Synthetic topology in Homotopy Type Theory for probabilistic programming

Florian Faissole[1]* and Bas Spitters[2]**

[1] Inria, Université Paris-Saclay, F-91120 Palaiseau
LRI, CNRS & Université Paris-Sud, F-91405 Orsay
florian.faissole@lri.fr
[2] Aarhus University spitters@cs.au.dk

Abstract. The ALEA Coq library formalizes discrete measure theory using a variant of the Giry monad, as a submonad of the CPS monad: $(A \to [0,1]) \to [0,1]$. This allows one to use Moggi's monadic meta-language to give an interpretation of a language, *Rml*, into type theory. *Rml* is a functional language with a primitive for probabilistic choice. This formalization was the semantical basis for the Certicrypt system for verifying security protocols. The Easycrypt proof assistant is still based on the same semantics. We improve on the formalization by using homotopy type theory which provides e.g. quotients and functional extensionality. Moreover, homotopy type theory allows us to use synthetic topology to present a theory which also includes non-discrete ('continuous') data types, like $[0,1]$. Such data types are relevant, for instance, in machine learning and differential privacy. Our axioms are justified by Kleene-Vesley realizability, a standard model for computation with continuous data types.

Keywords: Probabilistic computation, Synthetic topology, Homotopy type theory, Coq

1 Introduction

In her fundamental work on categorical measure theory, Giry [18] proved that the set of all measures $\mathcal{M}(X)$ on a measure space X is itself a measure space, and that $\mathcal{M}$ is in fact a monad on the category of measure spaces. Jones and Plotkin [25] showed how to use this as a semantics for probabilistic programming, using Moggi's monadic meta-language. The ALEA [3] Coq library formalizes discrete measure theory using a variant of the Giry monad, as a submonad of the CPS monad: $(A \to [0,1]) \to [0,1]$. This allows one to use Moggi's monadic meta-language to give an interpretation of a language *Rml* into type theory. *Rml* is a functional language with a primitive for probabilistic choice. To be precise,

* Two page abstracts of this work appeared at CoqPL'17 and PPS'17.

** This research was partially supported by the Guarded homotopy type theory project, funded by the Villum Foundation, project number 12386 and Digiteo.

the type of the monad of measures also requires monotonicity, summability and linearity. However, Coq cannot prove this to be a monad, as the equality on distributions is not the intensional equality of Coq. We solve this technical issue by using homotopy type theory (HoTT). More importantly, this allows us to use synthetic topology to present a theory which also includes non-discrete ('continuous') datatypes like $[0, 1]$.

Contributions We combine homotopy type theory and synthetic topology to provide a new development of parts of measure theory. This is then used as an axiomatic semantics for probabilistic computation using the monadic meta-language. This semantics has features of both denotational and operational (sampling) semantics, as we will explain. Our main insight is the extension of the Giry monad from locales to synthetic topology. We provide a substantial Coq development including synthetic topology, a theory of valuations and lower integrals, and an interpretation of probabilistic programs via a shallow embedding. This is based on the HoTT library [6] and the experimental induction-recursion branch of Coq. The development of synthetic topology dependents on a dominance, which we define by using the partiality monad. Finally, we show how to execute partial functions in Coq using the tactic language. This gives us a, currently slow, executable semantics.

Plan We assume the reader has basic familiarity with homotopy type theory [41]. Section 2 presents probability theory and synthetic topology. Section 3 introduces homotopy type theory and the univalent foundations. Section 4 describes our Coq development. Section 5 describes a small functional probabilistic language and its monadic interpretation. Section 6 deals with our computable denotational semantics and operational semantics. Section 7 contrasts our work with alternative approaches. Section 8 concludes.

2 Probability theory and synthetic topology

The Giry monad [18] can be constructed on various categories, for instance on the category of Polish spaces, on the category of measurable spaces, and on a category of domains [25]. We could first formalize these categories in Coq and then build the model on top of that. Such an approach is taken in Isabelle/HOL; e.g. [21]. However, such categories are not cartesian closed, so they do not allow function types; see subsection 7.2. We can solve this problem and stay closer to the ALEA library by using synthetic topology.

Synthetic topology follows the philosophy of synthetic domain theory [24], a successful tool in semantics. In synthetic topology, one uses a domain specific language for topology, or more precisely, a category which is sufficiently like that of topological spaces; see [14,7] for an overview. We use a topos with a special object $\mathbb{S}$ which classifies the (enumerable) opens. I.e. open subsets are replaced by maps to $\mathbb{S}$. One may think of $\mathbb{S}$ as the Sierpinski space, since in the category of topological spaces there is a bijective correspondence between open subsets and the inverse image of a continuous map at $1 \in \mathbb{S}$. In synthetic computability, one would take $\mathbb{S}$ to be the semi-decidable truth values. An abstract theory can be developed based on the assumption that $\mathbb{S}$ is a *dominance* — a certain subset of the set Ω of propositions. This axiom implies that monomorphisms classified by $\mathbb{S} \subset \Omega$ compose; e.g. [15, 2.6]. Topologically, $\mathbb{S}$ being a dominance implies that open subsets of open subsets are open.

Our Coq development is not maximally abstract, in that we are working with the specific dominance of semi-decidable propositions. This seems fitting though, as measure theory deals with *countable* unions as opposed to general ones. Moreover, most spaces we care about have a countable basis. We do, however, abstract from standard realizability presentations of computations with continuous datatypes, such as Kleene's second algebra [26], also known as TTE [43,5], or domain theory.

3 Homotopy type theory and univalent foundations

Coq's type theory lacks quotients and functional extensionality. To address this ALEA uses so-called setoids, a type together with an equivalence relation. However, they are not used consistently. For instance, the carrier of a distribution is a bare type. This works for the discrete types, but for continuous types, such as $[0, 1]$, or the type of valuations on the booleans, an equivalence relation should be taken into account. Setoids make the library quite heavy since one needs to prove that all functions actually preserve this relation. Even though better support has been developed [38], there is now a more principled solution. Homotopy type theory [41] provides a consistent way of adding such features while preserving the good computational properties [12]. Although details of a full integration of these ideas in the Coq proof assistant need to be worked out, the HoTT library [6] adds these features axiomatically. This is done in such a way that terms in quotients types will reduce as expected. Many Coq users have a naive view of Coq's types as some kind of sets. HoTT makes this precise, the so-called hSets behave much like structural sets. To be precise, they form a ΠW-pretopos [33]. The universe is an example of a type that is not an hSet. Voevodsky's univalence axiom is an extensionality axiom for the universe: all maps out of the universe should respect equivalence of types.

On top of HoTT, we add the axioms for synthetic topology. More precisely, we add them only for the hSets, based on the fact that they form a ΠW-pretopos. NuPrl [31] provides an extensional type theory in which the axioms for synthetic topology are provable. However, we prefer Coq, as it is a more mature system and we also get some benefits from homotopy type theory, as discussed below.

The sheaf topos models for synthetic topology can naturally be extended to higher toposes, and hence to models of homotopy type theory [37]. There are some technical issues connected to so-called weak Tarski universes, however they do not concern us here, as even for (the implementation of) Coq itself there does not seem to be a definitive treatment about the distinction between Russell and Tarski style universes. Similarly, one may extend realizability models to homotopy type theory, for instance by interpreting the constructive cubical model of HoTT in the extensional type theory of a realizability topos, or by considering cubical assemblies[3]. In fact, the realizabilty models embed into topological models [4]. We acknowledge that the definitive treaty on the semantics of HoTT still needs to be written, but following e.g. Shulman [36] we here investigate whether such models are useful. We avoid the axiom of countable choice which is not available in HoTT, although it does hold in the realizability models we are interested in.

In comparison with extensional type theory, as e.g. in NuPrl, HoTT (with univalence and higher inductive types) gives us a few benefits. For instance, the univalence axiom is well-suited for algebraic and categorical reasoning [41]; see e.g. subsection 4.2. Moreover, it facilitates the formalization of free (algebraic) structures. For instance, the partiality monad [1] is the free ω-cpo completion, a quotient inductive inductive type (QIIT). This is a type-theoretic encoding of the 'flat domain' in domain theory. For any type A, $A_\perp$ is defined with constructors, $\eta, \perp, \bigcup$ and a relation $\subseteq$ (and its expected properties [1]):

$$A_\perp : hSet \qquad \subseteq_{A_\perp} : A_\perp \to A_\perp \to hProp.$$
$$\eta : A \to A_\perp \qquad \perp : A_\perp$$
$$\bigcup : \prod_{f:\mathbb{N}\to A_\perp} (\prod_{n:\mathbb{N}} f(n) \subseteq_{A_\perp} f(n+1)) \to A_\perp$$

When termination is decidable, $A_\perp = A + \perp$. In general, elements of $A_\perp$ can be viewed as semi-decidable subsingleton subsets of A.

Gilbert [17] defined the partiality monad in Coq on top of the HoTT library [6] by simulating QIITs based on the experimental induction-recursion branch[4] of Coq. One then defines the semi-decidable propositions as $\mathbb{S} := \mathbf{1}_\perp$, where $\mathbf{1}$ is the unit type. This is a distibutive lattice. We will show that it is in fact a the dominance. This seems like the natural way to present the Rosolini dominance in HoTT, a context where countable choice is not present. Gilbert [17] also formalized the Cauchy and Dedekind reals in HoTT based on an adaptation of the MathClasses library [27,39]. MathClasses provides an abstract approach to continuous computation, using type classes. It contains, for instance, the algebraic and order-theoretic properties of the exact real numbers. Dedekind reals are pairs of predicates of type $\mathbb{Q} \to \mathbb{S}$ with the expected properties. In our development, we will reuse this approach to define their lower semi-continuous counterpart: the lower reals.

4 Implementation in HoTT

This section presents the layers of our underlying development of synthetic topology. In Section 4.1, we connect the partiality monad with our adaptation of ALEAs theory of ω-cpos and check that partiality is the free ω-cpo completion. We also check that $\mathbb{S}$ is a dominance in Section 4.2 and provide techniques to compute with partial types in Section 4.3. Section 4.4 presents our formalization

[3] E.g. `www.mat.uc.pt/~ct2017/slides/frey_j.pdf`

[4] `github.com/mattam82/coq/commits/IR`

of the space of lower reals. Section 4.5 defines a bijection between valuations and lower integrals.

4.1 Partiality and free ω-cpos

Altenkirch, Danielsson and Krauss [1] define $A_\bot$ to be the free ω-cpo on A. That is, given any ω-cpo C and a map $f : A \to C$, there is a (unique) map of type $f_\bot : A_\bot \to C$ such that $f_\bot \circ \eta = f$. This connects (our adaptation of) ALEAs theory with the partiality monad.

4.2 The Sierpiski space is a dominance

The notion of dominance is central in synthetic topology. A dominance is a suitable subset of the hpropositions. In our Coq development, we define a dominance S together with the coercion `IsTop`$(s) := (s = \top_S)$ from S to hProp.

Definition 1 *A type S endowed with a map to hProp is a **dominance** iff* $\top \in S$ *and:* $\forall u : hProp, \forall s : S, (\mathit{IsTop}(s) \Rightarrow$
$\exists z \in S, \mathit{IsTop}(z) = u) \Rightarrow \exists m \in S, \mathit{IsTop}(m) = u \wedge \mathit{IsTop}(s)$.

This can also be formulated [15], as a predicate $d(x) := \exists z \in \mathbb{S}.x = z$ such that dominant types are propositions, $d(\top)$ and

$$\prod_{P:U}\ \prod_{Q:P\to U} d(P) \to \prod_{p:P} d(Q(p)) \to d(\sum_{p:P} Q(p)).$$

Lemma 1 $\mathbb{S}$ *is a dominance.*

Proof. As $\mathbb{S}$ is $\mathbf{1}_\bot$, we proceed by induction. The cases $u = \top$ and $u = \bot$ are trivial. When u is the supremum of an increasing sequence u_n in $\mathbb{S}$ we need to pick a sequence of witnesses m_n. A priori, defining such a sequence would require countable choice. However, each m_n is *uniquely* determined by u_n, so we use unique choice property from the HoTT library, a constructive counterpart of the Hilbert's iota axiom. □

Alternatively, we could use the monad axioms to prove the equivalent characterization [15] of dominance: $d(U) \Rightarrow (U \to d(V)) \Rightarrow d(U \times V)$.

Dominance and univalence We now show explain how dominances and univalent type theory interact naturally. We have not formalized this part. Escardó and Knapp [15] provide a general construction of a lifting defined by a dominance. We present it here, specialized to $\mathbb{S}$. Consider the unique map $! : X \to 1$ and its lifting $!_\bot : X_\bot \to 1_\bot = \mathbb{S}$. Given $y : X_\bot$, $!_\bot y$ is called the *extent* of y and the map $\mathit{val} :!_\bot y \to X$, defined by $\mathbb{S}$-induction, is called the *value* of y. The extent and the value together give a partial map $1 \rightharpoonup X$, a *partial element.* The lifting $\mathcal{L}X$ is the collection of partial elements $\Sigma_{\sigma:\mathbb{S}}\sigma \to X$. They ask whether $X_\bot \to \mathcal{L}X$ is an equivalence. We give a positive answer:

Proposition 1 *The map $F : X_\perp \to \mathcal{L}X$ is an equivalence.*

To prove this we use the observation that on an ω-cpo Y we have a restriction operation $y|_\sigma$ for $\sigma : \mathbb{S}$.

Proof. We consider the statement $P(\sigma) := \prod_{f:\sigma\to X} \exists! y : X_\perp . F(y) = (\sigma; f)$. We prove this statement by induction. For this, it is important that the existential is a proposition. The base cases are trivial. For the induction case, let $\sigma = \sup \sigma_n$. Then each $f|_{\sigma_n}$ defines an element $y_n : X_\perp$ by unqiue choice. The sequence y_n is increasing, because $f|_{\sigma_n}$ is increasing. So, $\sup y_n$ has the desired properties. To prove uniqueness, consider z satisfying the specification. Then $z|_{\sigma_n} = y_n$ and hence $z = \sup_n z|_{\sigma_n} = \sup y_n = y$. □

In fact, $\operatorname{ext} y$ is the pullback of $y : 1 \to X_\perp$ and $\eta : X \to X_\perp$, with the unique map $! : exty \to 1$ and $\operatorname{val} y$ as projections. The equivalence above shows that for any map $\sigma \to X$, $! : \sigma \to 1$ fits in a pullback square with an element $z : 1 \to X_\perp$. This generalizes to show that $X_\perp$ is the *classifier for partial maps with open domain.* Such a partial map $A \to X$ consists of an open embedding $D \rightarrowtail A$ and a map $D \to X$. By considering fibers we see that any such pair of maps determines a map $A \to X_\perp$ such that D is the pullback along $\eta : X \to X_\perp$. Partial map classifiers for the dominance hProp are fundamental in topos theory.

A dominance allows one to define a category of monomorphisms. In synthetic topology, it is the class of open inclusions, which we may identify with open subsets, by *univalence.* By a minor generalization of [33, Thm 3.31], we have that any mere property P of types classifies the collection of P-fibred maps. Being modal [41] is such an mere property. Examples include the truncations and the identity. This gives for instance the object classifier and the subobject classifier. Moreover, this also includes the mere predicate d. This gives us for each type A an equivalence between dominantly-fibered families and maps into $\mathbb{S}$. More concretely, this formalizes the categorical practice of identifying (open) subsets with equivalence classes of (open) maps. As a practical example, consider the coercion (an embedding) of $[0,1]$ into $\mathbb{R}$. Then we are used to identify the embeddings $(0,1)_{[0,1]} \rightarrowtail \mathbb{R}$ and $(0,1)_\mathbb{R} \rightarrowtail \mathbb{R}$. The classification theorem tells us these maps are equal, as families over $\mathbb{R}$, in type theory.

4.3 Computing with Partial values

All functions in Coq are terminating. This is a very strong guarantee and facilitates the semantics. However, occasionally one wants to write a non-terminating function, so there has been a quest to add them *safely* to type theory. Here we combine two ideas. First, one uses the partiality monad [1] to obtain an ω-cpo enriched Kleisli category. Second, based on the partiality monad, we have the lower real numbers which allow us to develop computable domain theory, which in turn allows us to interpret general fixpoints in the probabilistic semantics. Since the valuations on an hSet A form an ω-cpo, as we will show below, we have a unique embedding from $A_\perp$. This shows that partiality is a special case of probabilistic partiality.

Our semantics below will be semi-decidable; see section 6. One may wonder how to actually evaluate such propositions. Unfortunately, there is no way to implement a non-constant function from $\mathbb{S}$ to a discrete type, such as bool, since this would allow us to decide all semi-decidable propositions. However, we consider two other options. First we could treat such functions by using Coq's extraction mechanism to obtain a program in a general programming language, like ocaml, haskell or scheme, and run the program there. We have no guarantee that the program terminates, but if it does we have a formal guarantee that the result is correct. We have used a second method: we have defined a tactic which tries to prove that a concrete semi-decidable truth value is true. It takes a concrete natural number m as fuel. Our tactic `S_compute m` applies to goals of the form `IsTop(s)`: if `s` is $\top$ then it trivially proves the goal; if `s` $= \sup(\lambda n.(f(n))$, it tries to prove $f(m)$. If it fails, it rewrites $\sup(\lambda n.(f(n)) = \sup(\lambda n.(f(n+m))$ and finishes by a reflexivity proof. We are thus simulating possibly non-terminating functions by computing approximations. This may be compared to the reflection of decidable types into the booleans, which is so characteristic for ssreflect [19]. Here we reflect into semi-decidable propositions. A simular tactic, but without the reflection into $\mathbb{S}$, has been used effectively to semi-decide real number inequalities [29, 5.5].

4.4 Lower Reals

Starting from $\mathbb{S}$ we define the lower reals $\mathbb{R}_l$. These are lower *open* cuts in the rational numbers, i.e. maps $\mathbb{Q} \to \mathbb{S}$. These are called the *lower* reals because maps $X \to \mathbb{R}_l$ correspond to lower semi-continuous functions in synthetic topology. Similarly, we can define the upper reals. A consistent pair of an upper and a lower real defines a Dedekind real. Defining the various real number structures using $\mathbb{S}$ helps to keep the universe levels down, as $\mathbb{S}$ is at the lowest level. We thus obtain a *predicative* definition of real numbers. We construct both the lower reals and the upper reals as instances of a general theory of rounded cuts in a full pseudo order. Hence we define $\mathbb{R}_l$ as the instance of the rounded cuts `RC` with `A` $= \mathbb{Q}$ and `Rlt` $= \leq_{\mathbb{Q}}$.

```
Definition QPred := Q → Sier.
Variable elt : QPred.

Class IsRlow : Type := {
    is_inhab : hexists (fun q ⇒ elt q);
    is_rounded : forall q, elt q ↔
                hexists (fun r ⇒ q < r ∧ elt r) }.
```

We obtain a fully-constructive formalization of the lower reals. The algebraic manipulations are slightly more difficult than in the classical axiomatized reals of the Coq standard library, or the unit interval in ALEA, as both contain axioms that break the computational interpretation. One main difficulty was an engineering issue. Rational numbers, being decidable, have the same theory classically and constructively. However, we could not use the standard library rationals, since Coq's Prop is inconsistent with univalence; see [6] for details. Fortunately, the rationals from the math-classes library had already been ported [17]. A tactic for dealing with ring equalities is also available. However, more automation would be welcome.

The development is substantial (about 1.800 lines of Coq) and we provide various operations on lower reals, such as addition, multiplication by a positive rational number, a cast from rational numbers to lower reals and so on, together with proofs of expected properties and theorem proving machinery for lower reals. We focus on positive lower reals $\mathbb{R}_l^+$ (lower reals containing every negative rational number) and formally prove the following result.

Lemma 2 $\mathbb{R}_l^+$ *is a distributive lattice ordered semi-group and an ω-cpo with* $\bot = 0_{\mathbb{R}_l^+}$.

In fact, it is a semi-ring. However, so far we have only defined (and needed) multiplication by a rational scalar.

4.5 Integrals and Valuations

We define open subsets and positive lower real valued functions on a set A. Then we define lower integrals and valuations, respectively defined on open subsets and on functions of A.

Layers of the formalization We define *open subset*s of $A : hSet$ ($hSet$ corresponds to homotopy 0-type) as the maps of type $OS(A) := A \to \mathbb{S}$ and the positive real functions from A as the maps of type $mf(A) := A \to \mathbb{R}_l^+$. The notation refers to ALEA's measurable functions. We prove that the space of opens is a distributive lattice and an ω-cpo, and that the space of functions is a semi-group for the operation $+$, and is an ω-cpo too.

From these we can define valuations and integrals on $A : hSet$. Valuations are maps from the open subsets to the positive lower reals:

Definition 2 *A (probability)* ***valuation*** *on $A : hSet$ is a map $\mu : OS(A) \to \mathbb{R}_l^+$ such that:*

- *Definiteness:* $\mu(\emptyset) = 0$;
- *Modularity:* $\forall U, V \in OS(A), \mu(U) + \mu(V) = \mu(U \cup V) + \mu(U \cap V)$;
- *Monotonicity:* $\forall U, V \in OS(A), U \subseteq V \Rightarrow \mu(U) \leq \mu(V)$;
- *Sub-probability:* $\mu(A) \leq 1$;
- *Continuity:* $\forall f : \mathbb{N} \to OS(A), \forall n, f_n \leq f_{n+1} \Rightarrow \mu(\sup(\lambda n.f(n)) \leq \sup(\lambda n.\mu(f(n)))$.

For locales, Vickers [42] uses lower integrals, which map to the lower reals. We generalize this construction to hSets.

Definition 3 *A (probability)* ***positive lower integral*** *on A is a map $\mathcal{I} : mf(A) \to \mathbb{R}_l^+$ such that:*

- *Definiteness:* $\mathcal{I}(0_{mf(A)}) = 0$;

- *Additivity:* $\forall f, g \in mf(A), \mathcal{I}(f) + \mathcal{I}(g) = \mathcal{I}(f+g)$*;*
- *Monotonicity:* $\forall f, g \in mf(A), f \leq g \Rightarrow \mathcal{I}(f) \leq \mathcal{I}(g)$*;*
- *Sub-probability:* $\mathcal{I}(1_{mf(A)}) \leq 1$*;*
- *Continuity:* $\forall f : \mathbb{N} \to mf(A), \forall n, f_n \leq f_{n+1} \Rightarrow \mu(\sup(\lambda n.f(n)) \leq \sup(\lambda n.\mu(f(n)))$*.*

As in ALEA, both intergrals and valuations carry an ω-cpo structure. This allows us to interpret recursive probabilistic programs (see Section 5.2). The equality on valuations (or on integrals) are hpropositions, due to functional extensionality.

Riesz Theorem Coquand and Spitters provide a constructive Riesz' Theorem [13]: a bijection between integrals and valuations for compact regular locales. These integrals are more informative, as they map to the positive *Dedekind* reals $\mathbb{R}^+_D$; see [13]. This will allow us to develop a good constructive probability theory for spaces including $[0, 1]$. However, to obtain a monad on *all hSets*, we need to consider lower integrals $Int^+_l(A) = (A \to \mathbb{R}^+_l) \to \mathbb{R}^+_l$. Vickers [42] proves another variant of Riesz's Theorem: a homeomorphism between lower integrals and valuations, for all locales. We carry out a similar construction in synthetic topology. Actually, we provide a formal proof of one direction of Riesz' Theorem. From any lower integral $\mathcal{I}$, we build a valuation $\mu_{\mathcal{I}}$. Thus we prove that this construction verifies the properties of an integral. Symmetrically, we provide machinery to build an integral $\mathcal{I}_\mu$ from a measure.

From integral to valuation

Definition 4 *The valuation* μ_I *is defined as* $\mu_{\mathcal{I}}(U) := \mathcal{I}(\mathbb{1}_U)$*.*

In our context, the construction of the map $\mathbb{1}_U$ of type $mf(A)$ from $U : OS(A)$ comprises several steps. We build (inductively) an embedding of $\mathbb{S}$ into $\mathbb{R}^+_l$. This also motivates our choice to use $\mathbb{S}$ to encode lower reals. For $U : A \to \mathbb{S}$, we obtain $\mathbb{1}_U : A \to \mathbb{R}^+_l$ by composition. We prove modularity of this embedding.

Lemma 3 *Let* $U : OS(A)$*. The map* $\mathbb{1}_U$ *verifies the following properties:*

- *(1)* $\forall U, V \in OS(A), \mathbb{1}_{U \cap V} = \mathbb{1}_U \cap \mathbb{1}_V$*;*
- *(2)* $\forall U, V \in OS(A), \mathbb{1}_{U \cup V} = \mathbb{1}_U \cup \mathbb{1}_V$*;*
- *(3)* $\forall U, V \in OS(A), \mathbb{1}_{U \cup V} + \mathbb{1}_{U \cap V} = \mathbb{1}_U + \mathbb{1}_V$

Properties (1) and (2) are proved by showing the cast of $\mathbb{S}$ into $\mathbb{R}^+_l$ preserves joins and meets, which is proved using properties of $\mathbb{S}$ as the distributivity of meets and joins over supremums. Property (3) is derived from (1) and (2). A characterization lemma follows:

Lemma 4 *Suppose* $U \in OS(A)$*,* $x \in A$*. Then:* $\mathbb{1}_U(x) = 1_{\mathbb{R}^+_l} \Leftrightarrow U(x) = \top$*.*

This lemma is used to prove a first part of the Riesz Theorem:

Theorem 1 *Let* $\mathcal{I}$ *an integral. Then* μ_I *is a valuation.*

Proof.
- **Definiteness**: by definition of $\mathbb{1}$, $\mu_{\mathcal{I}}(\emptyset) = \mathcal{I}(\mathbb{1}_{\emptyset}) = \mathcal{I}(0_{mf(A)})$.
  Hence, as $\mathcal{I}$ is an integral, $\mu_I(\emptyset) = 0$;
- **Monotonicity**: suppose $U \subseteq V$. We know $\mu_{\mathcal{I}}(U) = \mathcal{I}(\mathbb{1}_U)$ and $\mu_{\mathcal{I}}(V) = \mathcal{I}(\mathbb{1}_V)$.
  We prove that $U \subseteq V \Rightarrow \mathbb{1}_U \leq \mathbb{1}_V$.
  Thus by monotonicity of $\mathcal{I}$, $\mathcal{I}(\mathbb{1}_U) \leq \mathcal{I}(\mathbb{1}_V)$ i.e. $\mu_{\mathcal{I}}(U) \leq \mu_{\mathcal{I}}(V)$;
- **Modularity**: $\mu_{\mathcal{I}}(U \cap V) + \mu_{\mathcal{I}}(U \cup V) =$
  $\mathcal{I}(\mathbb{1}_{U\cap V}) + \mathcal{I}(\mathbb{1}_{U\cup V}) = \mathcal{I}(\mathbb{1}_{U\cap V} + \mathbb{1}_{U\cup V})$ (as $\mathcal{I}$ additive).
  Hence, by Lemma 3, $\mu_{\mathcal{I}}(U \cap V) + \mu_{\mathcal{I}}(U \cup V) = \mathcal{I}(\mathbb{1}_U + \mathbb{1}_V)$
  Then, as $\mathcal{I}$ is additive, $\mu_{\mathcal{I}}(U \cap V) + \mu_I(U \cup V) = \mu_{\mathcal{I}}(U) + \mu_{\mathcal{I}}(V)$;
- **Probability**: as $\mathcal{I}$ is an integral, $\mu_{\mathcal{I}}(A) = \mathcal{I}(\mathbb{1}_A) \leq \mathcal{I}(1_{mf(A)}) \leq 1$;
- **Continuity**: Let $f : \mathbb{N} \to mf(A)$ in increasing sequence.
  By definition, $\mu_{\mathcal{I}}(\sup(\lambda n.f(n))) = \mathcal{I}(\mathbb{1}_{\sup(\lambda n.f(n))})$.
  We prove that: $\mathbb{1}_{\sup(\lambda n.f(n)))} = \sup(\lambda n.\mathbb{1}_{f(n)})$ and by continuity of $\mathcal{I}$:
  $\mathcal{I}(\sup(\lambda n.\mathbb{1}_{f(n)})) \leq \sup(\lambda n.\mathcal{I}(\lambda n.\mathbb{1}_{f(n)})) = \sup(\lambda n.\mu_{\mathcal{I}}(f(n)))$.

From valuation to integral Lower integrals directly generalize valuations. This allows us to define probabilistic programs as lower integrals and to derive valuations from these integrals, which is possible by Theorem 1. To construct the multiplication of the monad, one usually defines an integral $\mathcal{I}_\mu$ from a valuation μ. In classical integration theory, this corresponds to the construction of Lebesgue's integral using simple functions. We formalize some basic constructions in the Riesz's Theorem, following [13] we define:

Definition 5 *Let $f \in mf(A)$, $q \in \mathbb{Q}_+$. We define the open subset $\mathcal{D}(f,q) \in OS(A)$ as:*

$$\mathcal{D}(f,q) := \lambda x.(q < f(x)).$$

To complete a formal proof of Riesz theorem, we would then define $\mathcal{I}_\mu(f)$ using a rational subdivision of the codomain:

$$\mathcal{I}_\mu(f) = \sup_{n\in\mathbb{N}} \left(\frac{1}{n} \sum_{i=0}^{n} \mu\left(\mathcal{D}(f, \frac{i}{n})\right)\right).$$

This sequence is directed, by least common multiple, but not increasing.

One can prove the following bijection both for lower reals and for Dedekind reals, using the aforementioned references as template.

$$(\forall \mathcal{J}, \mathcal{I}_{\mu_{\mathcal{J}}} = \mathcal{J}) \wedge (\forall \nu, \mu_{\mathcal{I}_\nu} = \nu).$$

However, we have taken a slightly different route and formalized the Giry monad as a (haskell-style) Kleisli triple, where instead of the using valuations we take the (more general) lower integrals. This connects neatly with the interpretation of a probabilistic programming language:

5 A probabilistic language

In this section, we describe how we export some features of the ALEA library to our constructive formalism, by providing a similar monadic construction.

5.1 Formalization of a Giry monad on integrals

From Riesz' Theorem, one obtains a monad on the category $hSet$:

- unit: $\eta_x(u) := \delta_x(u)$
- bind:
$$\mu_\varphi(u) := \underline{\int_{s\in M(X)}} s(u)\, d\varphi(s).$$

The Dirac δ-function reduces to $\delta_x(u) := u(x)$, since $u : A \to \mathbb{S}$. Interestingly, we prove the monad axioms directly for integrals without the need to go via valuations. More precisely, valuations are the restriction of integrals to the functions $A \to \mathbb{S}$. Hence, we get a construction which is very similar to the one in the ALEA library [3]:

Definition 6 *We define the following probability Kleisli triple* $(\mathcal{M}, unit, bind)$*:*

- $\mathcal{M}(A : hSet) := IntPos(A)$ *;*
- $unit(A : hSet) : A \to IntPosA := \lambda\ x : A.\ \lambda\ f : mf(A) \Rightarrow f(x)$ *;*
- $bind(A : hSet)(B : hSet) : IntPos(A) \to (A \to IntPosB) \to IntPosB :=$ $\lambda\ \mathcal{I} \Rightarrow \lambda\ \mathcal{F} \Rightarrow (\lambda\ f : mf(A) \Rightarrow \mathcal{I}(\lambda\ x : A \Rightarrow (\mathcal{F}(x))(f)))$.

5.2 Interpretation of a probabilistic language

Like in ALEA, we can now use the monad to interpret *Rml* using Moggi's computational λ-calculus, because hSet is Cartesian closed, by functional extensionality. This is in contrast with the semantics in, e.g., the category of Polish spaces which is not Cartesian closed. Moreover, since the Kleisli category is ω-cpo enriched, as we use subprobability valuations, we can interpret fixed points and recursive functions. Like ALEA, we use a shallow embedding of a programming language in Coq: it includes conditional (if), local binding and application of functions:

- $v \rightsquigarrow unit(v)$;
- let $x = a$ in $b \rightsquigarrow bind\ [\![a]\!]\ (\lambda\ x.[\![b]\!])$;
- $f(e_1, \ldots ,e_n) \rightsquigarrow bind\ [\![e_1]\!]\ (\lambda\ x_1 \ldots\ bind\ [\![e_n]\!]\ (\lambda\ x_n.[\![f]\!](x_1, \ldots ,x_n))$;
- if b then c_1 else $c_2 \rightsquigarrow bind\ [\![b]\!]\ (\lambda\ x : bool$. if x then $[\![c_1]\!]$ else $[\![c_2]\!])$;
- let rec $f(x) = e \rightsquigarrow fix\ (\lambda\ [\![f]\!] \Rightarrow \lambda\ x \Rightarrow [\![e]\!])$.

5.3 Probabilistic programming

As illustration, we have defined a few programs from the ALEA library such as the coin flip, the random walk and the random number generator.

Coin flip We define the coin flip program as a lower integral on **bool**, which is proved to be an $hSet$ in the HoTT library.

```
Definition flip : IntPos bool.
exists (fun f : mf bool ⇒ (1/2) ∗ (f true) + (1/2) ∗ (f false)).
(* proof that it is actually a lower integral *) Defined.
```

The coin flip program is defined as the convex combination of two point masses. We prove that the convex combination of two integrals is an integral (if $\mathcal{I}$ and $\mathcal{J}$ are two integrals, and $p, q \in \mathbb{Q}_+$ such that $p+q \leq 1$ then $p\mathcal{I} + q\mathcal{J}$ is an integral). We verify some properties of `flip` in order to check that the definition is correct.

Random number Similarly, we define the random sampling of a natural number between 0 and a given positive number N. It is the uniform distribution on $[0, N]$ presented as a lower integral on **nat**.

Random walk As we have recursive functions, we can build a random walk program for flip coin as in the ALEA library [3]:

```
let rec walk x = if flip() then x else walk (S x)
```

5.4 Towards continuous measures

Perhaps the easiest way to obtain the uniform valuation on the unit interval is via Riesz' theorem and the Riemann integral on the Dedekind valued continuous functions. The latter has been formalized in Coq before HoTT [30,28]. This can be connected to the current development when all functions $X \to \mathbb{R}_D$ are uniformly continuous. This is not directly provable from our axioms for synthetic topology, as the unit interval is not compact in Kleene's first realizability model. This makes recursive measure theory counter intuitive, for instance, there is a cover of the unit interval which has size less than $\frac{1}{2}$; see [11]. However, in our intended models such as Kleene's second realizability model, the unit interval *is* compact. So we can add this as an axiom (this is a consequence of Brouwer's 'fan theorem'). This has not been formalized in Coq.

5.5 Discussion about the formalization

Our development profited from the organization of the HoTT-library [6]. For instance, the rigorous development's rules and naming conventions were quite useful. However, there is room for improvement of support for HoTT in Coq, as already discussed in [6]. For instance, HoTT cannot reuse very basic parts of the Coq standard library such as automation for rational numbers, including field equalities and linear inequalities. Since these parts are decidable this is a proof engineering issue, not a mathematical one.

6 Computability

Implementing probabilistic programs in a host functional programming language, like in ALEA, has a long history, e.g. [32], and has recently shown that it can be as efficient as special purpose implementations [35]. Since Coq is evolving towards an efficient dependently typed programming language [10,2], it is an interesting choice of host language. Currently, speed is not our main aim though.

For probabilistic languages there are two types of semantics, a denotational one based on classical measure theory and an operational one based on the monadic meta-language embedded in a functional host language. Our formalization has both features, as we will now explain.

6.1 Denotational semantics

We provide a *computable* semantics based on computable valuations using realizability, which we present through the internal language.

Not only have we developed the new semantics in the internal language, but also formalized large parts of it in Coq. To do so, we have used axioms from both synthetic topology and homotopy type theory. This means that we no longer have a guarantee that our evaluation terminates in Coq. However, there are implementations of these axioms in NuPrl [31] and cubical [12], respectively. Moreover, it is reasonable to expect that these features can be combined. One approach[5] implements the cubical model in NuPrl. An alternative would be to add the theory of names and effects from NuPrl to the cubical proof assistant in a way similar to the addition of guarded recursion to cubical [9]. So, we improve on ALEA which uses an axiomatized unit interval with both suprema and an inverse $(1 - \_)$ operation without an (obvious) constructive model. This brings us to the next semantics.

6.2 Operational sampling semantics

At the same time, our work is similar to the work on embedded DSLs in functional languages.

If a program p contains randomness from only a type T, then the semantics is a valuation on T. Hence, we obtain a program which can semi-decide questions of the form $[\![p]\!](f) > r$, when $f : T \to \mathbb{R}_l$, where r is a rational number. Based on this we can provide a simple sampling semantics for probabilistic programs; not unlike the one in e.g. [35]. A sampler for a distribution on a type T takes a seed and returns an element of type T, usually using a pseudorandom generator in the process. For a discrete type, an element induces an open: $f : T \to \mathbb{S}$ which we can use as input for the semantics.

For non-discrete types, we can randomly sample a (basic) open. For example, for the unit interval, these would be rational intervals. This would allow us to use a similar method as in the discrete case.

[5] `http://www.math.ias.edu/vladimir/files/Bickford_Slides.pdf`

Generally, the valuations on continuous datatypes will only give lower reals as outputs. That this is the best possible can be seen by considering point masses. If we could compute the Dirac measure, δ_x, on the unit interval, we would be able to decide whether an open set contains x, i.e. we would be able to decide equality on the unit interval. This is impossible. In case we limit recursion to obtain terminating functions *and* restrict to a class of ‘compact regular’ spaces/types one may expect a stronger result when integrating a (continuous) function with respect to the measure $[\![p]\!]$, since in that case, the value of an integral is a Dedekind real, not just a lower real.

It is clear that this operational semantics, based on Coq’s operational semantics and the partiality monad, is far from being practical. However, it is already an improvement over ALEA, which only provides an axiomatic semantics.

7 Discussion

7.1 Non-measurable functions

Adding classical logic ($\Sigma = \mathbf{2}$) a valuation on $[0,1]$ assigns a measure to *all* its subsets. The Vitali set shows that this contradicts the axiom of choice. Fortunately, continuously, the elements of $[0,1] \to \Sigma$ are precisely the opens, hence we can model the Lebesgue valuation — the uniform distribution on $[0,1]$.

On the other hand, valuations on, for instance, compact Hausdorff spaces, are in bijective correspondence with regular measures; see e.g. [25]. Hence, we capture one of the standard categories of spaces for the Giry monad. This puts our work in the *structural* approach to probability theory. We have a very good category of probability spaces, including the unit interval, while avoiding set theoretic anomalies.

7.2 Presheaves

There is an interesting analogy with the semantics for higher order probabilistic programming in [40,20]. They first consider a fairly standard model for first-order probabilistic computation, say, the Giry monad on standard Borel spaces. This category is not Cartesian closed, so to model function types, they use a variant of the Yoneda embedding.

A similar problem exists in synthetic topology, the category **Top** is not Cartesian closed. A common solution is to consider a convenient super-category. Escardó [14, Ch10] presents a number of subcategories of presheaves over **Top** for this purpose. In our case, it is more natural to consider the *sheaves* for the open cover topology and, in fact, we could take a ”big topos” on a topological site [16]. In this light, one could consider our construction as first completing with (dependent) function types and then defining the monad on the bigger category.

8 Conclusions and future work

We have combined homotopy type theory and synthetic topology to provide a new axiomatic semantics for probabilistic computation. This simplifies the ALEA library by the use of quotients and functional extensionality from HoTT and allows the addition of continuous data types. Our main insight is the extension of the Giry monad from locales to synthetic topology. We have formalized most of the constructions[6]. Presently, we have more than **5.500** LOC consisting of the main constructions and definitions. For instance, we have a theory of the lower reals, of integrals and valuations, and their ω-cpo structure. On top of this we interpret some probabilistic programs. The discrete parts of the ALEA library, e.g. binomial coefficients, can be ported to the HoTT library in the same way it has been done for other discrete mathematics.

The ALEA library forms the basis for the Certicrypt system [8] to verify security protocols. Easycrypt proof assistant, its successor, is still based on the same ideas. Sato [34] extends the semantics of easycrypt's apWhile language with continuous data types, using a classical meta-theory. Since we have extended ALEA, which underlies easycrypt, with continuous data types, we conjecture that our work can be used as a framework for the formalization of such semantics.

We have followed ALEA's axiomatic semantics for *Rml*. It would also be interesting to deeply embed *Rml* into Coq. This would make it possible to formally connect an operational and the denotational semantics. Huang and Morrisett [23] use *computable* distributions as the semantics of the probabilistic programming language AugurV2. Discussion between Huang and Spitters shows that their semantics is in fact connected to ours by realizability [22, sec 4.8]. This opens the way to use our work for verification of parts of their compiler.

Acknowledgements. The questions in this paper originated from discussions with Christine Paulin in 2014, when Spitters held a Digiteo chair at LRI, Inria. We also benefited from Faissole's internship with Paulin about lower reals in Coq. We are grateful for both.

References

1. Thorsten Altenkirch, Nils Anders Danielsson, and Nicolai Kraus. Partiality, revisited - the partiality monad as a quotient inductive-inductive type. In *FOSSACS 2017*, pages 534–549, 2017. `doi:10.1007/978-3-662-54458-7_31`.
2. Abhishek Anand, Andrew W Appel, Greg Morrisett, Zoe Paraskevopoulou, Randy Pollack, Olivier Savary Bélanger, Matthieu Sozeau, and Matthew Weaver. Certicoq: A verified compiler for coq. In *CoqPL'17*, 2017.
3. Philippe Audebaud and Christine Paulin-Mohring. Proofs of randomized algorithms in Coq. In *MPC*, 2006.
4. Steven Awodey and Andrej Bauer. Sheaf toposes for realizability. *Archive for Mathematical Logic*, 47(5):465–478, 2008. `doi:10.1007/s00153-008-0090-6`.

[6] `https://github.com/FFaissole/Valuations/`

5. Andrej Bauer. Realizability as the connection between computable and constructive mathematics. In *Proceedings of CCA*, 2005.
6. Andrej Bauer, Jason Gross, Peter LeFanu Lumsdaine, Michael Shulman, Matthieu Sozeau, and Bas Spitters. The hott library: A formalization of homotopy type theory in coq. In *Proceedings of the 6th ACM SIGPLAN Conference on Certified Programs and Proofs*, CPP 2017, pages 164–172. ACM, 2017. URL: `http://doi.acm.org/10.1145/3018610.3018615`, `doi:10.1145/3018610.3018615`.
7. Andrej Bauer and Davorin Lesnik. Metric spaces in synthetic topology. *Ann. Pure Appl. Logic*, 163:87–100, 2012.
8. Santiago Zanella Béguelin. *Formal certification of game-based cryptographic proofs.* PhD thesis, 2010.
9. Lars Birkedal, Aleš Bizjak, Ranald Clouston, Hans Bugge Grathwohl, Bas Spitters, and Andrea Vezzosi. Guarded Cubical Type Theory. *ArXiv:1611.09263*, 2016.
10. Mathieu Boespflug, Maxime Dénès, and Benjamin Grégoire. Full reduction at full throttle. In *Certified Programs and Proofs*. Springer, Springer, 2011. URL: `http://hal.inria.fr/hal-00650940`, `doi:10.1007/978-3-642-25379-9\_26`.
11. Douglas Bridges and Fred Richman. *Varieties of constructive mathematics*, volume 97. Cambridge University Press, 1987.
12. Cyril Cohen, Thierry Coquand, Simon Huber, and Anders Mörtberg. Cubical type theory: a constructive interpretation of the univalence axiom. *Proc. Types for Proofs and Programs (TYPES 2015)*, 2016.
13. Thierry Coquand and Bas Spitters. Integrals and valuations. *Journal of Logic and Analysis*, 1(3):1–22, 2009. `doi:10.4115/jla.2009.1.3`.
14. Martín Escardó. Synthetic topology: of data types and classical spaces. *ENTCS*, 87:21–156, 2004.
15. Martín Escardó and Cory Knapp. Partial elements and recursion via dominances in univalent type theory. 2017. URL: `http://www.cs.bham.ac.uk/~mhe/papers/partial-elements-and-recursion.pdf`.
16. Michael Fourman. Continuous truth II: Reflections. In *WoLLIC*, 2013.
17. Geätan Gilbert. Formalising real numbers in homotopy type theory. *CPP'17*, 2016.
18. Michele Giry. A categorical approach to probability theory. In *Categorical aspects of topology and analysis*, pages 68–85. 1982.
19. Georges Gonthier and Assia Mahboubi. An introduction to small scale reflection in Coq. *Journal of Formalized Reasoning*, 3(2):95–152, 2010.
20. Chris Heunen, Ohad Kammar, Sam Staton, and Hongseok Yang. A convenient category for higher-order probability theory. *CoRR*, abs/1701.02547, 2017.
21. Johannes Hölzl, Andreas Lochbihler, and Dmitriy Traytel. A formalized hierarchy of probabilistic system types - proof pearl. In Christian Urban and Xingyuan Zhang, editors, *ITP*, volume 9236 of *LNCS*, pages 203–220. Springer, 2015.
22. Daniel Huang. *On Programming Languages for Probabilistic Modeling.* PhD thesis, Harvard, 2017.
23. Daniel Huang and Greg Morrisett. An application of computable distributions to the semantics of probabilistic programming languages. In *European Symposium on Programming Languages and Systems*, pages 337–363. Springer, 2016.
24. Martin Hyland. First steps in synthetic domain theory. In *Category Theory*, pages 131–156. Springer, 1991.
25. Claire Jones and Gordon Plotkin. A probabilistic powerdomain of evaluations. In *LICS*, 1989.
26. Stephen Kleene and Richard Vesley. *The foundations of intuitionistic mathematics*, volume 1. North-Holland, 1965.

27. Robbert Krebbers and Bas Spitters. Type classes for efficient exact real arithmetic in Coq. *LMCS*, 9(1:1):1–27, 2013. `doi:10.2168/LMCS-9(1:01)2013`.
28. Evgeny Makarov and Bas Spitters. The Picard algorithm for ordinary differential equations in Coq. In Blazy, Paulin-Mohring, and Picardie, editors, *Interactive Theorem Proving (ITP2013)*, pages 463–468. Springer, 2013.
29. Russell O'Connor. *Certified Exact Transcendental Real Number Computation in Coq*, pages 246–261. Springer, 2008. `doi:10.1007/978-3-540-71067-7_21`.
30. Russell O'Connor and Bas Spitters. A computer verified, monadic, functional implementation of the integral. *TCS*, 411(37):3386–3402, 2010. `doi:10.1016/j.tcs.2010.05.031`.
31. Vincent Rahli and Mark Bickford. A nominal exploration of intuitionism. In *CPP*, 2016.
32. Norman Ramsey and Avi Pfeffer. Stochastic lambda calculus and monads of probability distributions. In *ACM SIGPLAN Notices*, volume 37, pages 154–165. ACM, 2002.
33. Egbert Rijke and Bas Spitters. Sets in homotopy type theory. In *MSCS, special issue: From type theory and homotopy theory to univalent foundations*, 2014. `arXiv:1305.3835`.
34. Tetsuya Sato. Approximate relational hoare logic for continuous random samplings. *ENTCS*, 325:277–298, 2016.
35. Adam Ścibior, Zoubin Ghahramani, and Andrew Gordon. Practical probabilistic programming with monads. In *ACM SIGPLAN Notices*, volume 50, pages 165–176. ACM, 2015.
36. Michael Shulman. Brouwer's fixed-point theorem in real-cohesive homotopy type theory. *arXiv:1509.07584*, 2015.
37. Michael Shulman. Homotopy type theory: the logic of space. In Gabriel Catren and Mathieu Anel, editors, *New spaces for mathematics and physics*. 2016. To appear.
38. Matthieu Sozeau. A new look at generalized rewriting in type theory. *Journal of Formalized Reasoning*, 2(1):41–62, 2010.
39. Bas Spitters and Eelis van der Weegen. Type classes for mathematics in type theory. *MSCS, special issue on 'Interactive theorem proving and the formalization of mathematics'*, 21:1–31, 2011. `doi:10.1017/S0960129511000119`.
40. Sam Staton, Hongseok Yang, Chris Heunen, Ohad Kammar, and Frank Wood. Semantics for probabilistic programming: higher-order functions, continuous distributions, and soft constraints. *CoRR*, abs/1601.04943, 2016.
41. The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations for Mathematics*. `http://homotopytypetheory.org/book`, Institute for Advanced Study, 2013.
42. Steven Vickers. A monad of valuation locales. 2011. URL: `http://www.cs.bham.ac.uk/~sjv/Riesz.pdf`.
43. Klaus Weihrauch. *Computable analysis: an introduction*. 2012.