Formalizing concentration inequalities in Rocq: infrastructure and automation

Reynald Affeldt [1] Alessandro Bruni [2] Cyril Cohen [3]
Pierre Roux [4] Takafumi Saikawa [5]

[1]AIST Japan [2]ITU Copenhagen [3]INRIA Lyon [4]ONERA/DTIS Toulouse [5]Nagoya University

ITP, 28 September 2025

A Political Statement

A Political Statement

- Our original motivation is to certify elections

A Political Statement

- Our original motivation is to certify elections
- Risk-limiting audits are statistical procedures that give high confidence in elections
  - with paper trail (i.e., physical ballots or EVMs with voter-verified paper audits)
  - and software independence (i.e., we do not trust the software used to count the votes)

A Political Statement

- Our original motivation is to certify elections
- Risk-limiting audits are statistical procedures that give high confidence in elections
  - with paper trail (i.e., physical ballots or EVMs with voter-verified paper audits)
  - and software independence (i.e., we do not trust the software used to count the votes)
- The statistical procedure itself now becomes the new trusted computing base

A Political Statement

- Our original motivation is to certify elections
- Risk-limiting audits are statistical procedures that give high confidence in elections
  - with paper trail (i.e., physical ballots or EVMs with voter-verified paper audits)
  - and software independence (i.e., we do not trust the software used to count the votes)
- The statistical procedure itself now becomes the new trusted computing base
- Their correctness is essentially a concentration inequality, i.e., of the form:

$$\Pr[\mathbf{X} \geq t] \leq b$$

A Political Statement

- Our original motivation is to certify elections
- Risk-limiting audits are statistical procedures that give high confidence in elections
  - with paper trail (i.e., physical ballots or EVMs with voter-verified paper audits)
  - and software independence (i.e., we do not trust the software used to count the votes)
- The statistical procedure itself now becomes the new trusted computing base
- Their correctness is essentially a concentration inequality, i.e., of the form:

$$\Pr[\mathbf{X} \geq t] \leq b$$

- Hence the first part of our title:

"Formalizing concentration inequalities in Rocq"

Contributions

- Infrastructure and Automation:

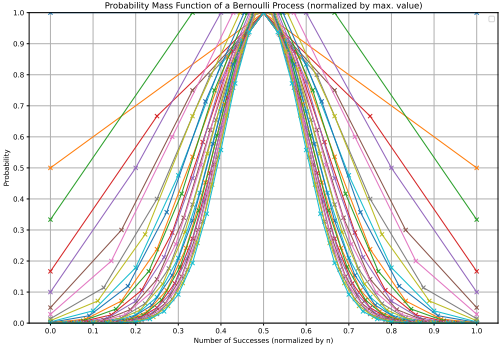
  We extend MATHCOMP and MATHCOMP-ANALYSIS so that we can:
  - automatically discharge range checks using lightweight abstract interpretation
  - reduce assumptions in probability proofs using $\mathcal{L}^p$ spaces
  - introduce seminorms in the structure hierarchy while maintaining compatibility
- We formalize concentration inequalities:
  - a library of general lemmas (Markov, Chebyshev, Chernoff, Cantelli)
  - a showcase with Bernoulli sampling

Motivating example: Bernoulli sampling

Theorem [Mitzenmacher and Upfal, 2005, Exercise 4.5]

Given independent Boolean random variables X_i with $\Pr[X_i = 1] = p$, confidence $0 < \delta \leqslant 1$, and accuracy $0 < \theta < p$,

$$n \geqslant \frac{3}{\theta^2} \ln\left(\frac{2}{\delta}\right) \implies \Pr\left[\left| \frac{\sum_{i=1}^{n} X_i}{n} - p \right| \leqslant \theta\right] \geqslant 1 - \delta.$$



Outline

Background

Changing the hierarchy to accomodate seminorms

Automatic handling of interval types

Improve handling of hypotheses in probability theory

Prove concentration inequalities

Putting it all together

Conclusion

Outline

Background

Changing the hierarchy to accomodate seminorms

Automatic handling of interval types

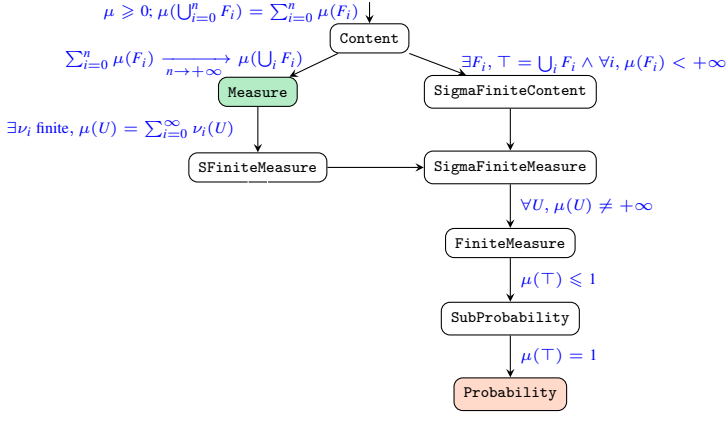
Improve handling of hypotheses in probability theory

Prove concentration inequalities

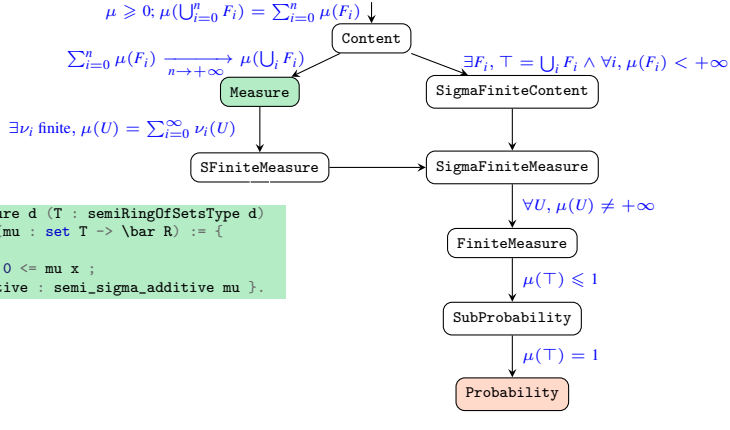
Putting it all together

Conclusion

Background: Measure theory in MATHCOMP-ANALYSIS

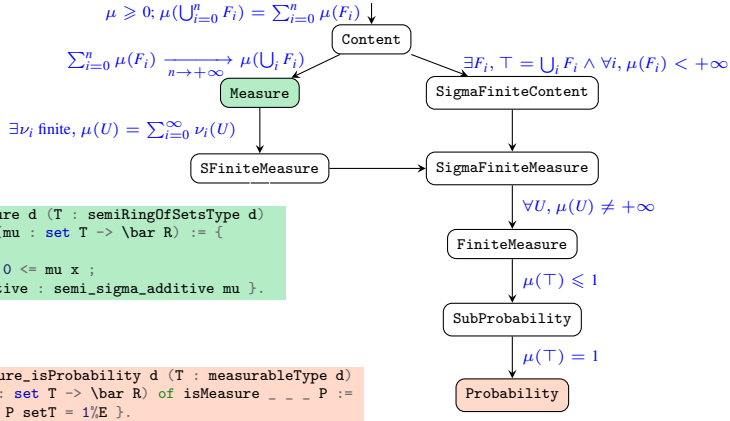


Background: Measure theory in MATHCOMP-ANALYSIS




```
HB.factory Record isMeasure d (T : semiRingOfSetsType d)
    (R : realFieldType) (mu : set T -> \bar R) := {
 measure0 : mu set0 = 0 ;
 measure_ge0 : forall x, 0 <= mu x ;
 measure_semi_sigma_additive : semi_sigma_additive mu }.
```

Background: Measure theory in MATHCOMP-ANALYSIS




```
HB.factory Record isMeasure d (T : semiRingOfSetsType d)
    (R : realFieldType) (mu : set T -> \bar R) := {
 measure0 : mu set0 = 0 ;
 measure_ge0 : forall x, 0 <= mu x ;
 measure_semi_sigma_additive : semi_sigma_additive mu }.
```

```
HB.factory Record Measure_isProbability d (T : measurableType d)
    (R : realType) (P : set T -> \bar R) of isMeasure _ _ _ P :=
  { probability_setT : P setT = 1%E }.
```

Random variables and expectations

Let T be a measurable space and P a probability measure on T:

```
Context d (T : measurableType d) (R : realType) (P : probability T R).
```

Definition (Random variables)

Random variables are measurable functions $X : T \to \mathbb{R}$ with an "ambient distribution" P:

```
Definition random_variable := {mfun T >-> R}.
Notation "{ 'RV' P >-> R }" := (@random_variable _ _ R P).
```

Definition (Expectation)

The *expectation* of a random variable $X : T \to \mathbb{R}$ is the integral of X with measure P:

```
Definition expectation (X : {RV P >-> R}) := \int[P]_w (X w)%:E.
Notation "'E_ P [ X ]" := (@expectation _ _ _ P X).
```

Outline

Background

Changing the hierarchy to accomodate seminorms

Automatic handling of interval types

Improve handling of hypotheses in probability theory

Prove concentration inequalities

Putting it all together

Conclusion

A p-norm is not a norm

$$N_p[f] \stackrel{\text{def}}{=} \begin{cases} \left(\int_\mu |f|^p\right)^{\frac{1}{p}} & 0 < p < \infty \\ \operatorname{esssup}_\mu |f| & p = \infty, \mu(T) > 0 \\ 0 & p = \infty, \mu(T) = 0 \end{cases}$$

```
Definition Lnorm (p : \bar R) (f : T -> \bar R) :=
match p with
| p%:E => (\int[P]_x `|f x| `^ p) `^ p^-1
| +oo => if P [set: T] > 0 then ess_sup P (abse \o f) else 0
| -oo => if P [set: T] > 0 then ess_inf P (abse \o f) else 0
end.
Notation "'N[ P ]_ p [ f ]" := (Lnorm P p f).
```

A p-norm is not a norm

$$N_p[f] \stackrel{\text{def}}{=} \begin{cases} \left(\int_\mu |f|^p\right)^{\frac{1}{p}} & 0 < p < \infty \\ \operatorname{esssup}_\mu |f| & p = \infty, \mu(T) > 0 \\ 0 & p = \infty, \mu(T) = 0 \end{cases}$$

```
Definition Lnorm (p : \bar R) (f : T -> \bar R) :=
match p with
| p%:E => (\int[P]_x `|f x| `^ p) `^ p^-1
| +oo => if P [set: T] > 0 then ess_sup P (abse \o f) else 0
| -oo => if P [set: T] > 0 then ess_inf P (abse \o f) else 0
end.
Notation "'N[ P ]_ p [ f ]" := (Lnorm P p f).
```

Normed Zmodule:

- $\|\cdot\| : T \to \mathbb{R}$
- $\|x + y\| \leq \|x\| + \|y\|$
- $\|n \cdot x\| = n \cdot \|x\|$
- $\|-x\| = \|x\|$
- $\|x\| = 0 \to x = 0$

```
HB.mixin Record Zmodule_isNormed
  (R : POrderedZmodule.type)
  M of GRing.Zmodule M := {
 norm : M -> R;
 ler_normD :
  forall x y, norm (x + y) <= norm x + norm y;
 normrMn : forall x n, norm (x *+ n) = norm x *+ n;
 normrN : forall x, norm (- x) = norm x;
 normr0_eq0 : forall x, norm x = 0 -> x = 0 }.
```

A p-norm is not a norm

$$N_p[f] \stackrel{\text{def}}{=} \begin{cases} \left(\int_\mu |f|^p\right)^{\frac{1}{p}} & 0 < p < \infty \\ \operatorname{esssup}_\mu |f| & p = \infty, \mu(T) > 0 \\ 0 & p = \infty, \mu(T) = 0 \end{cases}$$

```
Definition Lnorm (p : \bar R) (f : T -> \bar R) :=
match p with
| p%:E => (\int[P]_x `|f x| `^ p) `^ p^-1
| +oo => if P [set: T] > 0 then ess_sup P (abse \o f) else 0
| -oo => if P [set: T] > 0 then ess_inf P (abse \o f) else 0
end.
Notation "'N[ P ]_ p [ f ]" := (Lnorm P p f).
```

Normed Zmodule:

- $\|\cdot\| : T \to \mathbb{R}$
- $\|x + y\| \leq \|x\| + \|y\|$
- $\|n \cdot x\| = n \cdot \|x\|$
- $\|-x\| = \|x\|$
- $\|x\| = 0 \to x = 0$

```
HB.mixin Record Zmodule_isNormed
  (R : POrderedZmodule.type)
  M of GRing.Zmodule M := {
 norm : M -> R;
 ler_normD :
  forall x y, norm (x + y) <= norm x + norm y;
 normrMn : forall x n, norm (x *+ n) = norm x *+ n;
 normrN : forall x, norm (- x) = norm x;
 normr0_eq0 : forall x, norm x = 0 -> x = 0 }.
```

$N_p[\cdot]$ does not satisfy the last law if there exists a nonempty μ-null set

Backwards compatible hierarchy refinement

Normed Zmodule:

- $\|\cdot\| : T \to \mathbb{R}$
- $\|x + y\| \leq \|x\| + \|y\|$
- $\|n \cdot x\| = n \cdot \|x\|$
- $\|-x\| = \|x\|$
- $\|x\| = 0 \to x = 0$

```
HB.mixin Record Zmodule_isNormed
  (R : POrderedZmodule.type)
 M of GRing.Zmodule M := {
norm : M -> R;
ler_normD :
 forall x y, norm (x + y) <= norm x + norm y;
normrMn : forall x n, norm (x *+ n) = norm x *+ n;
normrN : forall x, norm (- x) = norm x;
normr0_eq0 : forall x, norm x = 0 -> x = 0 }.
```

Backwards compatible hierarchy refinement

Seminormed Zmodule:

- $\|\cdot\| : T \to \mathbb{R}$
- $\|x + y\| \leq \|x\| + \|y\|$
- $\|n \cdot x\| = n \cdot \|x\|$
- $\|-x\| = \|x\|$

```
HB.mixin Record Zmodule_isSemiNormed
  (R : POrderedZmodule.type)
  M of GRing.Zmodule M := {
 norm : M -> R;
 ler_normD :
  forall x y, norm (x + y) <= norm x + norm y;
 normrMn : forall x n, norm (x *+ n) = norm x *+ n;
 normrN : forall x, norm (- x) = norm x }.
```

Normed Zmodule:

- $\|\cdot\| : T \to \mathbb{R}$
- $\|x + y\| \leq \|x\| + \|y\|$
- $\|n \cdot x\| = n \cdot \|x\|$
- $\|-x\| = \|x\|$
- $\|x\| = 0 \to x = 0$

```
HB.mixin Record Zmodule_isNormed
  (R : POrderedZmodule.type)
  M of GRing.Zmodule M := {
 norm : M -> R;
 ler_normD :
  forall x y, norm (x + y) <= norm x + norm y;
 normrMn : forall x n, norm (x *+ n) = norm x *+ n;
 normrN : forall x, norm (- x) = norm x;
 normr0_eq0 : forall x, norm x = 0 -> x = 0 }.
```

Backwards compatible hierarchy refinement

Seminormed Zmodule:

- $\|\cdot\| : T \to \mathbb{R}$
- $\|x+y\| \leq \|x\| + \|y\|$
- $\|n \cdot x\| = n \cdot \|x\|$
- $\|-x\| = \|x\|$

```
HB.mixin Record Zmodule_isSemiNormed
  (R : POrderedZmodule.type)
  M of GRing.Zmodule M := {
 norm : M -> R;
 ler_normD :
  forall x y, norm (x + y) <= norm x + norm y;
 normrMn : forall x n, norm (x *+ n) = norm x *+ n;
 normrN : forall x, norm (- x) = norm x }.
```

Normed Zmodule:

- $\|\cdot\| : T \to \mathbb{R}$
- $\|x+y\| \leq \|x\| + \|y\|$
- $\|n \cdot x\| = n \cdot \|x\|$
- $\|-x\| = \|x\|$
- $\|x\| = 0 \to x = 0$

```
HB.mixin Record Zmodule_isNormed
  (R : POrderedZmodule.type)
  M of GRing.Zmodule M := {
 norm : M -> R;
 ler_normD :
  forall x y, norm (x + y) <= norm x + norm y;
 normrMn : forall x n, norm (x *+ n) = norm x *+ n;
 normrN : forall x, norm (- x) = norm x;
 normr0_eq0 : forall x, norm x = 0 -> x = 0 }.
```

Positive Definite:

- $\|x\| = 0 \to x = 0$

```
HB.mixin Record SemiNormedZmodule_isPositiveDefinite
  (R : POrderedZmodule.type)
  M of @SemiNormedZmodule R M := {
 normr0_eq0 : forall x : M, norm x = 0 -> x = 0 }.
```

Backwards compatible hierarchy refinement

Seminormed Zmodule:

- $\|\cdot\| : T \to \mathbb{R}$
- $\|x+y\| \leq \|x\| + \|y\|$
- $\|n \cdot x\| = n \cdot \|x\|$
- $\|-x\| = \|x\|$

```
HB.mixin Record Zmodule_isSemiNormed
  (R : POrderedZmodule.type)
  M of GRing.Zmodule M := {
 norm : M -> R;
 ler_normD :
  forall x y, norm (x + y) <= norm x + norm y;
 normrMn : forall x n, norm (x *+ n) = norm x *+ n;
 normrN : forall x, norm (- x) = norm x }.
```

Normed Zmodule:

- $\|\cdot\| : T \to \mathbb{R}$
- $\|x+y\| \leq \|x\| + \|y\|$
- $\|n \cdot x\| = n \cdot \|x\|$
- $\|-x\| = \|x\|$
- $\|x\| = 0 \to x = 0$

```
HB.mixin Record Zmodule_isNormed
  (R : POrderedZmodule.type)
  M of GRing.Zmodule M := {
 norm : M -> R;
 ler_normD :
  forall x y, norm (x + y) <= norm x + norm y;
 normrMn : forall x n, norm (x *+ n) = norm x *+ n;
 normrN : forall x, norm (- x) = norm x;
 normr0_eq0 : forall x, norm x = 0 -> x = 0 }.
```

Positive Definite:

- $\|x\| = 0 \to x = 0$

```
HB.mixin Record SemiNormedZmodule_isPositiveDefinite
  (R : POrderedZmodule.type)
  M of @SemiNormedZmodule R M := {
 normr0_eq0 : forall x : M, norm x = 0 -> x = 0 }.
```

$\mathcal{L}^p$ spaces and L^p spaces

$\mathcal{L}^p$ spaces

An $\mathcal{L}^p$ set ($1 \leqslant p \leqslant \infty$) is the set of measurable functions $f : T \to \mathbb{R}$ with $N_p[f] < \infty$:

```
HB.mixin Record isLfunction (p : \bar R) (p1 : 1 <= p)
    (f : T -> R) of @MeasurableFun d _ T R f
  := { Lfunction_finite : 'N[ P ]_p [ EFin \o f ] < +oo }.

#[short(type=LfunType)]
HB.structure Definition Lfunction (p : \bar R) (p1 : 1 <= p) :=
  {f of @MeasurableFun d _ T R f & @isLfunction d T R P p p1 f}.
```

L^p spaces

An L^p space is the set of equivalence classes of $\mathcal{L}^p$ functions modulo $f \sim g \stackrel{\text{def}}{=} f \stackrel{\text{a.e.}}{=} g$:

```
Definition Lequiv (f g : LfunType mu p1) := `[< f = g %[ae mu] >].

Definition LspaceType := {eq_quot Lequiv}.
```

Minkowski's Inequality

The main lemma to show that $N_p[\cdot]$ is a seminorm is **Minkowski's inequality**.

Minkowski's inequality

```
Lemma eminkowski f g (p : \bar R) :
    measurable_fun [set: T] f -> measurable_fun [set: T] g -> 1 <= p ->
  'N_p[f + g] <= 'N_p[f] + 'N_p[g].
```

This establishes the triangle inequality for $N_p[\cdot]$ required by the seminorm interface.

Minkowski's Inequality

The main lemma to show that $N_p[\cdot]$ is a seminorm is **Minkowski's inequality**.

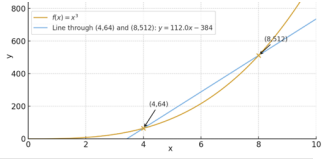
Minkowski's inequality

```
Lemma eminkowski f g (p : \bar R) :
    measurable_fun [set: T] f -> measurable_fun [set: T] g -> 1 <= p ->
  'N_p[f + g] <= 'N_p[f] + 'N_p[g].
```

This establishes the triangle inequality for $N_p[\cdot]$ required by the seminorm interface.

Convexity

To prove Minkowski we need convexity of the power function:




```
Definition convex_function (D : set R) (f : R -> R) :=
  forall t : {i01 R}, {in D &, forall x y, f (x <| t |> y) <= f x <| t |> f y}.

Lemma convex_powR p : 1 <= p -> convex_function `[0, +oo[ (fun x => powR x p).
```

Outline

Background

Changing the hierarchy to accomodate seminorms

Automatic handling of interval types

Improve handling of hypotheses in probability theory

Prove concentration inequalities

Putting it all together

Conclusion

Interval Inference

Many sidegoals about sign of simple expressions, e.g.:

`0 <= k%:R / 2 ^+ n`	$0 \leqslant \frac{k}{2^n}$
`0 < (2 ^ n)%:R`	$0 < 2^n$
`0 < 1 + `|r|`	$0 < 1 + \|r\|$
`0 < (3 + d.*2).*2.+2%:R * (3 + d.*2).*2.+1%:R`	$0 < (2(3+2d)+2)(2(3+2d)+1)$
`0 < 1 / (2 ^ i.+1)%:R`	$0 < 1/2^{i+1}$
`0 < n.+1%:R^-1 / 2`	$0 < (n+1)^{-1}/2$
`0 < (2 / e%:num) ^+ 2` with `e` non negative	$0 < \left(\frac{2}{e}\right)^2$ with $0 \leqslant e$

Interval Inference

Many sidegoals about sign of simple expressions, e.g.:

`0 <= k%:R / 2 ^+ n`	$0 \leqslant \frac{k}{2^n}$
`0 < (2 ^ n)%:R`	$0 < 2^n$
`0 < 1 + `|r|` `	$0 < 1 + \|r\|$
`0 < (3 + d.*2).*2.+2%:R * (3 + d.*2).*2.+1%:R`	$0 < (2(3+2d)+2)(2(3+2d)+1)$
`0 < 1 / (2 ^ i.+1)%:R`	$0 < 1/2^{i+1}$
`0 < n.+1%:R^-1 / 2`	$0 < (n+1)^{-1}/2$
`0 < (2 / e%:num) ^+ 2` with `e` non negative	$0 < \left(\frac{2}{e}\right)^2$ with $0 \leqslant e$

We'd like them automatically discharged ⇒ light interval inference

Interval Inference (implementation ideas)

A type of values in some interval:

```
Structure def T (I : interval int) :=
    Def { r : T; P : r \in I }.
```

Interval Inference (implementation ideas)

A type of values in some interval:

```
Structure def T (I : interval int) :=
    Def { r : T; P : r \in I }.
```

Two notations `%:num` from `def` to `T` (above projection `r`)
and `%:itv` to infer interval, e.g.:

```
Check n.+1%:itv.
    : def nat `[1, +oo[
Check (1 / 2 ^ n.+1)%:itv.
    : def R `]0, +oo[
Check fun (R : realDomainType) (x : {posnum R}) => (x%:num + 1)%:itv.
    : forall R : realDomainType, {posnum R} -> def R `]1, +oo[
```

Interval Inference (conclusion)

- automatically called 1604 times in the Analysis library
  (on subgoals of shape `0 <= _` and `0 < _`)
  solves 729 subgoals
- implemented via canonical structures (kind of typeclasses)
  one instance per operator, lightweight, no global reification

Interval Inference (conclusion)

- automatically called 1604 times in the Analysis library
  (on subgoals of shape `0 <= _` and `0 < _`)
  solves 729 subgoals
- implemented via canonical structures (kind of typeclasses)
  one instance per operator, lightweight, no global reification
- abstract interpretation framework: only overapproximations
  - start with very coarse instances (a.k.a. $\top$ or $(-\infty, +\infty)$)
  - refine on demand

Interval Inference (conclusion)

- automatically called 1604 times in the Analysis library
  (on subgoals of shape `0 <= _` and `0 < _`)
  solves 729 subgoals
- implemented via canonical structures (kind of typeclasses)
  one instance per operator, lightweight, no global reification
- abstract interpretation framework: only overapproximations
  - start with very coarse instances (a.k.a. $\top$ or $(-\infty, +\infty)$)
  - refine on demand
- available in MATHCOMP

Outline

Background

Changing the hierarchy to accomodate seminorms

Automatic handling of interval types

Improve handling of hypotheses in probability theory

Prove concentration inequalities

Putting it all together

Conclusion

Inclusion of $\mathcal{L}^p$ spaces

Consequence of Hölder's inequality: $\mathcal{L}^q \subseteq \mathcal{L}^p$ when $p \leqslant q$

Inclusion of $\mathcal{L}^p$ spaces

Consequence of Hölder's inequality: $\mathcal{L}^q \subseteq \mathcal{L}^p$ when $p \leqslant q$
This allows to reduce integrability assumptions, e.g.:

```
Lemma covarianceDl (X Y Z : {RV P >-> R}) :
    P.-integrable setT (EFin \o X) -> P.-integrable setT (EFin \o (X ^+ 2)) ->
    P.-integrable setT (EFin \o Y) -> P.-integrable setT (EFin \o (Y ^+ 2)) ->
    P.-integrable setT (EFin \o Z) -> P.-integrable setT (EFin \o (Z ^+ 2)) ->
    P.-integrable setT (EFin \o (X * Z)) ->
    P.-integrable setT (EFin \o (Y * Z)) ->
  covariance P (X \+ Y) Z = covariance P X Z + covariance P Y Z.
```

Inclusion of $\mathcal{L}^p$ spaces

Consequence of Hölder's inequality: $\mathcal{L}^q \subseteq \mathcal{L}^p$ when $p \leqslant q$
This allows to reduce integrability assumptions, e.g.:

```
Lemma covarianceDl (X Y Z : {RV P >-> R}) :
    P.-integrable setT (EFin \o X) -> P.-integrable setT (EFin \o (X ^+ 2)) ->
    P.-integrable setT (EFin \o Y) -> P.-integrable setT (EFin \o (Y ^+ 2)) ->
    P.-integrable setT (EFin \o Z) -> P.-integrable setT (EFin \o (Z ^+ 2)) ->
    P.-integrable setT (EFin \o (X * Z)) ->
    P.-integrable setT (EFin \o (Y * Z)) ->
  covariance P (X \+ Y) Z = covariance P X Z + covariance P Y Z.
```

becomes:

```
Lemma covarianceDl (X Y Z : T -> R) :
    X \in Lfun P 2 -> Y \in Lfun P 2 -> Z \in Lfun P 2 ->
  covariance P (X \+ Y) Z = covariance P X Z + covariance P Y Z.
```

Inclusion of $\mathcal{L}^p$ spaces

Consequence of Hölder's inequality: $\mathcal{L}^q \subseteq \mathcal{L}^p$ when $p \leqslant q$
This allows to reduce integrability assumptions, e.g.:

```
Lemma covarianceDl (X Y Z : {RV P >-> R}) :
    P.-integrable setT (EFin \o X) -> P.-integrable setT (EFin \o (X ^+ 2)) ->
    P.-integrable setT (EFin \o Y) -> P.-integrable setT (EFin \o (Y ^+ 2)) ->
    P.-integrable setT (EFin \o Z) -> P.-integrable setT (EFin \o (Z ^+ 2)) ->
    P.-integrable setT (EFin \o (X * Z)) ->
    P.-integrable setT (EFin \o (Y * Z)) ->
  covariance P (X \+ Y) Z = covariance P X Z + covariance P Y Z.
```

becomes:

```
Lemma covarianceDl (X Y Z : T -> R) :
    X \in Lfun P 2 -> Y \in Lfun P 2 -> Z \in Lfun P 2 ->
  covariance P (X \+ Y) Z = covariance P X Z + covariance P Y Z.
```

Similarly, we simplify our concentration inequalities:

```
Lemma cantelli (X : {RV P >-> R}) (lambda : R) :
  X \in Lfun P 2 -> 0 < lambda ->
 P [set x | lambda%:E <= (X x)%:E - 'E_P[X]] <= 'V_P[X] / ('V_P[X] + (lambda^2)%:E).
```

Outline

Background

Changing the hierarchy to accomodate seminorms

Automatic handling of interval types

Improve handling of hypotheses in probability theory

Prove concentration inequalities

Putting it all together

Conclusion

Standard concentration inequalities

We prove standard concentration inequalities, including:

- Markov's inequality

```
Lemma markov (X : {RV P >-> R}) (f : R -> R) (eps : R) : 0 < eps ->
    measurable_fun [set: R] f ->                  (* measurable *)
    (forall r, 0 <= r -> 0 <= f r) ->             (* non-negative *)
    {in Num.nneg &, {homo f : x y / x <= y}} -> (* non-decreasing *)
  (f eps)%:E * P [set x | eps <= `| X x | ] <=
    'E_P[f \o (fun x => `| x |) \o X].
```

- Chernoff's inequality

```
Lemma chernoff (X : {RV P >-> R}) (r a : R) : 0 < r ->
  P [set x | X x >= a] <= 'M_P X r * (expR (- (r * a)))%:E.
```

- Chebyshev's inequality

```
Lemma chebyshev (X : {RV P >-> R}) (eps : R) : 0 < eps ->
  P [set x | eps <= `| X x - fine ('E_P[X])| ] <= (eps^-2)%:E * 'V_P[X].
```

Outline

Background

Changing the hierarchy to accomodate seminorms

Automatic handling of interval types

Improve handling of hypotheses in probability theory

Prove concentration inequalities

Putting it all together

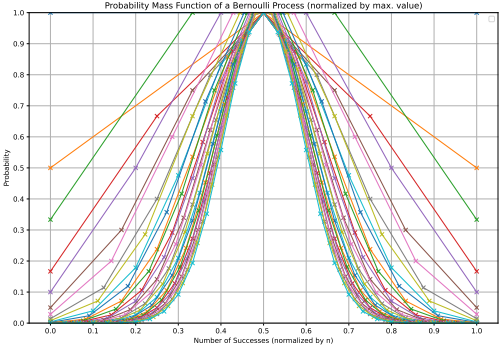
Conclusion

Sampling Theorem: Statement

Theorem [Mitzenmacher and Upfal, 2005, Exercise 4.5]

Given independent 0-1 random variables X_i with $Pr[X_i = 1] = p$, confidence $0 < \delta \leqslant 1$, and accuracy $0 < \theta < p$,

$$n \geqslant \frac{3}{\theta^2} \ln\left(\frac{2}{\delta}\right) \implies \Pr\left[\left| \frac{\sum_{i=1}^{n} X_i}{n} - p \right| \leqslant \theta\right] \geqslant 1 - \delta.$$



Formalization: Bernoulli Trials

Bernoulli random variable

A Bernoulli random variable is a measurable function X with distribution Bernoulli(p):

```
HB.mixin Record RV_isBernoulli ... :=
  { bernoulliP : distribution P X = bernoulli p }.

#[short(type=bernoulliRV)]
HB.structure Definition BernoulliRV ... := {X of @RV_isBernoulli ... X}.
```

Bernoulli trial

A Bernoulli trial is a sequence of n independent Bernoulli random variables. Its value is:

```
Definition trial_value n
    (X : n.-tuple {RV P >-> _}) : {RV (\X_n P) >-> R} :=
  \sum_(i < n) Tnth X i.

(* cast to real numbers *)
Definition real_of_bool n : _ -> n.-tuple _ :=
  map_tuple (bool_to_real R : bernoulliRV P p -> {mfun _ >-> _}).

Definition bool_trial_value n := @trial_value n \o @real_of_bool n.
```

Key Lemmas for the Proof

- **Moment generating function bound:**

```
Lemma mmt_gen_fun_expectation n (X_ : n.-tuple (bernoulliRV P p)) (t : R) :
  0 <= t ->
  let X = bool_trial_value X_ in
  'M_X t <= (expR (fine 'E_(\X_n P)[X] * (expR t - 1)))%:E.
```

- **Product of expectations:**

```
Lemma expectation_product n (X : n.-tuple {RV P >-> R}) :
  [set` X] `<=` lfun P 1 ->
  'E_(\X_n P)[ \prod_(i < n) Tnth X i ] = \prod_(i < n) 'E_P[ tnth X i ].
```

- **. . . and more Chernoff-type inequalities:**

$$P\left[X \geq (1+\delta)\mathbb{E}[X]\right] \leqslant e^{-\frac{\mathbb{E}[X]\delta^2}{3}}$$

$$P\left[X \leq (1-\delta)\mathbb{E}[X]\right] \leqslant e^{-\frac{\mathbb{E}[X]\delta^2}{2}}$$

$$P\left[|X - \mathbb{E}[X]| \geqslant \delta\mathbb{E}[X]\right] \leqslant 2e^{-\frac{\mathbb{E}[X]\delta^2}{3}}$$

Sampling Theorem in Rocq

```
Theorem sampling n (X : n.-tuple (bernoulliRV P p)) (theta delta : R) :
  let X' x := (bool_trial_value X x) / n%:R in
  0 < delta <= 1 ->
  0 < theta < p ->
  3 / theta ^+ 2 * ln (2 / delta) <= n%:R ->
  (\X_n P) [set i | `| X' i - p | <= theta] >= 1 - delta%:E.
```

- Proof applies the above inequalities and symbolic manipulations.
- Analytical subgoals (e.g., bounds on $x \ln(x)$) are handled using MathComp-Analysis and CoqInterval.

Proofs of numerical statements, in practice

Everybody knows that $1 + 1 = 2 \ldots$

Proofs of numerical statements, in practice

Everybody knows that $1 + 1 = 2$ . . . it only took Russell and Whitehead 360 pages . . .

Proofs of numerical statements, in practice

Everybody knows that $1 + 1 = 2$ . . . it only took Russell and Whitehead 360 pages . . .

But how long does it take to prove that $e^2 \leqslant 8$?

Proofs of numerical statements, in practice

Everybody knows that $1 + 1 = 2$ . . . it only took Russell and Whitehead 360 pages . . .

But how long does it take to prove that $e^2 \leqslant 8$?

- Example lemma needed in the proof (`xlnx_lbound_i12`)

$$\forall x \in]0, 1[,\ x + \frac{x^2}{3} \leqslant (1 + x)\ln(1 + x)$$

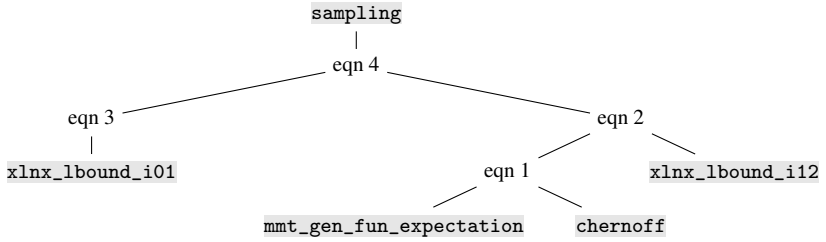
- Proof strategy:
  - Define $f(x) = (1 + x)\ln(1 + x) - (x + x^2/3)$.
  - Compute derivative, study sign, use monotonicity
  - Reduce to $e^2 \leqslant 8$, proved by CoqInterval:

```
Lemma exp2_le8 : exp 2 <= 8. Proof. interval. Qed.
```

  - Without `interval`, our first try took $\approx$ 100 lines
- MATHCOMP-ANALYSIS provides compatibility with CoqInterval for such goals.

```
Lemma exp2_le8_conversion : reflect (exp 2 <= 8) (expR 2 <= 8).
```

Proof Organization



Experiment tracked in PR#1645

Contents	Relevant file	Project
interval inference	`interval_inference.v`	MC
essential supremum/infimum	`ess_sup_inf.v`	MCA
$\mathcal{L}^p$, L^p spaces	`hoelder.v`	MCA
seminorms	`numdomain.v`	MC
basic probability theory	`probability.v`	MCA
power measure, sampling theorem	`sampling.v`	PR#1645
compatibility between reals	`Rstruct.v`, `Rstruct_topology.v`	MCA

Outline

Background

Changing the hierarchy to accomodate seminorms

Automatic handling of interval types

Improve handling of hypotheses in probability theory

Prove concentration inequalities

Putting it all together

Conclusion

Comparison with related work

Interval analysis

- Abstract interpretation of intervals in Rocq [Besson et al., 2009]
- Certified interval arithmetics [Melquiond et al., 2025, Boldo et al., 2015]
- Our novelty: lightweight semi-decision procedure for proof automation

Formalizations of probability theory in Rocq

- InfoTheo [InfoTheo, 2025]: has finitely-supported probabilities; used for information theory, error-correcting codes, robust statistics
- coq-proba [Tassarotti, 2023]: discrete probability spaces; used for program verification
- FormalML [FormalML, 2025]: restricted to finite probability spaces, advanced theorems in probability theory, e.g., stochastic approximation

Formalizations of probability theory in Isabelle and Lean

- Larger formalizations in both ISA-AFP [Gouezel, 2016] and mathlib [Mathlib 4, 2025]
- Compared to [Gouezel, 2016], we use dependent types to encode L^p spaces instead of sets of functions
- Compared to [Mathlib 4, 2025], the use of canonical structures, subtyping handled automatically using Hierarchy-Builder, and automation using intervals demonstrates Rocq's original features

Summary and Future Work

We proposed a formalization of probability leveraging Rocq and MathComp features:

- We used Rocq's type system to automate range inference of values
- We extended the MathComp hierarchy conservatively with $\mathcal{L}^p$ spaces
- We leveraged $\mathcal{L}^p$ spaces to enable succinct statements of basic probability lemmas
- We showed how this infrastructure can be used to formalize a sampling theorem
  - using several concentration inequalities and automation

Future Work

- Port results from existing Rocq probability libraries
  (e.g., conical spaces [Affeldt et al., 2020, Sect. 4])
- Generalize results towards formalization of complex statistical procedures
  (e.g., risk-limiting audits [Stark, 2009])
- Apply L^p spaces to give semantics to logics used in machine learning [Capucci, 2025]

References I

Affeldt, R., Garrigue, J., and Saikawa, T. (2020).

Formal adventures in convex and conical spaces.

In 13th Conference on Intelligent Computer Mathematics (CICM 2020), Bertinoro, Forli, Italy, July 26–31, 2020, volume 12236 of Lecture Notes in Artificial Intelligence, pages 23–38. Springer.

Besson, F., Cachera, D., Jensen, T. P., and Pichardie, D. (2009).

Certified static analysis by abstract interpretation.

In Foundations of Security Analysis and Design V (FOSAD 2007/2008/2009), Tutorial Lectures, volume 5705 of Lecture Notes in Computer Science, pages 223–257. Springer.

Boldo, S., Lelay, C., and Melquiond, G. (2015).

Coquelicot: A user-friendly library of real analysis for Coq.

Math. Comput. Sci., 9(1):41–62.

Capucci, M. (2025).

On quantifiers for quantitative reasoning.

FormalML (2025).

FormalML: Formalization of machine learning theory with applications to program synthesis.

`https://github.com/IBM/FormalML`.

Since 2019.

References II

Gouezel, S. (2016).

Lp spaces.

Archive of Formal Proofs.

`https://isa-afp.org/entries/Lp.html`, Formal proof development.

InfoTheo (2025).

InfoTheo: A Coq formalization of information theory and linear error-correcting codes.

`https://github.com/affeldt-aist/infotheo`.

Authors: Reynald Affeldt, Manabu Hagiwara, Jonas Sénizergues, Jacques Garrigue, Kazuhiko Sakaguchi, Taku Asai, Takafumi Saikawa, Naruomi Obata, and Alessando Bruni. Last stable release: 0.9.3 (2025).

Mathlib 4 (2025).

File `MeasureTheory/Function/LpSpace/Basic.lean`.

url.

Melquiond, G., Érik Martin-Dorel, Roux, P., and Sibut-Pinote, T. (2025).

CoqInterval.

`https://coqinterval.gitlabpages.inria.fr/`.

Since 2008. Version 4.11.1.

References III

Mitzenmacher, M. and Upfal, E. (2005).
Probability and Computing—Randomized Algorithms and Probabilistic Analysis.
Cambridge University Press.

Stark, P. B. (2009).
Risk-limiting postelection audits: conservative P-values from common probability inequalities.
IEEE Trans. Inf. Forensics Secur., 4(4):1005–1014.

Tassarotti, J. (2023).
A probability theory library for the Coq theorem prover.
`https://github.com/jtassarotti/coq-proba`.
Since 2020.