Application of Mathematical Components to probabilities and program formalization

Reynald Affeldt

Nat. Inst. of Advanced Industrial Science and Technology (AIST), Tokyo, Japan

December 7, 2022

This Talk

- We have been interested in applying formal verification to information security
- We have been led to look at topics such as information theory, error-correcting codes, probabilistic programs, etc.
- We would like to report on a few formal theories that might be of general interest
- The common topic is probability, the work spans several years
- Our message is that MathComp has been providing us with a reliable environment for our experiments

Outline

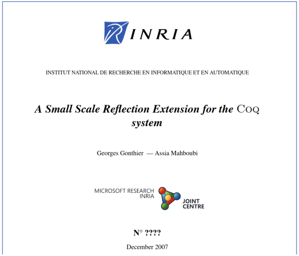
Information Theory using MathComp

Monadic Equational Reasoning for Probabilistic Programs

Formal Semantics for Statistical Modeling

SSReflect Early Adopters?

- Starting in 2004, we were working on formal proof of imperative programs in Coq using separation logic
- We were facing productivity issues that we failed to analyze correctly
- By chance, we ran into



INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

A Small Scale Reflection Extension for the Coq *system*

Georges Gonthier — Assia Mahboubi

MICROSOFT RESEARCH INRIA JOINT CENTRE

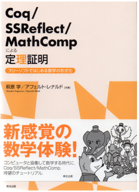
N° ????

December 2007

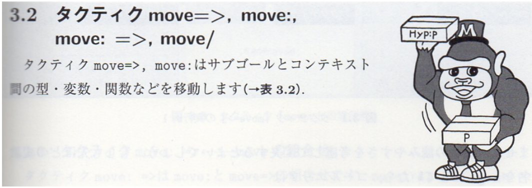
and realized that many of our concerns were addressed by SSReflect

SSReflect/MathComp in Japan

1. We have been trying to promote MathComp in Japan with lectures
2. These lectures turned into a book in 2018



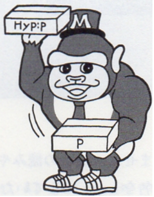
3. In this book, tactics are given mascots, e.g.:


3.2 タクティク move=>, move:, move: =>, move/

タクティク move=>, move:はサブゴールとコンテキスト間の型・変数・関数などを移動します(→表 3.2).

The SSReflect Zoo According to 

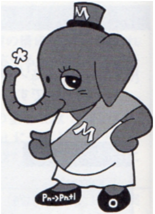
`move`



view

`case`

`rewrite`



`elim`

`apply`

Outline

Information Theory using MathComp

Monadic Equational Reasoning for Probabilistic Programs

Formal Semantics for Statistical Modeling

Fundamental Questions Answered Information Theory

1. What is the ultimate data compression rate[1]?
2. What is the ultimate encoding rate[2] for communication?

Setting:

- a source is a probability distribution
- a channel is a stochastic matrix
  (e.g., the binary symmetric channel $\begin{bmatrix} 1-p & p \\ p & 1-p \end{bmatrix}$)

Shannon answered the fundamental questions in 1948:

1. Source coding theorem: One cannot minimize the compression rate below the *entropy*
2. Channel coding theorem: One cannot maximize the encoding rate beyond the *capacity* of a channel

[1] compressed bitstring size / original message size

[2] original message size / encoded message size

Information Theory: Basic Definitions [CT01]

- The **entropy** of a random variable taking n values with probabilities p_i is $H = -\sum_{i=0}^{n-1} p_i \log p_i$
- Given two probability mass functions p and q, the *divergence* is $D(p||q) = \sum_{x \in \mathcal{X}} p(x) \log \frac{p(x)}{q(x)}$
- The *mutual information* between two random variables X, Y with joint probability mass function $p(x,y)$ and marginals $p(x)$ and $p(y)$ is $I(X;Y) = D(p(x,y)||p(x)p(y))$
- Consider an input X and an output Y. A channel is a conditional probability distribution $p_{Y|X}(y|x)$. The **capacity** is $C = \max_{p(x)} I(X;Y)$, $p(x)$ ranging over all the input distributions

$\Rightarrow$ It looks like a good fit for SSReflect's iterated operators [BGBP08]

Finite Probabilities with SSReflect and Coq

(We are in 2009)

Finite probability theory with the iterated operators and the finite sets of MathComp and the real numbers of Coq:

- Distributions over a finite type:

```
Record fdist (A : finType) := mk {
  f :> A ->R+ ;
  _ : \sum_(a in A) f a == 1 :> R }.
```

- Probability of an event `E` given a distribution `P`:

```
Definition Pr P (E : {set A}) := \sum_(a in E) P a.
```

- Random variables:

```
Definition RV U T (P : fdist U) := U -> T.
```

  - Distribution of a random variable `X`: `` `Pr[ X = a ] ``, shortcut for `Pr P (X @^-1: [set a])` or for `fdistmap X P a`

Information Theory with MathComp

Overview

Starting from:

- $H = -\sum_{i=0}^{n-1} p_i \log p_i$

```
Definition entropy := - \sum_(a in A) P a * log (P a).
```

- $\sum_{x \in \mathcal{X}} p(x) \log \frac{p(x)}{q(x)}$

```
Definition div := \sum_(a in A) P a * log (P a / Q a).
```

- $D(p(x,y) \| p(x)p(y))$

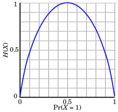
```
Definition mutual_info := D(PQ || PQ`1 `x PQ`2).
```

We completely formalized an introductory textbook to information theory:

- Shannon's theorems [AH12, AHS14]
- Error-correcting codes (Hamming, BCH, Reed-Solomon, LDPC) [AG15, AGS20b]
- Presentations to information theorists [OHA14, AGS16, AGS18]

Convexity of Information-theoretic Definitions

Statements from [CT01]



Entropy of a binary distribution

Theorem

$H(p)$ is a concave function of p.

Theorem

$D(p||q)$ is convex in the pair (p, q), i.e., if (p_1, q_1) and (p_2, q_2) are pairs of probability mass functions, then

$$D(\lambda p_1 + (1-\lambda)p_2 || \lambda q_1 + (1-\lambda)q_2) \leq \lambda D(p_1||q_1) + (1-\lambda)D(p_2||q_2)$$

for all $0 \leq \lambda \leq 1$.

Theorem

Let $(X, Y) \sim p(x, y) = p(x)p(y|x)$. The mutual information $I(X; Y)$ is a concave function of $p(x)$ for fixed $p(y|x)$ and a convex function of $p(y|x)$ for fixed $p(x)$.

Convex Space

A *convex space* is a carrier together with a family of binary operators $a \triangleleft\ p \triangleright b$ with $0 \leq p \leq 1$ such that [Sto49, Fri09]:

- $a \triangleleft\ 0 \triangleright b = b$
- $a \triangleleft\ p \triangleright a = a$ (idempotence)
- $a \triangleleft\ p \triangleright b = b \triangleleft\ 1 - p \triangleright a$ (skewed commutativity)
- $a \triangleleft\ p \triangleright (b \triangleleft\ q \triangleright b) = (a \triangleleft\ r \triangleright b) \triangleleft\ s \triangleright c$ (quasi-associativity)
  with $s = \overline{\bar{p}\bar{q}}$ and $r = \frac{p}{s}$ (where $\bar{x} = 1 - x$)

Examples: real numbers $(pa + (1-p)b)$, functions to a convex space, finite distributions, etc.

Allows for generic definitions (U convex space, V ordered convex space):

- $f : U \to U'$ is affine $\stackrel{def}{=} \forall a, b, 0 \leq p \leq 1, f(a \triangleleft\ p \triangleright b) = f(a) \triangleleft\ p \triangleright f(b)$
- $f : U \to V$ is convex $\stackrel{def}{=} \forall a, b, 0 \leq p \leq 1, f(a \triangleleft\ p \triangleright b) \leq f(a) \triangleleft\ p \triangleright f(b)$
- also convex sets and hulls

Convex Space in MathComp

Conveniently defined using Hierarchy-Builder [CST20]:

1. Declare an interface:

```
HB.mixin Record isConvexSpace (T : Type) := {
  _ <| _ |> _ : forall p, T -> T -> T ;
  conv1 : forall a b, a <| 1%:pr |> b = a ;
  convmm : forall p a, a <| p |> a = a ;
  convC : forall p a b, a <| p |> b = b <| p.~%:pr |> a;
  convA : forall (p q : prob) (a b c : T),
      a <| p |> (b <| q |> c) =
      (a <| [r_of p, q] |> b) <| [s_of p, q] |> c }.
```

2. Declare a structure:

```
#[short(type=convType)]
HB.structure Definition ConvexSpace := {T of isConvexSpace T }.
```

3. Build instances: any `lmodType` (and thus real numbers), the type “`fdist A`”, the type “`A -> fdist B`”, etc.

Application of Convex Spaces to Information Theory

- Short statements for convexity properties of information theoretic definitions:
  - ```
    Lemma entropy_concave :
      concave_function (fun P : fdist A => `H P).
    ```
  - ```
    Lemma mutual_information_concave W :
      concave_function (fun P => mutual_info (P `X W)).
    ```

    where `P `X W` is the product distribution $\lambda(x, y). P\, x \cdot W\, x\, y$
  - ```
    Lemma mutual_information_convex P :
      convex_function
        (fun W : A -> fdist B => mutual_info (P `X W)).
    ```
    ```

Real Cones

A practical tool to reason about convexity

- In a convex space, quasi-associativity and skewed commutativity make for cumbersome symbolic computations
- It is actually possible to transpose such computations into *real cones* where addition is commutative and associative [VW06]:

```
HB.mixin Record isQuasiRealCone A := {
  addpt : A -> A -> A ;
  zero : A ;
  addptC : commutative addpt ;
  addptA : associative addpt ;
  addpt0 : right_id zero addpt ;
  scalept : R -> A -> A ;
  scale0pt : forall x, scalept 0 x = zero ;
  scale1pt : forall x, scalept 1 x = x ;
  scaleptDr : forall r, {morph scalept r : x y / addpt x y >-> addpt x y}
  scaleptA : forall p q x, 0 <= p -> 0 <= q ->
    scalept p (scalept q x) = scalept (p * q) x }.

HB.mixin Record isRealCone A of isQuasiRealCone A := {
  scaleptDl : forall p q x, 0 <= p -> 0 <= q ->
    scalept (p + q) x = addpt (scalept p x) (scalept q x) }.
```

From Convex Spaces to Real Cones

Consider the following inductive type:

```
Inductive scaled (A : Type) := Scaled of Rpos & A | Zero.
```

When `A` is a convex space:

- `scaled A` can be equipped with a real cone structure (take `addpt (Scaled r x) (Scaled q y)` to be $(r+q)(x \triangleleft \frac{r}{r+q} \triangleright y)$)
- `scaled A` can be equipped with a convex space structure (take $x \triangleleft p \triangleright y$ to be `addpt (scalept p x) (scalept (1 - p) y)` [VW06])

We can transpose symbolic computations using the fact that `Scaled 1` is injective and affine:

- `a <|p|> b`
  → `Scaled 1 a <|p|> Scaled 1 b`
  → `addpt (scalept p (Scaled 1 a)) (scalept (1 - p) (Scaled 1 b))`
  where addition is associative and commutative

See InfoTheo online or [AGS20a] for details

Outline

Information Theory using MathComp

Monadic Equational Reasoning for Probabilistic Programs

Formal Semantics for Statistical Modeling

Monadic Equational Reasoning

This is an approach to verify programs with effects using equational reasoning [GH11]

- effects are represented by monad interfaces with typically:
  - an operator (failure, arbitrary choice, probabilistic choice, etc.)
  - rewriting laws in the form of equations
- monad interfaces can **inherit** from other interfaces and can be **combined**

Our starting idea:

- build a hierarchy of interfaces using packed classes [GGMR09]
- use SSReflect's `rewrite` [GT12] to perform equational reasoning

Example of Monadic Laws

Reminder

Monad laws (two operators: $ret(\cdot)$ and $\cdot \gg= \cdot$):

1. $ret(a) \gg= f = f\ a$ (left neutral)
2. $m \gg= (\lambda x.ret(x)) = m$ (right neutral)
3. $(m \gg= f) \gg= g = m \gg= (\lambda x.f\ x \gg= g)$ (associativity)

Arbitrary choice (one operator: $\cdot \Box \cdot$):

1. $(m_1 \Box m_2) \Box m_3 = m_1 \Box (m_2 \Box m_3)$ (associativity)
2. $(m_1 \Box m_2) \gg= k = (m_1 \gg= k) \Box (m_2 \gg= k)$
   (left-distributivity of bind w.r.t. arbitrary choice)

etc.

Functors and Monads with HIERARCHY-BUILDER

- We consider COQ's `Type` to be the category **Set** of sets and functions [TJ16]
- Let us start with functors:
  - action on objects: `F : Type -> Type` (carrier)
  - action on morphisms: `actm` below

```
HB.mixin Record isFunctor (F : Type -> Type) := {
  actm : forall A B, (A -> B) -> F A -> F B;
  functor_id : FunctorLaws.id actm ;
  functor_o : FunctorLaws.comp actm }.
```

- Next, monads (ret/bind interface):

```
HB.factory Record isMonad_ret_bind (F : Type -> Type) := {
  ret' : forall A, A -> F A ;
  bind : forall A B, F A -> (A -> F B) -> F B ;
  bindretf : BindLaws.left_neutral bind ret' ;
  bindmret : BindLaws.right_neutral bind ret' ;
  bindA : BindLaws.associative bind }.
```

The Interface of the Probability Monad

Probability monad:

- extends the type of `Monad`
- similar interface to convex spaces
- with left-distributivity of bind w.r.t. probabilistic choice

```
HB.mixin Record isMonadProb (M : Type -> Type) of Monad M := {
  _ <| _ |> _ : forall p T, M T -> M T -> M T ;
  choice0 : forall T a b, a <| 0 |> b = b ;
  choiceC : forall T p a b, a <| p |> b = b <| 1 - p |> a ;
  choicemm : forall T p, idempotent (_ <| p |> _) ;
  choiceA : forall T p q r s a b c,
    p = r * s  ->  1 - s = (1 - p) * (1 - q) ->
    a <| p |> (b <| q |> c) = (a <| r |> b) <| s |> c;

  choice_bindDl : forall p a b,
    (a <| p |> b) >>= f = (a >>= f) <| p |> (b >>= f) }.
```

Model of the Probability Monad

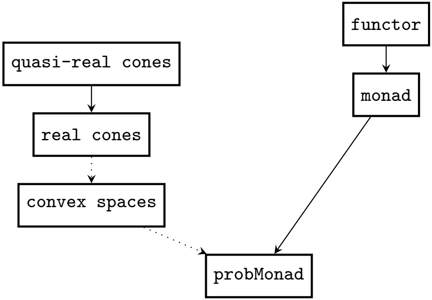
The interface do have an implementation

- Finite distributions do not form a monad because `fdist : finType -> Type` is not an endofunction
- Hence *finitely-supported distributions* with `finmap` [CS15]:

```
Record fsdist (A : choiceType) := mk {
  f :> {fsfun A -> R with 0} ;
  _ : all (fun x => 0 <b f x) (finsupp f) &&
      \sum_(a <- finsupp f) f a == 1}.
```

- The required operators ($ret(\cdot)$, $\cdot \gg= \cdot$, $\cdot \triangleleft \cdot \triangleright \cdot$):
  - `fsdist1 : forall A : choiceType, A -> {dist A}`
    $\stackrel{def}{=}$ `[fsfun b in [fset a] => 1 | 0]`
  - `fsdistbind : forall A B : choiceType,`
    `{dist A} -> (A -> {dist B}) -> {dist B}`
    $\stackrel{def}{=} \lambda b. \sum_{a \in \mathrm{supp}(d)} d(a) \times (f(a))(b)$ over $\bigcup_{x \in f(\mathrm{supp}(d))} \mathrm{supp}(x)$
  - `fsdist_conv : forall A : choiceType,`
    `prob -> {dist A} -> {dist A} -> {dist A}`
    $\stackrel{def}{=} \lambda a. p\, d_1(a) + (1 - p)\, d_2(a)$ over $\mathrm{supp}(d_1) \cup \mathrm{supp}(d_2)$

The Start of a Hierarchy of Effects



Solid arrow: inherits
Dotted arrow: uses

Probabilistic Program Verification using Rewriting

A biased coin with probability p:

```
Definition bcoin {M : probMonad} p : M bool := Ret T <| p |> Ret F.
```

Simple statement:

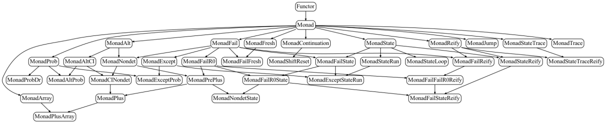
```
Definition two_coins p q : M (bool * bool) :=
  do a <- bcoin p; do b <- bcoin q; Ret (a, b).

Lemma two_coinsE p q : two_coins p q = two_coins q p.
```

Proof:

```
rewrite /two_coins /bcoin.
  (Ret T <|p|> Ret F) >>=
  (fun a => (Ret T <|q|> Ret F) >>= (fun b => Ret (a, b)))
rewrite ![in LHS](choice_bindDl,bindretf).
  (* choice_bindDl -> probability monad law *)
  (* bindretf = ret x >= f = f x -> monad law *)
  (Ret (T, T) <|q|> Ret (T, F)) <|p|> (Ret (F, T) <|q|> Ret (F, F))
rewrite -choiceACA.
  (* interchange <|p|> <|q|> -> real cones *)
  (Ret (T, T) <|p|> Ret (T, F)) <|q|> (Ret (F, T) <|p|> Ret (F, F))
...
```

Examples Formalized with The Monae Library



- tree relabeling [GH11], Spark aggregation [Mu19b], Monty-Hall problem [GH11, Gib12]
- n-queens [GH11], completed by [Mu19a] (we fixed an earlier version of the latter)
- quicksort [MC20] (we completed a pre-existing formalization in Agda)
- Jaskelioff's theory of modular monad transformers [Jas09] (we actually proposed a fix for this theory)

Experiments documented in the following papers [ANS19, AN21, AGNS21, SA22]

Combination of Monad Interfaces Can be Difficult

It was observed in [ASCG16] that [GH11] contains a mistake[3]:

- right-distributivity of bind over probabilistic choice
  $m \gg\!= \lambda x.(k\,x \lhd p \rhd k'\,x) = (m \gg\!= k) \lhd p \rhd (m \gg\!= k')$
  combined with
- distributivity of probabilistic choice over arbitrary choice
  $m \lhd p \rhd (a \,\Box\, b) = (m \lhd p \rhd a) \,\Box\, (m \lhd p \rhd b)$

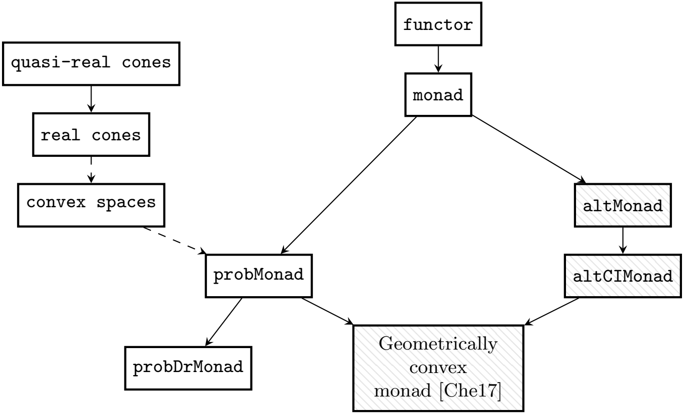
result in a degenerated theory:

- distributivity of arbitrary choice over probabilistic choice
  $m \,\Box\, (a \lhd p \rhd b) = (m \,\Box\, a) \lhd p \rhd (m \,\Box\, b)$
- which implies $a \lhd p \rhd b = a \lhd q \rhd b$ for all $p, q \in]0;1[$
  [KP17, Thm A.3]

$\Rightarrow$ It is important to provide implementations for interfaces

[3] We checked with MONAE that [GH11] was not relying on this mistake.

Hierarchy of Effects (cont'd)



The `probDrMonad` adds:

- $m \gg\!= \lambda x.(k\,x \triangleleft\ p\ \triangleright k'\,x) = (m \gg\!= k) \triangleleft\ p\ \triangleright (m \gg\!= k')$

The geometrically convex monad adds:

- $m \triangleleft\ p\ \triangleright (a \,\Box\, b) = (m \triangleleft\ p\ \triangleright a) \,\Box\, (m \triangleleft\ p\ \triangleright b)$

Model of the Geometrically Convex Monad

What is a computation in this monad?

- Gibbons observes that it should be a convex-closed sets of probability distributions [Gib12]
- Cheung provides a construction using adjunctions between categories [Che17]

We formalized Cheung's construction [AGNS21]:

$$\begin{tikzcd}
\begin{array}{c}\texttt{Type}\\ \mathcal{C}_T\end{array} \arrow[r, bend left, "F_C"] \arrow[r, phantom, "\perp"] & \begin{array}{c}\texttt{choiceType}\\ \mathcal{C}_C\end{array} \arrow[l, bend left, "U_C"] \arrow[r, bend left, "F_0"] \arrow[r, phantom, "\perp"] & \begin{array}{c}\texttt{convType}\\ \mathcal{C}_V\end{array} \arrow[l, bend left, "U_0"] \arrow[r, bend left, "F_1"] \arrow[r, phantom, "\perp"] & \begin{array}{c}\texttt{semiCompSemi-}\\ \texttt{-LattConvType}\\ \mathcal{C}_S\end{array} \arrow[l, bend left, "U_1"]
\end{tikzcd}$$

This relies on an extension of MONAE with *concrete categories* (to go beyond **Set**)

Ask Takafumi here in this room! →

Outline

Information Theory using MathComp

Monadic Equational Reasoning for Probabilistic Programs

Formal Semantics for Statistical Modeling

Statistical Model as Probabilistic Programs

- Example: guessing whether or not today's a weekday by looking at the number of buses passing by [Sta20]

```
normalize (
  let x = sample (bernoulli (2 / 7)) in
  let r = if x then 3 else 10 in
  let _ = score (r ^ 4 / 4! * e ^ (- r)) in
  return x)
```

- Intuitive explanation:
  - `sample` takes a probability measure
  - `normalize` returns a probability measure
  - `score (f x)` means that we observe `x` from the distribution corresponding to the density `f`
    - here, observe 4 from the Poisson distribution (of density $\frac{r^k}{k!}e^{-r}$)
- Problem: existing formalizations in Coq use axioms [HcS19, ZA22]

Formalization of Kernels using MathComp-Analysis

Staton proposed a semantics for programs with sampling, scoring, and normalization using *s-finite kernels* [Sta17]

Definition:

- A *kernel* $X \rightsquigarrow Y$ is a function $k : X \to \Sigma_Y \to [0, \infty]$ such that
  1. for all x, $k\,x$ is a measure
  2. for all measurable set U, $x \mapsto k\,x\,U$ is measurable

Reminder: measure theory in MathComp-Analysis [AC22]

measurable spaces	type `measurableType`
measure	type `{measure set T -> \bar R}`
measurable functions	predicate `measurable_fun`

Formal definition of kernel (notation `R.-ker X ~> Y`):

```
HB.mixin Record isKernel
    X Y R (k : X -> {measure set Y -> \bar R}) :=
  { measurable_kernel :
    forall U, measurable U -> measurable_fun setT (fun x => k x U) }.
```

S-Finite and Finite Kernels

A circular-looking definition

Definition:

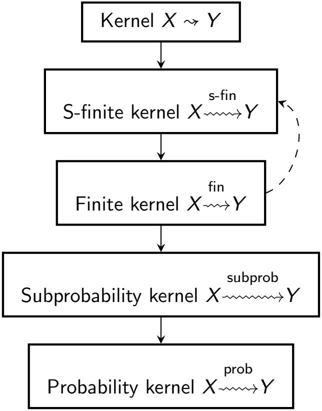
- A kernel $k : X \rightsquigarrow Y$ is *finite* when $\exists r$ s.t. $\forall x$, $k\,x\,Y < r$ (uniformly upper bounded)
- A kernel k is *s-finite* when there exists a sequence of finite kernels s such that $k = \sum_{i=0}^{\infty} s_i$

Circularity?

- s-finite kernels are more general than finite kernels (so they should be defined first)
- finite kernels are needed to define s-finite kernels...

Wanted: Hierarchy of Kernels

To implement Staton's semantics of probabilistic programs



S-Finite and Finite Kernels

A recipe using Hierarchy-Builder

1. Interface for s-finite kernels using a predicate for finite kernels:

```
HB.mixin Record Kernel_isSFinite_subdef
    X Y R (k : X -> {measure set Y -> \bar R}) := {
  sfinite_subdef : exists2 s : (R.-ker X ~> Y)^nat,
    forall n, measure_fam_uub (s n) &
    forall x U, measurable U -> k x U = kseries s x U }.
```

   Notation: `R.-sfker X ~> Y`, inherits from `R.-ker X ~> Y`

2. Interface for finite kernels:

```
HB.mixin Record SFiniteKernel_isFinite
    X Y R (k : X -> {measure set Y -> \bar R}) :=
  { measure_uub : measure_fam_uub k }.
```

   Notation: `R.-fker X ~> Y`, inherits from `R.-sfker X ~> X`

3. Definitive interface for s-finite kernels:

```
HB.factory Record Kernel_isSFinite
    X Y R (k : X -> {measure set Y -> \bar R})
    of isKernel _ _ _ _ _ k := {
  sfinite : exists s : (R.-fker X ~> Y)^nat,
    forall x U, measurable U -> k x U = kseries s x U }.
```

Composition of S-finite Kernels

The main property of s-finite kernels is that they are stable by composition (this provides a semantics for `let x := e in e'`)

- Given $l : X \rightsquigarrow Y$ and $k : X \times Y \rightsquigarrow Z$, the composition $l \boxed{;} k$ is defined by

$$\lambda x\, U.\ \int_y k\,(x, y)\; U(\mathbf{d}\, l\, x)$$

- Reminder: integral theory in MathComp-Analysis [AC22]

$\int_{x \in A} f(x)(\mathbf{d}\,\mu)$	`\int[mu]_(x in A) f x`

- Formal definition of composition:

```
Definition kcomp l k x U := \int[l x]_y k (x, y) U.
```

- Staton proved that the composition of s-finite kernels is a s-finite kernel [Sta17]. He skipped the proof that it is a kernel. It is not trivial but it can be achieved it by adapting existing lemmas from Fubini's theorem available in MathComp-Analysis.

Semantics of Sampling using S-finite Kernels

For illustration

What is the semantics of `sample (bernoulli (2 / 7))`?

1. Build the measurable space of probability measures `pprobability Y R`
   - generated from the set of probability measures μ such that $\mu(U) < r$ for all measurable sets U and $0 \leq r \leq 1$
   - The type `X -> pprobability Y R` is essentially `X -> {measure set Y -> \bar R}`
2. `P : X -> pprobability Y R` is a kernel
   - for any measurable set `U`, `fun x => P x U` is measurable
3. `P : X -> pprobability Y R` is a probability kernel
   - because for all `x`, `P x setT = 1`
   - it is therefore automatically a s-finite kernel
4. For our example, take for `P` the (constant) Bernoulli probability measure (built out of Dirac measures)

Staton's Buses in Coq

```
normalize (
  let x = sample (bernoulli (2 / 7)) in
  let r = if x then 3 else 10 in
  let _ = score (r ^ 4 / 4! * e ^ (- r)) in
  return x)
```

$\Downarrow$

```
Definition kstaton_bus : R.-sfker T ~> mbool :=
  letin (sample (bernoulli p27))
  (letin
    (letin (ite var2of2 (ret k3) (ret k10))
      (score (measurable_fun_comp mh var3of3)))
    (ret var2of3)).
  (* NB: density function parameterized,
         "De Bruijn indices" for variables *)
Definition staton_bus := normalize kstaton_bus.
```

Symbolic Evaluation of Statistical Models

We can evaluate a model to a distribution:

```
Lemma staton_busE P (t : R) U :
  let N := ((2 / 7) * poisson4 3 + (5 / 7) * poisson4 10)%R in
  staton_bus mpoisson4 P t U =
  ((2 / 7)%:E * (poisson4 3)%:E * \d_true U +
   (5 / 7)%:E * (poisson4 10)%:E * \d_false U) * N^-1%:E.
```

(Proof by rewriting)

In mathematical notation:

$$\frac{\frac{2}{7}\frac{3^4}{4!}e^{-3}}{N}\delta_1 + \frac{\frac{5}{7}\frac{10^4}{4!}e^{-10}}{N}\delta_0 = 0.780369\delta_1 + 0.219631\delta_0$$

So it is more likely that we are in the weekend

Commutativity Property of Probabilistic Programs

The main motivation for Staton's work

Is the following program transformation correct?

```
let x := t in        ↔      let y := u in
let y := u in               let x := t in
ret (x, y)                  ret (x, y)
```

This is a consequence of Tonelli-Fubini's theorem for *s-finite measures*:

```
(* f measurable non-negative, m1, m2 s-finite *)
Lemma sfinite_fubini :
  \int[m1]_x \int[m2]_y f (x, y) =
  \int[m2]_y \int[m1]_x f (x, y).
```

(This is a consequence of Tonelli-Fubini's theorem for σ-*finite* measures—the one you find in a standard undergraduate textbook on integration).

Conclusion

The MathComp project has been providing us with

- good tactic support
  (e.g., `rewrite` in monadic equational reasoning)
- a rich, stable, flexible framework (we could combine MathComp and the Coq standard library)
- libraries (`finmap`, MathComp-Analysis)
- methodologies (packed classes, naming conventions)
- tools (Hierarchy-Builder)

which let us

- develop original formalizations (InfoTheo, Monae)
- develop libraries for (probabilistic) program verification
- fix existing pencil-and-paper proofs
- retrofit results to MathComp (in particular MathComp-Analysis)

[AC22] Reynald Affeldt and Cyril Cohen, *Measure construction by extension in dependent type theory with application to integration*, Sep 2022.

[AG15] Reynald Affeldt and Jacques Garrigue, *Formalization of error-correcting codes: from Hamming to modern coding theory*, 6th Conference on Interactive Theorem Proving (ITP 2015), Nanjing, China, August 24–27, 2015, Lecture Notes in Computer Science, vol. 9236, Springer, Aug 2015, pp. 17–33.

[AGNS21] Reynald Affeldt, Jacques Garrigue, David Nowak, and Takafumi Saikawa, *A trustful monad for axiomatic reasoning with probability and nondeterminism*, J. Funct. Program. **31** (2021), e17.

[AGS16] Reynald Affeldt, Jacques Garrigue, and Takafumi Saikawa, *Formalization of Reed-Solomon codes and progress report on formalization of LDPC codes*, International Symposium on Information Theory and Its Applications (ISITA 2016), Monterey, California, USA, October 30–November 2, 2016, IEICE. IEEE Xplore, Oct 2016, pp. 537–541.

[AGS18] ______, *Examples of formal proofs about data compression*, International Symposium on Information Theory and Its Applications (ISITA 2018), Singapore, October 28–31, 2018, IEEE, Oct 2018, pp. 633–637.

[AGS20a] ______, *Formal adventures in convex and conical spaces*, 13th International Conference on Intelligent Computer Mathematics (CICM 2020), Bertinoro, Italy, July 26–31, 2020 (Christoph Benzmüller and Bruce R. Miller, eds.), Lecture Notes in Computer Science, vol. 12236, Springer, 2020, pp. 23–38.

[AGS20b] ______, *Reasoning with conditional probabilities and joint distributions in Coq*, Computer Software **37** (2020), no. 3, 79–95.

[AH12] Reynald Affeldt and Manabu Hagiwara, *Formalization of Shannon's theorems in SSReflect-Coq*, 3rd Conference on Interactive Theorem Proving (ITP 2012), Princeton, New Jersey, USA, August 13–15, 2012, Lecture Notes in Computer Science, vol. 7406, Springer, Aug 2012, pp. 233–249.

[AHS14] Reynald Affeldt, Manabu Hagiwara, and Jonas Sénizergues, *Formalization of Shannon's theorems*, Journal of Automated Reasoning **53** (2014), no. 1, 63–103.

[AN21] Reynald Affeldt and David Nowak, *Extending equational monadic reasoning with monad transformers*, 26th International Conference on Types for Proofs and Programs (TYPES 2020), Leibniz International Proceedings in Informatics, vol. 188, Schloss Dagstuhl, Jun 2021, pp. 2:1–2:21.

[ANS19] Reynald Affeldt, David Nowak, and Takafumi Saikawa, *A hierarchy of monadic effects for program verification using equational reasoning*, 13th International Conference on Mathematics of Program Construction (MPC 2019), Porto, Portugal, October 7–9, 2019, Lecture Notes in Computer Science, vol. 11825, Springer, 2019, pp. 226–254.

[ASCG16] Faris Abou-Saleh, Kwok-Ho Cheung, and Jeremy Gibbons, *Reasoning about probability and nondeterminism*, POPL workshop on Probabilistic Programming Semantics, January 2016.

[BGBP08] Yves Bertot, Georges Gonthier, Sidi Ould Biha, and Ioana Pasca, *Canonical big operators*, 21st International Conference on Theorem Proving in Higher Order Logics (TPHOLs 2008(, Montreal, Canada, August 18–21, 2008, Lecture Notes in Computer Science, vol. 5170, Springer, 2008, pp. 86–101.

[Che17] Kwok-He Cheung, *Distributive interaction of algebraic effects*, Ph.D. thesis, Merton College, University of Oxford, 2017.

[CS15] Cyril Cohen and Kazuhiko Sakaguchi, *A finset and finmap library: Finite sets, finite maps, multisets and order*, Available at `https://github.com/math-comp/finmap`, 2015, Last stable release: 1.5.0 (2020).

[CST20] Cyril Cohen, Kazuhiko Sakaguchi, and Enrico Tassi, *Hierarchy builder: Algebraic hierarchies made easy in coq with elpi (system description)*, 5th International Conference on Formal Structures for Computation and Deduction (FSCD 2020), June 29–July 6, 2020, Paris, France (Virtual Conference), LIPIcs, vol. 167, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020, pp. 34:1–34:21.

[CT01] Thomas M. Cover and Joy A. Thomas, *Elements of information theory*, Wiley, 2001.

[Fri09] Tobias Fritz, *Convex spaces I: Definition and examples*, 2009.

[GGMR09] François Garillot, Georges Gonthier, Assia Mahboubi, and Laurence Rideau, *Packaging mathematical structures*, 22nd International Conference on Theorem Proving in Higher Order Logics (TPHOLs 2009), Munich, Germany, August 17–20, 2009, Lecture Notes in Computer Science, vol. 5674, Springer, 2009, pp. 327–342.

[GH11] Jeremy Gibbons and Ralf Hinze, *Just do it: simple monadic equational reasoning*, Proceeding of the 16th ACM SIGPLAN international conference on Functional Programming, ICFP 2011, Tokyo, Japan, September 19–21, 2011 (Manuel M. T. Chakravarty, Zhenjiang Hu, and Olivier Danvy, eds.), ACM, 2011, pp. 2–14.

[Gib12] Jeremy Gibbons, *Unifying theories of programming with monads*, Revised Selected Papers of the 4th International Symposium on Unifying Theories of Programming (UTP 2012), Paris, France, August 27–28, 2012 (Burkhart Wolff, Marie-Claude Gaudel, and Abderrahmane Feliachi, eds.), Lecture Notes in Computer Science, vol. 7681, Springer, 2012, pp. 23–67.

[GT12] Georges Gonthier and Enrico Tassi, *A language of patterns for subterm selection*, Interactive Theorem Proving - Third International Conference, ITP 2012, Princeton, NJ, USA, August 13-15, 2012. Proceedings (Lennart Beringer and Amy P. Felty, eds.), Lecture Notes in Computer Science, vol. 7406, Springer, 2012, pp. 361–376.

[HcS19] Matthew Heimerdinger and Chung chieh Shan, *Verified equational reasoning on a little language of measures*, Workshop on Languages for Inference (LAFI 2019), Cascais, Portugal, January 15, 2019, Jan 2019.

[Jas09] Mauro Jaskelioff, *Modular monad transformers*, 18th European Symposium on Programming (ESOP 2009), York, UK, March 22–29, 2009. Proceedings, Lecture Notes in Computer Science, vol. 5502, Springer, 2009, pp. 64–79.

[KP17] Klaus Keimel and Gordon D. Plotkin, *Mixed powerdomains for probability and nondeterminism*, Logical Methods in Computer Science **13** (2017), no. 1:2, 1–84.

[MC20] Shin-Cheng Mu and Tsung-Ju Chiang, *Declarative pearl: Deriving monadic quicksort*, 15th International Symposium on Functional and Logic Programming (FLOPS 2020), Akita, Japan, September 14–16, 2020, Lecture Notes in Computer Science, vol. 12073, Springer, 2020, pp. 124–138.

[Mu19a] Shin-Cheng Mu, *Calculating a backtracking algorithm: An exercise in monadic program derivation*, Tech. Report TR-IIS-19-003, Institute of Information Science, Academia Sinica, Jun 2019.

[Mu19b] ______, *Equational reasoning for non-deterministic monad: A case study of Spark aggregation*, Tech. Report TR-IIS-19-002, Institute of Information Science, Academia Sinica, Jun 2019.

[OHA14] Ryosuke Obi, Manabu Hagiwara, and Reynald Affeldt, *Formalization of variable-length source coding theorem: Direct part*, International Symposium on Information Theory and Its Applications (ISITA 2014), Melbourne, Australia, October 26–29, 2014, IEICE. IEEE Xplore, Oct 2014, pp. 201–205.

[SA22] Ayumu Saitou and Reynald Affeldt, *Towards a practical library for monadic equational reasoning in Coq*, 14th International Conference on Mathematics of Program Construction (MPC 2022), Tbilisi, Georgia, September 26–28, 2022, Lecture Notes in Computer Science, vol. 13544, Springer, 2022, pp. 151–177.

[Sta17] Sam Staton, *Commutative semantics for probabilistic programming*, 26th European Symposium on Programming (ESOP 2017), Uppsala, Sweden, April 22–29, 2017, Lecture Notes in Computer Science, vol. 10201, Springer, 2017, pp. 855–879.

[Sta20] Sam Staton, *Foundations of probabilistic programming*, ch. Probabilistic Programs as Measures, pp. 43–74, Cambridge University Press, 2020, Editors: Barthe, Gilles and Katoen, Joost-Pieter and Silva, Alexandra.

[Sto49] Marshall Harvey Stone, *Postulates for the barycentric calculus*, Annali di Matematica Pura ed Applicata **29** (1949), 25–30.

[TJ16] Amin Timany and Bart Jacobs, *Category theory in Coq 8.5*, 1st International Conference on Formal Structures for Computation and Deduction (FSCD 2016), June 22–26, 2016, Porto, Portugal (Delia Kesner and Brigitte Pientka, eds.), LIPIcs, vol. 52, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016, pp. 30:1–30:18.

[VW06] Daniele Varacca and Glynn Winskel, *Distributing probability over nondeterminism*, Mathematical Structures in Compuer Science **16** (2006), no. 1, 87–113.

[ZA22] Yizhou Zhang and Nada Amin, *Reasoning about "reasoning about reasoning": semantics and contextual equivalence for probabilistic programs with nested queries and recursion*, Proc. ACM Program. Lang. **6** (2022), no. POPL, 1–28.