Available online at www.sciencedirect.com
ScienceDirect
ELSEVIER
Electronic Notes in Theoretical Computer Science 249 (2009) 193–217
Electronic Notes in Theoretical Computer Science
www.elsevier.com/locate/entcs

Categories of Timed Stochastic Relations

Daniel Brown[1] Riccardo Pucella[2]

College of Computer and Information Science
Northeastern University
Boston MA, USA

Abstract

Stochastic behavior—the probabilistic evolution of a system in time—is essential to modeling the complexity of real-world systems. It enables realistic performance modeling, quality-of-service guarantees, and especially simulations for biological systems. Languages like the stochastic pi calculus have emerged as effective tools to describe and reason about systems exhibiting stochastic behavior. These languages essentially denote continuous-time stochastic processes, obtained through an operational semantics in a probabilistic transition system. In this paper we seek a more descriptive foundation for the semantics of stochastic behavior using categories and monads. We model a first-order imperative language with stochastic delay by identifying probabilistic choice and delay as separate effects, modeling each with a monad, and combining the monads to build a model for the stochastic language.

Keywords: probability, stochastic behavior, category theory, monads, partial additivity

1 Introduction

Stochastic temporal behavior is crucial for modeling real-world systems with non-functional requirements like quality-of-service guarantees [11]. Such requirements often take the form of soft real-time constraints such as "do a before time t with probability 0.99". Multimedia applications and collaborative virtual environments are well-known examples of systems exhibiting such characteristics.

To model and program systems with soft constraints, we need languages to express probability distributions over the delays experienced during the evolution of the system. PEPA [26] and the stochastic pi calculus [43] are two examples of languages that express this kind of stochastic temporal behavior. The semantics of these languages is operational, given in terms of a labelled probabilistic transition system. The transition systems themselves denote continuous-time stochastic processes, often continuous-time Markov stochastic processes.

[1] Email: dbrown@ccs.neu.edu
[2] Email: riccardo@ccs.neu.edu

doi:10.1016/j.entcs.2009.07.091

An operational semantics in terms of probabilistic transition systems, however, does not directly describe stochastic temporal behavior—it delegates the task to the metatheory. In this paper, we initiate a study of the *foundation* of stochastic temporal behavior, working towards a natural denotational semantics for stochastic languages. In particular, we are interested in relating the kind of probabilistic behavior found in languages with probabilistic choice with the kind of stochastic temporal behavior found in the stochastic pi calculus. Following Giry's [21] approach to the categorical foundation of discrete- and continuous-time Markov processes, we explore a categorical model of stochastic temporal behavior. This approach has two immediate advantages. First, the resulting model is sufficiently abstract to be generally applicable. Second, we obtain a principled derivation of semantic models for stochastic languages.

To ground our intuitions, we study stochastic temporal behavior in the context of a simple language of while loops [24,52]. Languages of while loops are relatively simple, yet structured and Turing-complete. Moreover, being first-order, their denotational semantics requires no heavy-duty machinery. In §2, we review the standard categorical semantics for such languages, taking advantage of Moggi's insight that monads can be used to lift the semantics of a pure language to an extension with effects [37,38]. We illustrate the approach with two different effects: iteration and probabilistic choice.

Stochastic temporal behavior ultimately amounts to adding delay to computations. In §3, we present an abstract approach for adding delay to the categorical semantics of our imperative language by introducing a monad to express timed computations. We examine how this monad interacts with monads capturing other effects in the language. In particular, we investigate conditions under which adding timed computations to a semantic model that correctly handles iteration yields an extended semantic model that also correctly handles iteration.

In §4, we instantiate our abstract approach to derive semantic models for a language of while loops extended with a deterministic delay operator and for a language of while loops extended with a probabilistic delay operator. In §5, we instantiate our abstract approach to derive semantic models for a *probabilistic* language of while loops [32] extended with a deterministic delay operation. We call the resulting semantic models categories of $\mathcal{M}$-timed stochastic relations $\mathbf{TSRel}_{\mathcal{M}}$, extending the category **SRel** of stochastic relations commonly used to give semantics to probabilistic languages of while loops. In these categories, we draw a relationship between probabilistic choice and stochastic temporal behavior by showing that both are in fact derivable from a primitive that lets us sample probability distributions.

We review related work in §6 and conclude in §7. Due to space restrictions, proofs of our technical results are only sketched where useful, and full proofs have been relegated to the full version of the paper [12].

2 Categories for Imperative Languages with Effects

Standard denotational semantics for languages of while loops are state-transformer semantics: the meaning of a statement is a partial function from states to states, where states are assignments of values to variables. Partial functions are necessary because loops need not terminate.

This sort of semantics can be given abstractly in any category with the right structure. We review such a categorical semantics below. The material in this section is well known and we claim no novelty, but our presentation of it may be unfamiliar: because our approach to adding delay in §3–5 relies on first separating iteration from the model, we treat nontermination as an effect and model it with a monad in the style of Moggi [37,38].

We define a family of typed *imperative sequential languages* ISL_{ext}, where *ext* represents some language extensions that carry effects. The base language in this family, ISL_0, is an imperative sequential language without iteration.

Syntax of ISL_0**:**

τ ::=	type
$\quad\mathsf{bool}$	
$\quad\ldots$	
E ::=	expression
$\quad\ldots$	
S ::=	statement
$\quad\mathsf{skip}$	skip
$\quad S_1; S_2$	sequencing
$\quad\mathsf{let}\ v : \tau = E\ \mathsf{in}\ S$	allocation
$\quad v := E$	assignment
$\quad\mathsf{if}\ E\ \mathsf{then}\ S_1\ \mathsf{else}\ S_2$	conditional

Our focus is on statements and their semantics, so we elide the details of the expression language E and types other than bool; we assume only that expressions are effect free. In examples we freely use expressions that include variable reference, Boolean operations, and rational arithmetic. We assume a countable set $\mathcal{V}$ of variables, ranged over by u, v, w.

We use a standard type system (e.g., [42,24]) to simplify our semantics. Judgement $\Gamma \vdash E : \tau$ states that the expression E has type τ in typing context Γ. Judgement $\Gamma \vdash S$ states that the statement S is well formed in typing context Γ. A typing context Γ is a sequence of pairs $v : \tau$.

Typing Rules of ISL_0**:**

$$\frac{}{\Gamma \vdash \mathsf{skip}} \qquad \frac{\Gamma \vdash S_1 \qquad \Gamma \vdash S_2}{\Gamma \vdash S_1; S_2} \qquad \frac{\Gamma \vdash E : \tau \qquad v : \tau, \Gamma \vdash S}{\Gamma \vdash \mathsf{let}\ v : \tau = E\ \mathsf{in}\ S}$$

$$\frac{\Gamma, v : \tau, \Gamma' \vdash E : \tau}{\Gamma, v : \tau, \Gamma' \vdash v := E}\ {\scriptstyle (v \notin \Gamma)} \qquad \frac{\Gamma \vdash E : \mathsf{bool} \qquad \Gamma \vdash S_1 \qquad \Gamma \vdash S_2}{\Gamma \vdash \mathsf{if}\ E\ \mathsf{then}\ S_1\ \mathsf{else}\ S_2}$$

The only subtlety in the typing rules is the side condition for assignment: v must not occur in Γ, preventing non-unique types, but may occur in Γ', permitting bound variables to be shadowed by inner let bindings. In other words, typing contexts "grow on the left".

The standard state-transformer semantics for ISL_0 can be given in any distributive category, that is, a category with finite products (for state spaces), finite coproducts (for Booleans), and distributivity of products over coproducts given by an inverse to the canonical map $X{\times}Y{+}X{\times}Z \xrightarrow{[1\times\iota_1,1\times\iota_2]} X{\times}(Y{+}Z)$ (for conditionals [20]). Following Moggi, we model effectful extensions of ISL_0 in the *Kleisli category* of a suitable monad, allowing the semantics for the pure language ISL_0 to remain uniform as the extensions vary. We thus parameterize the semantics for ISL_0 over an arbitrary monad.

Let $\mathbf{C}$ be a distributive category and $T : \mathbf{C} \to \mathbf{C}$ a monad with unit $1 \xrightarrow{\eta^T} T$ and multiplication $TT \xrightarrow{\mu^T} T$. We define the monadic semantics of ISL_0 via a pair of maps: $[\![-]\!]^\Gamma$ assigns to every well-formed statement a Kleisli arrow on states and $[\![- : \tau]\!]^\Gamma$ assigns to every well-typed expression a pure arrow from states to values:

$$\begin{aligned}
[\![S]\!]^\Gamma &: [\![\Gamma]\!] \to T[\![\Gamma]\!] && (\text{if } \Gamma \vdash S)\\
[\![E:\tau]\!]^\Gamma &: [\![\Gamma]\!] \to [\![\tau]\!] && (\text{if } \Gamma \vdash E : \tau)
\end{aligned}$$

Since expressions have no effects, their semantics is given by arrows in the base category $\mathbf{C}$. Types denote objects; in particular,

$$[\![\mathsf{bool}]\!] \triangleq 1+1$$

the object representing the two truth values. The state object $[\![\Gamma]\!]$ denoted by the typing context Γ is the product of the objects denoted by the types in the context:

$$[\![v_1:\tau_1,\ldots,v_n:\tau_n]\!] \triangleq [\![\tau_1]\!] \times \cdots \times [\![\tau_n]\!]$$

We write $X_T \xrightarrow{f_T} Y_T$ or $\left(X \xrightarrow{f} TY\right)_T$ for a Kleisli arrow in $\mathbf{C}_T$ with underlying arrow $X \xrightarrow{f} TY$ in $\mathbf{C}$, and we abbreviate components of natural transformations $FX \xrightarrow{\varphi_X} GX$ as $FX \xrightarrow{\varphi} GX$ when the object is clear from context. We abbreviate η^T and μ^T as η and μ when the monad T is clear from context.

Monadic Semantics of ISL_0:

$$\begin{aligned}
\left([\![\mathsf{skip}]\!]^\Gamma\right)_T &\triangleq \left([\![\Gamma]\!] \xrightarrow{\eta} T[\![\Gamma]\!]\right)_T\\
&= [\![\Gamma]\!]_T \xrightarrow{1} [\![\Gamma]\!]_T\\
\left([\![S_1;S_2]\!]^\Gamma\right)_T &\triangleq \left([\![\Gamma]\!] \xrightarrow{[\![S_1]\!]^\Gamma} T[\![\Gamma]\!] \xrightarrow{T[\![S_2]\!]^\Gamma} TT[\![\Gamma]\!] \xrightarrow{\mu} T[\![\Gamma]\!]\right)_T\\
&= [\![\Gamma]\!]_T \xrightarrow{[\![S_1]\!]^\Gamma_T} [\![\Gamma]\!]_T \xrightarrow{[\![S_2]\!]^\Gamma_T} [\![\Gamma]\!]_T
\end{aligned}$$

$$\llbracket \mathsf{let}\ v : \tau = E\ \mathsf{in}\ S \rrbracket^{\Gamma} \triangleq \llbracket\Gamma\rrbracket \xrightarrow{\langle \llbracket E:\tau\rrbracket^{\Gamma}, 1\rangle} \llbracket\tau\rrbracket\times\llbracket\Gamma\rrbracket \xrightarrow{\llbracket S\rrbracket^{v:\tau,\Gamma}} T(\llbracket\tau\rrbracket\times\llbracket\Gamma\rrbracket) \xrightarrow{T\pi_2} T\llbracket\Gamma\rrbracket$$

$$\llbracket v := E \rrbracket^{\Gamma,v:\tau,\Gamma'} \triangleq \llbracket\Gamma\rrbracket\times\llbracket\tau\rrbracket\times\llbracket\Gamma'\rrbracket \xrightarrow{\langle \pi_1, \llbracket E:\tau\rrbracket^{\Gamma,v:\tau,\Gamma'}, \pi_3\rangle} \llbracket\Gamma\rrbracket\times\llbracket\tau\rrbracket\times\llbracket\Gamma'\rrbracket \xrightarrow{\eta} T(\llbracket\Gamma\rrbracket\times\llbracket\tau\rrbracket\times\llbracket\Gamma'\rrbracket)$$

$$\llbracket \mathsf{if}\ E\ \mathsf{then}\ S_1\ \mathsf{else}\ S_2 \rrbracket^{\Gamma} \triangleq \llbracket\Gamma\rrbracket \xrightarrow{\llbracket E:\mathsf{bool}\rrbracket^{\Gamma}?} \llbracket\Gamma\rrbracket+\llbracket\Gamma\rrbracket \xrightarrow{[\llbracket S_1\rrbracket^{\Gamma}, \llbracket S_2\rrbracket^{\Gamma}]} T\llbracket\Gamma\rrbracket$$

Identities and composition in the Kleisli category model skip and sequencing. Standard product constructions in the base category model let and assignment. Standard coproduct constructions along with guards [20] model conditionals, where guards map every predicate $X \xrightarrow{p} 1+1$ to the arrow

$$X \xrightarrow{p?} X+X \triangleq X \xrightarrow{\langle 1,p\rangle} X\times(1+1) \xrightarrow{[1\times\iota_1, 1\times\iota_2]^{-1}} X\times 1+X\times 1 \xrightarrow{\pi_1+\pi_1} X+X$$

The inverse $[1\times\iota_1, 1\times\iota_2]^{-1}$ exists because $\mathbf{C}$ is distributive. The standard denotational model of ISL_0 can be recovered using the category **Set** with the identity monad.

We illustrate the use of a monad T with our first extension, iteration, and its associated effect, nontermination.

Iteration Extension: while

Syntax:

$S\ ::=\ \cdots \mid \mathsf{while}\ E\ \mathsf{do}\ S$

Typing Rules:

$$\frac{\Gamma\vdash E:\mathsf{bool} \qquad \Gamma\vdash S}{\Gamma\vdash \mathsf{while}\ E\ \mathsf{do}\ S}$$

$\mathrm{ISL}_{\mathsf{while}}$ is the standard language of while loops, often called IMP [24,52]. To model while, we need a monad that imposes enough structure on its Kleisli category to support iteration. Following Manes and Arbib [35], we take this to mean that the Kleisli category should be *partially additive*.

Intuitively, a loop is the limit of the finite unrollings of its body. Partial additivity models this limiting process through an infinite summation operator on hom-sets. Partial additivity is the combination of a few simple structures (see [35]) but we present it as one large, aggregate definition that suffices for our purposes. The subtleties of this definition are less relevant to our goals than how it enables us to interpret loops, which we give below.

Definition 2.1 A category $\mathbf{D}$ is *partially additive* if and only if

(1) $\mathbf{D}$ has countable coproducts.

(2) Every hom-set $\mathbf{D}(X,Y)$ is a partially additive monoid. That is, it has a partial function $\sum_{X,Y}$ from countable subsets of $\mathbf{D}(X,Y)$ to $\mathbf{D}(X,Y)$—we say the family $\{f_i\}_{i\in I}$ is *summable* if $\sum\{f_i\}_{i\in I}$ is defined—subject to:

- *Partition-associativity axiom:* Given a countable family $\{f_i\}_{i\in I}$ and a countable partition $\{I_j\}_{j\in J}$ of its indexing set,

$$\sum \{f_i\}_{i\in I} = \sum \left\{ \sum \{f_i\}_{i\in I_j} \right\}_{j\in J}$$

In particular, the sum on the left is defined if and only if all of the sums on the right are defined.

- *Unary sum axiom:* Singleton families are summable with $\sum \{f\} = f$.
- *Limit axiom:* A countable family is summable if every finite sub-family is summable.

We abbreviate $\sum \{f_i\}_{i\in I}$ variously as $\sum_{i\in I} f_i$, $\sum_I f_i$, or $\sum f_i$, depending on context.

(3) *Composition distributes over sum*: Given $\left\{ X \xrightarrow{f_i} Y \right\}_{i\in I}$ summable,

- $\{g; f_i\}_{i\in I}$ is summable and $g; (\sum f_i) = \sum g; f_i$, for all $W \xrightarrow{g} X$;
- $\{f_i; h\}_{i\in I}$ is summable and $(\sum f_i); h = \sum f_i; h$, for all $Y \xrightarrow{h} Z$.

(4) *Compatible sum axiom:* A countable family $\left\{ X \xrightarrow{f_i} Y \right\}_{i\in I}$ is summable if some $X \xrightarrow{f} I\cdot Y$ makes all diagrams

$$\begin{tikzcd} X \arrow[r, "f"] \arrow[dr, "f_i"'] & I\cdot Y \arrow[d, "\rho_i"] \\ & Y \end{tikzcd}$$

commute, where $I \cdot Y = \coprod_I Y$ and the family $\left\{ I\cdot Y \xrightarrow{\rho_i} Y \right\}_{i\in I}$ is an instance of the more general family of *quasi-projections* $\left\{ \coprod X_i \xrightarrow{\rho_i} X_i \right\}_{i\in I}$ defined by

$$\rho_i \triangleq [0_{X_1,X_i}, \ \ldots, \ 0_{X_{i-1},X_i}, \ 1_{X_i}, \ 0_{X_{i+1},X_i}, \ \ldots]$$

where the arrows $0_{X,Y}$ are zeroes for composition and units for sum, which exist as $0_{X,Y} = \sum_{X,Y} \varnothing$ in any category satisfying (2) and (3).

(5) *Untying axiom:* If the two arrows $X \xrightarrow{f} Y$ and $X \xrightarrow{g} Y$ are summable, then so are $X \xrightarrow{f;\iota_1} Y+Y$ and $X \xrightarrow{g;\iota_2} Y+Y$.

Two familiar examples of partially additive categories are the category **Par** of sets with partial functions and the category **CPO** of complete partial orders and continuous functions. In **Par**, a family of partial functions is summable if and only if the functions are defined on mutually disjoint subsets of the domain, and the sum is the union of their graphs. Partial additivity in **CPO** is even more familiar: a family is summable if and only if it is a directed subset of the function space, and the sum is the least upper bound.

The key consequence of partial additivity in our setting is that every arrow $X \xrightarrow{f} X+Y$ decomposes into arrows $X \xrightarrow{f_1} X$ and $X \xrightarrow{f_2} Y$ such that $f = \sum \{f_1; \iota_1, \ f_2; \iota_2\}$, and the *iterate* of f

$$f^\dagger \triangleq \sum_{i<\omega} f_1^i; f_2$$

is defined, where $g^0 = 1$ and $g^{i+1} = g; g^i$. The morphism $f^\dagger$ satisfies the equation

$$f^\dagger = \sum\{f_1; f^\dagger,\, f_2\}$$

which can be seen as the defining equation of the while loop.

Given a monad $T : \mathbf{C} \to \mathbf{C}$ whose Kleisli category $\mathbf{C}_T$ is partially additive, we model $\mathrm{ISL}_{\mathsf{while}}$ by extending the ISL_0 semantics with an interpretation for loops:

Monadic Semantics of while**:**

$$[\![\mathsf{while}\ E\ \mathsf{do}\ S]\!]^\Gamma \triangleq [\![\Gamma]\!] \xrightarrow{\left([\![\Gamma]\!] \xrightarrow{[\![E:\mathsf{bool}]\!]^\Gamma?} [\![\Gamma]\!]+[\![\Gamma]\!] \xrightarrow{[\![S]\!]^\Gamma + \eta} T[\![\Gamma]\!]+T[\![\Gamma]\!]\right)^\dagger} T[\![\Gamma]\!]$$

A nice property of this semantics is that it does not rely on any particular monad but is defined abstractly over the class of monads that yield partially additive Kleisli categories. So when we consider additional effects and monads to model them, we have a canonical interpretation for while as long as we have partial additivity. The standard categorical model of $\mathrm{ISL}_{\mathsf{while}}$ can be recovered with **Set** as the base category and the partiality monad $-_\perp = - + 1$. The resulting Kleisli category $\mathbf{Set}_{-_\perp}$ is isomorphic to **Par**, which is partially additive as we already noted.

Probabilistic extensions of ISL_0 form the basis of our study. Throughout the paper we assume the reader is familiar with basic probability and measure theory [9,5,19,29]. The simplest way to model probabilistic behavior is to use a probabilistic choice operator:

Probabilistic Choice Extension: $+_p$

Syntax:

$$S \ ::= \ \cdots \mid S_1 +_p S_2$$

Typing Rules:

$$\frac{\Gamma \vdash S_1 \qquad \Gamma \vdash S_2}{\Gamma \vdash S_1 +_p S_2}$$

The statement $S_1 +_p S_2$ reads "execute S_1 with probability $1-p$ and S_2 with probability p". This operator is nicely modeled with Markov kernels: functions that map states to *probability distributions* over states. Markov kernels are the Kleisli arrows for Giry's [21] probability monad over measurable spaces but, since they fail to be partially additive, the monad is only suitable for modeling ISL_+ and not the richer language $\mathrm{ISL}_{\mathsf{while},+}$ that includes iteration.

Panangaden [41] solves this problem by considering *sub*-Markov kernels obtained from the monad Π of *sub*probability distributions. A subprobability distribution is like a probability distribution except it allows the probability of the whole space to be any value between 0 and 1; this relaxation enables the partiality inherent in modeling iteration. The subprobability functor Π over the category of measurable spaces can be summarized as:

$$\begin{aligned} \Pi \ :\ &\mathbf{Meas} \to \mathbf{Meas} \\ &(X, \Sigma_X) \mapsto (\Pi X, \Sigma_X^\bullet) \\ &X \xrightarrow{f} Y \mapsto \left(\Sigma_X \xrightarrow{\nu} [0,1] \mapsto \Sigma_Y \xrightarrow{f^{-1}} \Sigma_X \xrightarrow{\nu} [0,1] \right) \end{aligned}$$

The measurable space $(\Pi X, \Sigma_X^\bullet)$ is the set of all subprobability distributions over X equipped with the smallest σ-algebra that makes measurable all evaluation functions $\epsilon_A : \Pi X \to [0,1]$, where $A \in \Sigma_X$ and $\epsilon_A(\nu) = \nu(A)$. The arrow action produces a measurable map on distributions $(\nu \in \Pi X) \mapsto (f^{-1}; \nu \in \Pi Y)$. The functor is a monad with unit and multiplication:

$$\begin{aligned} \eta_X : X &\to \Pi X & \qquad \mu_X : \Pi^2 X &\to \Pi X \\ x &\mapsto \delta_x & P, A &\mapsto \int_{\Pi X} \nu(A)\ P(d\nu) \end{aligned}$$

The unit η maps a point x to its point-mass distribution δ_x and multiplication μ evaluates a distribution over distributions down to its average distribution. A comment about notation: when defining functions into spaces of distributions, we find it convenient to take a measurable set as an extra argument—that is, we define a map $X \to \Pi Y$ in its uncurried form, $X \times \Sigma_Y \to [0,1]$.

Panangaden presents $\mathbf{Meas}_\Pi$ more directly as the category of sub-Markov kernels or *stochastic relations*, **SRel**. Its objects are the same as **Meas**, and an arrow $X \xrightarrow{f} Y$ is a function $f : X \times \Sigma_Y \to [0,1]$ such that every $f(x, -)$ is a subprobability distribution and every $f(-, B)$ is measurable. We can think of f as a probabilistic variant of a relation: it gives the probability that a point in X is "related to" a measurable subset of Y. Arrow composition $X \xrightarrow{f} Y \xrightarrow{g} Z$ is then defined as

$$(f; g)(x, C) = \int_Y f(x, dy)\ g(y, C) \tag{1}$$

which can be read: the probability that $(f; g)$ relates x to the measurable set C is the probability that f relates x to something that g then relates to C. It is easy to see that **SRel** and $\mathbf{Meas}_\Pi$ are isomorphic: currying a stochastic relation $X \times \Sigma_Y \xrightarrow{f} [0,1]$ gives a Kleisli arrow $X \xrightarrow{\tilde{f}} \Pi Y$, and Kleisli composition is just a curried version of (1). Throughout the paper we will freely interchange $\mathbf{Meas}_\Pi$ and **SRel**.

To support a semantics for $\mathrm{ISL}_{\mathsf{while}}$, the base category **Meas** must be distributive and $\mathbf{Meas}_\Pi$ must be partially additive. Panangaden [41] establishes partial additivity:[3] the sum of a family $\left\{ X \times \Sigma_Y \xrightarrow{f_i} [0,1] \right\}_{i \in I}$ is the pointwise sum of the functions if the sum is a valid stochastic relation—it does not exceed 1 anywhere—otherwise the family is not summable. For distributivity, we first need products and coproducts. Like the category of topological spaces, **Meas** is topological over **Set** [2] and

[3] Abramsky [1] first observed that **SRel** is a *traced monoidal category*. Panangaden refined this by fleshing out its partially additive structure—see Haghverdi's thesis [25] for details on how the iteration operator in a partially additive category induces a trace.

inherits both completeness and cocompleteness. Limit spaces are limits from **Set** equipped with the initial σ-algebra, and colimit spaces are colimits from **Set** with the final σ-algebra. In the cases of products and coproducts, this means that the product space $(X,\Sigma_X)\times(Y,\Sigma_Y)$ is $(X\times Y,\Sigma_X\otimes\Sigma_Y)$, where $\Sigma_X\otimes\Sigma_Y$ is the smallest σ-algebra that makes the projections $X\times Y \xrightarrow{\pi_1} X$ and $X\times Y \xrightarrow{\pi_2} Y$ measurable. Similarly, the coproduct space $(X,\Sigma_X)+(Y,\Sigma_Y)$ is $(X+Y,\Sigma_X\oplus\Sigma_Y)$, where $\Sigma_X\oplus\Sigma_Y$ is the largest σ-algebra making the injections $X \xrightarrow{\iota_1} X+Y$ and $Y \xrightarrow{\iota_2} X+Y$ measurable. Thus,

$$\Sigma_X \otimes \Sigma_Y = \sigma(\{A \times B : A \in \Sigma_X, B \in \Sigma_Y\})$$

is the σ-algebra generated by the "rectangles" $A \times B$ with measurable sides, and

$$\Sigma_X \oplus \Sigma_Y = \{A \uplus B : A \in \Sigma_X, B \in \Sigma_Y\}$$

is the set of disjoint unions of pairs of measurable sets from X and Y. Distributivity then follows by an elementary argument that the inverse in **Set** for the canonical map $X\times Y+X\times Z \xrightarrow{[1\times\iota_1,1\times\iota_2]} X\times(Y+Z)$ is measurable.

It only remains to describe how to interpret probabilistic choice. It is easy to see that **SRel** is closed under subconvex combinations of morphisms: given a sequence $a_i \in \mathbb{R}^+$ such that $\sum a_i \leq 1$, any family of arrows $\left\{X \xrightarrow{f_i} \Pi Y\right\}_{i\in\mathbb{N}}$ becomes summable when each is scaled (pointwise) as $\left\{X \xrightarrow{a_i f_i} \Pi Y\right\}_{i\in\mathbb{N}}$ because the bound on the series guarantees that the pointwise sum of the family does not exceed 1. Specializing this to families with two arrows defines an abstract interpretation for probabilistic choice in any category whose hom-sets are closed under subconvex combinations:

Monadic Semantics of $+_p$:

$$[\![S_1 +_p S_2]\!]^\Gamma \triangleq [\![\Gamma]\!] \xrightarrow{\sum\left\{(1-p)[\![S_1]\!]^\Gamma,\ p[\![S_2]\!]^\Gamma\right\}} T[\![\Gamma]\!]$$

As Panangaden notes, this semantics for $\mathrm{ISL}_{\mathsf{while},+}$ in **SRel** is the same as the semantics of Kozen's probabilistic language of while loops [32,33].

3 Adding Delay

We begin by showing how to abstractly add delay to a monadic semantics for ISL. We are ultimately interested in modeling ISL_0 extended with delay and other effects; the method we present in this section is parameterized over the monad that models the class of effects to which delay should be added. This parametric story is instantiated in §4 to add delay to the **Par** model of $\mathrm{ISL}_{\mathsf{while}}$ and in §5 to add delay to the **SRel** model of $\mathrm{ISL}_{\mathsf{while},+}$.

Modeling delay requires a notion of time which can be conveniently and abstractly captured using a monoid (M, e, m) [4]. Common examples include the naturals $(\mathbb{N}, 0, +)$ and nonnegative reals $(\mathbb{R}^+, 0, +)$. Since we are defining semantics categorically, we abstract our model of time one step further and use a *monoid in a category* **C**: an object M in a category **C** with finite products equipped with a unit arrow $1 \xrightarrow{e} M$ and a multiplication arrow $M \times M \xrightarrow{m} M$ subject to the commutativity of two diagrams corresponding to the unit laws and associativity.

$$\begin{tikzcd} M \arrow[r, "1 \times e"] \arrow[dr, "1"'] & M^2 \arrow[d, "m"] & M \arrow[l, "e \times 1"'] \arrow[dl, "1"] \\ & M & \end{tikzcd} \qquad \begin{tikzcd} M^3 \arrow[r, "m \times 1"] \arrow[d, "1 \times m"'] & M^2 \arrow[d, "m"] \\ M^2 \arrow[r, "m"'] & M \end{tikzcd}$$

In outline, we add delay to a semantics for $\text{ISL}_{\mathsf{while}}$ as follows. We start with a monadic semantics of $\text{ISL}_{\mathsf{while},ext}$, for some language extension ext, in a partially additive category $\mathbf{C}_T$. Given a monoid (M, e, m) in $\mathbf{C}$ to represent time and a strength for T, we get a distributive law of the monad $- \times M$ over T, making the composition $T(- \times M)$ a monad. We then identify reasonable assumptions under which $\mathbf{C}_{T(-\times M)}$ inherits partial additivity from $\mathbf{C}_T$, enabling a natural extension of the $\mathbf{C}_T$ semantics to one in $\mathbf{C}_{T(-\times M)}$ that also models delay.

Let (M, e, m) be a monoid in a distributive category $\mathbf{C}$. Our notion of delay is expressed simply:

Delay Extension: wait

Syntax:	**Typing Rules:**
$\tau \ ::= \cdots \mid \mathsf{time}$ $S \ ::= \ \cdots \mid \mathsf{wait}\ E$	$\dfrac{\Gamma \vdash E : \mathsf{time}}{\Gamma \vdash \mathsf{wait}\ E}$

The statement wait E delays execution by E time units, where E has type time. We use the monoid M to interpret values of type time:

$$[\![\mathsf{time}]\!] \triangleq M$$

This means that time expressions denote arrows into M:

$$[\![E : \mathsf{time}]\!]^\Gamma : [\![\Gamma]\!] \to M$$

But what monad is appropriate to model delay? Or, more directly, what Kleisli arrows should interpret statements in the timed language?

To guide our intuition, we first consider adding wait to an effect-free model of ISL_0 in $\mathbf{C}$ so that statements are interpreted just as arrows $X \to Y$. To associate a delay with a pure computation, we can use arrows $X \to Y \times M$ that compute a time in addition to a new state:

$$[\![S]\!]^\Gamma : [\![\Gamma]\!] \to [\![\Gamma]\!] \times M$$

The statement wait E should then denote an arrow that computes the delay E and leaves the state unchanged:

$$[\![\mathsf{wait}\ E]\!]^\Gamma = [\![\Gamma]\!] \xrightarrow{\langle 1, [\![E : \mathsf{time}]\!]^\Gamma \rangle} [\![\Gamma]\!] \times M$$

Statements with no substatements, like skip, should denote arrows that record no delay, and sequenced statements should combine their delays:

$$[\![\mathsf{skip}]\!]^\Gamma = {}_{[\![\Gamma]\!]} \xrightarrow{\langle 1, ! \rangle} {}_{[\![\Gamma]\!] \times 1} \xrightarrow{1 \times e} {}_{[\![\Gamma]\!] \times M}$$
$$[\![S_1; S_2]\!]^\Gamma = {}_{[\![\Gamma]\!]} \xrightarrow{[\![S_1]\!]^\Gamma; ([\![S_2]\!]^\Gamma \times 1)} {}_{([\![\Gamma]\!] \times M) \times M} \xrightarrow{\alpha} {}_{[\![\Gamma]\!] \times (M \times M)} \xrightarrow{1 \times m} {}_{[\![\Gamma]\!] \times M}$$

where ${}_X \xrightarrow{!} {}_1$ is the terminal arrow from X and ${}_{(X \times Y) \times Z} \xrightarrow{\alpha} {}_{X \times (Y \times Z)}$ associates products. If we continued considering interpretations for the rest of the statements in ISL_0, we would see that it is exactly the monadic semantics over the well-known monad of monoid actions $- \times M : \mathbf{C} \to \mathbf{C}$ extended with an interpretation for wait.

Proposition 3.1 *Given a monoid (M, e, m) in a category $\mathbf{C}$ with finite products, the functor $- \times M : \mathbf{C} \to \mathbf{C}$ is a monad with*

$$\eta_X = {}_X \xrightarrow{\langle 1, !; e \rangle} {}_{X \times M} \qquad \mu_X = {}_{(X \times M) \times M} \xrightarrow{\alpha} {}_{X \times (M \times M)} \xrightarrow{1 \times m} {}_{X \times M}$$

where ${}_X \xrightarrow{!} {}_1$ is the terminal arrow from X.

The above gives a semantics for $\mathrm{ISL}_{\mathsf{wait}}$ in $\mathbf{C}_{-\times M}$, but it leaves no room for other monads capturing additional effects. Given a semantics for some extension ISL_{ext} in a Kleisli category $\mathbf{C}_T$, we want to combine the monads T and $- \times M$ to obtain a monad modeling the effects in both languages. Combining monads is difficult in general, but in our setting the straightforward approach of functor composition and distributive laws [6] suffices.

If we want to combine T and $- \times M$ by composition, which order is appropriate? Intuitively, we think of partial additivity as giving a way to take partially defined arrows and combine them into a single arrow that aggregates all of their partial information. This suggests that the $(T-) \times M$ order is inappropriate since a Kleisli arrow $X \to TY \times M$ decomposes into an arrow $X \to TY$ giving the partial state transformation from X to Y and an arrow $X \to M$ giving the non-partial delay computation. So, unless we put heavier demands on the monoid M, it seems futile to look for partial additivity on hom-sets $X \to TY \times M$. On the other hand, hom-sets $X \to T(Y \times M)$ give T the "last word" by framing the $Y \times M$ result—a new state and a delay value—within T. For example, consider $\mathbf{Set}_{-_\perp}$: arrows $X \to (Y \times M)_\perp$ have a natural notion of failure whereas arrows $X \to Y_\perp \times M$ do not, unless we invent a second notion of failure within M.

For the composition $T(- \times M) : \mathbf{C} \to \mathbf{C}$ to form a monad, a distributive law

$$\lambda : (T-) \times M \to T(- \times M)$$

suffices to define the combined multiplication and satisfy the monad laws.

Proposition 3.2 (Beck [7]) *If $S, T : \mathbf{C} \to \mathbf{C}$ are monads with a distributive law of S over T*

$$\lambda : ST \to TS$$

then $TS : \mathbf{C} \to \mathbf{C}$ *is a monad with unit and multiplication*

$$\eta^{TS} = {}_1 \xrightarrow{\eta^T \circ \eta^S} {}_{TS} \qquad\qquad \mu^{TS} = {}_{TSTS} \xrightarrow{T\lambda S} {}_{TTSS} \xrightarrow{\mu^T \circ \mu^S} {}_{TS}$$

where $\circ$ *is horizontal composition.*

Further, distributive laws for $- \times M$ arise from strong monads as defined by Moggi [38]. Even though any distributive law of $- \times M$ over T suffices to achieve our goals, the monads we use in §4 and §5 are both strong and strength is somewhat more familiar than distributive laws, so we restrict our attention to distributive laws arising from strong monads.

Proposition 3.3 *If* $\mathbf{C}$ *is a category with finite products,* $T : \mathbf{C} \to \mathbf{C}$ *is a strong monad, and* (M, e, m) *is a monoid in* $\mathbf{C}$*, then a tensorial strength for* T

$$t_{X,Y} : X \times TY \to T(X \times Y)$$

gives a distributive law of $- \times M$ *over* T

$$\lambda_X = \bar{t}_{X,M} : TX \times M \to T(X \times M)$$

where $\bar{t}_{X,Y} = {}_{TX\times Y} \xrightarrow{\gamma} {}_{Y\times TX} \xrightarrow{t_{Y,X}} {}_{T(Y\times X)} \xrightarrow{T\gamma} {}_{T(X\times Y)}$ *is* $t_{Y,X}$ *commuted.*

Not only does the distributive law give the monad $T(- \times M) : \mathbf{C} \to \mathbf{C}$ that extends the pure semantics for ISL_0 to model delay in addition to the effects in the original language ISL_{ext}, but it also induces a monad $\overline{- \times M} : \mathbf{C}_T \to \mathbf{C}_T$ that directly extends the effectful semantics of ISL_{ext}.

Proposition 3.4 *If* $S, T : \mathbf{C} \to \mathbf{C}$ *are monads and* S *distributes over* T*, then* S *lifts to a monad* $\overline{S} : \mathbf{C}_T \to \mathbf{C}_T$ *where*

$$\begin{aligned}
\overline{S}\left({}_{X_T} \xrightarrow{f_T} {}_{Y_T}\right) &= {}_{(SX)_T} \xrightarrow{\left({}_{SX} \xrightarrow{Sf} {}_{STY} \xrightarrow{\lambda_Y} {}_{TSY}\right)_T} {}_{(SY)_T} \\
{}_{X_T} \xrightarrow{\eta^{\overline{S}}_{X_T}} {}_{(SX)_T} &= \left({}_X \xrightarrow{\eta^{TS}_X} {}_{TSX}\right)_T \\
{}_{(SSX)_T} \xrightarrow{\mu^{\overline{S}}_{X_T}} {}_{(SX)_T} &= \left({}_{SSX} \xrightarrow{(\eta^T \circ \mu^S)_X} {}_{TSX}\right)_T
\end{aligned}$$

where $\circ$ *is horizontal composition. Further,*

$$(\mathbf{C}_T)_{\overline{S}} \cong \mathbf{C}_{TS}$$

This is an instance of a *Kleisli lifting* of a functor [39,40] where, since the natural transformation classifying the lifting is a distributive law, the lifted functor is a monad and its Kleisli category coincides with the one for the composite monad.

We can now give a monadic semantics for the wait statement. Let $T : \mathbf{C} \to \mathbf{C}$ a monad such that $- \times M$ distributes over T. This yields a monad $T(- \times M) : \mathbf{C} \to \mathbf{C}$ with which we instantiate the monadic semantics of §2 and extend with an interpretation for wait:

Monadic Semantics of wait**:**

$$[\![\mathsf{wait}\ E]\!]^\Gamma \triangleq {}_{[\![\Gamma]\!]} \xrightarrow{\langle 1, [\![E : \mathsf{time}]\!]^\Gamma \rangle} {}_{[\![\Gamma]\!] \times M} \xrightarrow{\eta} {}_{T([\![\Gamma]\!] \times M)}$$

We must also ensure, however, that the new models in $\mathbf{C}_{T(-\times M)}$ can still interpret T's original effects.

Our languages become uninteresting without iteration, so we seek conditions to ensure that if $\mathbf{C}_T$ is partially additive then $\mathbf{C}_{T(-\times M)}$ is partially additive as well. In particular, we seek a set of conditions much smaller than the somewhat cumbersome set of properties in Definition 2.1. If we keep in mind the lifted monad $\overline{-\times M} : \mathbf{C}_T \to \mathbf{C}_T$ while trying to prove partial additivity for $\mathbf{C}_{T(-\times M)}$, we arrive at a pair of simple, sufficient conditions:

Definition 3.5 A functor $S : \mathbf{D} \to \mathbf{D}'$ between partially additive categories *preserves partial additivity* [4] if and only if

(a) $\{f_i\}_{i\in I}$ summable implies $\{Sf_i\}_{i\in I}$ summable

(b) $S(\sum f_i) = \sum(Sf_i)$

Proposition 3.6 *If a monad* $S : \mathbf{D} \to \mathbf{D}$ *preserves partial additivity, then* $\mathbf{D}_S$ *is partially additive where*

(a) $\left\{ {}_{X_S} \xrightarrow{(f_i)_S} {}_{Y_S} \right\}_{i\in I}$ *is summable if and only if* $\left\{ {}_X \xrightarrow{f_i} {}_{SY} \right\}_{i\in I}$ *is summable*

(b) $\sum {}_{X_S} \xrightarrow{(f_i)_S} {}_{Y_S} = \left(\sum {}_X \xrightarrow{f_i} {}_{YS} \right)_S$

Checking conditions (2)–(5) in the definition of partial additivity is lengthy but straightforward. Condition (1), that $\mathbf{D}_S$ has countable coproducts, follows immediately from $\mathbf{D}$ having countable coproducts [49] since it is partially additive.

The result of all of this is that we can model delay in a monoid from the base category of a monadic model of ISL_0 and interpret the extended language, given that we can establish a distributive law and preservation of additivity. Modeling delay is the easy part—all the work goes into making sure we can still model iteration.

Theorem 3.7 *Let* $S, T : \mathbf{C} \to \mathbf{C}$ *be monads with* $\mathbf{C}_T$ *partially additive. If* S *distributes over* T *and* $\overline{S} : \mathbf{C}_T \to \mathbf{C}_T$ *perserves partial additivity, then* $\mathbf{C}_{TS}$ *is partially additive.*

Corollary 3.8 *Let* $\mathbf{C}$ *have finite products, let* $T : \mathbf{C} \to \mathbf{C}$ *be a strong monad with* $\mathbf{C}_T$ *partially additive, and let* M *be a monoid in* $\mathbf{C}$*. Then* $T(-\times M) : \mathbf{C} \to \mathbf{C}$ *is a monad and, if* $\overline{-\times M} : \mathbf{C}_T \to \mathbf{C}_T$ *preserves partial additivity, then* $\mathbf{C}_{T(-\times M)}$ *is partially additive.*

The rest of the paper studies two applications of this theorem.

As a sanity check, we can verify that using the trivial one-element monoid to model time gives back the original semantics. Let $(M, e, m) = (1, !_1, !_{1\times 1})$ and ob-

---

[4] Haghverdi [25] observes that this is the same as a functor enriched over the category of *partially additive monoids* and calls such a functor *additive*.

serve that, since terminal objects are the unit for product, $X \times 1 \cong X$, the extended Kleisli category collapses to the original one: $\mathbf{C}_{T(-\times 1)} \cong \mathbf{C}_T$. Intuitively, since a terminal object has exactly one point, modeling delay in the trivial monoid amounts to throwing away the language's information about the duration of computations.

4 Adding Delay to Par

Before extending a probabilistic variant of ISL_0 with delay, we first consider $\mathrm{ISL}_{\mathsf{while}}$ since it has a simple semantics over partial functions. Then, to obtain a useful intermediate between the two, we specialize the deterministic semantics to model probabilistic delays while retaining deterministic behavior on states.

4.1 Deterministic Delay

Consider the $\mathbf{Par} \cong \mathbf{Set}_{-_\perp}$ semantics for $\mathrm{ISL}_{\mathsf{while}}$ mentioned in §2. To extend the semantics with delay by following the program outlined in §3, we need a few things: a monoid M in $\mathbf{Set}$, a strength for $-_\perp$ to make $(- \times M)_\perp$ a monad, and preservation of partial additivity for the lifted monad $\overline{- \times M}$ on $\mathbf{Par}$. These things are easy to obtain.

Fix a monoid (M, e, m) in $\mathbf{Set}$ to model time. It is well known that $-_\perp$ is strong [38], making the composite functor $(- \times M)_\perp$ a monad.

Proposition 4.1 $-_\perp : \mathbf{Set} \to \mathbf{Set}$ *is a strong monad with tensorial strength*

$$\begin{aligned} t_{X,Y} : X \times Y_\perp &\to (X \times Y)_\perp \\ (x, y) &\mapsto (x, y) \\ (x, \perp) &\mapsto \perp \end{aligned}$$

Corollary 4.2 *The functor* $(- \times M)_\perp : \mathbf{Set} \to \mathbf{Set}$ *is a monad with unit and multiplication*

$$\begin{aligned} \eta_X : X &\to (X \times M)_\perp \\ x &\mapsto (x, e) \end{aligned} \qquad \begin{aligned} \mu_X : ((X \times M)_\perp \times M)_\perp &\to (X \times M)_\perp \\ ((x, b), a) &\mapsto (x, m(a, b)) \\ (\perp, a) &\mapsto \perp \\ \perp &\mapsto \perp \end{aligned}$$

The lifted monad can be shown to preserve partial additivity by straightforward reasoning with sums of partial functions.

Proposition 4.3 *The monad* $\overline{- \times M} : \mathbf{Set}_{-_\perp} \to \mathbf{Set}_{-_\perp}$ *preserves partial additivity.*

Corollary 4.4 *The category* $\mathbf{Set}_{(-\times M)_\perp}$ *is partially additive.*

Instantiating the monadic semantics of $\mathrm{ISL}_{\mathsf{while,wait}}$ from §2 and §3 with $\mathbf{C}_T = \mathbf{Set}_{(-\times M)_\perp}$, we see that statements are interpreted as partial functions where the result contains a delay component capturing the cumulative delay incurred by the statement:

$$[\![S]\!]^\Gamma : [\![\Gamma]\!] \to ([\![\Gamma]\!] \times M)_\perp$$

In particular, the wait statement terminates and records the specified delay:

$$[\![\mathsf{wait}\ E]\!]^\Gamma(\overline{x}) = (\overline{x}, [\![E : \mathsf{time}]\!]^\Gamma(\overline{x}))$$

Sequenced statements combine their delays with the monoid multiplication m, and pure statements represent the fact that they incur no delay with the monoid unit e.

4.2 Probabilistic Delay

Deterministic delays are too simple to model systems with complex time behavior. A more expressive language would be able to represent the duration of complex computations stochastically by sampling delays from probability distributions. Here we consider a language with probabilistic delays and deterministic state behavior before moving to a fully probabilistic language in the next section.

Consider a variation on $\mathrm{ISL}_{\mathsf{while,wait}}$ where delays are sampled from probability distributions. This is easily achieved by using the time type to classify expressions describing *distributions* over time. One way to understand such an expression language is to view expressions as *deterministically* specifying the *probability* distribution over their possible delays. For the sake of examples consider including expressions $\mathsf{exp}(E)$, an exponential distribution with parameter E, and $\mathsf{bern}(E)$, a Bernoulli distribution yielding false with probability E and true with probability $1 - E$ (with suitable default behavior if E is out of range).

Since expressions E : time now describe distributions instead of what we were previously thinking of as deterministic durations, the language is easily modeled within the **Par** semantics just presented: use a monoid of probability distributions over time. But what kind of monoidal structure is meaningful? In particular, what should we use for multiplication?

Consider how Kleisli composition should operate on a pair of sample denotations in $\mathbf{Set}_{(-\times\Pi\mathbb{R}^+)_\perp}$, where we abbreviate $T = (- \times \Pi\mathbb{R}^+)_\perp$:

$$[\![\Gamma]\!]_T \xrightarrow{[\![v := v+1; \mathsf{wait}\ \mathsf{exp}(2)]\!]^v_T} [\![\Gamma]\!]_T \xrightarrow{[\![v := v+1; \mathsf{wait}\ \mathsf{exp}(4)]\!]^v_T} [\![\Gamma]\!]_T$$

Since the mean of $\mathsf{exp}(\lambda)$ is $\frac{1}{\lambda}$, we expect this composition to increment v twice with mean delay $\frac{1}{2} + \frac{1}{4} = \frac{3}{4}$. Put differently, if we define a random variable observing the delay of each program, then we want the random variable of the composite to be the sum of those for the first and second program. This suggests using *convolution of measures* as our monoid multiplication.

Definition 4.5 Given $\mu, \nu \in \Pi\mathbb{R}^+$, their *convolution* $\mu * \nu \in \Pi\mathbb{R}^+$ is:

$$\mu * \nu = (\mu \times \nu)_+ = A \mapsto \iint \chi_A(x+y)\ \mu(dx)\ \nu(dy)$$

where $\chi_A(x) = 1$ if $x \in A$ and 0 otherwise.

Proposition 4.6 $(\Pi\mathbb{R}^+, \delta_0, *)$ *is a monoid in* **Set***, where the unit* δ_0 *is the point mass at* 0 *and multiplication* $*$ *is convolution.*

Generalizing this from $\mathbb{R}^+$ to arbitrary monoids is straightforward: replace the monoid $(\mathbb{R}^+, 0, +)$ with any monoid (M, e, m) that is also a measurable space, and the integral will be defined if multiplication m is measurable. More simply, we can just ask for a monoid in **Meas** since the unit e is always measurable: given a monoid $(|\mathcal{M}|, e, m)$ in **Set** where $\mathcal{M}$ is a measurable space, $1 \xrightarrow{e} |\mathcal{M}|$ is measurable since both subsets of the terminal object 1 are measurable.

Definition 4.7 A *measurable monoid* is a monoid in **Meas**.

We write $\mathcal{M}$ for measurable monoids and M for monoids in **Set**.

We can now generalize convolution to arbitrary measurable monoids and get a monoidal structure on their spaces of probability distributions.

Definition 4.8 Given $\mu, \nu \in \Pi\mathcal{M}$ over a measurable monoid $(\mathcal{M}, e, m)$, their *convolution* $\mu *_m \nu \in \Pi\mathcal{M}$ is:

$$\mu *_m \nu = (\mu \times \nu)_m = A \mapsto \iint_{\mathcal{M}^2} \chi_A(m(a,b))\ \mu(da)\ \nu(db)$$

We write $\mu *_m \nu$ as $\mu * \nu$ when the monoid is clear from context.

Proposition 4.9 *If* $(\mathcal{M}, e, m)$ *is a measurable monoid then* $(\Pi\mathcal{M}, \delta_e, *_m)$ *is a monoid in* **Set** *where the unit* δ_e *is the point mass at* e *and multiplication* $*_m$ *is convolution of measures.*

Equipped with a compelling monoidal structure over distributions, we can now instantiate the **Par** semantics from §4.1 and derive a model for the (deterministic) language with probabilistic delay. Type time now corresponds to distributions:

$$[\![\mathsf{time}]\!] \ \triangleq\ \Pi\mathcal{M}$$

Since expressions $E : \mathsf{time}$ now represent distributions and are interpreted as

$$[\![E : \mathsf{time}]\!]^\Gamma : [\![\Gamma]\!] \to \Pi\mathcal{M}$$

the abstract semantics for wait becomes:

$$[\![\mathsf{wait}\ E]\!]^\Gamma = {}_{[\![\Gamma]\!]} \xrightarrow{\langle 1, [\![E : \mathsf{time}]\!]^\Gamma \rangle} {}_{[\![\Gamma]\!] \times \Pi\mathcal{M}} \xrightarrow{\eta^T} {}_{T([\![\Gamma]\!] \times \Pi\mathcal{M})}$$

With $T = -_\perp$, the resulting $\mathbf{Set}_{(-\times\Pi\mathcal{M})_\perp}$ semantics is very close to the $\mathbf{Set}_{(-\times M)_\perp}$ semantics except it uses a monoid of distributions to interpret stochastic delay alongside deterministic behavior on states. The interpretation of statements is now

$$[\![S]\!]^\Gamma : [\![\Gamma]\!] \to ([\![\Gamma]\!] \times \Pi\mathcal{M})_\perp$$

The wait statement terminates and records the expressed distribution over time:

$$[\![\mathsf{wait}\ E]\!]^{\Gamma}(\overline{x}) = (\overline{x}, [\![E : \mathsf{time}]\!]^{\Gamma}(\overline{x}))$$

Sequenced statements combine their delay distributions by convolution. Pure statements represent the fact that they incur no delay with δ_e, the point mass at the monoid unit e.

5 Adding Delay to SRel

The language in the previous section expresses stochastic computations with probabilistic delay but fails to capture systems that also have probabilistic behavior on states. To achieve both we add delay to the probabilistic language $\mathrm{ISL}_{\mathsf{while},+}$ from §2; probabilistic delay falls out of the combination. This section uses the method from §3 to extend the **SRel** semantics for $\mathrm{ISL}_{\mathsf{while},+}$ to also model delay, giving a semantics for a probabilistic language with stochastic temporal behavior.

Consider the $\mathbf{SRel} \cong \mathbf{Meas}_{\Pi}$ semantics for $\mathrm{ISL}_{\mathsf{while},+}$ described in §2. Following the method in §3 is again straightforward: take a monoid $\mathcal{M}$ in **Meas**, construct a strength for Π to make $\Pi(- \times \mathcal{M})$ a monad, and establish that the lifted monad $\overline{- \times \mathcal{M}} : \mathbf{SRel} \to \mathbf{SRel}$ preserves partial additivity.

Fix a measurable monoid $\mathcal{M}$ to model time. Strength for Π is straightforward:

Proposition 5.1 $\Pi : \mathbf{Meas} \to \mathbf{Meas}$ *is a strong monad with tensorial strength*

$$\begin{aligned} t_{X,Y} : X \times \Pi Y &\to \Pi(X \times Y) \\ (x, \nu), C &\mapsto \nu(C_x) \end{aligned}$$

where $C_x = \{y : (x, y) \in C\}$. *Equivalently,* t *maps to the product measure*

$$t_{X,Y}(x, \nu) = \delta_x \times \nu$$

The equation follows easily by considering both sides' action on measurable rectangles, which uniquely determines product measures. Working with product measures then enables easy proofs of measurability and naturality, the former because the product map $\Pi X \times \Pi Y \xrightarrow{\times} \Pi(X \times Y)$ is measurable. Finally, proving the required equalities for strength is straightforward.

Proposition 5.1 gives a monad combining probability and delay:

Corollary 5.2 *The functor* $\Pi(- \times \mathcal{M}) : \mathbf{Meas} \to \mathbf{Meas}$ *is a monad with unit and multiplication*

$$\begin{aligned} \eta^{\Pi(-\times\mathcal{M})} &= 1 \xrightarrow{\eta^{\Pi} \circ \eta^{-\times\mathcal{M}}} \Pi(-\times\mathcal{M}) \\ \mu^{\Pi(-\times\mathcal{M})} &= \Pi(\Pi(-\times\mathcal{M})\times\mathcal{M}) \xrightarrow{\Pi\lambda(-\times\mathcal{M})} \Pi^2(-\times\mathcal{M})^2 \xrightarrow{\mu^{\Pi} \circ \mu^{-\times\mathcal{M}}} \Pi(-\times\mathcal{M}) \end{aligned}$$

where $\lambda = \overline{t}$ *is the distributive law obtained from strength for* Π*, and* $\circ$ *is horizontal composition.*

It is worthwhile to spell these out in detail. The unit just introduces the point-mass distribution and the monoid's unit: $\eta_X^{\Pi(-\times\mathcal{M})}(x) = \delta_{(x,e)}$. Multiplication is more interesting:

$$\mu_X^{\Pi(-\times\mathcal{M})}(P)(C) = \int_{\Pi(X\times\mathcal{M})\times\mathcal{M}} \nu(\{(x,b) : (x, m(b,a)) \in C\})\ P(d\nu, da)$$

where $P \in \Pi(\Pi(X \times \mathcal{M}) \times \mathcal{M})$ and $C \in \Sigma_{X\times\mathcal{M}}$. The behavior of $\mu^{\Pi(-\times\mathcal{M})}$ is similar to μ^{Π}, which averages a distribution over distributions down to a single distribution, except $\mu^{\Pi(-\times\mathcal{M})}$ must also incorporate the monoid action.

Although the above multiplication is complicated, it corresponds to a nice Kleisli composition and supports a satisfying direct presentation analogous to **SRel**:

Definition 5.3 The category $\mathbf{TSRel}_{\mathcal{M}}$ of $\mathcal{M}$-*timed stochastic relations* has measurable spaces as objects and an arrow $X \xrightarrow{f} Y$ is a function $f : X \times \Sigma_{Y\times\mathcal{M}} \to [0,1]$ such that every $f(x,-)$ is a subprobability measure and every $f(-,C)$ is measurable. The identity arrow $X \xrightarrow{1_X} X$ is $1_X(x,C) = \delta_{(x,e)}(C)$, and the composition $X \xrightarrow{f} Y \xrightarrow{g} Z$ is

$$(f;g)(x,C) = \int_{Y\times\mathcal{M}} \int_{Z\times\mathcal{M}} f(x, d(y,a))\ g(y, d(z,b))\ \chi_C((z, m(b,a)))$$

We think of a stochastic relation $X \xrightarrow{f} Y$ as giving the probability that a point in X relates to a measurable subset of Y; similarly, we think of a timed stochastic relation $X \xrightarrow{f} Y$ as doing the same for measurable subsets of $Y \times \mathcal{M}$—points in Y and values in the monoid $\mathcal{M}$, which we interpret as time delay. We can then read composition as: $f;g$ relates x to C if f relates x to y with delay a, g relates y to z with delay b, and z paired with the aggregate delay $m(b,a)$ is in C. The probability that $f;g$ relates x to C is then the sum of the probabilities of each of these sufficient cases.

Since currying a timed stochastic relation $X\times\Sigma_{Y\times\mathcal{M}} \xrightarrow{f} [0,1]$ produces a Kleisli arrow $X \xrightarrow{\tilde{f}} \Pi(Y\times\mathcal{M})$ we expect to also have the isomorphism $\mathbf{TSRel}_{\mathcal{M}} \cong \mathbf{Meas}_{\Pi(-\times\mathcal{M})}$. Indeed, using change of variables and Fubini's theorem it is a straightforward calculation to show that Kleisli composition is just a curried version of composition for timed stochastic relations, and the isomorphism is then easy to construct. We freely interchange $\mathbf{Meas}_{\Pi(-\times\mathcal{M})}$ and $\mathbf{TSRel}_{\mathcal{M}}$ to take advantage of both the curried and uncurried forms of timed stochastic relations.

Now that we have a category $\mathbf{TSRel}_{\mathcal{M}}$ capable of modeling probabilistic choice and delay, the last step is to show that it can also interpret iteration by establishing partial additivity. Because **SRel** is partially additive, it suffices to show that the lifted monad $\overline{-\times\mathcal{M}}$ on **SRel** preserves partial additivity. This also follows by an elementary measure-theoretic argument.

Proposition 5.4 *The monad* $\overline{-\times\mathcal{M}} : \mathbf{Meas}_{\Pi} \to \mathbf{Meas}_{\Pi}$ *preserves partial additivity.*

Corollary 5.5 *The category* $\mathbf{Meas}_{\Pi(-\times\mathcal{M})}$ *is partially additive.*

Additionally, we recover the **SRel** semantics as $\mathbf{TSRel}_1$ which ignores delay by collapsing everything in the one-element monoid.

$\mathbf{TSRel}_{\mathcal{M}}$ interprets delay statements $\mathsf{wait}\ E$ for deterministic durations as

$$[\![\mathsf{wait}\ E]\!]^\Gamma = {}_{[\![\Gamma]\!]} \xrightarrow{\langle 1, [\![E : \mathsf{time}]\!]^\Gamma\rangle} {}_{[\![\Gamma]\!]\times\mathcal{M}} \xrightarrow{\eta} {}_{\Pi([\![\Gamma]\!]\times\mathcal{M})}$$

but what about probabilistic delays like in §4.2? The monad $\Pi(- \times \mathcal{M})$ gives distributions over both state and time, so we expect to be able to model these as well *without* taking $\mathcal{M}$ itself to be a space of distributions.

So that deterministic and probabilistic delay can coexist, we introduce a second delay statement, pwait, and a new family of types:

Probabilistic Delay Extension for $\mathbf{TSRel}_{\mathcal{M}}$: pwait

Syntax:	**Typing Rules:**
$\tau ::= \cdots \mid \mathsf{prob}\ \tau$ $S ::= \cdots \mid \mathsf{pwait}\ E$	$\dfrac{\Gamma \vdash E : \mathsf{prob\ time}}{\Gamma \vdash \mathsf{pwait}\ E}$

The statement $\mathsf{pwait}\ E$ samples the time distribution $E : \mathsf{prob\ time}$ and delays execution by the resulting number of time units. Types $\mathsf{prob}\ \tau$ denote spaces of probability distributions over values of type τ:

$$[\![\mathsf{prob}\ \tau]\!] \triangleq \Pi[\![\tau]\!]$$

This means that $\mathsf{prob}\ \tau$ expressions denote arrows into these spaces of distributions:

$$[\![E : \mathsf{prob}\ \tau]\!]^\Gamma : [\![\Gamma]\!] \to \Pi[\![\tau]\!]$$

As in §4.2, we assume expressions for exponential distributions $\mathsf{exp}(E)$ and Bernoulli distributions $\mathsf{bern}(E)$, but now with type $\mathsf{prob}\ \tau$ where $E : \tau$.

We expect $[\![\mathsf{pwait}\ E]\!]^\Gamma$ to be a timed stochastic relation that relates a state $\overline{x}$ only to itself with delay sampled from $[\![E : \mathsf{prob\ time}]\!]^\Gamma(\overline{x})$:

$$\begin{aligned}[\![\mathsf{pwait}\ E]\!]^\Gamma(\overline{x}, C) &= \int_{\mathcal{M}} \chi_C(\overline{x}, a)\ [\![E : \mathsf{prob\ time}]\!]^\Gamma(\overline{x}, da) \\ &= (\delta_{\overline{x}} \times [\![E : \mathsf{prob\ time}]\!]^\Gamma(\overline{x}))(C)\end{aligned}$$

This semantics is concisely expressible using strength for Π.

$\mathbf{TSRel}_{\mathcal{M}}$ Semantics of pwait:

$$[\![\mathsf{pwait}\ E]\!]^\Gamma \triangleq {}_{[\![\Gamma]\!]} \xrightarrow{\langle 1, [\![E : \mathsf{prob\ time}]\!]^\Gamma\rangle} {}_{[\![\Gamma]\!]\times\Pi\mathcal{M}} \xrightarrow{t} {}_{\Pi([\![\Gamma]\!]\times\mathcal{M})}$$

We can even characterize probabilistic delay in $\mathbf{TSRel}_{\mathcal{M}}$ in terms of the original probabilistic delay in $\mathbf{Set}_{(-\times\Pi\mathcal{M})_\perp}$ given in §4.2 where state transitions were deterministic:

$$[\![\mathsf{pwait}\ E]\!]^\Gamma = {}_{[\![\Gamma]\!]} \xrightarrow{[\![\mathsf{wait}\ E]\!]^\Gamma} {}_{([\![\Gamma]\!]\times\Pi\mathcal{M})_\perp} \xrightarrow{\varphi} {}_{\Pi([\![\Gamma]\!]\times\Pi\mathcal{M})} \xrightarrow{\Pi t} {}_{\Pi^2([\![\Gamma]\!]\times\mathcal{M})} \xrightarrow{\mu^\Pi} {}_{\Pi([\![\Gamma]\!]\times\mathcal{M})}$$

where $[\![\mathsf{wait}\ E]\!]^\Gamma$ is the interpretation in $\mathbf{Set}_{(-\times\Pi\mathcal{M})_\perp}$ of $\mathsf{wait}\ E$ from the language of §4.2. We bridge the two semantic categories with a map $\varphi : -_\perp \to \Pi$ that sends points to their point-mass distributions and failure to the distribution that measures everything as 0:

$$\begin{aligned}\varphi_X\ :\ X_\perp &\to \Pi X\\ x &\mapsto \delta_x\\ \perp &\mapsto 0\end{aligned}$$

$\mathbf{TSRel}_\mathcal{M}$ models probabilistic choice and probabilistic delay, and both operators are based on sampling a probability distribution. This suggests that extending the language with a construct to sample probability distributions should enable us to express both operators.

Sampling Extension: $\leftarrow$

Syntax:

$$S\ ::=\ \cdots \mid v \leftarrow E$$

Typing Rules:

$$\frac{\Gamma, v:\tau, \Gamma' \vdash E : \mathsf{prob}\ \tau}{\Gamma, v:\tau, \Gamma' \vdash v \leftarrow E}\ (v \notin \Gamma)$$

The statement $v \leftarrow E$ samples the distribution $E : \mathsf{prob}\ \tau$ and assigns the result to v. It is tempting to formulate this as an expression, like $\mathsf{sample}(E)$, but doing so would introduce effects into the expression language and complicate our framework.

The sampling operator is easily modeled in our probabilistic categories. We define $[\![v \leftarrow E]\!]^\Gamma : [\![\Gamma]\!] \to \Pi S[\![\Gamma]\!]$ for any monad S on **Meas** that composes with Π; taking S to be 1 gives a denotation in **SRel**, and $- \times \mathcal{M}$ gives one in $\mathbf{TSRel}_\mathcal{M}$. Recall that distributions are interpreted as $[\![E : \mathsf{prob}\ \tau]\!]^\Gamma : [\![\Gamma]\!] \to \Pi[\![\tau]\!]$.

Monadic Semantics of $\leftarrow$:

$$[\![v \leftarrow E]\!]^{\Gamma,v:\tau,\Gamma'} \triangleq$$

$${}_{[\![\Gamma]\!]\times[\![\tau]\!]\times[\![\Gamma']\!]} \xrightarrow{\langle \pi_1, [\![E : \mathsf{prob}\ \tau]\!]^{\Gamma,v:\tau,\Gamma'}, \pi_3\rangle} {}_{[\![\Gamma]\!]\times\Pi[\![\tau]\!]\times[\![\Gamma']\!]} \xrightarrow{\hat{t}} {}_{\Pi([\![\Gamma]\!]\times[\![\tau]\!]\times[\![\Gamma']\!])} \xrightarrow{\Pi\eta^S} {}_{\Pi S([\![\Gamma]\!]\times[\![\tau]\!]\times[\![\Gamma']\!])}$$

The key is the arrow

$$\hat{t}_{X,Y,Z} = {}_{X\times\Pi Y\times Z} \xrightarrow{t\times 1} {}_{\Pi(X\times Y)\times Z} \xrightarrow{\bar{t}} {}_{\Pi(X\times Y\times Z)}$$

which reifies the distribution produced by $[\![E : \mathsf{prob}\ \tau]\!]^\Gamma$ as a probability distribution over the whole state space.

We can now express $\mathsf{pwait}\ E$ simply by sampling E and then waiting the length of time specified by the result. Sampling also generalizes probabilistic choice: sample a Bernoulli distribution and branch. The following proposition captures these intuitions using the $\mathbf{TSRel}_{\mathcal{M}}$ semantics, illustrating how our model can validate equivalences between stochastic programs.

Proposition 5.6

(a) $[\![\mathsf{pwait}\ E]\!]^\Gamma = [\![\mathsf{let}\ v : \mathsf{time} = 0\ \mathsf{in}\ v \leftarrow E; \mathsf{wait}\ v]\!]^\Gamma \qquad (v \notin \Gamma)$

(b) $[\![S_1 +_p S_2]\!]^\Gamma = [\![\mathsf{let}\ v : \mathsf{bool} = \mathsf{true}\ \mathsf{in}\ v \leftarrow \mathsf{bern}(p); \mathsf{if}\ v\ \mathsf{then}\ S_1\ \mathsf{else}\ S_2]\!]^\Gamma\ (v \notin \Gamma)$

6 Related Work

Related work broadly falls into three categories: models for stochastic temporal behavior, languages for expressing stochastic temporal behavior, and applications of the probability monad to develop semantic models.

Several frameworks exist to describe and model stochastic temporal behavior, including queueing systems [30], stochastic automata [15,16], generalised stochastic petri-nets [36], and generalised semi-Markov processes [22]. Our approach shares much in common with stochastic automata. Roughly speaking, stochastic automata extend standard deterministic automata with clock variables, just like timed automata [3]. Upon entering a state, some of those clocks are set by sampling a probability distribution, and then all clocks decrement at the same rate. Transitions are labeled with an input symbol and a set of clocks, and a transition is enabled once its labeling clocks reach 0. Stochastic automata are usually interpreted using a probabilistic transition system with two classes of states, states from which nondeterministic choices are made, and states from which probabilistic choices are made, the latter essentially corresponding to probabilistic delays. It is possible to view our work as a partial reframing of stochastic automata in a categorical setting, providing them with a direct transition semantics.

As far as languages for stochastic temporal behavior are concerned, much of the original impetus came from finding reasonable languages in which to compositionally and finitely represent models for the study of stochastic temporal behavior in systems with soft constraints. Stochastic process calculi, with their support for concurrency and their ready compositionality, have proved popular [23,43,26,8,27]. Stochastic process calculi, especially derived from the stochastic pi calculus [43], are especially popular for biological modeling [46,45,14,10,17]. In the tradition of process calculi, the semantics of those languages is operational, using an annotated reduction semantics that records the rate of reaction (which correspond, roughly, to the time delays introduced in the reduction). Stochastic process calculi generally use exponential distributions to model delays, and the reduction semantics can be shown to yield continuous-time Markov processes. Restricting to Markov processes implies that we can reason more efficiently about the resulting processes expressed in the stochastic pi calculus, or stochastic automata, for that matter; see, for instance, Bryans et al. [13]. Priami [44] shows how to extend the stochastic pi calculus to general distributions. Recently, Klin and Sassone [31] developed a general operational reduction semantics for stochastic process calculi that unifies much of the ad hoc presentation in earlier papers. Our work is essentially denotational and can be seen as complementary. We have not yet applied it to process calculi.

Variants of the Giry probability monad [21], based on earlier work by Lawvere [34], have been the basis of most denotational semantics for probabilistic languages [48,28,41,50]. Doberkat [18] offers an exhaustive overview of the probability monad and stochastic relations from a categorical perspective. Doberkat extends stochastic relations with monoids to model software architectures, but he considers component pipelines without cycles, whereas iteration is central to our study. Ramsey and Pfeffer [47] use the probability monad as a semantic foundation for a stochastic lambda calculus. The interaction between the probability monad and other monads has been studied in a few contexts. Breugel [50] shows that a distributive law between the Giry probability monad and the partiality monad $-_\perp$ gives rise to the subprobability monad Π. Other distributive laws relate the probability monad to nondeterminism—see Varacca and Winskel [51] and references therein. Our work can be seen as studying the interaction of the probability monad with various forms of monoid tensor addition.

7 Conclusion

Our paper presents an approach to adding delay to a categorical semantics for languages of while loops by generalizing the category of stochastic relations $\mathbf{SRel}$ to a family of categories of timed stochastic relations $\mathbf{TSRel}_\mathcal{M}$. Our approach is suitable for modeling both probabilistic choice and stochastic temporal behavior in a single categorical framework.

Our work is preliminary, and several questions remain. For instance, $\mathbf{TSRel}_\mathcal{M}$ is parameterized by a monoid $\mathcal{M}$; what is the role of the "re-timing" functors $\mathbf{TSRel}_\mathcal{M} \to \mathbf{TSRel}_\mathcal{N}$ induced by monoid homomorphisms? What is the exact relationship between $\mathbf{TSRel}_\mathcal{M}$ and continuous-time Markov chains, which must appear in $\mathbf{TSRel}_\mathcal{M}$ in some form? Another question relates to time dependence. Our transitions cannot depend on the time at which a transition occurs since time is not provided in the domains of arrows in our categories. Making transitions time dependent is not difficult, but in some sense everything would then collapse down to $\mathbf{SRel}$, at least in the probabilistic case: time-dependent transitions can be encoded by including time as part of the state and restricting to morphisms that update the time correctly. We have not explored the exact relationship between time-dependent models and our own. In addition, it would be interesting to explore extensions to higher-order recursive languages.

Finally, we need to examine the relationship between our models and the more operational models used in the stochastic process calculus literature. A starting point is to use our categories or variants thereof to give a semantics to stochastic process calculi. We hope to report on this research in the near future.

Acknowledgement

Thanks to Jesse Tov and Aaron Turon for comments on a preliminary version of this paper.

References

[1] Abramsky, S., *Retracing some paths in process algebra*, in: *Proc. 7th International Conference on Concurrency Theory (CONCUR'96)*, Lecture Notes in Computer Science **1119** (1996), pp. 1–17.

[2] Adámek, J., H. Herrlich and G. E. Strecker, "Abstract and Concrete Categories: The Joy of Cats," John Wiley & Sons, 1990.

[3] Alur, R. and D. L. Dill, *A theory of timed automata*, Theoretical Computer Science **126** (1994), pp. 183–235.

[4] Asarin, E., P. Caspi and O. Maler, *Timed regular expressions*, Journal of the ACM **49** (2002), pp. 172–206.

[5] Ash, R. B. and C. A. Doléans-Dade, "Probability & Measure Theory," Academic Press, 1999.

[6] Barr, M. and C. Wells, "Toposes, Triples and Theories," Grundlehren der mathematischen Wissenschaften **278**, Springer-Verlag, New York, 1985.

[7] Beck, J., *Distributive laws*, in: *Seminar on Triples and Categorical Homology Theory*, Lecture Notes in Mathematics **80** (1969), pp. 119–140.

[8] Bernardo, M. and R. Gorrieri, *A tutorial on EMPA: A theory of concurrent processes with nondeterminism, priorities, probabilities and time*, Theoretical Computer Science **202** (1998), pp. 1–54.

[9] Billingsley, P., "Probability and Measure," Wiley-Interscience, 1995.

[10] Bortolussi, L. and A. Policriti, *Modeling biological systems in stochastic concurrent constraint programming*, Constraints **13** (2008), pp. 66–90.

[11] Bowman, H., L. Blair, G. S. Blair and A. Chetwynd, "Formal Specification of Distributed Multimedia Systems," University College London Press, 1998.

[12] Brown, D. and R. Pucella, *Categories of timed stochastic relations*, Technical Report NU-CCIS-09-02, Northeastern University (2009).

[13] Bryans, J., H. Bowman and J. Derrick, *Model checking stochastic automata*, ACM Transactions on Computational Logic **4** (2003), pp. 452–492.

[14] Ciocchetta, F. and J. Hillston, *Bio-PEPA: An extension of the process algebra PEPA for biochemical networks*, Electronic Notes in Theoretical Computer Science **194** (2008), pp. 103–117.

[15] D'Argenio, P., "Algebras and automata for timed and stochastic systems," Ph.D. thesis, University of Twente, Enschede, The Netherlands (1999).

[16] D'Argenio, P. R. and J.-P. Katoen, *A theory of stochastic systems part I: Stochastic automata*, Information and Computation **203** (2005), pp. 1–38.

[17] Demattè, L., C. Priami and A. Romanel, *Modelling and simulation of biological processes in BlenX*, SIGMETRICS Performance Evaluation Review **35** (2008), pp. 32–39.

[18] Dobertkat, E.-E., "Stochastic Relations: Foundations for Markov Transition Systems," Chapman & Hall/CRC, 2007.

[19] Folland, G. B., "Real Analysis: Modern Techniques and Their Applications," Wiley-Interscience, 1984.

[20] Gibbons, J., *Conditionals in distributive categories*, Technical report, Oxford Brookes University (1997).

[21] Giry, M., *A categorical approach to probability theory*, in: B. Banaschewski, editor, *Categorical Aspects of Topology and Analysis*, Lecture Notes in Mathematics **915** (1981), pp. 68–85.

[22] Glynn, P. W., *A GSMP formalism for discrete event simulation*, Proceedings of the IEEE **77** (1989), pp. 14–23.

[23] Götz, N., U. Herzog and M. Rettelbach, *Multiprocessor and distributed system design: The integration of functional specification and performance analysis using stochastic process algebras*, in: *Performance/SIGMETRICS Tutorials*, 1993, pp. 121–146.

[24] Gunter, C. A., "Semantics of Programming Languages: Structures and Techniques," MIT Press, 1992.

[25] Haghverdi, E., "A Categorical Approach to Linear Logic, Geometry of Proofs and Full Completeness," Ph.D. thesis, University of Ottawa (2000).

[26] Hillston, J., "A Compositional Approach to Performance Modelling," Distinguished Dissertations in Computer Science, Cambridge University Press, 1996.

[27] Hillston, J., *Process algebras for quantitative analysis*, in: *Proc. 20th Annual IEEE Symposium on Logic in Computer Science (LICS'05)* (2005), pp. 239–248.

[28] Jones, C. and G. D. Plotkin, *A probabilistic powerdomain of evaluations*, in: *Proc. 4th Annual IEEE Symposium on Logic in Computer Science (LICS'89)* (1989), pp. 186–195.

[29] Kallenberg, O., "Foundations of Modern Probability," Springer-Verlag, 2002.

[30] Kleinrock, L., "Queueing Systems, Volume I: Theory," John Wiley & Sons, 1975.

[31] Klin, B. and V. Sassone, *Structural operational semantics for stochastic process calculi*, in: *Proc. 11th International Conference on Foundations of Software Science and Computation Structures (FOSSACS'08)*, Lecture Notes in Computer Science **4962** (2008), pp. 428–442.

[32] Kozen, D., *Semantics of probabilistic programs*, Journal of Computer and Systems Sciences **22** (1981), pp. 328–350.

[33] Kozen, D., *A probabilistic PDL*, Journal of Computer and Systems Sciences **30** (1985), pp. 162–178.

[34] Lawvere, F. W., *The category of probabilistic mappings* (1962), unpublished manuscript.

[35] Manes, E. and M. Arbib, "Algebraic Approaches to Program Semantics," Springer-Verlag, 1986.

[36] Marsam, M. A., G. Conte and G. Balbo, *A class of generalised stochastic petri nets for the performance evaluation of multiprocessor systems*, ACM Transactions on Computer Systems **2** (1984), pp. 93–122.

[37] Moggi, E., *Computational lambda-calculus and monads*, in: *Proc. 4th Annual IEEE Symposium on Logic in Computer Science (LICS'89)* (1989), pp. 14–23.

[38] Moggi, E., *Notions of computation and monads*, Information and Computation **93** (1991), pp. 55–92.

[39] Mulry, P. S., *Lifting theorems for Kleisli categories*, in: *Proc. 9th International Conference on Mathematical Foundations of Programming Semantics (MFPS 9)*, Lecture Notes in Computer Science **802** (1994), pp. 304–319.

[40] Mulry, P. S., *Lifting results for categories of algebras*, Theoretical Computer Science **278** (2002), pp. 257–269.

[41] Panangaden, P., *The category of Markov kernels*, in: *Proc. 1st International Workshop on Probabilistic Methods in Verification (PROBMIV'98)*, Electronic Notes in Theoretical Computer Science **22** (1999), pp. 171–187.

[42] Pierce, B. C., "Types and Programming Languages," MIT Press, Cambridge, MA, USA, 2002.

[43] Priami, C., *Stochastic pi calculus*, The Computer Journal **38** (1995), pp. 578–589.

[44] Priami, C., *Stochastic pi-calculus with general distributions*, in: *Proc. 4th Workshop on Process Algebras and Performance Modelling (PAPM '96)*, 1996, pp. 41–57.

[45] Priami, C. and P. Quaglia, *Modeling the dynamics of bio-systems*, Briefings in Bioinformatics **5** (2004), pp. 259–269.

[46] Priami, C., A. Regev, E. Shapiro and W. Silverman, *Application of a stochastic name-passing calculus to representation and simulation of molecular processes*, Information Processing Letters **80** (2001), pp. 25–31.

[47] Ramsey, N. and A. Pfeffer, *Stochastic lambda calculus and monads of probability distributions*, in: *Proc. 29th Annual ACM Symposium on Principles of Programming Languages (POPL'02)* (2002), pp. 154–165.

[48] Saheb-Djahromi, N., *CPOs of measures for nondeterminism*, Theoretical Computer Science **12** (1980), pp. 19–37.

[49] Szigeti, J., *On limits and colimits in the Kleisli category*, Cahiers de Topologie et Géométrie Différentielle Catégoriques **24** (1983), pp. 381–391.

[50] van Breugel, F., *The metric monad for probabilistic nondeterminism* (2005), unpublished manuscript, available from http://www.cse.yorku.ca/~franck/research/drafts.

[51] Varacca, D. and G. Winskel, *Distributing probability over non-determinism*, Mathematical Structures in Computer Science **16** (2006), pp. 87–113.

[52] Winskel, G., "The Formal Semantics of Programming Languages," MIT Press, 1993.