SEMIRING MODULE POWERDOMAINS

by

Michael G. Main*

CU-CS-286-84 December, 1984

*Department of Computer Science, University of Colorado,
Boulder, Colorado 80309

ANY OPINIONS, FINDINGS, AND CONCLUSIONS OR RECOMMENDATIONS EXPRESSED IN THIS PUBLICATION ARE THOSE OF THE AUTHOR AND DO NOT NECESSARILY REFLECT THE VIEWS OF THE NATIONAL SCIENCE FOUNDATION.

SEMIRING MODULE POWERDOMAINS*

Michael G. Main
Department of Computer Science
University of Colorado
Boulder, CO 80309 USA

ABSTRACT

One approach to the semantics of nondeterministic computations is to provide a CPO with an associative binary operation. An element $x+y$ in such a CPO is a nondeterministic choice between x and y. Different aspects of nondeterminism have been studied by imposing various axioms on the binary operation (*e.g.,* the "must" semantics requires $x+y \sqsubseteq x$). An alternative to the axiom systems is proposed here: equipping a CPO with the algebraic structure of a semiring module. By choosing different underlying semirings we can study different aspects of nondeterminism, and the algebraic properties of the modules can supplement the usual order-theoretic techniques. A detailed example is given for discrete probabilistic nondeterminism. The paper also shows how to add the module structure in a universal way, using a previously known tensor construction.

KEYWORDS: nondeterminism, powerdomains, probabilistic semantics.

*This research has been supported in part by National Science Foundation grant DCR-8402341.

1. INTRODUCTION

In the order-theoretic semantics of programming languages, state spaces of computations are ordered domains, such as ω-complete partial orders (CPOs). One approach to nondeterministic computations is to equip a CPO with an associative binary operation (written here as $+$). A nondeterministic computation can have a result such as $x+y$, which indicates that the computation may produce x under some conditions and produce y at other times.

Different aspects of nondeterminism are studied by imposing different axioms on the binary operation. For example, Plotkin's original powerdomain employed a commutative, continuous and idempotent operation [10]. An alternative powerdomain, originally given by Smyth [13] uses the additional axiom $x+y \sqsubseteq x$, to capture the intuition that "better information" corresponds to "less nondeterminism". The opposite axiom $x \sqsubseteq x+y$ arises when we study what a computation *may* do ($x+y$ has more possibilities than x alone). These three powerdomains and some variations have attracted considerable recent interest [1,3,4,9,14,15].

This paper further develops the idea of CPOs for nondeterministic computations. In contrast to the "axiom" approach, different aspects of nondeterminism are examined by equipping a CPO with the algebraic structure of a semiring module. The classical powerdomains are treated as special cases, and additional aspects such as probabilistic programs and fairness can be captured. The prospect of the new approach is that the algebraic structure of the modules can supplement the usual order-theoretic techniques in proofs of program properties. The basic ideas of this approach are given in section 2. Section 3 gives a detailed treatment of discrete probabilistic nondeterminism. In section 4, a universal (or free) construction of a CPO semiring-module is given in terms of a previously known tensor product construction.

Throughout the paper, the term CPO refers to an ω-complete partial order, which is a partially ordered set (the order denoted $\sqsubseteq$) with a least element and least upper bounds of all countably infinite increasing chains (called ω-chains). A function $f: C \rightarrow D$ between two CPOs is *strict* if it preserves the least element, and *monotonic* if $x \sqsubseteq y$ in C implies $f(x) \sqsubseteq f(y)$ in D. A *continuous* function is monotonic and preserves least upper bounds of ω-chains.

2. ALGEBRAIC STRUCTURES FOR CPOS

This section introduces three algebraic structures to CPOs in order to study various aspects of nondeterminism. The starting point is an approach which has commonly been taken in the past decade: equipping CPOs with an associative binary operator, written $+$. For example, Hennessy and Plotkin considered CPOs with an associative, commutative, continuous and idempotent operation called the "union operation" [4]. As described in the introduction, an element $x+y$ indicates a nondeterministic choice between the states x and y.

Several variations on the union operation have been studied. One option is to drop the idempotence requirement, so that sometimes $x+x \neq x$. This is useful if we are counting the different ways that a certain output may arise (e.g. Benson [2]). Another alternative is to require a zero element with the property that $x+0 = x = 0+x$, for all x. Intuitively, zero is an impossible (or maybe just uninteresting) element of the domain, so that it may be ignored in a result like $x+0$. For example, in partial correctness semantics (e.g., Hoare [5]), it is useful to ignore non-terminating computations, so that zero can be the uninteresting state of nontermination.

2.1. CPO-MONOIDS.

The above ideas motivate this definition:

Definition 1: A *CPO-monoid* is a CPO D together with an associative, continuous binary operation $(+)$ and a zero element $0 \in D$ such that for all $x \in D$: $x+0 = 0 = 0+x$. A CPO-monoid is *commutative* provided that the binary operation is commutative. □

The problem of constructing a commutative CPO-monoid from a given CPO has been the subject of several studies initiated by Plotkin [10] and Smyth [13]. The idea is to add the monoid structure in a universal way, "so that every other solution is obtained from this one by a unique morphism" [7, page v]. Specifically, if D is a CPO, then the *free commutative CPO-monoid generated by D* is a commutative CPO-monoid F together with a strict continuous function $\eta{:}D \to F$. The function η must be *universal* in this sense: if M is any commutative CPO-monoid and $g{:}D \to M$ is a strict continuous function, then there is a unique monoid morphism $\hat{g}{:}F \to M$ making this diagram commute:

$$\begin{tikzcd}
D \arrow[r, "\eta"] \arrow[dr, "g"'] & F \arrow[d, dashed, "\hat{g}"] \\
& M
\end{tikzcd}$$

This sort of universal property is exactly the property possessed by Plotkin's and Smyth's original powerdomain constructions. Hennessy and Plotkin have observed that this universal construction (and other similar ones) can be obtained from categorical considerations [4]. More convenient detailed constructions have been given elsewhere [10,13,11], including examples that include non-strict functions and elimination of the zero element.

2.2. CPO-SEMIRINGS. We want to add some additional structure to CPO-monoids in order to study aspects of nondeterminism. The motivation is the observation that sometimes we have additional information about a nondeterministic program which can reduce the degree of nondeterminism. We want to capture this information in an algebraic framework so that the algebraic properties may supplement the usual order-theoretic properties.

The sort of information used in this paper is related to *execution paths*. An execution path is a sequence of "choices" made by a nondeterministic program during a specific execution of the program, resulting in a specific output. For each possible output x, there is a set of execution paths which result in x, called the *path-set* of x. The idea proposed here is this: with each possible output, we associate some property of its path-set. Here are some questions that the path-set properties might answer:

P1. Is the path-set empty? This is the usual property that is used when we are interested only in determining which outputs are possible.

P2. What is the cardinality of the path-set?

P3. What is the probability that an execution from the path-set will occur?

P4. Which paths in the set are fair? (Choose your favorite definition of "fair".)

With this in mind, we start with a set K of "properties" of path-sets. If we are interested only in P1, then K can simply be {**true**,**false**}. For P2, we might take K as the natural numbers (possibly extended to infinity).

What algebraic structures does K have? First, we require that K is itself is a CPO, with the intuitive meaning of $r \sqsubseteq s$ being that the property r provides less information than the property s. At this point we cannot specify $\sqsubseteq$ any more exactly, since the meaning of "less information" depends on what sort of semantics we are pursuing. Considerations from linear algebra suggest that we also provide K with these two associative binary operations:

Addition(+): If r and s are properties associated with the mutually exclusive path-sets R and S, then $r+s$ is the property for the union $R \cup S$. If K is {**true**,**false**} (for P1), then addition is logical **or**. When K is the natural numbers (for P2), then addition is the usual integer addition. K must have an element 0, which is the identity for +.

Multiplication($\cdot$): If r and s are properties associated with path-sets R and S, then $r \cdot s$ is the property for the concatenation RS. If K is {**true**,**false**}, then multiplication is logical **and**. When K is the natural numbers, then multiplication is the usual integer multiplication. K must also have an identity for this operation, denoted 1.

We require that the operations meet the axioms of a *CPO-semiring, defined here:*

Definition 2: A *CPO-semiring* is a CPO K together with two distinguished elements 0 and 1 and two binary operations + and $\cdot$ such that:

(1) $<K,+,0>$ is a commutative CPO-monoid.

(2) $<K,\cdot,0>$ is a CPO-monoid.

(3) Multiplication distributes over addition, *i.e.*, for any $r,s,t \in K$:

$$r\cdot(s+t) = (r\cdot s)+(r\cdot t) \text{ and } (s+t)\cdot r = (s\cdot r)+(t\cdot r)$$

(4) $0\cdot r = r\cdot 0 = 0$, for all $r \in K$. □

In writing expressions, multiplication has priority over addition so that $r+s\cdot t$ is $r+(s\cdot t)$.

2.3. CPO-MODULES. Let M be a CPO-monoid intended for use as the domain for nondeterministic computations. The availability of a CPO-semiring such as K allows us to qualify outputs of a nondeterministic program. For example, suppose a program produces an element $x \in M$ with a path-set property of $r \in K$, and it can also produce $y \in M$ with path-set property $s \in K$. We can denote this output as $rx+sy$. So, the CPO-monoid M must contain "qualified" elements like rx and sy and $rx+sy$. This interaction between a CPO-monoid and a CPO-semiring is formalized as follows:

Definition 3: Let K be a CPO-semiring. A *CPO-(K-module)* is a commutative CPO-monoid M, together with a continuous function from $K \times M$ to M (the image of (r,x) being written rx). The operations are subject to these axioms:

$$(r+s)x = (rx)+(sx)$$

$$r(x+y) = (rx)+(ry)$$

$$(r\cdot s)x = r(sx)$$

$$0x = 0$$

$$1x = x$$

K is called the semiring of *scalars,* and the function from $K \times M$ to M is called *scalar multiplication;* in expressions, scalar multiplication has precedence over addition so that $rx+y = (rx)+y \neq r(x+y)$. Note that the symbols + and 0 are each used in two different ways: the addition and zero for the semiring K and also the monoid operation and identity for M. $\square$

2.4. TWO SPECIAL CASES. Two aspects of nondeterminism have been previously studied by restricting attention to certain subclasses of CPO-monoids. The *must semantics* focuses on the properties which the output of a computation must possess, regardless of the nondeterministic choices taken by the computation. Since an output $x+y$ has fewer guaranteed properties than x, the axiom $x+y \sqsubseteq x$ is imposed on the CPO-monoids. If the monoids are also idempotent, then they are called Smyth monoids. The opposite view of nondeterminism, called the *may semantics,* focuses on the properties which some output of a computation may possess, via some sequence of nondeterministic choices. In this case, an output $x+y$ has more possibilities than x alone, so the axiom $x \sqsubseteq x+y$ is imposed on the CPO-monoids. Because of the connection with partial correctness semantics studied by Hoare, the idempotent monoids with this axiom are called Hoare monoids.

Both of these cases can be studied using CPO-modules, by making an appropriate choice of the underlying semiring:*

> Must Semantics: The axiom $x+y \sqsubseteq x$ holds in all modules provided that $1 \sqsubseteq 0$ in the semiring of scalars, for then $x+y = x+1y \sqsubseteq x+0y = x$. The simplest case is the two-element Boolean semiring, $\mathbf{B} = \{0,1\}$, with $1+1 = 1$ and $1 \sqsubseteq 0$. The resulting modules are exactly the Smyth monoids.
>
> May Semantics: The axiom $x \sqsubseteq x+y$ holds in all modules provided that $0 \sqsubseteq 1$ in the semiring of scalars. For the two-element Boolean semiring with $0 \sqsubseteq 1$, the resulting modules are exactly the Hoare monoids.

The next section gives a more detailed example of CPO-modules in use.

*These correspondences were pointed out by Gordon Plotkin.

3. DISCRETE PROBABILISTIC NONDETERMINISM

This section describes an approach to the semantics of iterative programs, with discrete probabilistic choices. The approach takes its motivation from Dexter Kozen's handling of probabilistic while programs [6] (although continuous probability distributions are not covered here). The purpose is to illustrate the use of CPO-modules to study a non-trivial aspect of nondeterminism.

3.1. SYNTAX. The programs considered have variables $x_1, \cdots, x_n$, which range over some fixed set X of values. There are four types of statements:

Assignment: $x_i := f(x_1, \cdots, x_n)$.

The function $f{:}X^n \rightarrow X$ may be any function.

Composition: $S;T$.

S and T may be any statements.

Conditional: if $p(x_1, \cdots, x_n)$ then S else T fi.

S and T are any statements, and $p{:}X^n \rightarrow [0..1]$ is a function to the closed interval $[0..1]$. The function p gives the probability that statement S is executed, and T is executed when S is not. The usual conditional statement is obtained (with probability 1) by restricting the range of p to $\{0,1\}$.

Iteration: repeat S break_probability $p(x_1, \cdots, x_n)$.

S and p are as above. The statement S is iterated indefinitely. At the conclusion of each execution of S, the function p gives the probability that the iteration will end.

3.2. The CPO-module. For each statement S, we will give an interpretation $I[\![S]\!]$, which will be a continuous function from a CPO-module M to itself. In fact, the function will be a *module morphism,* meaning that it preserves the zero element, addition and scalar multiplication.

Since we are studying probabilities, the underlying semiring of scalars will be the non-negative real numbers, including an infinity element ∞, and having the usual operations and the usual order. This semiring is denoted $\overline{\mathrm{R}}_+$. Recall that the intuitive meaning of a scalar r is a "property" of a path-set. In this case, the property given by $r \in \overline{\mathrm{R}}_+$ is the probability that the execution will be in the given path-set*. The CPO-module M is defined as follows:

> Elements of M are functions from the set of values X^n to the semiring $\overline{\mathrm{R}}_+$. We restrict these to countably non-zero functions $f:X^n \to \overline{\mathrm{R}}_+$, *i.e.*, $\{\overline{x} \in X^n \mid f(\overline{x}) \neq 0\}$ is countable. Intuitively, such a function is a probabilistic state, with $f(\overline{x})$ giving the probability that the state is $\overline{x}$.
>
> Operations and Order are pointwise. For example, if $f,g \in M$ and $\overline{x} \in X^n$, then $(f+g)(\overline{x}) = f(\overline{x}) + g(\overline{x})$.

Each element $\overline{x} \in X^n$ has an associated element $\epsilon_{\overline{x}} \in M$ with $\epsilon_{\overline{x}}(\overline{x}) = 1$, and $\epsilon_{\overline{x}}(\overline{y}) = 0$ elsewhere. Sometimes we write $\overline{x} \in M$ instead of $\epsilon_{\overline{x}} \in M$. thus considering X^n as a subset of M. M is freely generated by the subset X^n in the usual way: if N is any CPO-($\overline{\mathrm{R}}_+$-module) and $\alpha:X^n \to N$ is a function, then there is a unique module morphism $\dot{\alpha}:M \to N$ which extends α to all of M. This property will be used to give the semantics of iterative programs.

3.3. SEMANTICS. For each program S, we give an interpretation $I[\![S]\!]:M \to M$, which is a module morphism:

Assignment:

> Let S be the assignment $x_i := f(x_1, \cdots, x_n)$, and let $\alpha:X^n \to X^n$ take each tuple $(x_1, \cdots, x_i, \cdots, x_n)$ to $(x_1, \cdots, f(x_1, \cdots, x_n), \cdots, x_n)$. Since $X^n \subseteq M$, we can consider α to be a function from X^n to M. $I[\![S]\!]$ is the unique extension of α to a module morphism $\dot{\alpha}:M \to M$.

* The interval [0..1] might suffice, but including all of $\overline{\mathrm{R}}_+$ makes the algebra more straight-forward.

Composition:

$I[\![S;T]\!]$ is $I[\![T]\!]\circ I[\![S]\!]$.

Conditional:

Let $p{:}X^n \to [0..1]$ be any function. We define $true_p{:}M \to M$ to be the unique module morphism with $true_p(\epsilon_{\bar{x}}) = [p(\bar{x})]\epsilon_{\bar{x}}$, and $false_p{:}M \to M$ to be the unique module morphism with $false_p(\epsilon_{\bar{x}}) = [1-p(\bar{x})]\epsilon_{\bar{x}}$. The interpretation $I[\![$if $p(x_1, \cdots, x_n)$ then S else T fi$]\!]$ is

$$I[\![S]\!]\circ true_p \; + \; I[\![T]\!]\circ false_p$$

(where + is the pointwise addition of module morphisms from M to M).

Iteration:

Let p, $true_p$ and $false_p$ be as above. $I[\![$repeat S break_probability $p(x_1, \cdots, x_n)]\!]$ is the least fixed-point of the equation $\alpha = true_p\circ I[\![S]\!] + \alpha\circ false_p\circ I[\![S]\!]$. Equivalently, this is the pointwise infinite sum of $\sum_{i=0}^{\infty} true_p\circ I[\![S]\!]\circ(false_p\circ I[\![S]\!])^i$.

Here are some notes about the interpretation: For any $f\in M$, we define MASS(f) to be $\sum_{\bar{x}\in X^n} f(\bar{x})$. The elements $\epsilon_{\bar{x}}$ have a mass of 1, and for any program S, the interpretation $I[\![S]\!]$ does not increase mass, *i.e.*, MASS$(f) \geq$ MASS$(I[\![S]\!](f))$. A decrease in mass may occur in a repeat loop, since nonterminating branches have zero mass.

This gives a way to determine the probability that a program S will terminate for an input $\bar{x}\in X^n$. The probability is MASS$(I[\![S]\!](\bar{x}))$. The probability of meeting more restrictive post-conditions can be calculated similarly.

3.4. EXAMPLE. Let x be a program variable ranging over the natural numbers $N = \{0,1,2,...\}$, and let p be a function from the natural numbers to the half-open interval $[0..1)$. Define $LOOP(p)$ to be the program:

$$LOOP(p) = \text{repeat } x := x+1 \text{ break\_probability } p(x),$$

in which $p(i)$ is the probability that the loop will terminate when x is i. We will show that $LOOP(p)$ terminates (with probability 1) iff $\sum_{i=0}^{\infty} p(i)$ diverges.

We have only one program variable, x, so the module M consists of functions from N to $\overline{R}_+$. As usual, $\epsilon_i:N\to\overline{R}_+$ denotes the function which is 1 at i and 0 elsewhere. To define the interpretation of $LOOP(p)$, we need these functions:

$q:N\to(0..1]$ defined by $q(i) = 1 - p(i)$.

$succ:M\to M$ is the unique module morphism with $succ(\epsilon_i) = \epsilon_{i+1}$ for all i.

$true_p:M\to M$ is the unique module morphism with $true_p(\epsilon_i) = [p(i)]\epsilon_i$.

$false_p:M\to M$ is the unique module morphism with $false_p(\epsilon_i) = [1-p(i)]\epsilon_i = [q(i)]\epsilon_i$.

Now, $I[\![LOOP(p)]\!]$ is $\sum_{i=0}^{\infty} true_p \circ succ \circ (false_p \circ succ)^i$, which implies:

$$(I[\![LOOP(p)]\!])(\epsilon_0) \;=\; \sum_{i=0}^{\infty} [p(i)\cdot q(i-1)\cdot q(i-2)\cdot\ldots\cdot q(0)]\epsilon_i.$$

and

$$\mathrm{MASS}((I[\![LOOP(p)]\!])(\epsilon_0)) \;=\; \sum_{i=0}^{\infty} [p(i)\cdot q(i-1)\cdot q(i-2)\cdot\ldots\cdot q(0)].$$

This latter quantity is the probability that $LOOP(p)$ halts with input ϵ_0.

Theorem: *If the series $p(0)+p(1)+p(2)+\cdots$ converges, then the probability that $LOOP(p)$ terminates on input ϵ_0 is less than 1.*

Proof: Let $t = \sum_{i=0}^{\infty} [p(i)\cdot q(i-1)\cdot q(i-2)\cdot\ldots\cdot q(0)]$ be the probability that $LOOP(p)$ terminates on input ϵ_0. We define a sequence of approximations

$$t_m = \sum_{i=0}^{m} [p(i)\cdot q(i-1)\cdot q(i-2)\cdot\ldots\cdot q(0)],$$

and note (by induction on m) that $1-t_m = q(m)\cdot\ldots\cdot q(0) > 0$ for any m. This also implies $t_m < 1$ for any m.

Next, choose an n so that $\sum_{i=n+1}^{\infty} p(i) < 1$. Such an n exists, since $p(0)+p(1)+p(2)+\ldots$ converges. Define b to be $\sum_{i=n+1}^{\infty} p(i)$ and note that $t < 1$, since:

$$\begin{aligned} t &= t_n + \sum_{i=n+1}^{\infty} [p(i)\cdot q(i-1)\cdot q(i-2)\cdot\ldots\cdot q(0)] \\ &\leq t_n + \left(\sum_{i=n+1}^{\infty} p(i)\right)\cdot\left(q(n)\cdot\ldots\cdot q(0)\right) \\ &= t_n + b\cdot(1-t_n) \\ &< 1. \end{aligned}$$

The final inequality follows from $t_n < 1$ and $b < 1$. □

Theorem: *If the series $p(0)+p(1)+p(2)+\cdots$ diverges, then the probability that $LOOP(p)$ terminates on input ϵ_0 is 1.*

Proof: Define t and t_m as in the last proof, and note that $t = \lim_{m\to\infty} t_m$. We want to show that $t = 1$, or equivalently, $\lim_{m\to\infty}(1-t_m) = 0$. This is shown as follows:

$$\begin{aligned} 0 &< 1-t_m \\ &= q(m)\cdot\ldots\cdot q(0) \\ &= \frac{q(m)\cdot\ldots\cdot q(0)}{(q(m)+p(m))\cdot\ldots\cdot(q(0)+p(0))} \end{aligned}$$

$$\leq \frac{q(m)\cdot\ldots\cdot q(0)}{[q(m)\cdot\ldots\cdot q(0)]\cdot[1+p(m)+\cdots+p(0)]}$$

$$= \frac{1}{[1+p(m)+\cdots+p(0)]}$$

As m goes to infinity, this last equation goes to zero since the denominator diverges. Hence $\lim_{m\to\infty}(1-t_m) = 0$, as required. □.

The theorems of this section guarantee that this program will terminate (with probability 1):

```
$x := 0;$
repeat $x := x+1$ break_probability $\frac{1}{x+1}$
```

On the other hand, this program has a non-zero probability of non-termination:

```
$x := 0;$
repeat $x := x+1$ break_probability $\frac{1}{x^2+1}$
```

4. THE UNIVERSAL CONSTRUCTION

This section shows how to build a CPO-(K-module) from a commutative CPO-monoid in a universal way. The construction makes use of another universal construction: the tensor product, similar to the tensor product used by Hennessy and Plotkin [4]. We begin with a description of the properties of this tensor product.

4.1. TENSOR PRODUCTS. Let B, C and D be commutative CPO-monoids, and let $B\times C$ be the Cartesian product of B and C. A continuous function $g{:}B\times C\to D$ is called *bilinear* provided that for all $b,b'\in B$ and $c,c'\in C$:

$$g(b+b',c) = g(b,c)+g(b',c)$$

$$g(b,c+c') = g(b,c)+g(b,c')$$

$$g(0,c) = 0 = g(b,0)$$

This is how a nondeterministic function of two arguments should behave [4,8,12].

The purpose of the tensor product is to turn bilinear functions into linear functions. Specifically, the tensor product of B and C is a commutative CPO-monoid $B\otimes C$ together with a bilinear function $\otimes : B\times C \to B\otimes C$, which takes each pair $(b,c)\in B\times C$ to an element $b\otimes c \in B\otimes C$ (note the infix notation). The function $\otimes$ is universal in this sense: suppose D is a commutative CPO-monoid and $g: B\times C\to D$ is a continuous bilinear function. Then there is a unique continuous monoid morphism $\overline{g}: B\otimes C\to D$ making this triangle commute:

$$\begin{tikzcd}
B\times C \arrow[r, "\otimes"] \arrow[dr, "g"'] & B\otimes C \arrow[d, dashed, "\overline{g}"] \\
 & D
\end{tikzcd}$$

For commutative CPO-modules B and C, the tensor product $B\otimes C$ always exists. The construction is similar to that referred to by Hennessy and Plotkin [4]. The only property needed in this paper is the universality of $\otimes$, so we omit the detailed construction of $B\otimes C$.

The tensor product also provides a way of combining morphisms. Suppose $f: B\to B'$ and $g: C\to C'$ are continuous monoid morphisms. They may be combined to form a bilinear function $h: B\times C\to B'\otimes C'$ defined by $h(b,c) = f(b)\otimes g(c)$. The tensor product of f and g is the unique morphism $f\otimes g$ making this diagram commute:

$$\begin{tikzcd}
B\times C \arrow[r, "\otimes"] \arrow[dr, "h"'] & B\otimes C \arrow[d, dashed, "f\otimes g"] \\
 & B'\otimes C'
\end{tikzcd}$$

Note that for any pair $(b,c)\in B\times C$, $(f\otimes g)(b\otimes c) = f(b)\otimes g(c)$.

4.2. FREE CPO-MODULES. Throughout this section, M is a commutative CPO-monoid and K is a CPO-semiring. A *free CPO-(K-module) over M* is a CPO-(K-module), F, together with a continuous monoid morphism $\eta:M\to F$ which is universal in this sense: Suppose A is any other CPO-(K-module) and $g:M\to A$ is a continuous monoid morphism. Then there is a unique continuous module morphism $\hat{g}:F\to A$ making this diagram commute:

$$\begin{tikzcd}
M \arrow[r, "\eta"] \arrow[dr, "g"'] & F \arrow[d, dashed, "\hat{g}"] \\
 & A
\end{tikzcd}$$

(Note that $\hat{g}$ will be strict if g is.)

In order to construct F, note that the CPO-semiring K is also a commutative CPO-monoid (by forgetting about the multiplication). So, the tensor product $K\otimes M$ (taking K as a CPO-monoid) is a commutative CPO-monoid. We can make $K\otimes M$ into a CPO-(K-module) by defining scalar multiplication on $K\otimes M$ as follows: for each $r\in K$, define $\mu_r:K\to K$ to be the "multiply by r" function ($\mu_r(s) = r\cdot s$). From the axioms of a CPO-semiring, μ_r is a continuous monoid morphism, so the function $\mu_r\otimes 1_M:K\otimes M\to K\otimes M$ is also a continuous monoid morphism (where 1_M is the identity function). Now we can define scalar multiplication on $K\otimes M$: for any $r\in K$ and $x\in K\otimes M$, define define $rx = (\mu_r\otimes 1_M)(x)$. It is straight-forward to verify that this definition meets the requirements of a CPO-(K-module).

Theorem: *Let M be a commutative CPO-module and let K be a CPO-semiring. The free CPO-(K-module) over M is $K\otimes M$ with scalar multiplication defined above, and insertion function $\eta:M\to K\otimes M$ defined by $\eta(x) = 1\otimes x$.*

Proof: First note that η is a continuous monoid morphism, as required. (This follows from continuity and bilinearity of $\otimes$.) To show the universality of η, let A be a CPO-(K-module) and suppose $g:M\to A$ is a continuous monoid morphism. Define $\overline{g}:K\times M\to A$ to be the bilinear function $\overline{g}(r,x) = rg(x)$, and let $\hat{g}$ be the unique continuous monoid morphism making this triangle commute:

$$\begin{tikzcd}
K\times M \arrow[r, "\otimes"] \arrow[dr, "\overline{g}"'] & K\otimes M \arrow[d, dashed, "\hat{g}"] \\
 & A
\end{tikzcd}$$

We claim that $\hat{g}$ is also the unique continuous module morphism making this commute:

$$\begin{tikzcd}
M \arrow[r, "\eta"] \arrow[dr, "g"'] & K\otimes M \arrow[d, dashed, "\hat{g}"] \\
 & A
\end{tikzcd}$$

Note that the latter diagram does commute, since for any $x \in M$:

$\hat{g}(\eta(x)) = \hat{g}(1\otimes x) = \overline{g}(1,x) = 1g(x) = g(x)$. To show that $\hat{g}$ is a continuous module morphism, we need only show that it preserves scalar multiplication (since we already know it's a continuous monoid morphism). So, let $r \in K$ and define $\alpha_r : A \to A$ to be $\alpha_r(x) = rx$. We must show that $\alpha_r \circ \hat{g} = \hat{g} \circ (\mu_r \otimes 1)$. This latter equality follows from the fact that this diagram commutes:

$$\begin{tikzcd}
K\otimes M \arrow[rr, "\hat{g}"] \arrow[ddd, "m_r\otimes 1_M"'] & & A \arrow[ddd, "\alpha_r"] \\
& K\times M \arrow[ul, "\otimes"] \arrow[ur, "\bar{g}"'] \arrow[d, "m_r\times 1_M"] & \\
& K\times M \arrow[dl, "\otimes"'] \arrow[dr, "\bar{g}"] & \\
K\otimes M \arrow[rr, "\hat{g}"] & & A
\end{tikzcd}$$

The two triangles of the diagram commute from the definition of $\hat{g}$, and the left square commutes from the definition of $\otimes$. The right square commutes since for any $(s,x)\in K\times M$:

$$\bar{g}\circ(\mu_r\times 1_M)(s,x) = \bar{g}(r\cdot s,x) = (r\cdot s)g(x) = r(sg(x)) = \alpha_r(sg(x)) = \alpha_r\circ\bar{g}(s,x)$$

The commuting of these separate pieces immediately implies that $\alpha_r\circ\hat{g}\circ\otimes = \hat{g}\circ(\mu_r\otimes 1_M)\circ\otimes$, which from the property of $\otimes$, implies that the perimeter commutes.

Finally, to show that $\hat{g}$ is unique, suppose that $h{:}K\otimes M\rightarrow A$ is another continuous module morphism with $g = h\circ\eta$. Then for any pair $(r,x)\in K\times M$ we have:

$$h(r\otimes x) = r[h(1\otimes x)] = r[h\circ\eta(x)] = rg(x) = r[\hat{g}\circ\eta(x)] = \hat{g}(r\otimes x),$$

which implies that $h = \hat{g}$. □

5. PHILOSOPHY

The algebraic structures of section 2 have been proposed as a method for studying aspects of nondeterminism. Section 3 gave an illustration of this technique for programs with discrete probabilistic choices. The final section showed how the algebraic structure can be added in a universal way.

Semiring modules are not the only -- nor even the "best" -- algebraic structure for studying nondeterminism. The importance of the above results lies more in the philosophy that existing algebraic methods can successfully be applied to the study of formal semantics for nondeterminism.

References

(1) S. Abramsky. Experiments, powerdomains and fully abstract models for applicative multiprogramming, in: *Foundations of Computation Theory,* LNCS 158, (Springer-Verlag, 1983), 1-13.

(2) D.B. Benson. Counting paths: nondeterminism as linear algebra, Washington State University Technical Report CS-82-084, Pullman, WA 99164 (1982), to appear in *IEEE Trans. Software Engineering.*

(3) M.C.B. Hennessy. Powerdomains and nondeterministic recursive definitions, in: *International Symposium on Programming,* LNCS 137, (Springer-Verlag, 1982), 178-193.

(4) M.C.B. Hennessy and G.D. Plotkin. Full abstraction for a simple parallel programming language, in: *Mathematical Foundations of Computer Science 79,* LNCS 74, (Springer-Verlag, 1979), 108-120.

(5) C.A.R. Hoare. An axiomatic basis for computer programming, *CACM* 12 (1969), 576-583.

(6) D. Kozen. Semantics of probabilistic programs, *Journal of Computer and System Sciences* 22 (1981), 328-350.

(7) S. MacLane and G. Birkhoff. *Algebra,* (MacMillan Publishing Co., 1979).

(8) M.G. Main and D.B. Benson. Functional behavior of nondeterministic programs, in: *Foundations of Computation Theory,* LNCS 158, (Springer-Verlag, 1983), 290-301.

(9) R. de Nicola and M.C.B. Hennessy. Testing equivalences for processes, in: *Automata, Languages and Programming, 10th Colloquium,* LNCS 154, (Springer-Verlag, 1983), 548-560.

(10) G.D. Plotkin. A powerdomain construction, *SIAM J. Computing* 5 (1976), 452-487.

(11) G.D. Plotkin. *Computer Science Postgraduate Course Notes,* University of Edinburgh, 1980-81.

(12) A. Poigne. Using least fixed points to characterize formal computations of nondeterminate equations, in: *Formalizations of Programming Concepts* (J. Diaz and I. Ramos, Eds.), LNCS 107, (Springer-Verlag, 1981), 447-459.

(13) M. Smyth. Powerdomains, *Journal of Computer and System Sciences* 16 (1978), 23-36.

(14) M. Smyth. Power domains and predicate transformers: a topological view, in: *Automata, Languages and Programming, 10th Colloquium*, LNCS 154, (Springer-Verlag, 1983), 662-675.

(15) G. Winskel. Synchronisation trees, in: *Automata, Languages and Programming, 10th Colloquium*, LNCS 154, (Springer-Verlag, 1983), 696-711.