A Stochastic Lambda-Calculus for Probabilistic Programming

(Preliminary Report)

Radu Mardare,

Prakash Panangaden,

and

Dana Scott

Logical Structures in Computation
Reunion Workshop
December 2017

Randomizing a Model

Definition. Let $\langle A, \bullet \rangle$ be an ***abstract algebra*** with a binary operation, where A is a ***topological space***, and where $\bullet$ is ***continuous***. Suppose Ω is a ***probability space***, and let A^{Ω} be the set of ***measurable functions*** from the sample space of Ω into the set A. Define $(\alpha \circ \beta)(\omega) = \alpha(\omega) \bullet \beta(\omega)$ as an ***operation*** on A^{Ω}.

Proposition. The algebras $\langle A, \bullet \rangle$ and $\langle A^{\Omega}, \circ \rangle$ satisfy the same equations, even when, in the second algebra, equality is ***equality almost everywhere***.

- **Dana S. Scott.** *A proof of the independence of the continuum hypothesis*. **Mathematical Systems Theory**, vol. 1 (1967), pp. 89–111.
- **Dexter Kozen.** *Semantics of Probabilistic Programs*. **Journal of Computer and System Sciences**, vol. 22 (1981), pp. 328–350.
- **H. Jerome Keisler.** *Randomizing a Model*. **Advances in Mathematics**, vol. 143 (1999), pp. 124–158.
- **Dana S. Scott.** *Stochastic λ-calculi: An extended abstract*. **Journal of Applied Logic**, vol. 12 (2014), pp. 369–376.

Boolean-Valued Models

Definition. Let $\langle A, R \rangle$ be a relational structure, and let A^Ω be the set of measurable functions with values in A.
Give formulae Boolean-valued ***semantics*** in the algebra of measurable sets modulo Null sets:

$$[\![\alpha\, R^\Omega\, \beta]\!] = \{\omega \in A^\Omega \mid \alpha(\omega)\, R\, \beta(\omega)\} / \text{Null}$$

$$[\![\alpha = \beta]\!] = \{\omega \in A^\Omega \mid \alpha(\omega) = \beta(\omega)\} / \text{Null}$$

$$[\![\Phi \wedge \Psi]\!] = [\![\Phi]\!] \wedge [\![\Psi]\!]$$

$$[\![\Phi \vee \Psi]\!] = [\![\Phi]\!] \vee [\![\Psi]\!]$$

$$[\![\Phi \rightarrow \Psi]\!] = [\![\Phi]\!] \rightarrow [\![\Psi]\!]$$

$$[\![\exists x.\Phi(x)]\!] = \bigvee_{\alpha \in A^\Omega} [\![\Phi(\alpha)]\!]$$

$$[\![\forall x.\Phi(x)]\!] = \bigwedge_{\alpha \in A^\Omega} [\![\Phi(\alpha)]\!]$$

Theorem. Under very mild assumptions, the Boolean-valued structure $\langle A^\Omega, R^\Omega \rangle$ will satisfy the same first-order formulae as $\langle A, R \rangle$.

Church's λ-Calculus

Definition. λ-calculus — as a formal theory — has rules for the ***explicit definition*** of functions *via* well known equational axioms:

α-conversion

$$\lambda X.[\ldots X\ldots] = \lambda Y.[\ldots Y\ldots]$$

β-conversion

$$(\lambda X.[\ldots X\ldots])(T) = [\ldots T\ldots]$$

η-conversion

$$\lambda X.F(X) = F$$

NOTE: The third axiom will be dropped in favor of a theory employing properties of a partial ordering.

- **F. Cardone and J.R. Hindley.** *Lambda-Calculus and Combinators in the 20th Century*. In: Volume 5, pp. 723-818, of **Handbook of the History of Logic**, Dov M. Gabbay and John Woods (eds.), North-Holland/Elsevier Science, 2009.

Using Integers as Data

$\mathbb{N} = 1 + \mathbb{N}$, **the basic integers;**

$\mathbb{N} = 1 + \mathbb{N} \times \mathbb{N}$, **integers as binary trees;**

$\mathbb{N} = \mathbb{N}^*$, **integers as finite sequences;**

$\mathbb{N} = \mathcal{P}_f(\mathbb{N})$, **integers as finite sets.**

Definitions. (1) ***Pairs:*** $(n,m) = 2^n(2m+1)$.

(2) ***Sequences:*** $\langle\rangle = 0$ and $\langle n_0, \ldots, n_{k-1}, n_k \rangle = (\langle n_0, \ldots, n_{k-1} \rangle, n_k)$.

(3) ***Terms:*** $\mathbf{term}(0) = \varnothing$ and $\mathbf{term}((n,m)) = \mathbf{term}(n) \cup \{m\}$.

(4) ***Kleene star:*** $X^* = \{n \mid \mathbf{term}(n) \subseteq X\}$, for sets $X \subseteq \mathbb{N}$.

(5) ***Finite sets:*** $\mathbf{set}(0) = \varnothing$ and $\mathbf{set}((n,m)) = \{n\} \cup \{n+1+k \mid k \in \mathbf{set}(m)\}$.

NOTE: Because such numbering within $\mathbb{N}$ is so easy, there is no need for being more abstract.

Enumeration Operators as a Model

Definition. The ***λ-calculus model*** is formed using the powerset $\mathcal{P}(\mathbb{N}) = \{\, X \mid X \subseteq \mathbb{N} \,\}$ with a binary operation of ***application***:

Application:

$$F(X) = \{\, m \mid \exists n.\ \mathbf{set}(n) \subseteq X \ \&\ (n,m) \in F \,\}$$

This binary operation further permits a ***reverse*** procedure:

Abstraction:

$$\lambda X.[\ldots X \ldots] = \{\, (n,m) \mid m \in [\ldots \mathbf{set}(n) \ldots] \,\}$$

- Richard M. Friedberg and Hartley Rogers Jr., *Reducibility and completeness for sets of integers*, **Mathematical Logic Quarterly**, vol. 5 (1959), pp. 117-125. Some earlier results are presented in an abstract in **The Journal of Symbolic Logic**, vol. 22 (1957), p. 107.
- Hartley Rogers, Jr., **Theory of Recursive Functions and Effective Computability**, McGraw-Hill, 1967, xix + 482 pp.

What is the Secret?

(1) The powerset $\mathcal{P}(\mathbb{N}) = \{\, X \mid X \subseteq \mathbb{N} \,\}$ is a ***topological space*** with the sets $\mathcal{U}_n = \{\, X \mid \mathbf{set}(n) \subseteq X \,\}$ as a ***basis*** for the topology.

(2) Functions $\Phi : \mathcal{P}(\mathbb{N})^n \longrightarrow \mathcal{P}(\mathbb{N})$ are ***continuous*** iff, for all $m \in \mathbb{N}$, we have $\mathbf{set}(m) \subseteq \Phi(X_0, X_1, \ldots, X_{n-1})$ iff there exist $\mathbf{set}(k_i) \subseteq X_i$ for each $i<n$, such that $\mathbf{set}(m) \subseteq \Phi(\mathbf{set}(k_0), \mathbf{set}(k_1), \ldots, \mathbf{set}(k_{n-1}))$.

(3) The application operation $F(X)$ is continuous as a function of ***two*** variables.

(4) If the function $\Phi(X_0, X_1, \ldots, X_{n-1})$ is continuous, then the abstraction term $\lambda X_0 . \Phi(X_0, X_1, \ldots, X_{n-1})$ is continuous in all of the ***remaining variables***.

(5) If $\Phi(X)$ is continuous, then $\lambda X . \Phi(X)$ is the ***largest set*** F such that for all sets T, we have $F(T) = \Phi(T)$. And, therefore, generally $F \subseteq \lambda X . F(X)$.

NOTE: This model could easily have been defined in 1957!!
It clearly satisfies the rules of α, β-conversion (but not η).

Some Lambda Properties

Theorem. For all sets of integers F and G we have:

$$\lambda X.F(X) \subseteq \lambda X.G(X) \text{ iff } \forall X.F(X) \subseteq G(X),$$

$$\lambda X.(F(X) \cap G(X)) = \lambda X.F(X) \cap \lambda X.G(X),$$

and

$$\lambda X.(F(X) \cup G(X)) = \lambda X.F(X) \cup \lambda X.G(X).$$

Definition. A continuous operator $\Phi(X_0, X_1, \ldots, X_{n-1})$ is ***computable*** iff in the model this set is **RE**:

$$F = \lambda X_0 \lambda X_1 \ldots \lambda X_{n-1}.\Phi(X_0, X_1, \ldots, X_{n-1}).$$

How to do Recursion?

Three Basic Theorems.

- All pure λ-terms define ***computable*** operators.
- If $\Phi(X)$ is continuous and if we let $\nabla = \lambda X.\Phi(X(X))$, then the set $P = \nabla(\nabla)$ is the ***least fixed point*** of Φ.
- The least fixed point of a ***computable*** operator is computable.

A Principal Theorem. These computable operators:

$$\textbf{Succ}(X)=\{n+1 \mid n \in X\},$$

$$\textbf{Pred}(X)=\{n \mid n+1 \in X\}, \textbf{ and}$$

$$\textbf{Test}(Z)(X)(Y)= \{n \in X \mid 0 \in Z\} \cup \{m \in Y \mid \exists k.k+1 \in Z\},$$

together with λ-calculus, suffice for defining ***all RE sets***.

NOTE: So far we have a pure functional programming language for Recursion Theory. For probabilistic programming more is needed.

How to Randomize $\mathcal{P}(\mathbb{N})$?

Definition. Suppose Ω is a probability space, and let $\mathcal{P}(\mathbb{N})^{\Omega}$ be the set of measurable functions (random variables) from Ω into the topological space $\mathcal{P}(\mathbb{N})$.

Theorem. $\mathcal{P}(\mathbb{N})^{\Omega}$ forms a ***Boolean-valued model*** for the λ - calculus — expanding the two-valued model $\mathcal{P}(\mathbb{N})$.

NOTE: The random variables are closed under application.

We then define the first-order structure by:

$$[\![\mathbf{X} \subseteq \mathbf{Y}]\!] = \{ t \in \Omega \mid \forall n \in \mathbf{X}(t).\ n \in \mathbf{Y}(t) \} / \mathrm{Null}.$$

The validity of the first order properties is then automatic.

How to do Probabilistic Programming?

(1) The elements of $\mathcal{P}(\mathbb{N})$ are the ***non-random*** or ***stable*** objects.

(2) Suitable elements of $\mathcal{P}(\mathbb{N})^{\Omega}$ can function as ***random oracles.***

(3) For example, **C :** $\Omega \to \{\{0\},\{1\}\}$ represents a ***coin toss.***

(4) Objects **T** of $\mathcal{P}(\mathbb{N})^{\Omega}$ can also represent ***sequences*** of coins.

(5) A sequence of ***fair*** and ***mutually independent*** coins is a ***tossing***.

(6) A stable algorithm can take a tossing as its ***oracle***.

NOTE: In order to mimic probabilistic programming in the conventional sense, the application of the tossing in this model has to be controlled to be used in a specific order, and no coin can be used more than once.

This can be achieved by using a continuation-passing style style of semantics.

Axioms for Stochastic λ-Calculus

(α-conversion)	$\lambda x.M = \lambda y.M\{y/x\}$;
(β-reduction)	$(\lambda x.M)(N) = M\{N/x\}$, for M and N classical terms;
(Commutativity)	$M \oplus N = N \oplus M$;
(Idempotence)	$M \oplus M = M$;
(L-distributivity)	$(M_1 \oplus M_2)(N) = M_1(N) \oplus M_2(N)$;
(R-distributivity)	$N(M_1 \oplus M_2) = N(M_1) \oplus N(M_2)$;
(λ-distributivity)	$\lambda x.(M_1 \oplus M_2) = \lambda x.M_1 \oplus \lambda x.M_2$;
(Entropic equality)	$(M_1 \oplus M_2) \oplus (N_1 \oplus N_2) = (M_1 \oplus N_1) \oplus (M_2 \oplus N_2)$;
(Fixed point)	$\mu x.M = (\lambda x.M)(\mu x.M)$;
(Recursive choice)	$\mu x.(x \oplus M) = \mu x.M$.

NOTE: We now prefer to call terms without any occurrence of ⊕ "stable terms" rather than "classical terms"; however, the fixed-point operator μ must be carefully introduced to be stable.

To have simple rules the Boolen-valued logic has to be invoked to make the meaning of equations independent of the choice of a tossing as the oracle.

Some Future Projects

(1) Explore using Boolean-valued logic in proving ***properties*** of programs.

(2) Expand the use of ***types*** in stable λ-calculus to stochastic λ-calculus.

(3) Show how the $\mathcal{P}(\mathbb{N})^{\Omega}$ model corresponds to ***operational*** semantics.

Stay Tuned!