Stochastic λ-Calculi: An Extended Abstract

Dana S. Scott

University Professor, Emeritus
Carnegie Mellon University

Visiting Scholar in Mathematics
University of California, Berkeley

Abstract. It is shown how the operators in the "graph model" for λ-calculus (which can function as a programming language for Recursive Function Theory) can be expanded to allow for "random combinators". The result then is a model for a new language for random algorithms.

1. Background. In the mid-1970s the author in **[1]** defined a "graph model" for the untyped **λ**-calculus making use of *enumeration operators* from Recursive Function Theory. It turned out that Gordon Plotkin had earlier defined in set-theoretical terms a closely related construction. In 1993 Plotkin published his previously un-published technical report along with extensive commentary in **[2]**. A full history of **λ**-calculus — along with expository material and many, many references — can be found in **[3]**. *Enumeration operators* had been defined in the mid-1950s independently by Myhill and Shepherdson in **[4]** and by Friedberg and Rogers in **[5]**. The book **[6]** by Rogers popularized the idea of *enumeration degrees*, a subject on which there is now a very large literature. Neither team in **[4]** or **[5]** seemed to realize they had sufficient mechanics for defining a model for **λ**-calculus, however.

Recently several authors have made proposals for defining non-deterministic extensions of the untyped **λ**-calculus — generally explained using *operational semantics* rather than models for a *denotational semantics*. See **[7]** and **[8]** for proposals and many references. (There are unfortunately too many to mention here in this brief article.) More discussion and references can also be found in the very recent papers of Michael Mislove referenced on his web page **[9]** which also reference other approaches.

2. Basic definitions. Let $\mathbb{N}$ be the set of ***natural numbers***, on which we can use a very easy to define ***pairing function***: $(n,m) = 2^n(2m+1)$. This puts the pairs of natural numbers into a one-one correspondence with the ***positive integers***. ***Sequence numbers*** can now be defined as:

$$\langle\,\rangle = 0 \quad \text{and} \quad \langle n_0, n_1, \ldots, n_{k-1}, n_k\rangle = (\langle n_0, n_1, \ldots, n_{k-1}\rangle, n_k).$$

This notation puts ***finite sequences*** of elements of $\mathbb{N}$ into a one-one correspondence with the ***whole*** of the set $\mathbb{N}$. ***Finite sets*** of natural numbers can be numbered as follows:

$$\mathbf{set}(0) = \varnothing \quad \text{and} \quad \mathbf{set}((n,m)) = \mathbf{set}(n) \cup \{m\}.$$

It then follows that:

$$\mathbf{set}(\langle n_0, n_1, \ldots, n_{k-1}, n_k\rangle) = \{n_0, n_1, \ldots, n_{k-1}, n_k\}.$$

There are many other popular ways of numbering (Gödel-numbering) pairs, sequences and sets, but the choice of a numbering system is not so very important as long as the functions are ***computable***. And to these notations we add the ***Kleene star***: $X^* = \{\, n \mid \mathbf{set}(n) \subseteq X \,\}$, which gives the set X^* of all (numbers of) finite sequences of elements of a set $X \subseteq \mathbb{N}$. Note that $\mathbb{N}^* = \mathbb{N}$ and $\varnothing^* = \{0\}$.

Definition 2.1. The ***enumeration operators*** are identified with the sets F of natural numbers which operate on sets of integers through the binary operation of ***application***:

$$F(X) = \{ m \mid \exists n \in X^*. (n,m) \in F \}.$$

The idea here is that, as you start enumerating the elements of the set X, you can also then enumerate the elements of the set X*. Along with these enumerations, you can also enumerate the pairs in F. Every time you see a match between a sequence number $n \in X^*$ and the first term of a pair $(n,m) \in F$, you then ***output*** m as an element of F(X). The enumerations of the sets F and X thus generate the enumeration of the set F(X).

Except for a small detail in the use of Gödel numbers, this is the same as the definition in **[6]**. And, following Friedberg and Rogers, we say that a set B is ***enumeration reducible*** to a set A just in case there is a ***recursively enumerable*** set F such that B = F(A). The emphasis here is that the ***computable*** enumeration operators are those given by recursively enumerable sets F.

Next, note that the general enumeration operators have an important connection to ***topology***. The powerset

$$\mathcal{P}(\mathbb{N}) = \{ X \mid X \subseteq \mathbb{N} \}$$

can be considered as being a topological space with the sets

$$\mathcal{Q}_n = \{ X \mid n \in X^* \}$$

as a ***basis*** for the topology. This could be called the ***positive topology***, because it works with only the ***positive facts*** as to which finite sets are in fact contained in a "point" $X \in \mathcal{P}(\mathbb{N})$.

Definition 2.2. We say that a function $\Phi : \mathcal{P}(\mathbb{N})^n \longrightarrow \mathcal{P}(\mathbb{N})$ of n-arguments is ***continuous*** in this topology iff (for all integers m) we have:

$$m \in \Phi(X_0, X_1, \ldots, X_{n-1}) \text{ iff there are } k_i \in X_i^* \text{ (for all } i<n) \text{ where } m \in \Phi(\mathbf{set}(k_0), \mathbf{set}(k_1), \ldots, \mathbf{set}(k_{n-1})).$$

In words we can paraphrase this definition as saying that a function Φ is continuous just in case a finite amount of information about the function value $\Phi(X_0, X_1, \ldots, X_{n-1})$ is exactly determined by a finite amount of information about the arguments $X_0, X_1, \ldots, X_{n-1}$.

With these definitions in hand we can next introduce the λ-calculus model.

3. Modeling λ-calculus. The Church-Curry calculus is an algebraic system of operators allowing for ***application*** of one object to another (in a type-free manner) together with a variety of ***definition schemes*** — either by use of ***λ-abstraction*** (Church) or by invoking special ***combinators*** (Curry). We shall discuss both. The modeling of the calculus will be in terms of enumeration operators on $\mathcal{P}(\mathbb{N})$. The first fact that justifies the modeling concerns our definition of application.

Theorem 3.1. *The application operation* F(X) *is continuous as a function of two variables on* $\mathcal{P}(\mathbb{N})$.

The proof is very easy and can be left as an exercise. The power of the modeling, however, comes from the next theorem.

Theorem 3.2. *For every continuous function* $\Phi : \mathcal{P}(\mathbb{N}) \longrightarrow \mathcal{P}(\mathbb{N})$ *there is a* ***largest*** *set* F *such that for all* $X \subseteq \mathbb{N}$ *we have* $\Phi(X) = F(X)$.

Proof: (Note that we write $\Phi(X)$ for ordinary (mathematical) function application on the left side of the equation, and we write $F(X)$ for the binary operation in $\mathcal{P}(\mathbb{N})$ on the right-hand side. As the defined binary operation gives us exactly the continuous functions, there is no need for two different notations.) Note that we can characterize the F we need as follows: $F = \{0\} \cup \{ (n,m) \mid m \in \Phi(\mathbf{set}(n)) \}$, because by the definition of continuity this set uniquely determines Φ. And, from the definition of application, for any other set G where for all $X \subseteq \mathbb{N}$ we have $G(X) = \Phi(X)$, we find $G \subseteq F$.

Definition 3.3. For any expression [...X...] defining a continuous function, we use the ***abstraction notation***:

$$\lambda X.\, [...X...] = \{0\} \cup \{ (n,m) \mid m \in [... \mathbf{set}(n)...] \}.$$

Theorem 3.4. *If* $\Phi(X_0, X_1, \dots, X_{n-1})$ *is continuous in all the variables, then the* $\lambda X_0.\Phi(X_0, X_1, \dots, X_{n-1})$ *is continuous in all of the* ***remaining*** *variables.*

Corollary 3.5. *If* $\Phi(X_0, X_1, \dots, X_{n-1})$ *is continuous in all the variables, then there is a largest set* F *such that* $F\,(X_0)(X_1)\dots(X_{n-1}) = \Phi(X_0, X_1, \dots, X_{n-1})$ *for all the* $X_0, X_1, \dots, X_{n-1} \in \mathcal{P}(\mathbb{N})$.

Note, therefore, that generally $F \subseteq \lambda X.F(X)$. And here are some other easy-to-prove properties.

Theorem 3.6. *For all sets of integers* F *and* G *we have:*

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

$$\lambda X.\, (F(X) \cap G(X)) = \lambda X.\, F(X) \cap \lambda X.\, G(X) \quad \textit{and} \quad \lambda X.\, (F(X) \cup G(X)) = \lambda X.\, F(X) \cup \lambda X.\, G(X).$$

The powerset $\mathcal{P}(\mathbb{N})$ is a ***complete lattice***, as is the space $\mathbf{Cont}[\mathcal{P}(\mathbb{N}), \mathcal{P}(\mathbb{N})]$ of all continuous functions. The above theorem shows that the mapping $F \longmapsto (X \longmapsto F(X))$ has some nice lattice-structure-preserving properties.

Definition 3.7. A continuous operator $\Phi(X_0, X_1, \dots, X_{n-1})$ is ***computable*** iff in $\mathcal{P}(\mathbb{N})$ the following set is ***recursively enumerable***:

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

From this it follows that all pure λ-terms (which correspond to Curry's ***combinators***, see **[3]**) define computable operators in the model. And this is in line with the use of enumeration operators in **[6]**. But a special feature of the model is the way of obtaining ***new*** computable operators from given ones.

Theorem 3.8. *If* $\Phi(X)$ *is continuous and we let* $\nabla = \lambda X.\, \Phi(X(X))$, *then* $P = \nabla(\nabla)$ *is the* ***least fixed point*** *of* Φ. *Moreover, the least fixed point of a computable operator is always computable.*

Proof: Suppose Φ: $\mathcal{P}(\mathbb{N}) \longrightarrow \mathcal{P}(\mathbb{N})$ is continuous. Let $\nabla = \lambda X.\Phi(X(X))$ and $P = \nabla(\nabla) = \Phi(P)$, so P is indeed a fixed point of Φ. Let $Q = \Phi(Q)$ be ***another*** fixed point. We have to prove $P \subseteq Q$ to show that P is in fact the least fixed point of Φ.

Assume $m \in P$. We want to show $m \in Q$. By continuity of application, we can write:

$$P = \nabla(\nabla) = \bigcup \{ \mathbf{set}(k)(\mathbf{set}(k)) \mid \mathbf{set}(k) \subseteq \nabla \}.$$

So, for some k with **set**(k) $\subseteq \nabla$, we have that m $\in$ **set**(k)(**set**(k)). If, using ***The Principle of Strong Induction***, we could prove that for all k, if **set**(k) $\subseteq \nabla$, then **set**(k)(**set**(k)) $\subseteq$ Q, we would be done. Hence, ***assume*** this implication is true for all n < k, and now ***prove*** it for k.

Well, suppose **set**(k) $\subseteq \nabla$, and that q $\in$ **set**(k)(**set**(k)). By the definition of application we must have (n,q) $\in$ **set**(k) and n $\in$ **set**(k)* for some n. Thus, n < k and **set**(n) $\subseteq$ **set**(k). Whence, **set**(n) $\subseteq \nabla$, and hence, **set**(n)(**set**(n)) $\subseteq$ Q, by the inductive assumption.

But, note that (n,q) $\in \nabla = \lambda X.\ \Phi(X(X))$, and therefore we have q $\in \Phi($**set**(n)(**set**(n))). However, because we proved **set**(n)(**set**(n)) $\subseteq$ Q , we also have q $\in \Phi($**set**(n)(**set**(n))) $\subseteq \Phi(Q) = Q$. So, we have established that **set**(k)(**set**(k)) $\subseteq$ Q, and the whole proof is finally complete.

Definition 3.9. The ***basic arithmetic combinators*** are the following three recursively enumerable sets:

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

Using standard methods of λ-calculus and knowledge of primitive recursive functions, we can show that the arithmetic combinators suffice for defining ***all*** recursively enumerable sets. Indeed, every such set can be put into the form **RE**({n}), where we have a computable **RE** = $\lambda X.$ **RE**(X) so that

(i) **RE**({0}) = $\lambda Y.\lambda X.\ Y$,

(ii) **RE**({1}) = $\lambda Z.\lambda Y.\lambda X.\ Z(X)(Y(X))$,

(iii) **RE**({2}) = **Test**,

(iv) **RE**({3}) = **Succ**,

(v) **RE**({4}) = **Pred**, and

(vi) **RE**({4 + (n,m)}) = **RE**({n})(**RE**({m})), for all n and m.

The above equations (i) – (vi) could be put into one, rather large, fixed-point equation in order to show that **RE** is indeed recursively enumerable, and we can think of the n in **RE**({n}) as the Gödel number of the n^{th} recursively enumerable set. The effort of making this explicit is nothing more than an exercise to show that our notation here can be used as a ***programming language*** for computable enumeration operators. The special example of **RE** is in effect a definition of the analogue of the ***Universal Turing Machine*** — without having to use Turing machines at all. And the approach can then lead to easy definitions of ***non-recursive sets*** and to finding ***recursively inseparable sets***. The author feels there could be some pedagogical value in developing basic recursive function theory in this way.

Aside from the space $\mathcal{P}(\mathbb{N})$, a theory of computability on ***other spaces*** might also be desired. As long as attention is to be given to topological spaces with a countable basis for a topology, it is not necessary to look much beyond $\mathcal{P}(\mathbb{N})$.

Theorem 3.10. *Every countably based T_0-space is homeomorphic to a subspace of $\mathcal{P}(\mathbb{N})$.*

Proof: A T_0-space is one where points are ***uniquely determined*** by the open sets to which they belong. Assume $\mathfrak{T}$ is a such a space, and let the $\mathcal{U}_n$ be the elements of a ***countable basis*** for the topology of the space $\mathfrak{T}$. Define a mapping $\varepsilon : \mathfrak{T} \longrightarrow \mathcal{P}(\mathbb{N})$ by the equation: $\varepsilon(X) = \{\ n \mid X \in \mathcal{U}_n\ \}$ on elements $X \in \mathfrak{T}$. By the very ***definition*** of being a T_0-space, it follows that the mapping ε is ***one-one***.

The ***basis*** for the topology of $\mathcal{P}(\mathbb{N})$ consists of the sets $\mathcal{Q}_n = \{ X \mid n \in X^* \}$. Thus, the sets of the special form $\mathcal{Q}_{\langle m \rangle} = \{ X \mid m \in X \}$ provide a ***subbasis*** for the topology. To prove that ε is continuous, we need only check that the inverse-image sets $\{ X \in \mathfrak{T} \mid \mathfrak{E}(X) \in \mathcal{Q}_{\langle m \rangle} \} = \mathcal{U}_m$ are open. But that is obvious.

Finally, to show that ε is ***bicontinuous***, we need to show that the ***image*** of an open set is open onto the range of the function. But $\mathfrak{E}(\mathcal{U}_m) = \{ \mathfrak{E}(X) \mid X \in \mathcal{U}_m \} = \mathfrak{E}(\mathfrak{T}) \cap \mathcal{Q}_{\langle m \rangle}$, which ***is indeed*** open.

So the space $\mathcal{P}(\mathbb{N})$ has a very wide variety of subspaces. But, how do continuous functions fare?

Definition 3.11. A space $\mathfrak{S}$ is called ***injective*** iff whenever $\mathfrak{T}$ is a subspace of another space $\mathfrak{U}$ and when $\Phi : \mathfrak{T} \longrightarrow \mathfrak{S}$ is continuous, then Φ can be ***extended*** to a continuous function $\Psi: \mathfrak{U} \longrightarrow \mathfrak{S}$.

Theorem 3.12. *The space $\mathcal{P}(\mathbb{N})$ is injective.*

Proof: Assume $\mathfrak{T}$ is a subspace of $\mathfrak{U}$ and $\Phi : \mathfrak{T} \longrightarrow \mathcal{P}(\mathbb{N})$ is continuous. The sets

$$\mathcal{T}_n = \{ X \in \mathfrak{T} \mid n \in \Phi(X) \}$$

must be open subsets of $\mathfrak{T}$. By the definition of subspace topology, we can write $\mathcal{T}_n = T \cap \mathcal{U}_n$, where the $\mathcal{U}_n$ are open in $\mathfrak{U}$. And indeed, we can make a ***canonical choice*** by taking the $\mathcal{U}_n$ as large as possible.

Now define $\Psi: \mathfrak{U} \longrightarrow \mathcal{P}(\mathbb{N})$ by the equation: $\Psi(Y) = \{ n \mid Y \in \mathcal{U}_n \}$. By definition, Ψ is continuous, and, for $X \in \mathfrak{T}$, we find: $\Psi(X) = \{ n \mid X \in \mathcal{T}_n \} = \{ n \mid n \in \Phi(X) \} = \Phi(X)$. In other words, Ψ ***extends*** Φ, as desired.

An immediate consequence of this result is that when $\mathfrak{T}$ and $\mathfrak{U}$ are ***subspaces*** of $\mathcal{P}(\mathbb{N})$, then every continuous function $\Phi : \mathfrak{T} \longrightarrow \mathfrak{U}$ is just the ***restriction*** of a continuous function $\Psi : \mathcal{P}(\mathbb{N}) \longrightarrow \mathcal{P}(\mathbb{N})$. In other words, not only do we have that wide range of subspaces of $\mathcal{P}(\mathbb{N})$, but the continuous functions between subspaces are already "known" from the continuous functions on $\mathcal{P}(\mathbb{N})$. Thus, we can say that topology is ***inherited*** directly from $\mathcal{P}(\mathbb{N})$, and the techniques of the λ-calculus can be applied directly.

We would also like to say that ***computability notions*** can also be inherited from $\mathcal{P}(\mathbb{N})$. This is true, but we have to take some care, because a particular space $\mathfrak{T}$ might have ***several*** homeomorphic embeddings into $\mathcal{P}(\mathbb{N})$ by different choices of bases giving us different notions of computability. Usually, in the author's experience, there is a preferred embedding, however, and then the notions of computability are well behaved and have desirable properties as expected.

4. Adding randomness. Turing suggested adding ***oracles*** to Turing machines to define computability ***relative to*** given, external, possibly non-recursive processes. In our formulation here we can use as oracles any ***arbitrary*** sets $A \in \mathcal{P}(\mathbb{N})$. That is, instead of a computable enumeration operator producing values $F(X)$, we can use a combination $F(A)(X)$, with F recursively enumerable and with A arbitrary, but fixed. But we do not have to stop here, the oracle can be endowed with properties of ***randomness*** by using the following completely standard definition from Probability Theory.

Definition 4.1. By a ***random variable*** in our model we understand a function on the ***unit interval***

$$\mathbb{X} : [0,1] \longrightarrow \mathcal{P}(\mathbb{N}),$$

where, for each $n \in \mathbb{N}$, the set $\{ t \in [0,1] \mid n \in \mathbb{X}(t) \}$ is always Lebesgue measurable.

We use the unit interval here as a ***standard*** probability space. Any other convenient probability space (along with its given measure) could have been used for this discussion

Theorem 4.2. *Given two random variables* $\mathbb{X}, \mathbb{Y} : [0,1] \longrightarrow \mathcal{P}(\mathbb{N})$, the ***application operation*** *defined by*

$$\mathbb{X}(\mathbb{Y})(t) = \mathbb{X}(t)(\mathbb{Y}(t))$$

produces for us again a random variable.

The easy proof of this theorem is completely analogous to the theorem that the the ***sum*** of two ***numerical*** random variables is again a random variable — because numerical addition on the real numbers is a continuous and, therefore, a ***Borel-measurable*** operation. The same goes for ***application*** on $\mathcal{P}(\mathbb{N})$ owing to continuity. ***Thus λ-calculus can be applied to random variables as well as to constants in*** $\mathcal{P}(\mathbb{N})$**.** This also means that a combination $F(\mathbb{A})(X)$ can be considered as a randomized algorithm when A is a random "oracle".

Note, too, that for random variables $\mathbb{X}, \mathbb{Y} : [0,1] \longrightarrow \mathcal{P}(\mathbb{N})$, ***equations*** between them can be assigned a well defined ***probability***, because an "event" such as

$$[\![\, \mathbb{X} = \mathbb{Y} \,]\!] = \{\, t \in [0,1] \mid \forall n \in \mathbb{X}(t).\ n \in \mathbb{Y}(t) \wedge \forall n \in \mathbb{Y}(t).\ n \in \mathbb{X}(t) \,\}$$

is always Lebesgue measurable. And this agrees with our experience with numerical random variables.

As an example of randomization, we give some definitions appropriate to ***Automata Theory***.

Definition 4.3. Let $\mathsf{S} = \lambda F.\ \mathsf{S}(F)$ be a suitable recursively enumerable set where

$$\mathsf{S}(F)(\{0\}) = \lambda X.\ X \text{ and } \mathsf{S}(F)(\{(n,m)\}) = F(\{m\}) \circ \mathsf{S}(F)(\{n\}).$$

We can refer to the combinator S as the ***sequentializer***, for it takes any operator F and any sequence of integers $\langle n_0, n_1, \ldots, n_{k-1}, n_k \rangle$ and produces a corresponding sequence of ***compositions:***

$$\mathsf{S}(F)(\langle n_0, n_1, \ldots, n_{k-1}, n_k \rangle) = F(\{n_k\}) \circ F(\{n_{k-1}\}) \circ \ldots \circ F(\{n_1\}) \circ F(\{n_0\}).$$

Note the order on the right. We write as usual $F \circ G = \lambda X.\ F(G(X))$ for the composition of operators.

Theorem 4.4. *Let* $\Sigma \in \mathcal{P}(\mathbb{N})$ *be finite, then the* ***regular languages*** *contained in* Σ^* *are exactly the sets of the form*

$$\{\, \sigma \in \Sigma^* \mid 0 \in \mathsf{S}(A)(\{\sigma\})(Q) \,\},$$

where $A, Q \in \mathcal{P}(\mathbb{N})$ *are given finite sets.*

Here Σ is to be thought of as a fixed ***alphabet***, the operator A as defining an ***automaton***, and Q is the ***initial state*** for running the automaton. Note that we could also have let $Q = \{q\}$, but the larger sets give the same languages over the class of all such representations of automata. The sequentializer S then ***runs*** the automaton, and, by convention, hitting a state containing the number 0 indicates ***success*** or ***acceptance.*** Note that as A is finite, then all values $A(\{n\})(X)$ are also finite sets, so we never encounter more than a finite number of finite states of a computation.

In the form shown in the theorem, the definition of regular language looks a little different from what we are used to from the standard literature, but actually this is almost a direct translation of the usual definition of regular languages — we are just taking a little advantage of our model to employ enumeration operators. In order next to pass to ***probabilistic languages,*** a very small change is needed.

Theorem 4.5. *Let* $\Sigma, Q \in \mathcal{P}(\mathbb{N})$ *be finite, let* $\mathbb{A}$ *be a random variable taking values in a finite subset of finite elements of* $\mathcal{P}(\mathbb{N})$*, and let* $\delta \in [0,1]$. *Then the* ***probabilistic languages*** *contained in* Σ^* *are among the sets*

$$\{\, \sigma \in \Sigma^* \mid \mu\,([\![0 \in \mathsf{S}(\mathbb{A})(\{\sigma\})(Q)]\!]) > \delta \,\},$$

where μ *is Lebesgue measure.*

Of course, all we have illustrated here is a ***notation***, and we have not argued that ***proofs*** using $\mathcal{P}(\mathbb{N})$ are any better than standard presentations. The author did establish this in a classroom situation for elementary recursion theory, however, but not as yet for presenting probabilistic results.

5. Conclusion. It is an elementary exercise in measure theory to construct a random variable

$$\mathbb{T} : [0,1] \longrightarrow \mathcal{P}(\mathbb{N}),$$

where for all n we have $\mathbb{T}(\{n\}) \in \{\{0\}, \{1\}\}$ in such a way that the events $[\![\, \mathbb{T}(\{0\}) = \{0\} \,]\!]$, ... , $[\![\, \mathbb{T}(\{n\}) = \{0\} \,]\!]$ are each of probability 1/2 and are jointly ***probabilistically independent***. In other words, the infinite sequence $\mathbb{T}(\{0\})$, $\mathbb{T}(\{1\})$, $\mathbb{T}(\{2\})$, ... , $\mathbb{T}(\{n\})$, ... represents in the model the ***tossing of a fair coin***. Hence, when F is computable, a combination $F(\mathbb{T})(X)$ can represent a ***Monte Carlo algorithm*** with the probability of the outcome depending on the values taken by the coins during the steps of the computation. In this way, with the aid of the model $\mathcal{P}(\mathbb{N})$ and with random variables, $\boldsymbol{\lambda}$-calculus can be made ***stochastic***. But this is only ***one*** topological model, and, as catalogued in **[10]**, there are many, many others which can easily admit random variables. The author urges the further study of these models.

6. References.

[1] Dana S. Scott. *Data types as lattices.* **SIAM Journal on Computing**, vol. 5 (1976), pp. 522–587.

[2] Gordon D. Plotkin. *Set-theoretical and other elementary models of the λ-calculus.* **Theoretical Computer Science**, vol. 121 (1993), pp. 351–409.

[3] 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.

[4] John Myhill and John C. Shepherdson. *Effective operations on partial recursive functions.* **Zeitschrift für Mathematische Logik und Grundlagen der Mathematik**, vol. 1 (1955), pp. 310–317.

[5] Richard M. Friedberg and Hartley Rogers Jr. *Reducibility and completeness for sets of integers.* **Mathematical Logic Quarterly**, vol. 5 (1959), pp. 117–125. [Some of the results of this paper are presented in abstract in: **Journal of Symbolic Logic**, vol. 22 (1957), p. 107.]

[6] Hartley Rogers, Jr. **Theory of Recursive Functions and Effective Computability**, McGraw-Hill, 1967, xix + 482 pp.

[7] U. de'Liguoro and A. Piperno. *Nondeterministic extensions of untyped λ-calculus.* **Information and Computation**, vol. 122 (1995), pp. 149–177.

[8] Ugo Dal Lago and Margherita Zorzi. *Probabilistic operational semantics for the lambda calculus.* **RAIRO - Theoretical Informatics and Applications**, vol. 46 (2012), pp. 413–450.

[9] `http://www.cs.tulane.edu/~mwm/Michael_Misloves_Home_Page/Research.html`

[10] G. Gierz, K. H. Hofmann, K. Keimel, J. D. Lawson, M. Mislove, and D. S. Scott. **Continuous Lattices and Domains.** (Encyclopedia of Mathematics and its Applications) Cambridge University Press, 2003, 628 pp.

Please cite this paper as: Dana S. Scott. *Stochastic λ-calculi: An extended abstract.* **Journal of Applied Logic**, vol. 12 (2014), pp. 369–376.