a list | IntermState 'a list 'a

lemma run-state-induct:
  assumes 'P (FinalState [])
  assumes 'P (FinalState xs)  $\implies$  P (IntermState xs x)
  assumes 'P (IntermState xs x)  $\implies$  P (FinalState (xs@[x]))
  shows 'P result
proof -
  have 'P (FinalState xs)  $\wedge$  P (IntermState xs x) for xs x
  using assms by (induction xs rule:rev-induct) auto
  thus ?thesis by (cases result) auto
qed

locale cvm-algo-abstract =
  fixes n :: nat and f :: real and subsample :: 'a set  $\Rightarrow$  'a set pmf
  assumes n-gt-0: 'n > 0
  assumes f-range: 'f  $\in$  {1/2.. $1$ }
  assumes subsample:
    'P (U x. card U = n  $\implies$  x  $\in$  set-pmf (subsample U)  $\implies$  x  $\subseteq$  U)
  assumes subsample-inequality:
    'P (g U S. S  $\subseteq$  U
       $\implies$  card U = n
       $\implies$  range g  $\subseteq$  {0::real..}
       $\implies$  ( $\int \omega. (\prod_{s \in S. g(s \in \omega))} \partial \text{subsample } U) \leq (\prod_{s \in S. (\int \omega. g \omega \partial \text{bernoulli-pmf } f))$ )
begin

```

Line 1 of Algorithm 1:

```

definition initial-state :: 'a state where
  'initial-state  $\equiv$  State {} 1

```

Lines 3–7:

```

fun step-1 :: 'a  $\Rightarrow$  'a state  $\Rightarrow$  'a state pmf where
  'step-1 a (State  $\chi$  p) =
  do {
    b  $\leftarrow$  bernoulli-pmf p;
    let  $\chi =$  (if b then  $\chi \cup \{a\}$  else  $\chi - \{a\}$ );

    return-pmf (State  $\chi$  p)
  }

```

Lines 8–10:

```

fun step-2 :: ⟨'a state ⇒ 'a state pmf⟩ where
  ⟨step-2 (State χ p) = do {
    if card χ = n
    then do {
      χ ← subsample χ;
      return-pmf (State χ (p*f))
    } else do {
      return-pmf (State χ p)
    }
  }⟩

```

schematic-goal step-1-def: ⟨step-1 x σ = ?x⟩
by (subst state.collapse[symmetric], subst step-1.simps, rule refl)

schematic-goal step-2-def: ⟨step-2 σ = ?x⟩
by (subst state.collapse[symmetric], subst step-2.simps, rule refl)

Lines 1–10:

```

definition run-steps :: ⟨'a list ⇒ 'a state pmf⟩ where
  ⟨run-steps xs ≡ foldM-pmf (λx σ. step-1 x σ ≫ step-2) xs initial-state⟩

```

Line 11:

```

definition estimate :: ⟨'a state ⇒ real⟩ where
  ⟨estimate σ = card (state-χ σ) / state-p σ⟩

```

lemma run-steps-snoc: ⟨run-steps (xs @[x]) = run-steps xs ≫ step-1 x ≫ step-2⟩
unfolding run-steps-def foldM-pmf-snoc **by** (simp add:bind-assoc-pmf)

```

fun run-state-pmf where
  ⟨run-state-pmf (FinalState xs) = run-steps xs⟩ |
  ⟨run-state-pmf (IntermState xs x) = run-steps xs ≫ step-1 x⟩

```

```

fun len-run-state where
  ⟨len-run-state (FinalState xs) = length xs⟩ |
  ⟨len-run-state (IntermState xs x) = length xs⟩

```

```

fun run-state-set where
  ⟨run-state-set (FinalState xs) = set xs⟩ |
  ⟨run-state-set (IntermState xs x) = set xs ∪ {x}⟩

```

lemma finite-run-state-set[simp]: ⟨finite (run-state-set σ)⟩ **by** (cases σ) auto

lemma subsample-finite-pmf:
assumes ⟨card U = n⟩
shows ⟨finite (set-pmf (subsample U))⟩
proof –
have ⟨finite (Pow U)⟩ **unfolding** finite-Pow-iff **using** assms n-gt-0 card-gt-0-iff **by** blast
from finite-subset[OF - this] **show** ?thesis **using** subsample[OF assms] **by** auto
qed

lemma finite-run-state-pmf: ⟨finite (set-pmf (run-state-pmf ρ))⟩
proof (induction ρ rule:run-state-induct)
case 1 **thus** ?case **by** (simp add:run-steps-def)

```

next
  case (2 xs x) thus ?case by (simp add:step-1-def Let-def)
next
  case (3 xs x)
  have a: ⟨run-state-pmf (FinalState (xs@[x])) = run-state-pmf (IntermState xs x) ≫≡
step-2⟩
  by (simp add:run-steps-snoc)
  show ?case unfolding a using 3 subsample-finite-pmf
  by (auto simp:step-2-def simp del:run-state-pmf.simps)
qed

```

lemma *state-χ-run-state-pmf*: ⟨*AE* σ in *run-state-pmf* ρ . *state-χ* $\sigma \subseteq$ *run-state-set* ρ ⟩
proof (*induction* ρ *rule:run-state-induct*)

```

  case 1 thus ?case by (simp add:run-steps-def AE-measure-pmf-iff initial-state-def)

```

```

next

```

```

  case (2 xs x)
  then show ?case by (simp add:AE-measure-pmf-iff Let-def step-1-def) auto

```

```

next

```

```

  case (3 xs x)
  let ?p = ⟨run-state-pmf (IntermState xs x)⟩
  have b: ⟨run-state-pmf (FinalState (xs@[x])) = ?p ≫≡ step-2⟩
  by (simp add:run-steps-snoc)
  show ?case unfolding b using subsample 3 by (simp add:AE-measure-pmf-iff step-2-def
Let-def) blast

```

```

qed

```

lemma *state-χ-finite*: ⟨*AE* σ in *run-state-pmf* ρ . *finite* (*state-χ* σ)⟩

```

  using finite-subset[OF - finite-run-state-set]
  by (intro AE-mp[OF state-χ-run-state-pmf AE-I2]) auto

```

lemma *state-p-range*: ⟨*AE* σ in *run-state-pmf* ρ . *state-p* $\sigma \in \{0 <..1\}$ ⟩

proof (*induction* ρ *rule:run-state-induct*)

```

  case 1 thus ?case by (simp add:run-steps-def AE-measure-pmf-iff initial-state-def)

```

```

next

```

```

  case (2 xs x)
  then show ?case by (simp add:AE-measure-pmf-iff Let-def step-1-def)

```

```

next

```

```

  case (3 xs x)
  let ?p = ⟨run-state-pmf (IntermState xs x)⟩
  have b: ⟨run-state-pmf (FinalState (xs@[x])) = ?p ≫≡ step-2⟩
  by (simp add:run-steps-snoc)
  have ⟨ $x * f \leq 1$ ⟩ if ⟨ $x \in \{0 <..1\}$ ⟩ for  $x$  using f-range that by (intro mult-le-one) auto
  thus ?case unfolding b using 3 f-range by (simp add:AE-measure-pmf-iff step-2-def
Let-def)

```

```

qed

```

Lemma 1:

lemma *run-steps-preserves-expectation-le*:

```

  fixes  $\varphi :: \langle \text{real} \Rightarrow \text{bool} \Rightarrow \text{real} \rangle$ 
  assumes phi :
    ⟨ $\bigwedge x b. \llbracket 0 < x; x \leq 1 \rrbracket \Longrightarrow \varphi x b \geq 0$ ⟩
    ⟨ $\bigwedge p x. \llbracket 0 < p; p \leq 1; 0 < x; x \leq 1 \rrbracket \Longrightarrow (\int \omega. \varphi x \omega \partial \text{bernoulli-pmf } p) \leq \varphi (x / p)$ ⟩
  True

```

```

  ⟨mono-on {0..1} (λx. φ x False)⟩
defines ⟨aux ≡ λ S σ. (∏ x ∈ S. φ (state-p σ) (x ∈ state-χ σ))⟩
assumes ⟨S ⊆ run-state-set ρ⟩
shows ⟨measure-pmf.expectation (run-state-pmf ρ) (aux S) ≤ φ 1 True ^ card S⟩
using assms(5)
proof (induction ρ arbitrary; S rule: run-state-induct)
  case 1 thus ?case by (simp add:aux-def)
next
  case (2 xs x)

  have ⟨finite (set-pmf (run-steps xs))⟩
    using finite-run-state-pmf[where ρ=⟨FinalState xs⟩] by simp
  note [simp] = integrable-measure-pmf-finite[OF this]

  have fin-S: ⟨finite S⟩ using finite-run-state-set 2(2) finite-subset by auto

  have a: ⟨aux S ω = aux (S - {x}) ω * aux (S ∩ {x}) ω⟩ for ω :: ⟨'a state⟩
    unfolding aux-def using fin-S by (subst prod.union-disjoint[symmetric]) (auto intro:prod.cong)

  have b: ⟨finite (set-pmf (run-steps xs ≧ step-1 x))⟩
    using finite-run-state-pmf[where ρ=⟨IntermState xs x⟩] by simp

  have c: ⟨(∫ u. aux (S ∩ {x}) u ∂step-1 x τ) ≤ φ 1 True ^ (card (S ∩ {x}))⟩ (is ⟨?L ≤ ?R⟩)
    if ⟨τ ∈ set-pmf (run-steps xs)⟩ for τ
  proof(cases ⟨x ∈ S⟩)
    case True
      have p-range: ⟨state-p τ ∈ {0 <.. 1}⟩
        using state-p-range[where ρ=⟨FinalState xs⟩] that by (auto simp: AE-measure-pmf-iff)
      have ⟨?L = measure-pmf.expectation (bernoulli-pmf (state-p τ)) (λx. φ (state-p τ) x)⟩
        unfolding step-1-def Let-def map-pmf-def[symmetric] integral-map-pmf aux-def using True
        by (intro integral-cong-AE AE-pmfI simp-all)
      also have ⟨... ≤ φ (state-p τ / state-p τ) True⟩ using p-range by (intro phi(2)) auto
      also have ⟨... ≤ φ 1 True⟩ using p-range by simp
      also have ⟨... = φ 1 True ^ card (S ∩ {x})⟩ using True by simp
      finally show ?thesis by simp
    next
      case False thus ?thesis by (simp add:aux-def)
  qed

  have d: ⟨aux (S - {x}) τ ≥ 0⟩ if ⟨τ ∈ set-pmf (run-steps xs)⟩ for τ
    using state-p-range[where ρ=⟨FinalState xs⟩] that unfolding aux-def
    by (intro prod-nonneg phi(1) power-le-one (auto simp: AE-measure-pmf-iff))

  have ⟨(∫ τ. aux S τ ∂(bind-pmf (run-steps xs) (step-1 x))) =
    (∫ τ. (∫ u. aux (S - {x}) τ * aux (S ∩ {x}) u ∂step-1 x τ) ∂run-steps xs)⟩
    unfolding integral-bind-pmf[OF bounded-finite[OF b]] a
    by (intro integral-cong-AE AE-pmfI arg-cong2[where f=⟨(*)⟩] prod.cong)
    (auto simp:step-1-def aux-def Let-def)
  also have ⟨... = (∫ τ. aux (S - {x}) τ * (∫ u. aux (S ∩ {x}) u ∂step-1 x τ) ∂run-steps xs)⟩
    by simp

```

```

also have  $\langle \dots \leq (\int \tau. aux (S-\{x\}) \tau * (\varphi 1 True)^\wedge card (S\cap\{x\}) \partial run-steps xs) \rangle$ 
  by (intro integral-mono-AE AE-pmfI mult-left-mono c d) auto
also have  $\langle \dots = (\varphi 1 True)^\wedge card (S\cap\{x\}) * (\int \tau. aux (S-\{x\}) \tau \partial (run-state-pmf$ 
(FinalState xs))) \rangle
  by simp
also have  $\langle \dots \leq (\varphi 1 True)^\wedge card (S\cap\{x\}) * (\varphi 1 True)^\wedge card (S - \{x\}) \rangle$ 
  using phi(1) 2(2) by (intro mult-left-mono 2(1)) auto
also have  $\langle \dots = (\varphi 1 True)^\wedge (card ((S\cap\{x\}) \cup (S-\{x\}))) \rangle$ 
  using fin-S by (subst card-Un-disjoint) (auto simp add:power-add)
also have  $\langle \dots = (\varphi 1 True)^\wedge card S \rangle$  by (auto intro!:arg-cong2[where f= $\langle \lambda x y. x^\wedge$ 
(card y) \rangle])
  finally show ?case by simp
next
case ( $\exists xs x$ )
define p where  $\langle p = run-state-pmf (IntermState xs x) \rangle$ 
have a:  $\langle run-state-pmf (FinalState (xs@[x])) = p \ggg step-2 \rangle$ 
  by (simp add:run-steps-snoc p-def)

have fin-S:  $\langle finite S \rangle$  using finite-run-state-set 3(2) finite-subset by auto

have  $\langle finite (set-pmf p) \rangle$ 
  using finite-run-state-pmf[where  $\rho = \langle IntermState xs x \rangle$ ] by (simp add:p-def)
note [simp] = integrable-measure-pmf-finite[OF this]

have b:  $\langle finite (set-pmf (p \ggg step-2)) \rangle$ 
  using finite-run-state-pmf[where  $\rho = \langle FinalState (xs@[x]) \rangle$ ] a by simp

have c:  $\langle (\int u. (\prod s \in S. \varphi (f * state-p \tau) (s \in u)) \partial subsample (state-\chi \tau)) \leq aux S \tau \rangle$ 
  (is  $\langle ?L \leq ?R \rangle$ ) if that':  $\langle card(state-\chi \tau) = n \rangle \langle \tau \in set-pmf p \rangle$  for  $\tau$ 
proof -
  let ?q =  $\langle subsample (state-\chi \tau) \rangle$ 
  let ?U =  $\langle state-\chi \tau \rangle$ 

  define p' where  $\langle p' = f * state-p \tau \rangle$ 

  have d:  $\langle y \subseteq state-\chi \tau \rangle$  if  $\langle y \in set-pmf (subsample (state-\chi \tau)) \rangle$  for y
    using subsample[OF that'(1)] that by auto

  have e:  $\langle state-p \tau \in \{0 <.. 1\} \rangle$ 
    using that(2) unfolding p-def using state-p-range by (meson AE-measure-pmf-iff)
  hence f:  $\langle p' \in \{0 <.. 1\} \rangle$  unfolding p'-def using f-range by (auto intro:mult-le-one)

  have  $\langle ?L = (\int u. (\prod s \in (S-?U) \cup (S\cap?U). \varphi p' (s \in u)) \partial ?q) \rangle$ 
    using fin-S p'-def by (intro integral-cong-AE prod.cong AE-pmfI) auto
  also have  $\langle \dots = (\int u. (\prod s \in S-?U. \varphi p' (s \in u)) * (\prod s \in (S\cap?U). \varphi p' (s \in u)) \partial ?q) \rangle$ 
    using fin-S by (subst prod.union-disjoint) auto
  also have  $\langle \dots = (\int u. (\prod s \in S-?U. \varphi p' False) * (\prod s \in (S\cap?U). \varphi p' (s \in u)) \partial ?q) \rangle$ 
    using d by (intro integral-cong-AE AE-pmfI arg-cong2[where f= $\langle (*) \rangle$ ] prod.cong
      arg-cong2[where f= $\langle \varphi \rangle$ ]) auto
  also have  $\langle \dots = (\prod s \in S-?U. \varphi p' False) * (\int u. (\prod s \in S\cap?U. \varphi p' (s \in u)) \partial ?q) \rangle$ 
    by simp
  also have  $\langle \dots \leq (\prod s \in S-?U. \varphi p' False) * (\prod s \in S\cap?U. (\int u. \varphi p' u \partial bernoulli-pmf$ 
f)) \rangle

```

using *that f* **by** (*intro mult-left-mono subsample-inequality prod-nonneg*) (*auto intro!:**phi(1)*)
also have $\langle \dots \leq (\prod s \in S - ?U. \varphi p' \text{ False}) * (\prod s \in S \cap ?U. \varphi (p'/f) \text{ True}) \rangle$
using *ff-range*
by (*intro mult-left-mono prod-mono phi(2) conjI integral-nonneg-AE AE-pmfI phi(1)*)
prod-nonneg
auto
also have $\langle \dots \leq (\prod s \in S - ?U. \varphi (\text{state-}p \tau) \text{ False}) * (\prod s \in S \cap ?U. \varphi (\text{state-}p \tau) \text{ True}) \rangle$
using *ef f-range unfolding p'-def*
by (*intro mult-mono prod-mono conjI phi(1) mono-onD[OF phi(3)] prod-nonneg*)
power-le-one *auto*
also have $\langle \dots = (\prod s \in S - ?U. \varphi(\text{state-}p \tau) (s \in ?U)) * (\prod s \in S \cap ?U. \varphi(\text{state-}p \tau) (s \in ?U)) \rangle$
by *simp*
also have $\langle \dots = (\prod s \in (S - ?U) \cup (S \cap ?U). \varphi(\text{state-}p \tau) (s \in ?U)) \rangle$
using *fin-S* **by** (*subst prod.union-disjoint[symmetric]*) (*auto*)
also have $\langle \dots = \text{aux } S \tau \rangle$ **unfolding** *aux-def* **by** (*intro prod.cong*) *auto*
finally show *?thesis* **by** *simp*
qed

have $\langle (\int \tau. \text{aux } S \tau \partial \text{run-state-pmf } (\text{FinalState } (xs@[x]))) = (\int \tau. \text{aux } S \tau \partial \text{bind-pmf } p \text{ step-2}) \rangle$
unfolding *a* **by** *simp*
also have $\langle \dots = (\int \tau. (\int u. \text{aux } S u \partial \text{step-2 } \tau) \partial p) \rangle$ **by** (*intro integral-bind-pmf*)
bounded-finite b
also have $\langle \dots = (\int \tau. (\text{if } \text{card}(\text{state-}\chi \tau) = n \text{ then } (\int u. (\prod s \in S. \varphi (f * \text{state-}p \tau) (s \in u)) \partial \text{subsample } (\text{state-}\chi \tau)) \text{ else } \text{aux } S \tau) \partial p) \rangle$
unfolding *step-2-def map-pmf-def[symmetric]* *Let-def aux-def*
by (*simp add:if-distrib if-distribR ac-simps cong:if-cong*)
also have $\langle \dots \leq (\int \tau. (\text{if } \text{card}(\text{state-}\chi \tau) < n \text{ then } \text{aux } S \tau \text{ else } \text{aux } S \tau) \partial p) \rangle$
using *c* **by** (*intro integral-mono-AE AE-pmfI*) *auto*
also have $\langle \dots = (\int \tau. \text{aux } S \tau \partial p) \rangle$ **by** *simp*
also have $\langle \dots \leq \varphi 1 \text{ True } \wedge \text{card } S \rangle$ **using** *3(2) unfolding p-def* **by** (*intro 3(1)*) *auto*
finally show *?case* **by** *simp*
qed

Lemma 2:

lemma *run-steps-preserves-expectation-le'* :

fixes *q* :: *real* **and** *h* :: $\langle \text{real} \Rightarrow \text{real} \rangle$

assumes *h*:

$\langle 0 < q \rangle \langle q \leq 1 \rangle$

$\langle \text{concave-on } \{0 .. 1 / q\} h \rangle$

$\langle \bigwedge x. [0 \leq x; x * q \leq 1] \implies h x \geq 0 \rangle$

defines

$\langle \text{aux} \equiv \lambda S \sigma. (\prod x \in S. \text{of-bool } (\text{state-}p \sigma \geq q) * h (\text{of-bool } (x \in \text{state-}\chi \sigma) / \text{state-}p \sigma)) \rangle$

assumes $\langle S \subseteq \text{run-state-set } \varrho \rangle$

shows $\langle (\int \tau. \text{aux } S \tau \partial \text{run-state-pmf } \varrho) \leq (h 1) \wedge \text{card } S \rangle$ (**is** $\langle ?L \leq ?R \rangle$)

proof –

define φ **where** $\langle \varphi = (\lambda p e. \text{of-bool } (q \leq p) * h(\text{of-bool } e / p)) \rangle$

have φ -1: $\langle \varphi x b \geq 0 \rangle$ **if** $\langle x > 0 \rangle \langle x \leq 1 \rangle$ **for** *x b*

using *h(1,4)* **that** **unfolding** φ -*def* **by** *simp*

```

have  $\varphi$ -2:  $\langle \varphi x \text{ True} * p + \varphi x \text{ False} * (1 - p) \leq \varphi (x / p) \text{ True} \rangle$  if  $\langle x \in \{0..1\} \rangle$   $\langle p \in \{0..1\} \rangle$ 
  for  $x p$ 
proof (cases  $\langle 1 / x \in \{0..1 / q\} \rangle$ )
  case True
    hence  $a: \langle q \leq x \rangle$  using that(1) h(1) by (simp add:divide-simps)
    also have  $\langle \dots \leq x/p \rangle$  using that by (auto simp add:divide-simps)
    finally have  $b: \langle q \leq x / p \rangle$  by simp
    show ?thesis
      unfolding  $\varphi$ -def using that concave-onD[OF h(3) - - True, where t= $\langle p \rangle$  and x= $\langle 0 \rangle$ ] a b h(1)
      by (auto simp:algebra-simps)
  next
  case False
    hence  $a: \langle q > x \rangle$  using that h(1) by (auto simp add:divide-simps)
    hence  $\langle q \leq x/p \implies 0 \leq h (p / x) \rangle$  using that by (intro h(4)) (auto simp:field-simps)
    thus ?thesis using a by (simp add: $\varphi$ -def)
qed
have  $\varphi$ -3:  $\langle \varphi x \text{ False} \leq \varphi y \text{ False} \rangle$  if  $\langle x \leq y \rangle$  for  $x y$ 
  using that by (auto intro:h(4) simp add: $\varphi$ -def)

have  $\langle ?L = (\int \tau. (\prod x \in S. \varphi (\text{state-}p \tau) (x \in \text{state-}\chi \tau)) \partial \text{run-state-pmf } \varrho) \rangle$ 
  unfolding  $\varphi$ -def by (simp add:state-p-def aux-def)
also have  $\langle \dots \leq \varphi 1 \text{ True} \hat{\text{card}} S \rangle$  using  $\varphi$ -1  $\varphi$ -2  $\varphi$ -3
  by (intro run-steps-preserves-expectation-le assms) (auto intro:mono-onI)
also have  $\langle \dots = h 1 \hat{\text{card}} S \rangle$  using h unfolding  $\varphi$ -def by simp
finally show ?thesis by simp
qed

```

Analysis of the case where $n \leq \text{card} (\text{set } xs)$:

context

fixes $xs :: \langle 'a \text{ list} \rangle$

begin

private abbreviation $\langle A \equiv \text{real} (\text{card} (\text{set } xs)) \rangle$

context

assumes *set-larger-than-n*: $\langle \text{card} (\text{set } xs) \geq n \rangle$

begin

private definition $\langle q = \text{real } n / (4 * \text{card} (\text{set } xs)) \rangle$

lemma *q-range*: $\langle q \in \{0..1/4\} \rangle$

using *set-larger-than-n n-gt-0* **unfolding** *q-def* **by** *auto*

lemma *mono-nonnegI*:

assumes $\langle \bigwedge x. x \in I \implies h' x \geq 0 \rangle$

assumes $\langle \bigwedge x. x \in I \implies (h \text{ has-real-derivative } (h' x)) (at x) \rangle$

assumes $\langle x \in I \cap \{0..\} \rangle$ $\langle \text{convex } I \rangle$ $\langle 0 \in I \rangle$ $\langle h 0 \geq 0 \rangle$

shows $\langle h x \geq 0 \rangle$

proof –

have *h-mono*: $\langle h x \leq h y \rangle$ **if** *that'*: $\langle x \leq y \rangle$ $\langle x \in I \rangle$ $\langle y \in I \rangle$ **for** $x y$

proof –

have $\langle \{x..y\} \subseteq I \rangle$ **unfolding** *closed-segment-eq-real-ivl1* [*OF that(1),symmetric*]
by (*intro closed-segment-subset assms that*)
from *subsetD* [*OF this*]
show *?thesis using assms(1,2)* **by** (*intro DERIV-nonneg-imp-nondecreasing* [*OF that(1)*])
auto
qed

have $\langle 0 \leq h \ 0 \rangle$ **using** *assms* **by** *simp*
also have $\langle \dots \leq h \ x \rangle$ **using** *assms* **by** (*intro h-mono*) *auto*
finally show *?thesis* **by** *simp*
qed

lemma *upper-tail-bound-helper*:

assumes $\langle x \in \{0..1::real\} \rangle$
defines $\langle h \equiv (\lambda x. -q * x^2 / 3 - \ln(1 + q * x) + q * \ln(1 + x) * (1 + x)) \rangle$
shows $\langle h \ x \geq 0 \rangle$
proof –
define *h'* **where** $\langle h' \ x = -2*x*q/3 - q/(1+q*x) + q*\ln(1+x) + q \rangle$ **for** $x :: real$

have *a*: $\langle (h \text{ has-real-derivative } (h' \ x)) \ (at \ x) \rangle$ **if** $\langle x \geq 0 \rangle \langle x \leq 1 \rangle$ **for** x
proof –
have $\langle 0 < (1::real) + 0 \rangle$ **by** *simp*
also have $\langle \dots \leq 1 + q * x \rangle$ **using** *that q-range* **by** (*intro add-mono mult-nonneg-nonneg*)
auto

finally have $\langle 0 < 1 + q * x \rangle$ **by** *simp*
thus *?thesis*
using *that q-range* **unfolding** *h-def h'-def power2-eq-square*
by (*auto intro!:derivative-eq-intros*)
qed

have *b*: $\langle h' \ x \geq 0 \rangle$ **if** $\langle x \geq 0 \rangle \langle x \leq 1 \rangle$ **for** x
proof –
have $\langle \exp(2*x/3) = \exp((1-x) *_{\mathbb{R}} 0 + x *_{\mathbb{R}} (2/3)) \rangle$ **by** *simp*
also have $\langle \dots \leq (1-x) * \exp 0 + x * \exp(2/3) \rangle$
using *that* **by** (*intro convex-onD* [*OF exp-convex*]) *auto*
also have $\langle \dots = 1 + x * (\exp(2/3) - \exp 0) \rangle$ **by** (*simp add:algebra-simps*)
also have $\langle \dots \leq 1 + x * 1 \rangle$ **by** (*intro that add-mono order.refl mult-left-mono*)
(approximation 5)

finally have $\langle \exp(2*x/3) \leq \exp(\ln(1+x)) \rangle$ **using** *that* **by** *simp*
hence $\langle 0 \leq \ln(1+x) - 2 * x / 3 \rangle$ **by** *simp*
also have $\langle \dots = \ln(1+x) + 1 - 2*x/3 - 1 \rangle$ **by** *simp*
also have $\langle \dots \leq \ln(1+x) + 1 - 2*x/3 - 1/(1+q*x) \rangle$
using *q-range* **that** **by** (*intro add-mono diff-mono*) (*auto simp:divide-simps*)
finally have *c*: $\langle 0 \leq \ln(1+x) + 1 - 2*x/3 - 1/(1+q*x) \rangle$ **by** *simp*

have $\langle h' \ x = q * (-2*x/3 - 1/(1+q*x) + \ln(1+x) + 1) \rangle$
unfolding *h'-def* **by** (*simp add:algebra-simps*)
also have $\langle \dots \geq 0 \rangle$ **using** *c q-range* **by** (*intro mult-nonneg-nonneg*) *auto*
finally show *?thesis* **by** *simp*
qed

show *?thesis* **by** (*rule mono-nonnegI* [**where** $I = \langle 0..1 \rangle$, *OF b a*]) (*use assms(1) h-def in simp-all*)

qed

private definition ϑ **where** $\langle \vartheta \ t \ x = 1 + q * x * (\exp (t / q) - 1) \rangle$

lemma ϑ -concave: $\langle \text{concave-on } \{0..1 / q\} (\vartheta \ t) \rangle$

unfolding ϑ -def **by** (intro concave-on-linorderI) (auto simp:algebra-simps)

lemma ϑ -ge-exp-1:

assumes $\langle x \in \{0..1/q\} \rangle$

shows $\langle \exp (t * x) \leq \vartheta \ t \ x \rangle$

proof –

have $\langle \exp (t * x) = \exp ((1-q*x) * 0 + (q*x) * (t/q)) \rangle$ **using** q-range **by** simp

also have $\langle \dots \leq (1-q*x) * \exp 0 + (q*x) * \exp (t/q) \rangle$ **using** assms q-range

by (intro convex-onD[OF exp-convex]) (auto simp:field-simps)

also have $\langle \dots = \vartheta \ t \ x \rangle$ **unfolding** ϑ -def **by** (simp add:algebra-simps)

finally show ?thesis **by** simp

qed

lemma ϑ -ge-exp:

assumes $\langle y \geq q \rangle$

shows $\langle \exp (t / y) \leq \vartheta \ t \ (1 / y) \rangle$

using assms ϑ -ge-exp-1[where $x=\langle 1/y \rangle$ and $t=t$] q-range **by** (auto simp:field-simps)

lemma ϑ -nonneg:

assumes $\langle x \in \{0..1/q\} \rangle$

shows $\langle \vartheta \ t \ x \geq 0 \rangle \langle \vartheta \ t \ x > 0 \rangle$

proof –

have $\langle 0 < \exp (t * x) \rangle$ **by** simp

also have $\langle \dots \leq \vartheta \ t \ x \rangle$ **by** (intro ϑ -ge-exp-1 assms)

finally show $\langle \vartheta \ t \ x > 0 \rangle$ **by** simp

thus $\langle \vartheta \ t \ x \geq 0 \rangle$ **by** simp

qed

lemma ϑ -0: $\langle \vartheta \ t \ 0 = 1 \rangle$ **unfolding** ϑ -def **by** simp

lemma tail-bound-aux:

assumes $\langle \text{run-state-set } \varrho \subseteq \text{set } xs \rangle \langle c > 0 \rangle$

defines $\langle A' \equiv \text{real } (\text{card } (\text{run-state-set } \varrho)) \rangle$

shows $\langle \text{measure } (\text{run-state-pmf } \varrho) \{ \omega. \exp (t * \text{estimate } \omega) \geq c \wedge \text{state-p } \omega \geq q \} \leq \vartheta \ t$
 $1 \text{ powr } A'/c \rangle$

(is $\langle ?L \leq ?R \rangle$)

proof –

let $?p = \langle \text{run-state-pmf } \varrho \rangle$

note [simp] = integrable-measure-pmf-finite[OF finite-run-state-pmf]

let $?A' = \langle \text{run-state-set } \varrho \rangle$

let $?X = \langle \lambda i \omega. \text{of-bool } (i \in \text{state-}\chi \ \omega) / \text{state-p } \omega \rangle$

have a: $\langle 0 < \vartheta \ t \ 1 \rangle$ **using** q-range **by** (intro ϑ -nonneg) auto

have $\langle ?L \leq \mathcal{P}(\omega \text{ in } ?p. \text{of-bool}(\text{state-p } \omega \geq q) * \exp (t * \text{estimate } \omega) \geq c) \rangle$

by (intro pmf-mono) auto

also have $\langle \dots \leq (\int \omega. \text{of-bool}(\text{state-p } \omega \geq q) * \exp (t * \text{estimate } \omega) \vartheta ?p) / c \rangle$

by (intro integral-Markov-inequality-measure[**where** $A = \langle \{\} \rangle$] *assms*(2)) *simp-all*
also have $\langle \dots = (\int \omega. \text{of_bool}(\text{state-p } \omega \geq q) * \exp((\sum_{i \in ?A'} t * ?X i \omega)) \partial ?p) / c \rangle$
using *state- χ -run-state-pmf*[**where** $\varrho = \langle \varrho \rangle$] *Int-absorb1*
unfolding *sum-divide-distrib*[*symmetric*] *sum-distrib-left*[*symmetric*] *estimate-def*
by (intro integral-cong-AE arg-cong2[**where** $f = \langle (/) \rangle$])
(auto *simp:AE-measure-pmf-iff* intro!:arg-cong[**where** $f = \langle \text{card} \rangle$])
also have $\langle \dots \leq (\int \omega. (\prod_{i \in ?A'} \text{of_bool}(\text{state-p } \omega \geq q) * \exp(t * ?X i \omega)) \partial ?p) / c \rangle$
unfolding *exp-sum*[*OF finite-run-state-set*] *prod.distrib* **using** *assms*(2)
by (intro *divide-right-mono* *integral-mono-AE* *AE-pmfI*)
(auto intro!:*mult-nonneg-nonneg prod-nonneg*)
also have $\langle \dots \leq (\int \omega. (\prod_{i \in ?A'} \text{of_bool}(\text{state-p } \omega \geq q) * \vartheta t (?X i \omega)) \partial ?p) / c \rangle$
using *assms*(2) *ϑ -ge-exp ϑ -0* **by** (intro *divide-right-mono* *integral-mono-AE* *AE-pmfI*
prod-mono) *auto*
also have $\langle \dots \leq \vartheta t 1 \wedge \text{card } ?A' / c \rangle$ **using** *q-range ϑ -concave* *assms*(2)
by (intro *divide-right-mono* *run-steps-preserves-expectation-le'* *ϑ -nonneg*)
(auto intro!: *ϑ -nonneg simp:field-simps*)
also have $\langle \dots \leq ?R \rangle$
unfolding *A'-def* **using** *card-mono*[*OF - assms*(1)] *assms*(2) *a*
by (*subst powr-realpow*) (auto intro!:*power-increasing divide-right-mono*)
finally show *?thesis* **by** *simp*
qed

Lemma 3:

lemma *upper-tail-bound*:

assumes $\langle \varepsilon \in \{0 < .. 1 :: \text{real}\} \rangle$
assumes $\langle \text{run-state-set } \varrho \subseteq \text{set } xs \rangle$
shows $\langle \text{measure } (\text{run-state-pmf } \varrho) \{ \omega. \text{estimate } \omega \geq (1 + \varepsilon) * A \wedge \text{state-p } \omega \geq q \} \leq \exp(-\text{real } n / 12 * \varepsilon^2) \rangle$
(is $\langle ?L \leq ?R \rangle$)

proof –

let $?p = \langle \text{run-state-pmf } \varrho \rangle$
define t **where** $\langle t = q * \ln(1 + \varepsilon) \rangle$

have $t > 0$: $\langle t > 0 \rangle$ **unfolding** *t-def* **using** *q-range* *assms*(1) **by** *auto*

have *mono-exp-t*: $\langle \text{strict-mono } (\lambda(x::\text{real}). \exp(t * x)) \rangle$
using *t-gt-0* **by** (intro *strict-monoI*) *auto*

have a : $\langle \vartheta t 1 = 1 + q * \varepsilon \rangle$ **using** *assms*(1) **unfolding** *ϑ -def t-def* **by** *simp*

have b : $\langle \vartheta t 1 \geq 1 \rangle$ **unfolding** a **using** *q-range* *assms*(1) **by** *auto*

have c : $\langle \ln(\vartheta t 1) - t * (1 + \varepsilon) \leq -q * \varepsilon^2 / 3 \rangle$
using *upper-tail-bound-helper*[*OF assms*(1)]
unfolding a **unfolding** *t-def* **by** (*simp add:algebra-simps*)

have $\langle ?L = \text{measure } ?p \{ \omega. \exp(t * \text{estimate } \omega) \geq \exp(t * ((1 + \varepsilon) * A)) \wedge \text{state-p } \omega \geq q \} \rangle$
by (*subst strict-mono-less-eq*[*OF mono-exp-t*]) *simp*

also have $\langle \dots \leq \vartheta t 1 \text{ powr real } (\text{card } (\text{run-state-set } \varrho)) / \exp(t * ((1 + \varepsilon) * A)) \rangle$
by (intro *tail-bound-aux* *assms*) *auto*

also have $\langle \dots \leq \vartheta t 1 \text{ powr } A / \exp(t * ((1 + \varepsilon) * A)) \rangle$
using *card-mono*[*OF finite-set* *assms*(2)] b

by (intro *powr-mono* *divide-right-mono*) *auto*

also have $\langle \dots = \exp(A * (\ln(\vartheta t 1) - t * (1 + \varepsilon))) \rangle$

```

    using b unfolding powr-def by (simp add:algebra-simps exp-diff)
  also have ⟨... ≤ exp (A * (-q * ε2/3))⟩
    by (intro iffD2[OF exp-le-cancel-iff] mult-left-mono c) simp
  also have ⟨... = ?R⟩ using set-larger-than-n n-gt-0 unfolding q-def by auto
  finally show ?thesis by simp
qed

Lemma 4:
lemma low-p:
  shows ⟨measure (run-steps xs) {σ. state-p σ < q} ≤ real (length xs) * exp(-real n/12)⟩
    (is ⟨?L ≤ ?R⟩)
proof -
  define ϱ where ϱ = FinalState xs

  have ih: ⟨run-state-set ϱ ⊆ set xs⟩ unfolding ϱ-def by simp

  have ⟨?L = measure (run-state-pmf ϱ) {ω. state-p ω < q}⟩
    unfolding ϱ-def run-state-pmf.simps by simp
  also have ⟨... ≤ real (len-run-state ϱ) * exp(- real n/12)⟩
    using ih
proof (induction ϱ rule:run-state-induct)
  case 1
  then show ?case using q-range by (simp add:run-steps-def initial-state-def)
next
  case (2 ys x)
  let ?pmf = ⟨run-state-pmf (IntermState ys x)⟩
  have a:⟨run-state-set (FinalState ys) ⊆ set xs⟩ using 2(2) by auto

  have ⟨measure ?pmf {ω. state-p ω < q} = (∫ σ. of-bool (state-p σ < q) ∂run-steps ys)⟩
    unfolding run-state-pmf.simps step-1-def Let-def by (simp add:measure-bind-pmf
indicator-def)
  also have ⟨... = (∫ σ. indicator {ω. (state-p ω < q)} σ ∂run-steps ys)⟩
    by (intro integral-cong-AE AE-pmfI) simp-all
  also have ⟨... = measure (run-steps ys) {ω. (state-p ω < q)}⟩ by simp
  also have ⟨... ≤ real (len-run-state (IntermState ys x)) * exp (- real n / 12)⟩
    using 2(1)[OF a] by simp
  finally show ?case by simp
next
  case (3 ys x)
  define p where ⟨p = run-state-pmf (IntermState ys x)⟩

  have ⟨finite (set-pmf p)⟩ unfolding p-def by (intro finite-run-state-pmf)
  note [simp] = integrable-measure-pmf-finite[OF this]

  have a: ⟨run-state-pmf (FinalState (ys@[x])) = p ≫ step-2⟩ (is ⟨?pmf= -⟩)
    by (simp add:run-steps-snoc p-def)

  have b: ⟨run-state-set (IntermState ys x) ⊆ set xs⟩
    using 3(2) by simp

  have c:⟨measure (step-2 σ) {σ. state-p σ < q} ≤
    indicator {σ. state-p σ < q ∨ (card (state-χ σ) = n ∧ state-p σ ∈ {q..<q/f}) } σ⟩
    for σ :: ⟨'a state⟩

```

using *f-range*
by (*simp add:step-2-def Let-def indicator-def map-pmf-def[symmetric] divide-simps*)

have $d: \langle 2 * \text{real} (\text{card} (\text{set } xs)) \leq \text{real } n / \alpha \rangle$ **if** $\langle \alpha \in \{q..<q/f\} \rangle$ **for** α
proof –
have $\langle \alpha \leq q * (1/f) \rangle$ **using** *that* **by** *simp*
also have $\langle \dots \leq q * 2 \rangle$ **using** *q-range f-range* **by** (*intro mult-left-mono*) (*auto simp:divide-simps*)
finally have $\langle \alpha \leq 2*q \rangle$ **by** *simp*
hence $\langle \alpha \leq \text{real } n / (2 * \text{real} (\text{card} (\text{set } xs))) \rangle$
using *set-larger-than-n n-gt-0 unfolding q-def* **by** (*simp add:divide-simps*)
thus *?thesis*
using *set-larger-than-n n-gt-0 that q-range* **by** (*simp add:field-simps*)
qed

hence $\langle \text{measure } p \{ \sigma. \text{card} (\text{state-}\chi \sigma) = n \wedge \text{state-}p \sigma \in \{q..<q/f\} \} \leq$
 $\text{measure } p \{ \sigma. (1+1) * A \leq \text{estimate } \sigma \wedge q \leq \text{state-}p \sigma \} \rangle$
unfolding *estimate-def* **by** (*intro pmf-mono*) (*simp add:estimate-def*)
also have $\langle \dots \leq \exp (- \text{real } n / 12 * 1^2) \rangle$
unfolding *p-def* **by** (*intro upper-tail-bound b*) *simp*
finally have *e*:
 $\langle \text{measure } p \{ \sigma. \text{card} (\text{state-}\chi \sigma) = n \wedge \text{state-}p \sigma \in \{q..<q/f\} \} \leq \exp (- \text{real } n / 12) \rangle$
by *simp*

have $\langle \text{measure} (\text{run-state-pmf} (\text{FinalState} (ys @ [x]))) \{ \omega. \text{state-}p \omega < q \} =$
 $(\int s. \text{measure} (\text{step-}2 s) \{ \omega. \text{state-}p \omega < q \} \partial p) \rangle$
unfolding *a* **by** (*simp add:measure-bind-pmf*)
also have $\langle \dots \leq (\int s. \text{indicator} \{ \omega. \text{state-}p \omega < q \vee \text{card} (\text{state-}\chi \omega) = n \wedge \text{state-}p \omega \in$
 $\{q..<q/f\} \} s \partial p) \rangle$
by (*intro integral-mono-AE AE-pmfI c*) *simp-all*
also have $\langle \dots = \text{measure } p \{ \omega. \text{state-}p \omega < q \vee \text{card} (\text{state-}\chi \omega) = n \wedge \text{state-}p \omega \in$
 $\{q..<q/f\} \} \rangle$
by *simp*
also have $\langle \dots \leq \text{measure } p \{ \omega. \text{state-}p \omega < q \} + \text{measure } p \{ \omega. \text{card} (\text{state-}\chi \omega) = n \wedge \text{state-}p$
 $\omega \in \{q..<q/f\} \} \rangle$
by (*intro pmf-add*) *auto*
also have $\langle \dots \leq \text{length } ys * \exp (- \text{real } n / 12) + \exp (- \text{real } n / 12) \rangle$
using $\mathcal{B}(1)[OF b]$ **by** (*intro add-mono e*) (*simp add:p-def*)
also have $\langle \dots = \text{real} (\text{len-run-state} (\text{FinalState} (ys @ [x]))) * \exp (- \text{real } n / 12) \rangle$
by (*simp add:algebra-simps*)
finally show *?case* **by** *simp*
qed

also have $\langle \dots = \text{real} (\text{length } xs) * \exp (- \text{real } n / 12) \rangle$ **by** (*simp add:q-def*)
finally show *?thesis* **by** *simp*
qed

lemma *lower-tail-bound-helper*:

assumes $\langle x \in \{0 <..<1::\text{real}\} \rangle$

defines $\langle h \equiv (\lambda x. - q * x^2 / 2 - \ln (1 - q * x) + q * \ln (1 - x) * (1 - x)) \rangle$

shows $\langle h x \geq 0 \rangle$

proof –

define *h'* **where** $\langle h' x = -x*q + q/(1-q*x) - q*\ln(1-x) - q \rangle$ **for** *x*

have $a: \langle (h \text{ has-real-derivative } (h' x)) (at x) \rangle$ **if** $\langle x \geq 0 \rangle \langle x < 1 \rangle$ **for** x
proof –
have $\langle q * x \leq (1/4) * 1 \rangle$ **using** *that q-range* **by** *(intro mult-mono) auto*
also have $\langle \dots < 1 \rangle$ **by** *simp*
finally have $\langle q * x < 1 \rangle$ **by** *simp*
thus *?thesis*
using *that q-range unfolding h-def h'-def power2-eq-square*
by *(auto intro!:derivative-eq-intros)*
qed

have $b: \langle h' x \geq 0 \rangle$ **if** $\langle x \geq 0 \rangle \langle x < 1 \rangle$ **for** x
proof –
have $\langle q * x \leq (1/4) * 1 \rangle$ **using** *that q-range* **by** *(intro mult-mono) auto*
also have $\langle \dots < 1 \rangle$ **by** *simp*
finally have $a: \langle q * x < 1 \rangle$ **by** *simp*

have $\langle 0 \leq -\ln(1-x) - x \rangle$ **using** *ln-one-minus-pos-upper-bound[OF that]* **by** *simp*
also have $\langle \dots = -\ln(1-x) - 1 - x + 1 \rangle$ **by** *simp*
also have $\langle \dots \leq -\ln(1-x) - 1 - x + 1 / (1 - q * x) \rangle$
using *a q-range that by (intro add-mono diff-mono) (auto simp:divide-simps)*
finally have $b: \langle 0 \leq -\ln(1-x) - 1 - x + 1 / (1 - q * x) \rangle$ **by** *simp*

have $\langle h' x = q * (-x + 1 / (1 - q * x) - \ln(1-x) - 1) \rangle$
unfolding *h'-def* **by** *(simp add:algebra-simps)*
also have $\langle \dots \geq 0 \rangle$ **using** *b q-range* **by** *(intro mult-nonneg-nonneg) auto*
finally show *?thesis* **by** *simp*
qed

show *?thesis* **by** *(rule mono-nonnegI[where I= $\{0..<1\}$], OF b a)* *(use assms(1) h-def in simp-all)*
qed

Lemma 5:

lemma *lower-tail-bound*:

assumes $\langle \varepsilon \in \{0..<1::real\} \rangle$
shows $\langle \text{measure } (run\text{-steps } xs) \{ \omega. \text{ estimate } \omega \leq (1-\varepsilon) * A \wedge \text{state-p } \omega \geq q \} \leq \exp(-\text{real } n/8*\varepsilon^2) \rangle$
(is $\langle ?L \leq ?R \rangle$)

proof –

let $?p = \langle run\text{-state-pmf } (FinalState xs) \rangle$
define t **where** $\langle t = q * \ln(1-\varepsilon) \rangle$

have $t\text{-lt-0}: \langle t < 0 \rangle$

unfolding *t-def* **using** *q-range assms(1)* **by** *(intro mult-pos-neg ln-less-zero) auto*

have *mono-exp-t*: $\langle \exp(t * x) \leq \exp(t * y) \iff y \leq x \rangle$ **for** $x y$ **using** *t-lt-0* **by** *auto*

have $a: \langle \vartheta t 1 = 1 - q * \varepsilon \rangle$ **using** *assms(1) unfolding ϑ -def t-def* **by** *simp*
have $\langle \varepsilon * (q * 4) \leq 1 * 1 \rangle$ **using** *q-range assms(1)* **by** *(intro mult-mono) auto*
hence $b: \langle \vartheta t 1 \geq 3/4 \rangle$ **unfolding** *a* **by** *(auto simp:algebra-simps)*

have $c: \langle \ln(\vartheta t 1) - t * (1 - \varepsilon) \leq -q * \varepsilon \hat{=} 2 / 2 \rangle$

unfolding *a* **unfolding** *t-def* **using** *lower-tail-bound-helper[OF assms(1)]*

by (simp add:divide-simps)
 have $\langle ?L = \text{measure } ?p \{ \omega. \exp(t * \text{estimate } \omega) \geq \exp(t * ((1-\varepsilon) * A)) \wedge \text{state-}p \ \omega \geq q \} \rangle$
 by (subst mono-exp-t) simp
 also have $\langle \dots \leq \vartheta \ t \ 1 \ \text{powr } \text{card } (\text{run-state-set } (\text{FinalState } xs)) / \exp(t * ((1 - \varepsilon) * A)) \rangle$
 by (intro tail-bound-aux assms) auto
 also have $\langle \dots \leq \vartheta \ t \ 1 \ \text{powr } A / \exp(t * ((1 - \varepsilon) * A)) \rangle$ by simp
 also have $\langle \dots = \exp(A * (\ln(\vartheta \ t \ 1) - t * (1 - \varepsilon))) \rangle$
 using b unfolding powr-def by (simp add:algebra-simps exp-add[symmetric] exp-diff)
 also have $\langle \dots \leq \exp(A * (-q * \varepsilon^2 / 2)) \rangle$
 by (intro iffD2[OF exp-le-cancel-iff] mult-left-mono c) simp
 also have $\langle \dots = ?R \rangle$ using set-larger-than-n n-gt-0 unfolding q-def by auto
 finally show ?thesis by simp
 qed

lemma correctness-aux:

assumes $\langle \varepsilon \in \{0 < .. < 1 :: \text{real}\} \rangle$ $\langle \delta \in \{0 < .. < 1 :: \text{real}\} \rangle$
 assumes $\langle \text{real } n \geq 12 / \varepsilon^2 * \ln(3 * \text{real } (\text{length } xs) / \delta) \rangle$
 shows $\langle \text{measure } (\text{run-steps } xs) \{ \omega. |\text{estimate } \omega - A| > \varepsilon * A \} \leq \delta \rangle$
 (is $\langle ?L \leq ?R \rangle$)

proof -

let $?pmf = \langle \text{run-steps } xs \rangle$
 let $?pmf' = \langle \text{run-state-pmf } (\text{FinalState } xs) \rangle$
 let $?p = \langle \text{state-}p \rangle$
 let $?l = \langle \text{real } (\text{length } xs) \rangle$
 have l-gt-0: $\langle \text{length } xs > 0 \rangle$ using set-larger-than-n n-gt-0 by auto
 hence l-ge-1: $\langle ?l \geq 1 \rangle$ by linarith

have a: $\langle \ln(3 * \text{real } (\text{length } xs) / \delta) = - \ln(\delta / (3 * ?l)) \rangle$
 using l-ge-1 assms(2) by (subst (1 2) ln-div) auto

have $\langle \exp(-\text{real } n / 12 * 1) \leq \exp(-\text{real } n / 12 * \varepsilon^2) \rangle$
 using assms(1) by (intro iffD2[OF exp-le-cancel-iff] mult-left-mono-neg power-le-one)

auto

also have $\langle \dots \leq \delta / (3 * ?l) \rangle$
 using assms(1-3) l-ge-1 unfolding a by (subst ln-ge-iff[symmetric]) (auto simp:divide-simps)
 finally have $\langle \exp(-\text{real } n / 12) \leq \delta / (3 * ?l) \rangle$ by simp
 hence b: $\langle ?l * \exp(-\text{real } n / 12) \leq \delta / 3 \rangle$ using l-gt-0 by (auto simp:field-simps)

have $\langle \exp(-\text{real } n / 12 * \varepsilon^2) \leq \delta / (3 * ?l) \rangle$
 using assms(1-3) l-ge-1 unfolding a by (subst ln-ge-iff[symmetric]) (auto simp:divide-simps)
 also have $\langle \dots \leq \delta / 3 \rangle$ using assms(1-3) l-ge-1 by (intro divide-left-mono) auto
 finally have c: $\langle \exp(-\text{real } n / 12 * \varepsilon^2) \leq \delta / 3 \rangle$ by simp

have $\langle \exp(-\text{real } n / 8 * \varepsilon^2) \leq \exp(-\text{real } n / 12 * \varepsilon^2) \rangle$ by (intro iffD2[OF exp-le-cancel-iff])

auto

also have $\langle \dots \leq \delta / 3 \rangle$ using c by simp
 finally have d: $\langle \exp(-\text{real } n / 8 * \varepsilon^2) \leq \delta / 3 \rangle$ by simp

have $\langle ?L \leq \text{measure } ?pmf \{ \omega. |\text{estimate } \omega - A| \geq \varepsilon * A \} \rangle$ by (intro pmf-mono) auto
 also have $\langle \dots \leq \text{measure } ?pmf \{ \omega. |\text{estimate } \omega - A| \geq \varepsilon * A \wedge ?p \ \omega \geq q \} + \text{measure}$

```

?pmf { $\omega$ . ?p  $\omega < q$ }
  by (intro pmf-add) auto
  also have  $\langle \dots \leq \text{measure } ?\text{pmf } \{\omega. (\text{estimate } \omega \leq (1-\varepsilon) * A \vee \text{estimate } \omega \geq (1+\varepsilon) * A) \wedge ?p \omega \geq q\} +$ 
    ?l * exp(-real n/12) $\rangle$ 
  by (intro pmf-mono add-mono low-p) (auto simp:abs-real-def algebra-simps split:if-split-asm)
  also have  $\langle \dots \leq \text{measure } ?\text{pmf } \{\omega. \text{estimate } \omega \leq (1-\varepsilon) * A \wedge \text{state-p } \omega \geq q\} +$ 
    measure ?pmf'  $\{\omega. \text{estimate } \omega \geq (1+\varepsilon) * A \wedge \text{state-p } \omega \geq q\} + \delta/3 \rangle$ 
  unfolding run-state-pmf.simps by (intro add-mono pmf-add b) auto
  also have  $\langle \dots \leq \text{exp}(-\text{real } n/8 * \varepsilon^2) + \text{exp}(-\text{real } n/12 * \varepsilon^2) + \delta/3 \rangle$ 
  using assms(1) by (intro upper-tail-bound add-mono lower-tail-bound) auto
  also have  $\langle \dots \leq \delta/3 + \delta/3 + \delta/3 \rangle$  by (intro add-mono d c) auto
  finally show ?thesis by simp
qed

```

end

lemma deterministic-phase:

```

assumes  $\langle \text{card } (\text{run-state-set } \sigma) < n \rangle$ 
shows  $\langle \text{run-state-pmf } \sigma = \text{return-pmf } (\text{State } (\text{run-state-set } \sigma) 1) \rangle$ 
using assms
proof (induction  $\sigma$  rule:run-state-induct)
case 1 thus ?case by (simp add:run-steps-def initial-state-def)
next
case (2 xs x)
have  $\langle \text{card } (\text{set } xs) < n \rangle$  using 2(2) by (simp add:card-insert-if) presburger
moreover have  $\langle \text{bernoulli-pmf } 1 = \text{return-pmf } \text{True} \rangle$ 
by (intro pmf-eqI) (auto simp:bernoulli-pmf.rep-eq)
ultimately show ?case using 2(1) by (simp add:step-1-def bind-return-pmf)
next
case (3 xs x)
let ?p =  $\langle \text{run-state-pmf } (\text{IntermState } xs x) \rangle$ 
have a:  $\langle \text{card } (\text{run-state-set } (\text{IntermState } xs x)) < n \rangle$  using 3(2) by simp
have b:  $\langle \text{run-state-pmf } (\text{FinalState } (xs@[x])) = ?p \gg \text{step-2} \rangle$ 
by (simp add:run-steps-snoc)
show ?case
using 3(2) unfolding b 3(1)[OF a] by (simp add:step-2-def bind-return-pmf Let-def)
qed

```

Theorem 1:

theorem correctness:

```

fixes  $\varepsilon \delta :: \text{real}$ 
assumes  $\langle \varepsilon \in \{0 < .. < 1\} \rangle \langle \delta \in \{0 < .. < 1\} \rangle$ 
assumes  $\langle \text{real } n \geq 12 / \varepsilon^2 * \ln (3 * \text{real } (\text{length } xs) / \delta) \rangle$ 
shows  $\langle \text{measure } (\text{run-steps } xs) \{\omega. |\text{estimate } \omega - A| > \varepsilon * A\} \leq \delta \rangle$ 
proof (cases  $\langle \text{card } (\text{set } xs) \geq n \rangle$ )
case True
show ?thesis by (intro correctness-aux True assms)
next
case False
hence  $\langle \text{run-steps } xs = \text{return-pmf } (\text{State } (\text{set } xs) 1) \rangle$ 
using deterministic-phase[where  $\sigma = \langle \text{FinalState } xs \rangle$ ] by simp
thus ?thesis using assms(1,2) by (simp add:indicator-def estimate-def not-less)

```

qed

lemma *p-pos*: $\langle \exists M \in \{0 <..1\}. AE \ \omega \text{ in run-steps } xs. \text{ state-p } \omega \geq M \rangle$

proof –

have $\langle \text{finite } (\text{set-pmf } (\text{run-steps } xs)) \rangle$
using *finite-run-state-pmf* [**where** $\rho = \langle \text{FinalState } xs \rangle$] **by** *simp*
define *M* **where** $\langle M = (\text{MIN } \sigma \in \text{set-pmf } (\text{run-steps } xs). \text{ state-p } \sigma) \rangle$
have $\langle M \in \text{state-p } ' \text{set-pmf } (\text{run-steps } xs) \rangle$
using *fin set-pmf-not-empty* **unfolding** *M-def* **by** (*intro Min-in*) *auto*
also have $\langle \dots \subseteq \{0 <..1\} \rangle$
using *state-p-range* [**where** $\rho = \langle \text{FinalState } xs \rangle$]
by (*intro image-subsetI*) (*simp add: AE-measure-pmf-iff*)
finally have $\langle M \in \{0 <..1\} \rangle$ **by** *simp*

moreover have $\langle AE \ \omega \text{ in run-steps } xs. \text{ state-p } \omega \geq M \rangle$

using *fin unfolding AE-measure-pmf-iff M-def* **by** (*intro ballI Min-le*) *auto*
ultimately show *?thesis* **by** *auto*

qed

lemma *run-steps-expectation-sing*:

assumes *i*: $\langle i \in \text{set } xs \rangle$

shows $\langle \text{measure-pmf.expectation } (\text{run-steps } xs) (\lambda \omega. \text{ of-bool } (i \in \text{state-}\chi \ \omega) / \text{state-p } \omega) = 1 \rangle$

(**is** $\langle ?L = - \rangle$)

proof –

have $\langle \text{finite } (\text{set-pmf } (\text{run-steps } xs)) \rangle$
using *finite-run-state-pmf* [**where** $\rho = \langle \text{FinalState } xs \rangle$] **by** *simp*
note *int = integrable-measure-pmf-finite* [*OF this*]

obtain *M* **where** $\langle AE \ \sigma \text{ in run-steps } xs. M \leq \text{state-p } \sigma \rangle$ **and** *M-range*: $\langle M \in \{0 <..1\} \rangle$

using *p-pos* **by** *blast*

then have $\langle ?L = (\int \tau. (\prod x \in \{i\}. \text{ of-bool } (M \leq \text{state-p } \tau)) * (\text{ of-bool } (x \in \text{state-}\chi \ \tau) / \text{state-p } \tau)) \rangle$

$\partial \text{run-state-pmf } (\text{FinalState } xs) \rangle$

by (*auto intro!: integral-cong-AE*)

also have $\langle \dots \leq 1 \wedge \text{card } \{i\} \rangle$

using *M-range i* **by** (*intro run-steps-preserves-expectation-le'*) (*auto simp: concave-on-iff*)

finally have *le*: $\langle ?L \leq 1 \rangle$ **by** *auto*

have *concave*: $\langle \text{concave-on } \{0..1 / M\} ((-) (1 / M + 1)) \rangle$

unfolding *concave-on-iff*

using *M-range* **apply** (*clarsimp simp add: field-simps*)

by (*metis combine-common-factor distrib-right linear mult-1-left*)

have $\langle 1 / M + 1 - ?L = (\int \omega. 1 / M + 1 - \text{ of-bool } (i \in \text{state-}\chi \ \omega) / \text{state-p } \omega) \partial \text{run-steps } xs \rangle$

by (*auto simp:int*)

also have $\langle \dots = (\int \tau. (\prod x \in \{i\}. \text{ of-bool } (M \leq \text{state-p } \tau)) * (1 / M + 1 - \text{ of-bool } (x \in \text{state-}\chi \ \tau) / \text{state-p } \tau)) \partial \text{run-state-pmf } (\text{FinalState } xs) \rangle$

using \ast **by** (*auto intro!: integral-cong-AE*)

also have $\langle \dots \leq (1 / M + 1 - 1) \wedge \text{card } \{i\} \rangle$

using *i M-range*

```

    by (intro run-steps-preserves-expectation-le'[OF - - concave]) (auto simp: field-simps)
  also have ⟨... = 1 / M⟩ by auto
  finally have ge:⟨-?L ≤ -1⟩ by auto
  show ?thesis using le ge by auto
qed

```

Subsection A.3:

corollary unbiasedness:

fixes $\sigma :: \langle 'a \text{ run-state} \rangle$

shows $\langle \text{measure-pmf.expectation (run-steps xs) estimate} = \text{real (card (set xs))} \rangle$
 (is $\langle ?L = - \rangle$)

proof –

have $\langle \text{finite (set-pmf (run-steps xs))} \rangle$

using *finite-run-state-pmf*[**where** $\varrho = \langle \text{FinalState xs} \rangle$] **by** *simp*

note [*simp*] = *integrable-measure-pmf-finite*[OF *this*]

have $s: \langle \text{AE } \omega \text{ in run-steps xs. state-}\chi \ \omega \subseteq \text{set xs} \rangle$

by (*metis run-state-pmf.simps(1) run-state-set.simps(1) state-}\chi-run-state-pmf*)

have $\langle ?L = (\int \omega. (\sum_{i \in \text{set xs. of-bool (i \in state-}\chi \ \omega)}) / \text{state-p } \omega \ \partial \text{run-steps xs}) \rangle$

unfolding *estimate-def state-p-def*[*symmetric*]

by (*auto intro!*: *integral-cong-AE intro: AE-mp*[OF *s*] *simp add: Int-absorb1*)

also have $\langle \dots = (\int \omega. (\sum_{i \in \text{set xs. of-bool (i \in state-}\chi \ \omega)} / \text{state-p } \omega) \ \partial \text{run-steps xs}) \rangle$

by (*metis (no-types) sum-divide-distrib*)

also have $\langle \dots = (\sum_{i \in \text{set xs.} (\int \omega. \text{of-bool (i \in state-}\chi \ \omega)} / \text{state-p } \omega \ \partial \text{run-steps xs})) \rangle$

by (*auto intro: Bochner-Integration.integral-sum*)

also have $\langle \dots = (\sum_{i \in \text{set xs.} 1) \rangle$

using *run-steps-expectation-sing* **by** (*auto cong:sum.cong*)

finally show ?thesis **by** *auto*

qed

end

end

end

3 The Original CVM Algorithm

In this section, we verify the algorithm as presented by Chakraborty et al. [1] (repliated, here, in Algorithm 2), with the following caveat:

In the original algorithm the elements are removed with probability $f := \frac{1}{2}$ in the subsampling step. The version verified here allows for any $f \in [\frac{1}{2}, e^{-1/12}]$.

Algorithm 2 Original CVM algorithm.

Input: Stream elements a_1, \dots, a_l , $0 < \varepsilon$, $0 < \delta < 1$, f subsampling param.

Output: An estimate R , s.t., $\mathcal{P}(|R - |A|| > \varepsilon|A|) \leq \delta$ where $A := \{a_1, \dots, a_l\}$.

```

1:  $\chi \leftarrow \{\}, p \leftarrow 1, n \geq \lceil \frac{12}{\varepsilon^2} \ln(\frac{6l}{\delta}) \rceil$ 
2: for  $i \leftarrow 1$  to  $l$  do
3:    $b \stackrel{\$}{\leftarrow} \text{Ber}(p)$   $\triangleright$  insert  $a_i$  with probability  $p$  (and remove it otherwise)
4:   if  $b$  then
5:      $\chi \leftarrow \chi \cup \{a_i\}$ 
6:   else
7:      $\chi \leftarrow \chi - \{a_i\}$ 
8:   if  $|\chi| = n$  then
9:      $\chi \stackrel{\$}{\leftarrow} \text{subsample}(\chi)$   $\triangleright$  keep each element of  $\chi$  indep. with prob.  $f$ 
10:     $p \leftarrow pf$ 
11:   if  $|\chi| = n$  then
12:     return  $\perp$ 
13: return  $\frac{|\chi|}{p}$   $\triangleright$  estimate cardinality of  $A$ 

```

The first step of the proof is identical to the original proof [1], where the above algorithm is approximated by a second algorithm, where lines 11–12 are removed, i.e., the two algorithms behave identically, unless the very improbable event—where the subsampling step fails to remove any elements—occurs. It is possible to show that the total variational distance between the two algorithms is at most $\frac{\delta}{2}$.

In the second step, we verify that the probability that the second algorithm returns an estimate outside of the desired interval is also at most $\frac{\delta}{2}$. This, of course, works by noticing that it is an instance of the abstract algorithm we introduced in Section 2. In combination, we conclude a failure probability of δ for the unmodified version of the algorithm.

On the other hand, the fact that the number of elements in the buffer is at most n can be seen directly from Algorithm 2.

theory *CVM-Original-Algorithm*

imports *CVM-Abstract-Algorithm*

begin

context

fixes $f :: \text{real}$

fixes $n :: \text{nat}$

assumes $f\text{-range: } \langle f \in \{1/2..exp(-1/12)\} \rangle$

assumes $n\text{-gt-0: } \langle n > 0 \rangle$

begin

Line 1:

```
definition initial-state :: ⟨'a state⟩ where  
  ⟨initial-state = State {} 1⟩
```

Lines 3–7:

```
fun step-1 :: ⟨'a ⇒ 'a state ⇒ 'a state spmf⟩ where  
  ⟨step-1 a (State χ p) =  
    do {  
      b ← bernoulli-pmf p;  
      let χ = (if b then χ ∪ {a} else χ - {a});  
  
      return-spmf (State χ p)  
    }⟩
```

```
definition subsample :: ⟨'a set ⇒ 'a set spmf⟩ where  
  ⟨subsample χ =  
    do {  
      keep-in-χ ← prod-pmf χ (λ-. bernoulli-pmf f);  
      return-spmf (Set.filter keep-in-χ χ)  
    }⟩
```

Lines 8–10:

```
fun step-2 :: ⟨'a state ⇒ 'a state spmf⟩ where  
  ⟨step-2 (State χ p) =  
    do {  
      if card χ = n then do {  
        χ ← subsample χ;  
        return-spmf (State χ (p * f))  
      } else  
        return-spmf (State χ p)  
    }⟩
```

Lines 11–12:

```
fun step-3 :: ⟨'a state ⇒ 'a state spmf⟩ where  
  ⟨step-3 (State χ p) =  
    do {  
      if card χ = n  
      then fail-spmf  
      else return-spmf (State χ p)  
    }⟩
```

Lines 1–12:

```
definition run-steps :: ⟨'a list ⇒ 'a state spmf⟩ where  
  ⟨run-steps xs ≡ foldM-spmf (λx σ. step-1 x σ ≧≧ step-2 ≧≧ step-3) xs initial-state⟩
```

Line 13:

```
definition estimate :: ⟨'a state ⇒ real⟩ where  
  ⟨estimate σ = card (state-χ σ) / state-p σ⟩
```

```
definition run-algo :: ⟨'a list ⇒ real spmf⟩ where  
  ⟨run-algo xs = map-spmf estimate (run-steps xs)⟩
```

schematic-goal *step-1-m-def*: $\langle \text{step-1 } x \sigma = ?x \rangle$
by (*subst state.collapse[symmetric]*, *subst step-1.simps*, *rule refl*)

schematic-goal *step-2-m-def*: $\langle \text{step-2 } \sigma = ?x \rangle$
by (*subst state.collapse[symmetric]*, *subst step-2.simps*, *rule refl*)

schematic-goal *step-3-m-def*: $\langle \text{step-3 } \sigma = ?x \rangle$
by (*subst state.collapse[symmetric]*, *subst step-3.simps*, *rule refl*)

lemma *ord-spmf-remove-step3*:
 $\langle \text{ord-spmf } (=) (\text{step-1 } x \sigma \ggg \text{step-2} \ggg \text{step-3}) (\text{step-1 } x \sigma \ggg \text{step-2}) \rangle$
proof –
have $\langle \text{ord-spmf } (=) (\text{step-2 } x \ggg \text{step-3}) (\text{step-2 } x) \rangle$ **for** $x :: \langle 'a \text{ state} \rangle$
proof –
have $\langle \text{ord-spmf } (=) (\text{step-2 } x \ggg \text{step-3}) (\text{step-2 } x \ggg \text{return-spmf}) \rangle$
by (*intro bind-spmf-mono'*) (*simp-all add:step-3-m-def*)
thus *?thesis* **by** *simp*
qed
thus *?thesis* **unfolding** *bind-spmf-assoc* **by** (*intro bind-spmf-mono'*) *simp-all*
qed

lemma *ord-spmf-run-steps*:
 $\langle \text{ord-spmf } (=) (\text{run-steps } xs) (\text{foldM-spmf } (\lambda x \sigma. \text{step-1 } x \sigma \ggg \text{step-2}) xs \text{initial-state}) \rangle$
unfolding *run-steps-def*
proof (*induction xs rule:rev-induct*)
case *Nil*
then show *?case* **by** *simp*
next
case (*snoc x xs*)
show *?case*
unfolding *run-steps-def foldM-spmf-snoc*
by (*intro ord-spmf-remove-step3 bind-spmf-mono' snoc*)
qed

lemma *f-range-simple*: $\langle f \in \{1/2..<1\} \rangle$
proof –
have $\langle \text{exp } (- 1 / 12) < (1::\text{real}) \rangle$ **by** (*approximation 5*)
from *dual-order.strict-trans2[OF this]*
show *?thesis* **using** *f-range* **by** *auto*
qed

Main result:

theorem *correctness*:
fixes $xs :: \langle 'a \text{ list} \rangle$
assumes $\langle \varepsilon \in \{0 < .. < 1\} \rangle \langle \delta \in \{0 < .. < 1\} \rangle$
assumes $\langle \text{real } n \geq 12 / \varepsilon^2 * \ln (6 * \text{real } (\text{length } xs) / \delta) \rangle$
defines $\langle A \equiv \text{real } (\text{card } (\text{set } xs)) \rangle$
shows $\langle \mathcal{P}(\omega \text{ in run-algo } xs. \text{fails-or-satisfies } (\lambda R. |R - A| > \varepsilon * A) \omega) \leq \delta \rangle$
(is $\langle ?L \leq ?R \rangle$ **)**
proof –
define *abs-subsample* **where**
 $\langle \text{abs-subsample } \chi = \text{map-pmf } (\lambda \omega. \text{Set.filter } \omega \chi) (\text{prod-pmf } \chi (\lambda -. \text{bernoulli-pmf } f)) \rangle$

```

for  $\chi :: \langle 'a \text{ set} \rangle$ 

interpret abs:cvm-algo-abstract n f abs-subsample
  rewrites  $\langle \text{abs.estimate} = \text{estimate} \rangle$ 
proof -
show abs: $\langle \text{cvm-algo-abstract } n \text{ f abs-subsample} \rangle$ 
proof (unfold-locales, goal-cases)
  case 1 thus ?case by (rule n-gt-0)
next
  case 2 thus ?case using f-range-simple by auto
next
  case ( $\exists U x$ )
  then show ?case unfolding abs-subsample-def by auto
next
  case ( $\lambda g \chi S$ )
  hence fin-U:  $\langle \text{finite } \chi \rangle$  using n-gt-0 card-gt-0-iff by metis
  note conv = Pi-pmf-subset[OF this  $\lambda(1)$ ]

  have  $\langle (\int \omega. (\prod s \in S. g (s \in \omega)) \partial \text{abs-subsample } \chi) =$ 
     $(\int \omega. (\prod s \in S. g (s \in \chi \wedge \omega s)) \partial \text{prod-pmf } \chi (\lambda-. \text{bernoulli-pmf } f)) \rangle$ 
    unfolding abs-subsample-def by (simp cong:prod.cong)
  also have  $\langle \dots = (\int \omega. (\prod s \in S. g (s \in \chi \wedge \omega s)) \partial \text{prod-pmf } S (\lambda-. \text{bernoulli-pmf } f)) \rangle$ 
    unfolding conv by simp
  also have  $\langle \dots = (\prod s \in S. (\int \omega. g (s \in \chi \wedge \omega) \partial \text{bernoulli-pmf } f)) \rangle$ 
    using fin-U finite-subset[OF  $\lambda(1)$ ]
    by (intro expectation-prod-Pi-pmf integrable-measure-pmf-finite) auto
  also have  $\langle \dots = (\prod s \in S. (\int \omega. g \omega \partial \text{bernoulli-pmf } f)) \rangle$ 
    using  $\lambda(1)$  by (intro prod.cong refl) auto
  finally show ?case by simp
qed
show  $\langle \text{cvm-algo-abstract.estimate} = (\text{estimate} :: 'a \text{ state} \Rightarrow \text{real}) \rangle$ 
  unfolding cvm-algo-abstract.estimate-def[OF abs] estimate-def by simp
qed

have a:  $\langle \text{step-1 } \sigma x = \text{spmf-of-pmf } (\text{abs.step-1 } \sigma x) \rangle$  for  $\sigma x$ 
  unfolding step-1-m-def abs.step-1-def Let-def spmf-of-pmf-def by (simp add:map-bind-pmf)

have b:  $\langle \text{step-2 } \sigma = \text{map-pmf } \text{Some } (\text{abs.step-2 } \sigma) \rangle$  for  $\sigma$ 
  unfolding step-2-m-def abs.step-2-def subsample-def abs-subsample-def Let-def
  by (simp add:map-bind-pmf bind-pmf-return-spmf)

have c:  $\langle \text{abs.initial-state} = \text{initial-state} \rangle$ 
  unfolding initial-state-def abs.initial-state-def by simp

have d:  $\langle \text{subsample } \chi = \text{spmf-of-pmf } (\text{abs-subsample } \chi) \rangle$  for  $\chi$ 
  unfolding subsample-def abs-subsample-def map-pmf-def[symmetric]
  by (simp add:spmf-of-pmf-def map-pmf-comp)

define  $\alpha :: \text{real}$  where  $\langle \alpha = f \hat{\ } n \rangle$ 

have  $\alpha$ -range:  $\langle \alpha \in \{0..1\} \rangle$ 
  using f-range-simple unfolding  $\alpha$ -def by (auto intro:power-le-one)
hence [simp]:  $\langle |\alpha| \leq 1 \rangle$  by auto

```

have $\langle (\int x. (if\ card\ x = n\ then\ 1\ else\ 0)\ \partial abs\text{-}sample\ \chi) \leq \alpha \rangle$ **(is** $\langle ?L1 \leq - \rangle$)
if that': $\langle card\ \chi = n \rangle$ **for** χ
proof –
have $fin\text{-}U: \langle finite\ \chi \rangle$ **using** $n\text{-}gt\text{-}0$ **that** $card\text{-}gt\text{-}0\text{-}iff$ **by** $metis$

have $\langle (\prod s \in \chi. of\text{-}bool\ (s \in x)::real) = of\text{-}bool(card\ x = n) \rangle$
if $\langle x \in set\text{-}pmf\ (abs\text{-}sample\ \chi) \rangle$ **for** x
proof –
have $x\text{-}ran: \langle x \subseteq \chi \rangle$ **using** $that$ **unfolding** $abs\text{-}sample\text{-}def$ **by** $auto$

have $\langle (\prod s \in \chi. of\text{-}bool\ (s \in x)::real) = of\text{-}bool(x = \chi) \rangle$
using $fin\text{-}U\ x\text{-}ran$ **by** $(induction\ \chi\ rule:finite\text{-}induct)\ auto$
also have $\langle \dots = of\text{-}bool\ (card\ x = card\ \chi) \rangle$
using $x\text{-}ran\ fin\text{-}U\ card\text{-}subset\text{-}eq$ **by** $(intro\ arg\text{-}cong[where\ f = \langle of\text{-}bool \rangle])\ blast$
also have $\langle \dots = of\text{-}bool\ (card\ x = n) \rangle$ **using** $that'$ **by** $simp$
finally show $?thesis$ **by** $auto$

qed
hence $\langle ?L1 = (\int x. (\prod s \in \chi. of\text{-}bool(s \in x))\ \partial abs\text{-}sample\ \chi) \rangle$
by $(intro\ integral\text{-}cong\text{-}AE\ AE\text{-}pmfI)\ simp\text{-}all$
also have $\langle \dots \leq (\prod s \in \chi. (\int x. of\text{-}bool\ x\ \partial bernoulli\text{-}pmf\ f)) \rangle$
by $(intro\ abs\text{-}sample\text{-}inequality\ that)\ auto$
also have $\langle \dots = f \wedge card\ \chi \rangle$ **using** $f\text{-}range\text{-}simple$ **by** $simp$
also have $\langle \dots = \alpha \rangle$ **unfolding** $\alpha\text{-}def\ that$ **by** $simp$
finally show $?thesis$ **by** $simp$

qed
hence $e: \langle pmf\ (step\text{-}2\ \sigma \ggg step\text{-}3)\ None \leq \alpha \rangle$ **for** $\sigma :: \langle 'a\ state \rangle$
using $\alpha\text{-}range$ **unfolding** $step\text{-}2\text{-}m\text{-}def\ step\text{-}3\text{-}m\text{-}def\ d\ Let\text{-}def$
by $(simp\ add:pmf\text{-}bind\ bind\text{-}pmf\text{-}return\text{-}spmf\ if\text{-}distrib\ if\text{-}distribR\ cong:if\text{-}cong)$

have $\langle pmf\ (step\text{-}1\ x\ \sigma \ggg step\text{-}2 \ggg step\text{-}3)\ None \leq \alpha \rangle$ **for** σ **and** $x :: 'a$
proof –
have $\langle pmf\ (step\text{-}1\ x\ \sigma \ggg step\text{-}2 \ggg step\text{-}3)\ None \leq 0 + (\int -. \alpha\ \partial\ measure\text{-}spmf\ (step\text{-}1\ x\ \sigma)) \rangle$
unfolding $bind\text{-}spmf\text{-}assoc\ pmf\text{-}bind\text{-}spmf\text{-}None$ **[where** $p = \langle step\text{-}1\ x\ \sigma \rangle$ **]**
by $(intro\ add\text{-}mono\ integral\text{-}mono\text{-}AE\ measure\text{-}spmf.integrable\text{-}const\text{-}bound$ **[where** $B = \langle 1 \rangle$ **]**
 $iffD2$ **[OF** $AE\text{-}measure\text{-}spmf\text{-}iff$ **]** $ballI\ e)$
 $(simp\text{-}all\ add:pmf\text{-}le\text{-}1\ step\text{-}1\text{-}m\text{-}def\ map\text{-}pmf\text{-}def[symmetric])\ pmf\text{-}map\ vimage\text{-}def\ Let\text{-}def)$
also have $\langle \dots \leq \alpha \rangle$ **using** $\alpha\text{-}range$ **by** $(simp\ add: mult\text{-}left\text{-}le\text{-}one\text{-}le\ weight\text{-}spmf\text{-}le\text{-}1)$
finally show $?thesis$ **by** $simp$

qed
hence $\langle prob\text{-}fail\ (run\text{-}steps\ xs) \leq length\ xs * \alpha \rangle$
unfolding $run\text{-}steps\text{-}def$ **by** $(intro\ prob\text{-}fail\text{-}foldM\text{-}spmf\text{-}le$ **[where** $P = \langle \lambda -. True \rangle$ **])\ auto
also have $\langle \dots \leq \delta / 2 \rangle$
proof $(cases\ \langle xs = [] \rangle)$
case $True$
thus $?thesis$ **using** $assms(2)$ **by** $auto$**

next
case $False$
have $\langle \delta \leq 6 * 1 \rangle$ **using** $assms(2)$ **by** $simp$
also have $\langle \dots \leq 6 * real\ (length\ xs) \rangle$

using *False* **by** (*intro mult-mono order.refl*) (*cases xs, auto*)
finally have [*simp*]: $\langle \delta \leq 6 * \text{real} (\text{length } xs) \rangle$ **by** *simp*
have $\langle 2 * \text{real} (\text{length } xs) * f^n \leq 2 * \text{real} (\text{length } xs) * \exp (-1/12)^n \rangle$
using *f-range* **by** (*intro mult-left-mono power-mono*) *auto*
also have $\langle \dots = 2 * \text{real} (\text{length } xs) * \exp (-\text{real } n / 12) \rangle$
unfolding *exp-of-nat-mult[symmetric]* **by** *simp*
also have $\langle \dots \leq 2 * \text{real} (\text{length } xs) * \exp (-(12 / \varepsilon^2 * \ln (6 * \text{real} (\text{length } xs) / \delta)) / 12) \rangle$
using *assms(3)* **by** (*intro mult-left-mono iffD2[OF exp-le-cancel-iff] divide-right-mono*) *auto*
also have $\langle \dots = 2 * \text{real} (\text{length } xs) * \exp (-\ln (6 * \text{real} (\text{length } xs) / \delta) / \varepsilon^2) \rangle$
by *auto*
also have $\langle \dots \leq 2 * \text{real} (\text{length } xs) * \exp (-\ln (6 * \text{real} (\text{length } xs) / \delta) / 1) \rangle$
using *assms(1,2)* *False*
by (*intro mult-left-mono iffD2[OF exp-le-cancel-iff] divide-left-mono-neg power-le-one*)
(auto intro!:ln-ge-zero simp:divide-simps)
also have $\langle \dots = 2 * \text{real} (\text{length } xs) * \exp (\ln (\text{inverse} (6 * \text{real} (\text{length } xs) / \delta))) \rangle$
using *False* *assms(2)* **by** (*subst ln-inverse[symmetric]*) *auto*
also have $\langle \dots = 2 * \text{real} (\text{length } xs) / (6 * \text{real} (\text{length } xs) / \delta) \rangle$
using *assms(1,2)* *False* **by** (*subst exp-ln*) *auto*
also have $\langle \dots = \delta / 3 \rangle$ **using** *False* *assms(2)* **by** *auto*
also have $\langle \dots \leq \delta \rangle$ **using** *assms(2)* **by** *auto*
finally have $\langle 2 * \text{real} (\text{length } xs) * f^n \leq \delta \rangle$ **by** *simp*
thus *?thesis* **unfolding** *α -def* **by** *simp*
qed
finally have *f*: $\langle \text{prob-fail} (\text{run-steps } xs) \leq \delta / 2 \rangle$ **by** *simp*

have *g*: $\langle \text{spmf-of-pmf} (\text{abs.run-steps } xs) = \text{foldM-spmf} (\lambda x \sigma. \text{step-1 } x \sigma \gg \text{step-2}) xs \text{ initial-state} \rangle$

unfolding *abs.run-steps-def foldM-spmf-of-pmf-eq(2)[symmetric]*
unfolding *spmf-of-pmf-def map-pmf-def c b a*
by (*simp add:bind-assoc-pmf bind-spmf-def bind-return-pmf*)

have $\langle ?L \leq \text{measure} (\text{run-steps } xs) \{None\} + \text{measure} (\text{measure-spmf} (\text{run-steps } xs)) \{x. |\text{estimate } x - A| > \varepsilon * A\} \rangle$
unfolding *run-algo-def measure-measure-spmf-conv-measure-pmf measure-map-pmf*
by (*intro pmf-add*) (*auto split:option.split-asm*)
also have $\langle \dots \leq \delta / 2 + \text{measure} (\text{measure-spmf} (\text{run-steps } xs)) \{x. |\text{estimate } x - A| > \varepsilon * A\} \rangle$

unfolding *measure-pmf-single* **by** (*intro add-mono f order.refl*)
also have $\langle \dots \leq \delta / 2 + \text{measure} (\text{measure-spmf} (\text{spmf-of-pmf} (\text{abs.run-steps } xs))) \{x. |\text{estimate } x - A| > \varepsilon * A\} \rangle$
using *ord-spmf-eqD-emeasure[OF ord-spmf-run-steps]* **unfolding** *measure-spmf.emeasure-eq-measure* *g*

by (*intro add-mono*) *auto*
also have $\langle \dots \leq \delta / 2 + \text{measure} (\text{abs.run-steps } xs) \{x. |\text{estimate } x - A| > \varepsilon * A\} \rangle$
using *measure-spmf-map-pmf-Some* *spmf-of-pmf-def* **by** *auto*
also have $\langle \dots \leq \delta / 2 + \delta / 2 \rangle$
using *assms(1-3)* **unfolding** *A-def* **by** (*intro add-mono abs.correctness*) *auto*
finally show *?thesis* **by** *simp*

qed

lemma *space-usage*:

```

  ⟨AE σ in measure-spmf (run-steps xs). card (state-χ σ) < n ∧ finite (state-χ σ)⟩
proof (induction xs rule:rev-induct)
  case Nil thus ?case using n-gt-0 by (simp add:run-steps-def initial-state-def)
next
  case (snoc x xs)
  define p1 where ⟨p1 = run-steps xs ≫= step-1 x⟩
  define p2 where ⟨p2 = p1 ≫= step-2⟩
  define p3 where ⟨p3 = p2 ≫= step-3⟩

  have a:⟨run-steps (xs@[x]) = p3⟩
  unfolding run-steps-def p1-def p2-def p3-def foldM-spmf-snoc by (simp add:bind-assoc-pmf)

  have ⟨card (state-χ σ) ≤ n ∧ finite (state-χ σ)⟩ if ⟨σ ∈ set-spmf p1⟩ for σ
  using snoc that less-imp-le unfolding p1-def
  by (auto simp: step-1-m-def set-bind-spmf set-spmf-bind-pmf Let-def card-insert-if)+

  hence ⟨card (state-χ σ) ≤ n ∧ finite (state-χ σ)⟩ if ⟨σ ∈ set-spmf p2⟩ for σ
  using that card-filter-mono unfolding p2-def
  by (auto intro!:card-filter-mono simp:step-2-m-def set-bind-spmf set-spmf-bind-pmf
    subsample-def Let-def if-distrib)
    blast

  hence ⟨card (state-χ σ) < n ∧ finite (state-χ σ)⟩ if ⟨σ ∈ set-spmf p3⟩ for σ
  using that unfolding p3-def
  by (auto intro:le-neq-implies-less simp:step-3-m-def set-bind-spmf if-distrib)

  thus ?case unfolding a by simp
qed

end

end

```

4 The New Unbiased Algorithm

In this section, we introduce the new algorithm variant promised in the abstract. The main change is to replace the subsampling step of the original algorithm, which removes each element of the buffer independently with probability f . Instead, we choose a random nf -subset of the buffer (see Algorithm 3). (This means f, n must be chosen, such that nf is an integer.)

Algorithm 3 New CVM algorithm.

Input: Stream elements a_1, \dots, a_l , $0 < \varepsilon$, $0 < \delta < 1$, f subsampling param.

Output: An estimate R , s.t., $\mathcal{P}(|R - |A|| > \varepsilon|A|) \leq \delta$ where $A := \{a_1, \dots, a_l\}$.

```

1:  $\chi \leftarrow \{\}$ ,  $p \leftarrow 1$ ,  $n \geq \lceil \frac{12}{\varepsilon^2} \ln(\frac{3l}{\delta}) \rceil$ 
2: for  $i \leftarrow 1$  to  $l$  do
3:    $b \stackrel{\$}{\leftarrow} \text{Ber}(p)$   $\triangleright$  insert  $a_i$  with probability  $p$  (and remove it otherwise)
4:   if  $b$  then
5:      $\chi \leftarrow \chi \cup \{a_i\}$ 
6:   else
7:      $\chi \leftarrow \chi - \{a_i\}$ 
8:   if  $|\chi| = n$  then
9:      $\chi \stackrel{\$}{\leftarrow} \text{subsample}(\chi)$   $\triangleright$  Choose a random  $nf$ -subset of  $\chi$ 
10:     $p \leftarrow pf$ 
11: return  $\frac{|\chi|}{p}$   $\triangleright$  estimate cardinality of  $A$ 

```

The fact that this still preserves the required inequality for the subsampling operation (Eq. 1) follows from the negative associativity of permutation distributions [2, Th. 10].

(See also our formalization of the concept [3].)

Because the subsampling step always removes elements unconditionally, the second check, whether the subsampling succeeded of the original algorithm is not necessary anymore.

This improves the space usage of the algorithm, because the first reduction argument from Section 3 is now obsolete. Moreover the resulting algorithm is now unbiased, because it is an instance of the abstract algorithm of Section 2.

theory *CVM-New-Unbiased-Algorithm*

imports

CVM-Abstract-Algorithm

Probabilistic-Prime-Tests.Generalized-Primality-Test

Negative-Association.Negative-Association-Permutation-Distributions

begin

unbundle *no vec-syntax*

context

fixes $f :: \text{real}$ **and** $n :: \text{nat}$

assumes *f-range*: $\langle f \in \{1/2..<1\} \rangle$ $\langle n * f \in \mathbb{N} \rangle$ **and** *n-gt-0*: $\langle n > 0 \rangle$

begin

definition $\langle \text{initial-state} = \text{State } \{\} \ 1 \rangle$ — Setup initial state $\chi = \emptyset$ and $p = 1$.
fun *subsample* **where** — Subsampling operation: Sample random nf subset.
 $\langle \text{subsample } \chi = \text{pmf-of-set } \{S. S \subseteq \chi \wedge \text{card } S = n * f\} \rangle$

fun *step* **where** — Loop body.
 $\langle \text{step } a \ (\text{State } \chi \ p) = \text{do } \{$
 $\quad b \leftarrow \text{bernoulli-pmf } p;$
 $\quad \text{let } \chi = (\text{if } b \text{ then } \chi \cup \{a\} \text{ else } \chi - \{a\});$

 $\quad \text{if } \text{card } \chi = n \text{ then } \text{do } \{$
 $\quad \quad \chi \leftarrow \text{subsample } \chi;$
 $\quad \quad \text{return-pmf } (\text{State } \chi \ (p * f))$
 $\quad \quad \} \text{ else } \text{do } \{$
 $\quad \quad \quad \text{return-pmf } (\text{State } \chi \ p)$
 $\quad \quad \}$
 $\quad \}$
 \rangle

fun *run-steps* **where** — Iterate loop over stream xs .
 $\langle \text{run-steps } xs = \text{foldM-pmf } \text{step } xs \ \text{initial-state} \rangle$

fun *estimate* **where**
 $\langle \text{estimate } (\text{State } \chi \ p) = \text{card } \chi / p \rangle$

fun *run-algo* **where** — Run algorithm and estimate.
 $\langle \text{run-algo } xs = \text{map-pmf } \text{estimate } (\text{run-steps } xs) \rangle$

definition $\langle \text{subsample-size} = (\text{THE } x. \text{real } x = n * f) \rangle$

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

lemma *subsample-size-eq*:

$\langle \text{real } \text{subsample-size} = n * f \rangle$

proof —

obtain a **where** $a\text{-def} : \langle \text{real } a = \text{real } n * f \rangle$ **using** $f\text{-range}(2)$ **by** (*metis Nats-cases*)

show $?thesis$

unfolding *subsample-size-def* **using** $a\text{-def}$

by ($\text{rule } \text{theI2}[\text{where } a = \langle a \rangle]$) ($\text{use } a\text{-def}$ **in** *auto*)

qed

lemma *subsample-size*:

$\langle \text{subsample-size} < n \rangle \ \langle 2 * \text{subsample-size} \geq n \rangle$

proof (*goal-cases*)

case 1

have $\langle \text{real } \text{subsample-size} < \text{real } n \rangle$

unfolding *subsample-size-eq* **using** $f\text{-range}(1)$ $n\text{-gt-0}$ **by** *auto*

thus $?case$ **by** *simp*

next

case 2

have $\langle \text{real } n \leq 2 * \text{real } \text{subsample-size} \rangle$

using $f\text{-range}(1)$ $n\text{-gt-0}$ **unfolding** *subsample-size-eq* **by** *auto*

thus $?case$ **by** *simp*

qed

lemma *subsample-finite-nonempty*:
assumes $\langle \text{card } U = n \rangle$
shows
 $\langle \{T. T \subseteq U \wedge \text{card } T = \text{subsample-size}\} \neq \{\}\rangle$ (**is** $\langle ?C \neq \{\}\rangle$)
 $\langle \text{finite } \{T. T \subseteq U \wedge \text{card } T = \text{subsample-size}\} \rangle$
 $\langle \text{subsample } U = \text{pmf-of-set } \{T. T \subseteq U \wedge \text{card } T = \text{subsample-size}\} \rangle$
 $\langle \text{finite } (\text{set-pmf } (\text{subsample } U)) \rangle$

proof –
have *fin-U*: $\langle \text{finite } U \rangle$ **using** *assms subsample-size*
by (*meson card-gt-0-iff le0 order-le-less-trans order-less-le-trans*)
have *a*: $\langle \text{card } U \text{ choose subsample-size} > 0 \rangle$
using *subsample-size assms* **by** (*intro zero-less-binomial*) **auto**
show *b*: $\langle \text{subsample } U = \text{pmf-of-set } ?C \rangle$
using *subsample-size-eq unfolding subsample.simps*
by (*intro arg-cong[where f= $\langle \text{pmf-of-set} \rangle$] Collect-cong*) **auto**
with *assms subsample-size* **have** $\langle \text{card } ?C > 0 \rangle$
using *n-subsets[OF fin-U]* **by** *simp*
thus $\langle ?C \neq \{\}\rangle$ $\langle \text{finite } ?C \rangle$ **using** *card-gt-0-iff* **by** *blast+*
thus $\langle \text{finite } (\text{set-pmf } (\text{subsample } U)) \rangle$ **unfolding** *b* **by** *auto*

qed

lemma *int-prod-subsample-eq-prod-int*:
fixes *g* :: $\langle \text{bool} \Rightarrow \text{real} \rangle$
assumes $\langle \text{card } U = n \rangle$ $\langle S \subseteq U \rangle$ $\langle \text{range } g \subseteq \{0..\} \rangle$
shows $\langle (\int \omega. (\prod s \in S. g(s \in \omega))) \partial \text{subsample } U \leq (\prod s \in S. (\int \omega. g \omega \partial \text{bernoulli-pmf } f)) \rangle$
(is $\langle ?L \leq ?R \rangle$)

proof –
define *η* **where** $\langle \eta \equiv \text{if } g \text{ True} \geq g \text{ False} \text{ then } F \text{wd} \text{ else } R \text{ev} \rangle$

have *fin-U*: $\langle \text{finite } U \rangle$ **using** *assms subsample-size*
by (*meson card-gt-0-iff le0 order-le-less-trans order-less-le-trans*)

note *subsample* = *subsample-finite-nonempty*[*OF assms(1)*]

note [*simp*] = *integrable-measure-pmf-finite*[*OF subsample(4)*]

let *?C* = $\langle \{T. T \subseteq U \wedge \text{card } T = \text{subsample-size}\} \rangle$

have *subsample-size-le-card-U*: $\langle \text{subsample-size} \leq \text{card } U \rangle$
using *subsample-size* **unfolding** *assms(1)* **by** *simp*

have $\langle \text{measure-pmf.neg-assoc } (\text{subsample } U) (\lambda s \omega. (s \in \omega)) U \rangle$
using *subsample-size-le-card-U* **unfolding** *subsample*
by (*intro n-subsets-distribution-neg-assoc fin-U*)

hence *na*: $\langle \text{measure-pmf.neg-assoc } (\text{subsample } U) (\lambda s \omega. (s \in \omega)) S \rangle$
using *measure-pmf.neg-assoc-subset*[*OF assms(2)*] **by** *auto*

have *fin-S*: $\langle \text{finite } S \rangle$ **using** *assms(2)* *fin-U* *finite-subset* **by** *auto*

note *na-imp-prod-mono* = *has-int-thatD*(2)[*OF measure-pmf.neg-assoc-imp-prod-mono*[*OF fin-S na*]]

have *g-borel*: $\langle g \in \text{borel-measurable borel} \rangle$ **by** (*intro borel-measurable-continuous-onI*)

```

simp
  have g-mono-aux: ⟨g x ≤η g y⟩ if ⟨x ≤ y⟩ for x y
    unfolding η-def using that by simp (smt (verit, best))
  have g-mono: ⟨monotone (≤) (≤η) g⟩
    by (intro monotoneI) (auto simp:dir-le-refl intro!:g-mono-aux)

  have a: ⟨map-pmf (λω. s ∈ ω) (subsample U) = bernoulli-pmf f⟩ if ⟨s ∈ U⟩ for s
  proof -
    have ⟨measure (pmf-of-set ?C) {x. s ∈ x} = real subsample-size / card U⟩
      by (intro n-subsets-prob subssample-size-le-card-U that fin-U)
    also have ⟨... = f⟩ unfolding subsample-size-eq assms(1) using n-gt-0 by auto
    finally have ⟨measure (pmf-of-set ?C) {x. s ∈ x} = f⟩ by simp
    thus ?thesis
      unfolding subsample by (intro eq-bernoulli-pmfI) (simp add: pmf-map vimage-def)
  qed

  have ⟨?L ≤ (∏ s ∈ S. (∫ ω. g (s ∈ ω) ∂subsample U))⟩
    by (intro na-imp-prod-mono[OF - g-mono] g-borel assms(3)) auto
  also have ⟨... = (∏ s ∈ S. (∫ ω. g ω ∂map-pmf (λω. s ∈ ω) (subsample U)))⟩ by simp
  also have ⟨... = ?R⟩ using a assms(2) by (intro prod.cong refl) (metis in-mono)
  finally show ?thesis .
qed

schematic-goal step-n-def: ⟨step x σ = ?x⟩
  by (subst state.collapse[symmetric], subst step.simps, rule refl)

interpretation abs: cvm-algo-abstract n f subsample
  rewrites ⟨abs.run-steps = run-steps⟩ and ⟨abs.estimate = estimate⟩
proof -
  show abs:⟨cvm-algo-abstract n f subsample⟩
  proof (unfold-locales, goal-cases)
    case 1 thus ?case using subsample-size by auto
  next
    case 2 thus ?case using f-range by auto
  next
    case (3 U x) thus ?case using subsample-finite-nonempty[OF 3(1)] by simp
  next
    case (4 g U S) thus ?case by (intro int-prod-subsample-eq-prod-int) auto
  qed
  have a:⟨(λx σ. cvm-algo-abstract.step-1 x σ ≍ cvm-algo-abstract.step-2 n f subsample)
= step⟩
    unfolding cvm-algo-abstract.step-1-def[OF abs] cvm-algo-abstract.step-2-def[OF abs]
  step-n-def
    by (intro ext) (simp add: bind-assoc-pmf Let-def bind-return-pmf cong:if-cong)
  have c:⟨cvm-algo-abstract.initial-state = initial-state⟩
    unfolding cvm-algo-abstract.initial-state-def[OF abs] initial-state-def by auto
  show ⟨cvm-algo-abstract.run-steps n f subsample = run-steps⟩
    unfolding cvm-algo-abstract.run-steps-def[OF abs] run-steps.simps a c by simp
  show ⟨cvm-algo-abstract.estimate = estimate⟩
    unfolding cvm-algo-abstract.estimate-def[OF abs]
    by (intro ext) (metis estimate.simps state.collapse)
qed

```

theorem unbiasedness: $\langle \text{measure-pmf.expectation (run-algo xs) id} = \text{card (set xs)} \rangle$
unfolding *run-algo.simps integral-map-pmf using abs.unbiasedness by simp*

theorem correctness:

assumes $\langle \varepsilon \in \{0 < .. < 1 :: \text{real}\} \rangle$ $\langle \delta \in \{0 < .. < 1 :: \text{real}\} \rangle$
assumes $\langle \text{real } n \geq 12 / \varepsilon^2 * \ln (3 * \text{real (length xs)} / \delta) \rangle$
defines $\langle A \equiv \text{real (card (set xs))} \rangle$
shows $\langle \mathcal{P}(R \text{ in run-algo xs. } |R - A| > \varepsilon * A) \leq \delta \rangle$
using *assms(3) unfolding A-def using abs.correctness[OF assms(1,2)] by auto*

lemma space-usage:

$\langle AE \sigma \text{ in run-steps xs. card (state-}\chi \sigma) < n \wedge \text{finite (state-}\chi \sigma) \rangle$

proof –

define ρ **where** $\langle \rho = \text{FinalState xs} \rangle$
have $\langle \text{card (state-}\chi \sigma) < n + (\text{case } \rho \text{ of FinalState -} \Rightarrow 0 \mid \text{IntermState -} \Rightarrow 1) \rangle$
if $\langle \sigma \in \text{set-pmf (abs.run-state-pmf } \rho) \rangle$ **for** σ
using *that*
proof (*induction* ρ *arbitrary:* σ *rule:* *run-state-induct*)
case 1
then show *?case using n-gt-0 by (simp add:initial-state-def)*
next
case (2 $xs \ x$)
have $\langle \text{card (state-}\chi \tau) < n \wedge \text{finite (state-}\chi \tau) \rangle$
if $\langle \tau \in \text{set-pmf (abs.run-state-pmf (FinalState xs))} \rangle$ **for** τ
using 2(1) *abs.state-}\chi-finite[where* $\rho = \langle \text{FinalState xs} \rangle$ *that by (simp add:AE-measure-pmf-iff)*
thus *?case*
using 2(2) **unfolding** *abs.step-1-def abs.run-state-pmf.simps Let-def map-pmf-def[symmetric]*
by (*force simp: card-insert-if*)

next

case (3 $xs \ x$)
define p **where** $\langle p = \text{abs.run-state-pmf (IntermState xs x)} \rangle$

have $a: \langle \text{abs.run-state-pmf (FinalState (xs@[x]))} = p \ggg \text{abs.step-2} \rangle$
by (*simp add:p-def abs.run-steps-snoc del:run-steps.simps*)

have $b: \langle \text{card } \chi < \text{card (state-}\chi \tau) \rangle$
if $\langle \text{card (state-}\chi \tau) = n \rangle$ $\langle \chi \in \text{set-pmf (subsample (state-}\chi \tau)) \rangle$ $\langle \tau \in \text{set-pmf } p \rangle$ **for**

$\chi \ \tau$

proof –

from *subsample-finite-nonempty[OF that(1)]*
have $\langle \text{card } \chi = \text{subsample-size} \rangle$ **using** *that unfolding subsample-def by auto*
thus *?thesis using subsample-size(1) that by auto*
qed

have $\langle \text{card (state-}\chi \tau) < n \vee \text{card (state-}\chi \tau) = n \rangle$ $\langle \text{finite (state-}\chi \tau) \rangle$
if $\langle \tau \in \text{set-pmf } p \rangle$ **for** τ
using 3(1) *abs.state-}\chi-finite[where* $\rho = \langle \text{IntermState xs x} \rangle$ *that unfolding p-def*
by (*auto simp:AE-measure-pmf-iff less-Suc-eq*)

hence $\langle \text{card (state-}\chi \sigma) < n \rangle$
using 3(2) **unfolding** *a abs.step-2-def Let-def by (auto intro!:b simp:if-distrib if-distribR)*

thus *?case by simp*
qed

```
thus ?thesis using abs.state- $\chi$ -finite[where  $\rho = \langle \text{FinalState } xs \rangle$ ] unfolding  $\rho$ -def
  by (simp add:AE-measure-pmf-iff)
qed

end

end
```

References

- [1] S. Chakraborty, N. V. Vinodchandran, and K. S. Meel. Distinct elements in streams: An algorithm for the (text) book. In S. Chechik, G. Navarro, E. Rotenberg, and G. Herman, editors, *ESA*, volume 244 of *LIPICs*, pages 34:1–34:6. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.
- [2] D. P. Dubhashi, V. Priebe, and D. Ranjan. Negative dependence through the fkg inequality. *BRICS Report Series*, 3, 1996.
- [3] E. Karayel. Negatively associated random variables. *Archive of Formal Proofs*, January 2025. https://isa-afp.org/entries/Negative_Association.html, Formal proof development.

A Informal Proof

This section includes an informal version of the proof for the tail bounds and unbiasedness of the abstract algorithm (Algorithm 1) for interested readers.

This means we assume the $\text{subsample}(\chi)$ operation fulfills Eq. 1 and always returns a subset of χ .

Notation: For a finite set S , the probability space of uniformly sampling from the set is denoted by $U(S)$, i.e., for each $s \in S$ we have $\mathcal{P}_{U(S)}(s) = |S|^{-1}$. We write $\text{Ber}(p)$ for the Bernoulli probability space, over the set $\{0, 1\}$, i.e., $P_{\text{Ber}(p)}(\{1\}) = p$. $I(P)$ is the indicator function for a predicate P , i.e., $I(\text{true}) = 1$ and $I(\text{false}) = 0$.

Like in the formalization, we will denote the first five lines of the loop (3–7) as step 1, the last four lines (8–10) as step 2. For the distribution of the state of the algorithm after processing i elements of the sequence, we will write Ω_i . The elements of the probability spaces are pairs composed of a set and the number of subsampling steps, representing χ and p respectively.

For example: $\Omega_0 = U(\{(\emptyset, 1)\})$ is the initial state, $\Omega_1 = U(\{(\{a_1\}, 1)\})$, etc., and Ω_l denotes the final state. We introduce χ and p as random variables defined over such probability spaces Ω , in particular, χ (resp. p) is the projection to the first (resp. second) component.

The state of the algorithm after processing only step 1 of the i -th loop iteration is denoted by Ω'_i . So the sequence of states is represented by the distributions $\Omega_0, \Omega'_1, \Omega_1, \dots, \Omega'_l, \Omega_l$.

A.1 Loop Invariant

After these preliminaries, we can go to the main proof, whose core is a probabilistic loop invariant for Algorithm 1 that can be verified inductively.

Lemma 1. Let $\varphi : (0, 1] \times \{0, 1\} \rightarrow \mathbb{R}_{\geq 0}$ be a function, fulfilling the following conditions:

1. $(1 - \alpha)\varphi(x, 0) + \alpha\varphi(x, 1) \leq \varphi(x/\alpha, 1)$ for all $0 < \alpha < 1$, $0 < x \leq 1$, and
2. $\varphi(x, 0) \leq \varphi(y, 0)$ for all $0 < x < y \leq 1$.

Then for all $k \in \{0, \dots, l\}$, $S \subseteq \{a_1, \dots, a_k\}$, $\Omega \in \{\Omega_k, \Omega'_k\}$:

$$\mathbb{E}_{\Omega} \left[\prod_{s \in S} \varphi(p, I(s \in \chi)) \right] \leq \varphi(1, 1)^{|S|}$$

Proof. We show the result using induction over k . Note that we show the statement for arbitrary S , i.e., the induction statements are:

$$P(k) \quad :\Leftrightarrow \quad \left(\forall S \subseteq \{a_1, \dots, a_k\}. \mathbb{E}_{\Omega_k} \left[\prod_{s \in S} \varphi(p, I(s \in \chi)) \right] \leq \varphi(1, 1)^{|S|} \right)$$

$$Q(k) \quad :\Leftrightarrow \quad \left(\forall S \subseteq \{a_1, \dots, a_k\}. \mathbb{E}_{\Omega'_k} \left[\prod_{s \in S} \varphi(p, I(s \in \chi)) \right] \leq \varphi(1, 1)^{|S|} \right)$$

and we will show $P(0), Q(1), P(1), Q(2), P(2), \dots, Q(l), P(l)$ successively.

Induction start $P(0)$:

We have $S \subseteq \emptyset$, and hence

$$\mathbb{E}_{\Omega_0} \left[\prod_{s \in S} \varphi(p, I(s \in \chi)) \right] = \mathbb{E}_{\Omega_0} [1] = 1 \leq \varphi(1, 1)^0.$$

Induction step $P(k) \rightarrow Q(k+1)$:

Let $S \subseteq \{a_1, \dots, a_{k+1}\}$ and define $S' := S - \{a_{k+1}\}$. Note that Ω'_{k+1} can be constructed from Ω_k as a compound distribution, where a_{k+1} is included in the buffer, with the probability p , which is itself a random variable over the space Ω_k .

In particular, for example:

$$\mathcal{P}_{\Omega'_{k+1}}(P(\chi, p)) = \int_{\Omega_k} \int_{\text{Ber}(p(\omega))} P(\chi(\omega) - \{a_{k+1}\} \cup f(\tau), p(\omega)) d\tau d\omega$$

for all predicates P where we define $f(1) = \{a_{k+1}\}$ and $f(0) = \emptyset$.

We distinguish the two cases $a_{k+1} \in S$ and $a_{k+1} \notin S$. If $a_{k+1} \in S$:

$$\begin{aligned} & \mathbb{E}_{\Omega'_{k+1}} \left[\prod_{s \in S} \varphi(p, I(s \in \chi)) \right] \\ &= \int_{\Omega_k} \left(\prod_{s \in S'} \varphi(p, I(s \in \chi)) \right) \int_{\text{Ber}(p(\omega))} \varphi(p, \tau) d\tau d\omega \\ &= \int_{\Omega_k} \left(\prod_{s \in S'} \varphi(p, I(s \in \chi)) \right) ((1-p)\varphi(p, 0) + p\varphi(p, 1)) d\omega \\ &\stackrel{\text{Cond 1}}{\leq} \int_{\Omega_k} \left(\prod_{s \in S'} \varphi(p, I(s \in \chi)) \right) \varphi(1, 1) d\omega \\ &\stackrel{\text{IH}}{\leq} \varphi(1, 1)^{|S'|} \varphi(1, 1) = \varphi(1, 1)^{|S|} \end{aligned}$$

If $a_{k+1} \notin S$ then $S' = S$ and:

$$\mathbb{E}_{\Omega'_{k+1}} \left[\prod_{s \in S} \varphi(p, I(s \in \chi)) \right] = \int_{\Omega_k} \prod_{s \in S} \varphi(p, I(s \in \chi)) d\omega \stackrel{\text{IH}}{\leq} \varphi(1, 1)^{|S'|} = \varphi(1, 1)^{|S|}$$

Induction step $Q(k+1) \rightarrow P(k+1)$:

Let $S \subseteq \{a_1, \dots, a_{k+1}\}$.

Let us again note that Ω_{k+1} is a compound distribution over Ω'_{k+1} . In general, for all predicates P :

$$\mathcal{P}_{\Omega_{k+1}}(P(\chi, p)) = \int_{\Omega'_{k+1}} I(|\chi(\omega)| < n) P(\chi(\omega), p(\omega)) + I(|\chi(\omega)| = n) \int_{\text{subsample}(\chi(\omega))} P(\tau, fp(\omega)) d\tau d\omega.$$

With this we can now verify the induction step:

$$\begin{aligned}
& \mathbb{E}_{\Omega_{k+1}} \left[\prod_{s \in S} \varphi(p, I(s \in \chi)) \right] \\
&= \int_{\Omega'_{k+1}} I(|\chi| < n) \prod_{s \in S} \varphi(p, I(s \in \chi)) d\omega \\
&+ \int_{\Omega'_{k+1}} I(|\chi| = n) \prod_{s \in S \setminus \chi(\omega)} \varphi(pf, 0) \int_{\text{subsample}(\chi)} \prod_{s \in S \cap \chi} \varphi(pf, I(s \in \tau)) d\tau d\omega \\
&\leq \int_{\Omega'_{k+1}} I(|\chi| < n) \prod_{s \in S} \varphi(p, I(s \in \chi)) d\omega && \text{Eq. 1} \\
&+ \int_{\Omega'_{k+1}} I(|\chi| = n) \prod_{s \in S \setminus \chi(\omega)} \varphi(pf, 0) \prod_{s \in S \cap \chi} \int_{\text{Ber}(f)} \varphi(pf, \tau) d\tau d\omega \\
&\leq \int_{\Omega'_{k+1}} I(|\chi| < n) \prod_{s \in S} \varphi(p, I(s \in \chi)) d\omega && \text{Cond 2} \\
&+ \int_{\Omega'_{k+1}} I(|\chi| = n) \prod_{s \in S \setminus \chi(\omega)} \varphi(p, 0) \prod_{s \in S \cap \chi} ((1-f)\varphi(pf, 0) + f\varphi(pf, 1)) d\omega \\
&\leq \int_{\Omega'_{k+1}} I(|\chi| < n) \prod_{s \in S} \varphi(p, I(s \in \chi)) d\omega && \text{Cond 1} \\
&+ \int_{\Omega'_{k+1}} I(|\chi| = n) \prod_{s \in S \setminus \chi(\omega)} \varphi(p, 0) \prod_{s \in S \cap \chi} \varphi(p, 1) d\omega \\
&= \int_{\Omega'_{k+1}} I(|\chi| < n) \prod_{s \in S} \varphi(p, I(s \in \chi)) d\omega \\
&+ \int_{\Omega'_{k+1}} I(|\chi| = n) \prod_{s \in S} \varphi(p, I(s \in \chi)) d\omega \\
&= \mathbb{E}_{\Omega'_{k+1}} \left[\prod_{s \in S} \varphi(p, I(s \in \chi)) \right] \leq \varphi(1, 1)^{|S|} && \text{IH}
\end{aligned}$$

□

A corollary and more practical version of the previous lemma is:

Lemma 2. Let $q \leq 1$ and $h : [0, q^{-1}] \rightarrow \mathbb{R}_{\geq 0}$ be concave then for all $k \in \{0, \dots, l\}$, $S \subseteq \{a_1, \dots, a_k\}$, $\Omega \in \{\Omega_k, \Omega'_k\}$:

$$\mathbb{E}_{\Omega} \left[\prod_{s \in S} I(p > q) h(p^{-1} I(s \in \chi)) \right] \leq h(1)^{|S|}$$

Proof. Follows from Lemma 1 for $\varphi(p, \tau) := I(p > q) h(\tau p^{-1})$. We just need to check Conditions 1 and 2. Indeed,

$$\begin{aligned}
(1 - \alpha)\varphi(x, 0) + \alpha\varphi(x, 1) &= (1 - \alpha)I(x > q)h(0) + \alpha I(x > q)h(x^{-1}) \\
&\leq I(x > q)h(\alpha x^{-1}) \leq I(x > q\alpha)h(\alpha x^{-1}) = \varphi(x/\alpha, 1)
\end{aligned}$$

for $0 < \alpha < 1$ and $0 < x \leq 1$, where we used that $x > q$ implies $x > q\alpha$; and

$$\varphi(x, 0) = I(x > q)h(0) \leq I(y > q)h(0) = \varphi(y, 0)$$

for $0 < x < y \leq 1$, where we used that $x > q$ implies $y > q$. □

It should be noted that this is a probabilistic recurrence relation, but the main innovation is that we establish a relation, with respect to general classes of functions of the state variables.

A.2 Concentration

Let us now see how we can obtain concentration bounds using Lemma 2, i.e., that the result of the algorithm is concentrated around the cardinality of $A = \{a_1, \dots, a_l\}$. This will be done using the Cramér–Chernoff method for the probability that the estimate is above $(1 + \varepsilon)|A|$ (resp. below $(1 - \varepsilon)|A|$) assuming p is not too small and a tail estimate for p being too small.

It should be noted that concentration is trivial, if $|A| < n$, i.e., if we never need to do sub-sampling, so we assume $|A| \geq n$.

Define $q := n/(4|A|)$ and notice that $q \leq \frac{1}{4}$.

Let us start with the upper tail bound:

Lemma 3. For any $\Omega \in \{\Omega_0, \dots, \Omega_l\} \cup \{\Omega'_1, \dots, \Omega'_l\}$ and $0 < \varepsilon \leq 1$:

$$L := \mathcal{P}_\Omega (p^{-1}|\chi| \geq (1 + \varepsilon)|A| \wedge p \geq q) \leq \exp\left(-\frac{n}{12}\varepsilon^2\right)$$

Proof. By assumption there exists a k such that $\Omega \in \{\Omega_k, \Omega'_k\}$. Let $A' = A \cap \{a_1, \dots, a_k\}$. Moreover, we define:

$$\begin{aligned} t &:= q \ln(1 + \varepsilon) \\ h(x) &:= 1 + qx(e^{t/q} - 1) \end{aligned}$$

To get a tail estimate, we use the Cramér–Chernoff method:

$$\begin{aligned} L &\stackrel{t>0}{\leq} \mathcal{P}_\Omega (\exp(tp^{-1}|\chi|) \geq \exp(t(1 + \varepsilon)|A|) \wedge p \geq q) \\ &\leq \mathcal{P}_\Omega (I(p \geq q) \exp(tp^{-1}|\chi|) \geq \exp(t(1 + \varepsilon)|A|)) \\ &\stackrel{\text{Markov}}{\leq} \exp(-t(1 + \varepsilon)|A|) \mathbb{E}_\Omega [I(p \geq q) \exp(tp^{-1}|\chi|)] \\ &\leq \exp(-t(1 + \varepsilon)|A|) \mathbb{E}_\Omega \left[\prod_{s \in A'} I(p \geq q) \exp(tp^{-1}I(s \in \chi)) \right] \\ &\leq \exp(-t(1 + \varepsilon)|A|) \mathbb{E}_\Omega \left[\prod_{s \in A'} I(p \geq q) h(p^{-1}I(s \in \chi)) \right] \\ &\stackrel{\text{Lé. 2}}{\leq} \exp(-t(1 + \varepsilon)|A|) h(1)^{|A'|} \\ &\stackrel{h(1) \geq 1}{\leq} (\exp(\ln(h(1)) - t(1 + \varepsilon)))^{|A|} \end{aligned}$$

So we just need to show that (using $|A| = \frac{n}{4q}$):

$$\ln(h(1)) - t(1 + \varepsilon) \leq \frac{-q\varepsilon^2}{3}$$

The latter can be established by analyzing the function

$$f(\varepsilon) := -\ln(1 + q\varepsilon) + q \ln(1 + \varepsilon)(1 + \varepsilon) - \frac{q\varepsilon^2}{3} = -\ln(h(1)) + t(1 + \varepsilon) - \frac{q\varepsilon^2}{3}.$$

For which it is easy to check $f(0) = 0$ and the derivative with respect to ε is non-negative in the range $0 \leq q \leq 1/4$ and $0 < \varepsilon \leq 1$, i.e., $f(\varepsilon) \geq 0$. \square

Using the previous result we can also estimate bounds for p becoming too small:

Lemma 4.

$$\mathcal{P}_{\Omega_l}(p < q) \leq l \exp\left(-\frac{n}{12}\right)$$

Proof. We will use a similar strategy as in the Bad_2 bound from the original CVM paper [1]. Let j be maximal, s.t., $q \leq f^j$. Hence $f^{j+1} < q$ and:

$$f^j \leq 2ff^j < 2q = \frac{n}{2|A|}. \quad (2)$$

First, we bound the probability of jumping from $p = f^j$ to $p = f^{j+1}$ at a specific point in the algorithm, e.g., while processing k stream elements. It can only happen if $|\chi| = n$, $p = f^j$ in Ω'_k . Then

$$\begin{aligned} \mathcal{P}_{\Omega'_k}(|\chi| \geq n \wedge p = f^j) &\leq \mathcal{P}(p^{-1}|\chi| \geq f^{-j}n \wedge p \geq q) \\ &\stackrel{\text{Eq. 2}}{\leq} \mathcal{P}(p^{-1}|\chi| \geq 2|A| \wedge p \geq q) \\ &\stackrel{\text{Le. 3}}{\leq} \exp(-n/12) \end{aligned}$$

The probability that this happens ever in the entire process is then at most l times the above which proves the lemma. \square

Lemma 5. Let $0 < \varepsilon < 1$ then:

$$L := \mathcal{P}_{\Omega_l}(p^{-1}|\chi| \leq (1 - \varepsilon)|A| \wedge p \geq q) \leq \exp\left(-\frac{n}{8}\varepsilon^2\right)$$

Proof. Let us define

$$\begin{aligned} t &:= q \ln(1 - \varepsilon) < 0 \\ h(x) &:= 1 + qx(e^{t/q} - 1) \end{aligned}$$

Note that $h(x) \geq 0$ for $0 \leq x \leq q^{-1}$ (can be checked by verifying it is true for $h(0)$ and $h(q^{-1})$ and the fact that the function is affine.)

With these definitions we again follow the Cramér–Chernoff method:

$$\begin{aligned} L &\stackrel{t < 0}{=} \mathcal{P}_{\Omega_l}(\exp(tp^{-1}|\chi|) \geq \exp(t(1 - \varepsilon)|A|) \wedge p \geq q) \\ &\leq \mathcal{P}_{\Omega_l}(I(p \geq q) \exp(tp^{-1}|\chi|) \geq \exp(t(1 - \varepsilon)|A|) \wedge p > q) \\ &\stackrel{\text{Markov}}{\leq} \exp(-t(1 - \varepsilon)|A|) \mathbb{E}_{\Omega} [I(p \geq q) \exp(tp^{-1}|\chi|)] \\ &= \exp(-t(1 - \varepsilon)|A|) \mathbb{E}_{\Omega} \left[\prod_{s \in A} I(p \geq q) \exp(tp^{-1}I(s \in \chi)) \right] \\ &\leq \exp(-t(1 - \varepsilon)|A|) \mathbb{E}_{\Omega} \left[\prod_{s \in A} I(p \geq q) h(p^{-1}I(s \in \chi)) \right] \\ &\stackrel{\text{Le. 2}}{\leq} \exp(-t(1 - \varepsilon)|A|) (h(1))^{|A|} \\ &= \exp(\ln(h(1)) - t(1 - \varepsilon))^{|A|} \end{aligned}$$

Substituting t and h and using $|A| = \frac{n}{4q}$, we can see that the lemma is true if

$$f(\varepsilon) := q \ln(1 - \varepsilon)(1 - \varepsilon) - \ln(1 - q\varepsilon) - \frac{q}{2}\varepsilon^2 = t(1 - \varepsilon) - \ln(h(1)) - \frac{q}{2}\varepsilon^2$$

is non-negative for $0 \leq q \leq \frac{1}{4}$ and $0 < \varepsilon < 1$. This can be verified by checking that $f(0) = 0$ and that the derivative with respect to ε is non-negative. \square

We can now establish the concentration result:

Theorem 1. *Let $0 < \varepsilon < 1$ and $0 < \delta < 1$ and $n \geq \frac{12}{\varepsilon^2} \ln\left(\frac{3l}{\delta}\right)$ then:*

$$L = \mathcal{P}_{\Omega_l}(|p^{-1}|\chi| - |A| \geq \varepsilon|A|) \leq \delta$$

Proof. Note that the theorem is trivial if $|A| < n$. If not:

$$\begin{aligned} L &\leq \mathcal{P}_{\Omega_l}(|p^{-1}|\chi| \leq (1 - \varepsilon)|A| \wedge p \geq q) \\ &\quad + \mathcal{P}_{\Omega_l}(|p^{-1}|\chi| \geq (1 + \varepsilon)|A| \wedge p \geq q) + \mathcal{P}_{\Omega_l}(p < q) \\ &\stackrel{\text{L.e. 3-5}}{\leq} \exp\left(-\frac{n}{8}\varepsilon^2\right) + \exp\left(-\frac{n}{12}\varepsilon^2\right) + l \exp\left(-\frac{n}{12}\right) \\ &\leq \frac{\delta}{3} + \frac{\delta}{3} + \frac{\delta}{3} \end{aligned}$$

\square

A.3 Unbiasedness

Let M be large enough such that $p^{-1} \leq M$ a.s. (e.g., we can choose $M = f^{-l}$). Then we can derive from Lemma 2 using $h(x) = x$ and $h(x) = M + 1 - x$ that for all $s \in A$:

$$\begin{aligned} \mathbb{E}_{\Omega_l}[p^{-1}I(s \in \chi)] &= \mathbb{E}_{\Omega_l}[I(p \geq M^{-1})p^{-1}I(s \in \chi)] \leq 1 \\ \mathbb{E}_{\Omega_l}[M + 1 - p^{-1}I(s \in \chi)] &= \mathbb{E}_{\Omega_l}[I(p \geq M^{-1})(M + 1 - p^{-1}I(s \in \chi))] \leq M \end{aligned}$$

which implies $\mathbb{E}_{\Omega_l}[p^{-1}I(s \in \chi)] = 1$. By linearity of expectation we conclude

$$\mathbb{E}_{\Omega_l}[p^{-1}|\chi|] = \sum_{s \in A} \mathbb{E}_{\Omega_l}[p^{-1}I(s \in \chi)] = |A|.$$