# Notions of computation determine monads

Gordon Plotkin and John Power \*

Division of Informatics, University of Edinburgh, King's Buildings, Edinburgh EH9 3JZ, Scotland

**Abstract.** We give semantics for notions of computation, also called computational effects, by means of operations and equations. We show that these generate several of the monads of primary interest that have been used to model computational effects, with the striking omission of the continuations monad, demonstrating the latter to be of a different character, as is computationally true. We focus on semantics for global and local state, showing that taking operations and equations as primitive yields a mathematical relationship that reflects their computational relationship.

# 1 Introduction

Eugenio Moggi, in [10,12], introduced the idea of giving a unified category theoretic semantics for what he called notions of computation, but which we call computational effects. He modelled each computational effect in the Kleisli category for an appropriate strong monad on a base category C with finite products. The perspective of this paper is that computational effects determine monads but are not identified with monads. We regard a computational effect as being realised by operators, with a monad being generated by the equational theory of the operators.

Examples of computational effects are nondeterminism, probabilistic nondeterminism, side-effects, exceptions, interactive input/output, and continuations. Moggi's unified approach to modelling them has proved useful, especially in functional programming [2], but there has not been a precise mathematical basis on which to compare and contrast the various effects.

For instance, continuations are computationally of a different character to other computational effects, partly because they are inherently non-local phenomena, allowing a program to go outside its usual scope; partly also because they are a basis for building compilers, as opposed, for instance, to nondeterminism. For another example, computationally, the introduction of global state is a first step towards the introduction of local state. So we seek a mathematical description of features of the various monads that reflects the comparisons and contrasts between the corresponding computational phenomena.

An immediate observation is that the monad for continuations  $R^{(R^-)}$  does not have a rank (see [7] for a definition), while the monads for all the other

 $<sup>^{\</sup>star}$  This work is supported EPSRC grant GR/M56333 and a British Council grant.

above-mentioned computational effects do. There is a theorem to the effect that, because they have ranks, all the other monads are derivable from algebraic operations and equations in a precise sense [8]. So consideration of operations and equations might provide a basis on which to describe features of the computational effects: we regard it as positive that there are no algebraic operations and equations in the precise sense of the theorem, yielding the continuations monad, as that reflects their differing computational nature.

Operations are essential to giving any programming language along the lines of Moggi's computational  $\lambda$ -calculus, for instance FPC or fragments of ML. And the precise definition of operation in the above-mentioned theorem [8] is consistent with computationally natural operations associated with several of the above monads: for instance, one considers a nondeterministic binary choice operator in modelling nondeterminism; one considers a probabilistic binary choice operator for probabilistic nondeterminism; one considers lookup and update operations in modelling global state, and one adds a block operation to model local state; one has operations to raise exceptions; and one has read and write operations associated with interactive input/output. An analysis of several of these operations, regarded as operations associated with an already given monad, appears in [17], with a unified formulation of operational semantics given for some of them in [16].

Some of these examples, notably those associated with nondeterminism, are already well known; others, such as those of exceptions and interactive input/output, are easy; but global and local state required considerable thought. So most of the technical detail of the paper is devoted to a precise statement of the situation for state. At this point, we should like to thank Eugenio Moggi for suggesting to us that the monad for global state may be derived from computationally natural operations and equations, and we should like to thank Peter O'Hearn for showing us a monad for local state and explaining its significance to us. The relationship between global and local state is delicate. We characterise global state in terms of ordinary infinitary operations and equations, but two new features emerge in characterising local state. First, the arities must be allowed to be not just sets but presheafs; and second, the block operation, in contrast to lookup and update, is inherently linear, using symmetric monoidal closed structure rather than cartesian closed structure.

There are fundamentally two ways in which one can generate a monad from algebraic operations and equations, and they are equivalent. One of them is associated with the Kleisli construction on a monad, and the other is associated with its category of algebras. The first amounts to using the equations to give a normal form for each word generated by the operations, while the latter amounts to giving a left adjoint to the forgetful functor from the category of algebras generated by the operations and equations. The relationship between the two approaches is given by the relationship between generic effects and algebraic operations in [17]: to give a generic effect  $e: \mathbf{n} \longrightarrow T\mathbf{m}$  is equivalent to giving m n-ary operations. We formulate our results here in terms of algebras.

For future work, having made some progress here in describing computational effects in terms of operations and equations, it is natural to consider their combinations; we plan to address that issue shortly.

The paper is organised as follows. In Section 2, we give a general explanation of the notions of signature, operations, and equations, and we give some fairly routine examples. In Section 3, we give a careful explanation of how the monad  $(S \otimes -)^S$  for global state is generated by operations for lookup and update subject to computationally natural equations. And in Section 4, we extend the definitions of Section 3 to see how the addition of block subject to reasonable additional equations generates the monad for local state. The central point here is that this gives precise mathematics that reflects the computational relationship between global and local state.

# 2 Operations and equations

Given a finitary signature  $\Sigma$  in the usual universal algebraic sense, one can speak of a  $\Sigma$ -algebra in any category C with finite products: it consists of an object A of C together with, for each  $\sigma \epsilon \Sigma$ , a map

$$a_{\sigma}: A^{ar(\sigma)} \longrightarrow A$$

in C, where  $ar(\sigma)$  is the arity of  $\sigma$ . One can speak of derived operations as usual, and of equations between derived operations. We henceforth assume that C is fixed, leading examples being Set, Poset,  $\omega$ -Cpo, presheaf categories [W, Set], and functor categories of the form  $[W, \omega$ -Cpo] for a small category of worlds W, cf [13].

So, given a finitary signature  $\Sigma$  and equations E between derived operations, we have the notion of a  $(\Sigma, E)$ -algebra in C. This notion, with the evident definition of homomorphism of algebras, generates a category  $(\Sigma, E)$ -Alg with a forgetful functor

$$U: (\Sigma, E) - Alg \longrightarrow C$$

which, if C has all small coproducts, has a left adjoint F, inducing a monad T = UF on C. The category  $(\Sigma, E)$ -Alg is isomorphic to the category T-Alg of algebras for the monad T.

This is a considerably simplified version of the work in [8], which also assumes closedness of C, but the above version is sufficient for most of the work in this paper. One illuminating view of this is in terms of models for a Lawvere theory in a category other than Set, cf [17,18]. There is nothing special about the finitariness of  $\Sigma$ : everything holds equally for infinitary operations, providing C has correspondingly infinitary products, as all our examples do. It is, moreover, routine, to verify that the induced monad T always has a natural strength associated with it, induced by the universal property of products.

Straightforward examples are as follows.

Example 1. The monad -+E for exceptions on Set is induced by E nullary operations, with no equations. These operations model the raising of exceptions,

but do not model a handle operation. This distinction is consistent with the fact that raising exceptions is, in a precise sense, algebraic [17], while handling exceptions is not. In this paper, we only consider algebraic operations, as they induce the various monads, while non-algebraic operations such as handle are of a different character.

Example 2. The monad  $TX = \mu Y.(O \times Y + Y^I + X)$  for interactive input/output is induced by operations  $read: X^I \longrightarrow X$  and  $write: X \longrightarrow X^O$ , with no equations [11]. Here, one uses infinitary operations. A programming language only contains finitary operations, but it may use or generate infinite data, for instance, being fed by an oracle or producing an infinite stream of output; so one may model such phenomena by infinitary operations. This also holds of other classes of resumptions too.

Now consider nondeterminism. In order to explain it properly, we need the more sophisticated notions of signature, operations, and equations of [8], but first observe how a crude version of it fits into our setting here.

Example 3. Let C be the category of  $\omega$ -cpo's. Then the category of algebras for the convex power-domain [14,4,15,1] is, except for partiality, the category of semi-lattices in C, i.e., structures with an associative, commutative, idempotent binary operation. Similar facts are true of the upper and lower power-domains, except that each requires an additional equational axiom in the setting of [8].

In order to outline a complete modelling of nondeterminism in terms of operations and equations as developed in [8], we digress briefly to give an idea of how [8] applies.

One of the simpler examples of algebraic structure on  $\omega$ -Cpo as an instance of [8] is the structure required to give a least element to an  $\omega$ -cpo. It follows from [8] that the category  $\omega$ - $Cpo_{\perp}$  of  $\omega$ -cpo's with least element and maps preserving the least element is of the form L-Alg for a monad L with rank on  $\omega$ -Cpo. Moreover, this application of the general theory shows that the category L-Alg has a canonical enrichment, in fact a unique coherent one, in  $\omega$ -Cpo.

Given any additional algebraic structure  $(\Sigma, E)$  on  $\omega$ -Cpo, one can generate corresponding additional algebraic structure  $(\Sigma_{\perp}, E_{\perp})$  on  $\omega$ - $Cpo_{\perp}$ : an "arity" c in  $\omega$ -Cpo lifts to an arity Lc in  $\omega$ - $Cpo_{\perp}$ , and this induces a lifting of  $(\Sigma, E)$  to algebraic structure  $(\Sigma_{\perp}, E_{\perp})$ . A  $(\Sigma_{\perp}, E_{\perp})$ -algebra consists of a  $(\Sigma, E)$ -algebra with a least element. So the category T-Alg for the monad T on  $\omega$ -Cpo generated by  $(\Sigma, E)$  together with the algebraic structure for a least element is isomorphic to the category  $T_{\perp}$ -Alg for the monad  $T_{\perp}$  on  $\omega$ - $Cpo_{\perp}$  generated by  $(\Sigma_{\perp}, E_{\perp})$ . The monad T is the coproduct, in the category of monads on  $\omega$ -Cpo, of L with the monad generated by  $(\Sigma, E)$ , so there is a monad morphism from L to T.

This is the general situation, accounting for partiality combined with any algebraic structure. The situation for nondeterminism is slightly stronger. There, one has the monad Sl for semilattices on  $\omega$ -Cpo, the monad L for partiality, and a distributive law of L over Sl: this provides a monad structure on (Sl)L, which is also isomorphic to T, i.e., the monad (Sl)L is isomorphic to the coproduct

Sl+L in the category of monads on  $\omega$ -Cpo. This phenomenon reflects special features of the category  $\omega$ -Cpo and the monad L. In general, if all operations of an algebraic structure on  $\omega$ -Cpo are of the form  $P^n \longrightarrow P$  and satisfy the equation

$$f(x, x, \cdots, x) = x$$

then such a distributive law exists, necessarily uniquely.

Example 4. The probabilistic power-domain [5, 6] can be treated algebraically in a number of equivalent ways as described in [3], where three probabilistic choice operators are considered. Of the three, if studied on the category of  $\omega$ -cpo's rather than that of  $\omega$ -dcpo's, two fit within the framework of [8] while the other does not seem to do so. There does not seem to be a natural way to separate out axioms for partiality as we have done for nondeterminism.

#### 3 Global state

In this section, we show how the side-effects monad, which is used to model global state, is generated by infinitary operations for lookup and update subject to computationally natural equations.

Let L be a finite set, to be regarded as a set of locations, and let V be a countable set, to be regarded as the set of values. For instance, V may be taken to be the set of natural numbers. One defines the set of states of a language to be the set  $V^L$  of functions from locations to values. So S is a countable set.

Now assume we have a category C with countable products and coproducts. Consider the monad on C given by  $(S \otimes -)^S$ , where  $A^X$  means the product of X copies of the object A of C, and  $X \otimes A$  means the coproduct of X copies of A. This monad allows us to model global state, as a map in the Kleisli category from A to B is equivalent to giving a map in C from  $S \otimes A$  to  $S \otimes B$ , thus allowing a change of state. For a more general formulation of the monad in terms of tensors and cotensors, see Kelly's book [7]. This more general formulation may become useful when S is treated not merely as a set but as an  $\omega$ -cpo or perhaps an  $\omega$ -cpo with least element.

We seek to express the category  $(S \otimes -)^S$ -Alg as the category of  $(\Sigma, E)$ -algebras, for computationally natural  $\Sigma$  and E, in the category C. In order to give this result, we define a category GS(C), which, by its description, is the category of  $(\Sigma, E)$ -algebras in C for evident  $(\Sigma, E)$ , such that the evident forgetful functor  $U: GS(C) \longrightarrow C$  has a left adjoint given by  $(S \otimes -)^S$ . It follows that  $(\Sigma, E)$ -Alg is isomorphic to  $(S \otimes -)^S$ -Alg.

Our operations will consist of a "lookup" operation of the form  $l:A^V\longrightarrow A^L$  and an "update" operation of the form  $u:A\longrightarrow A^{L\times V}$ . Given a V-indexed family of elements of A, the lookup operation takes a location loc, finds out what its value is in the current state of the computation, and computes the element of A determined by that value. Given an element of A together with a location loc and a value v, the update operation updates the state by insisting that loc take value v, then allows the computation to run.

We take care here to give the result for a category C with axiomatic structure rather than just for Set, as we expect the more general result to be required for modelling the combination of side effects with other computational effects.

**Definition 1.** Given a category C with countable products, a finite set L, and a countable set V, we define the category GS(C) as follows: an object of GS(C) consists of

 $\begin{array}{ll} - \ an \ object \ A \ of \ C \\ - \ a \ "lookup" \ map \ l: A^V \longrightarrow A^L, \ and \\ - \ an \ "update" \ map \ u: A \longrightarrow A^{L \times V} \end{array}$ 

subject to commutativity of two classes of diagrams. First, we have four interaction diagrams as follows:

$$A \xrightarrow{u} A^{L \times V} \xrightarrow{\cong} (A^{V})^{L}$$

$$A^{t} \downarrow \qquad \qquad \downarrow l^{L}$$

$$A^{L} \xrightarrow{A^{\delta}} A^{L \times L} \xleftarrow{\cong} (A^{L})^{L}$$

where  $\delta: L \longrightarrow L \times L$  and  $t: L \longrightarrow 1$  are the diagonal and terminal maps, and the lower unlabelled isomorphism matches the outer L of  $(A^L)^L$  with the first L of  $A^{L \times L}$ ,

$$(A^{V})^{V} \xrightarrow{l^{V}} (A^{L})^{V} \xrightarrow{\cong} (A^{V})^{L}$$

$$\cong \downarrow \qquad \qquad \downarrow l^{L}$$

$$A^{V \times V} \qquad \qquad (A^{L})^{L}$$

$$A^{\delta} \downarrow \qquad \qquad \downarrow \cong$$

$$A^{V} \xrightarrow{l} A^{L} \xrightarrow{A^{\delta}} A^{L \times L}$$

where the unlabelled isomorphisms match the outer V of  $(A^V)^V$  with the first V of  $A^{V\times V}$  and similarly for L, cf [7],

$$A \xrightarrow{u} A^{L \times V} \xrightarrow{u^{L \times V}} (A^{L \times V})^{L \times V}$$

$$\downarrow u \qquad \qquad \downarrow \cong$$

$$A^{L \times V} \xrightarrow{A^{L \times \pi_{1}}} A^{L \times V \times V} \xrightarrow{A^{\delta \times V \times V}} A^{L \times L \times V \times V}$$

where the unlabelled isomorphism matches the outside L with the first L and similarly for V, and

$$A^{V} \xrightarrow{l} A^{L} \xrightarrow{u^{L}} (A^{L \times V})^{L}$$

$$u^{V} \downarrow \qquad \qquad \downarrow A^{\delta \times V}$$

$$(A^{L \times V})^{V} \xrightarrow{A^{L \times \delta}} A^{L \times V}$$

suppressing two isomorphisms. We also have three commutation diagrams as follows:

$$(A^{V})^{V} \xrightarrow{l^{V}} (A^{L})^{V} \xrightarrow{\cong} (A^{V})^{L} \xrightarrow{l^{L}} (A^{L})^{L}$$

$$\downarrow s$$

$$(A^{V})^{V} \qquad (A^{L})^{L}$$

$$l^{V} \downarrow \qquad \downarrow$$

$$(A^{L})^{V} \xrightarrow{\cong} (A^{V})^{L} \xrightarrow{l^{L}} (A^{L})^{L} \xrightarrow{A^{L_{2}}}$$

where s signifies "swap" maps and  $L_2$  denotes the set of ordered pairs of distinct elements of L, with the unlabelled maps both given by the same canonical map,

where s again signifies a "swap" map and with the unlabelled maps again given by the same canonical map, and

$$A^{V} \xrightarrow{l} A^{L} \xrightarrow{u^{L}} (A^{L \times V})^{L} \xrightarrow{\cong} (A^{L})^{L \times V}$$

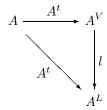
$$\downarrow u^{V} \qquad \qquad \downarrow \qquad \qquad \qquad \qquad \downarrow \qquad \qquad \qquad \downarrow \qquad \qquad \downarrow \qquad \qquad \qquad \downarrow \qquad \qquad \qquad \downarrow \qquad \qquad \qquad \downarrow \qquad \qquad \downarrow \qquad \qquad \downarrow \qquad \qquad \downarrow \qquad \qquad$$

where, again, the unlabelled maps are given by the same canonical map. The rest of the structure of GS(C) as a category is evident: for instance, a map from (A, u, l) to (A', u', l') is a map  $f: A \longrightarrow A'$  in C subject to commutativity of f with l and l' and commutativity of f with u and u'.

The above constitutes our formal definition of the category of algebras. However, if C were Set, one could equally write the equations as equations between terms generated by "lookup" and "update" functions applied to generic elements of the appropriate sets. More generally, for any category C with countable products, one can routinely give an equational language for which equations between infinitary terms of the language correspond to commutative diagrams in the category. So the seven commutative diagrams in the definition of GS(C) can be expressed equationally as the following seven axioms involving infinitary expressions respectively:

- 1.  $l(loc, (u(loc, v, a))_v) = a$
- 2.  $l(loc, (l(loc, (a_{vv'})_v))_{v'}) = l(loc, a_{vv})$
- 3. u(loc, v, u(loc, v', a)) = u(loc, v', a)
- 4.  $u(loc, v, l(loc, (a_v)_v)) = u(loc, v, a_v)$
- 5.  $l(loc, (l(loc', (a_{vv'})_v)_{v'}) = l(loc', l(loc, (a_{vv'})_{v'})_v)$  where  $loc \neq loc'$
- 6. u(loc, v, u(loc', v', a)) = u(loc', v', u(loc, v, a)) where  $loc \neq loc'$
- 7.  $u(loc, v, l(loc', (a_{v'})_{v'})) = l(loc', (u(loc, v, a_{v'}))_{v'})$  where  $loc \neq loc'$ .

**Proposition 1.** For any object (A, l, u) of GS(C), the diagram



commutes.

*Proof.* Use two applications of the first axiom and one application of the second axiom.

**Proposition 2.** If C has countable coproducts, for any object X of C, the object  $(S \otimes X)^S$  together with the maps

$$u: (S \otimes X)^S \longrightarrow ((S \otimes X)^S)^{L \times V}$$

determined by composition with the function

$$L\times V\times V^L\longrightarrow V^L$$

that, given  $(loc, v, \sigma)$ , "updates"  $\sigma: L \longrightarrow V$  by replacing its value at loc by v and

$$l: ((S \otimes X)^S)^V \longrightarrow ((S \otimes X)^S)^L$$

determined by composition with the function

$$L \times V^L \longrightarrow V \times V^L$$

that, given  $(loc, \sigma)$ , "lookups" loc in  $\sigma: L \longrightarrow V$  to determine its value, and is given by the projection into  $V^L$ , satisfy the commutative diagrams required to give an object of GS(C).

The definitions of u and l in the proposition correspond to the equations

$$u(loc, v, x)(\sigma) = x(\sigma[v/loc])$$

and

$$l(loc, (x_v)_v)(\sigma) = x_{\sigma(loc)}(\sigma).$$

**Theorem 1.** The forgetful functor  $U: GS(C) \longrightarrow C$  exhibits GS(C) as monadic over C, with monad  $(S \otimes -)^S$ .

*Proof.* We first show that the left adjoint to U is the functor  $(S \otimes -)^S$ , with algebra structure on  $(S \otimes X)^S$  given by the proposition, and with the unit of the adjunction given by the canonical map  $\eta_X : X \longrightarrow (S \otimes X)^S$ . Given an algebra (A, l, u) and a natural number n, let

$$u_n: (L \times V)^n \otimes A \longrightarrow A$$

denote the canonical map induced by n applications of u, and let

$$l_n: A^{V^n} \longrightarrow A^{L^n}$$

denote the canonical map induced by n applications of l.

Given an arbitrary map  $f: X \longrightarrow A$ , and recalling that  $S = V^L$ , define  $\overline{f}: (S \otimes X)^S \longrightarrow A$  to be the composite of  $(S \otimes f)^S: (S \otimes X)^S \longrightarrow (S \otimes A)^S$  with

$$(V^L \otimes A)^{V^L} \longrightarrow ((L \times V)^L \otimes A)^{V^L} \xrightarrow{u_L^{V^L}} A^{V^L} \xrightarrow{l_L} A^{L^L} \xrightarrow{} A$$

where the unlabelled maps are the evident structural maps. We need to prove four commutativities: the commutativity showing that  $\overline{f}$  composed with  $\eta_X$  is f, two commutativities to show that  $\overline{f}$  is an algebra map, and a final one to show that, given any algebra map  $q:(S\otimes X)^S\longrightarrow A$ , the map  $\overline{q\eta}$  equals q.

For the unit axiom, first observe that the commutation axioms generalise to allow l and u to be replaced by  $l_n$  and  $u_m$  for arbitrary natural numbers n and m. The unit axiom follows by induction on the size of L from these generalised versions of the first two commutation axioms together with the first interaction axiom.

One can see the proof of commutativity of  $\overline{f}$  with u by first considering the case where L has precisely one element, for which the proof is easy, using the third and fourth interaction axioms, together with Proposition 1. The proof for

arbitrary L is essentially the same, but also requires the generalised commutation axioms.

Commutativity of  $\overline{f}$  with l is straightforward: it requires the second interaction diagram together with generalised versions of the first commutation diagram. And the final commutativity follows from routine calculation, just using naturality. So  $(S \otimes -)^S$  is indeed left adjoint to U.

Finally, it follows routinely from Beck's monadicity theorem that U is monadic.

### 4 Local state

We now consider local state and see how its models can be seen in terms of operations and equations, extending those for global state in a principled fashion. In order to do that, we first need a natural statement of what the models of local state are. We follow [13], as further studied in [9]. Following [13], we do not model local state in terms of a category with axiomatically given structure as we have done for global state, but rather we restrict attention to a particular presheaf category. Our results may generalise to functor categories [I, C] where C has axiomatically given structure.

Let I be the category of finite sets and injections. So I is equivalent to the category of natural numbers and monomorphisms. Note that I does not have finite coproducts, and in particular, the sum of two natural numbers does not act as their binary coproduct. However, I does have an initial object, so by duality,  $I^{op}$  has a terminal object. The Yoneda embedding embeds  $I^{op}$  into the presheaf category [I, Set], which is cartesian closed as a locally finitely presentable category, allowing us to use the general theory of operations and equations of [8].

Finite products in [I, Set] are given pointwise, and the closed structure, given functors  $X, Y : I \longrightarrow Set$ , is given by

$$(\boldsymbol{Y}^{\boldsymbol{X}})\boldsymbol{n} = [\boldsymbol{I}, \boldsymbol{Set}](\boldsymbol{X} - \times \boldsymbol{I}(\boldsymbol{n}, -), \boldsymbol{Y} -)$$

i.e., the set of natural transformations from  $X - \times I(n, -)$  to Y. Observe that the terminal object of [I, Set] is I(0, -), which is the constant functor at 1, exactly as it should be because the Yoneda embedding preserves limits.

There is an additional symmetric monoidal closed structure on [I,Set] that is convenient for us to use here. It is a convolution monoidal structure. Using the notation for closed structure of [7], the closed structure corresponding to the convolution monoidal product is

$$[X, Y]n = [I, Set](X -, Y(n + -))$$

In particular, by the Yoneda lemma, [I(m, -), Y]n = Y(n + m). Moreover, the functor [X, -] has a canonical strength with respect to the cartesian closed structure of [I, Set]. We shall use a combination of both symmetric monoidal closed structures here. It takes a little thought to fit the combination exactly into the formalism of [8], but that work is an established part of general theory.

The category [I, Set] is a suitable category in which to model local state [13]. The category [F, Set], where F is the category of finite sets and all functions, is

less suitable, as one wishes to add or ignore new state but not to alter it; the exponent category is regarded as the category of worlds.

As we are modelling local rather than global state here, we can no longer model state by a set; instead, we must index it according to the world in which it is defined. So, given a set of values V, state is modelled by the functor  $S: I^{op} \longrightarrow Set$  given by  $Sn = V^n$ . Note that S is not an object of [I, Set]. But that is not of substantial difficulty to us as, when we studied global state, S was a set, so was not an object of C either. Observe that the functor S is the composite of the inclusion  $I^{op} \longrightarrow Set^{op}$  with  $V^{(-)}: Set^{op} \longrightarrow Set$ .

The monad for local state is

$$(TX)n = (\int^{m\epsilon(n/I)} (Sm \times Xm))^{Sn}$$

where  $\int$  denotes a coend, which is a complicated form of colimit [7]. This construction is a simplified version of one in Levy's thesis [9]. The behaviour of T on injective maps  $f: n \longrightarrow n'$  is as follows: decompose n' as the sum n + n'', note that  $S(p + n'') = Sp \times Sn''$ , and use covariance of X. So the map

$$(\int^{m\epsilon(n/I)} (Sm \times Xm)])^{Sn} \times Sn \times Sn'' \longrightarrow \int^{m''\epsilon((n+n'')/I)} (Sm'' \times Xm'')y$$

evaluates at Sn, then maps the m-th component of the first coend into the (m+n'')-th component of the second, using the above isomorphism for S and functoriality of X. The monad T routinely has strengths with respect to both symmetric monoidal closed structures.

We denote the inclusion of I into Set, which is I(1,-), by the notation L, as it represents locations, and we overload notation by letting  $V:I\longrightarrow Set$  denote the constant functor at V, representing values. As L is not a mere set but rather a set indexed by a world, we need more refined notions of signature, operations, and equations in order to allow L and V to be arities as we had for global state. Note that our definition of  $L_2$  as in the previous section extends here, where  $L_2$  may be seen as the functor from I to Set that sends a finite set to the set of ordered pairs of distinct elements.

We leave the precise general definitions of signature, operations, and equations implicit, and proceed by analogy with global state, by defining a category LS([I,Set]) which is of the form  $(\Sigma,E)$ -Alg for these extended notions. The relationship between our modelling of global and local state will be clear. One can fit these extended notions of arity within the remit of [8], but the particular combination of cartesianness and linearity we use here suggests a more delicate analysis can probably be made than a mere reference to [8] suggests. Observe that, as  $V:I\longrightarrow Set$  is a constant functor, we have

$$(A^V)$$
 –  $= [V, A]$  –  $= (A-)^V$ 

We shall only use the notation  $A^V$ .

**Definition 2.** We define the category LS([I, Set]) as follows: an object of LS([I, Set]) consists of

```
- an object A of [I, Set]
```

$$-a$$
 "lookup" map  $l: A^V \longrightarrow A^L$ 

$$-$$
 an "update" map  $u: A \longrightarrow A^{L \times V}$ 

$$-\ a\ "block"\ map\ b: [L,A] \longrightarrow A^V$$

subject to commutativity of six interaction diagrams and six commutativity diagrams. The interaction diagrams consist of the four interaction diagrams for global state, together with

$$\begin{array}{c|c} [L,A] & \xrightarrow{[L,u]} & [L,A^{L\times V}] \stackrel{\cong}{\longrightarrow} & [L,A^L]^V \longrightarrow [L\times L,A]^V \\ \downarrow & & \downarrow & \\ A^V & \xrightarrow{(A^t)^V} & (A^V)^V \xrightarrow{b^V} & [L,A]^V \end{array}$$

where the horizontal unlabelled map is given by a canonical distributivity law of  $X \otimes -$  over product together with the fact that the unit for the tensor product is the terminal object, and

$$[L, A^{V}] \xrightarrow{[L, l]} [L, A^{L}] \longrightarrow [L \times L, A]$$

$$\cong \downarrow \qquad \qquad \downarrow [\delta, A]$$

$$[L, A]^{V} \qquad \qquad [L, A]$$

$$b^{V} \downarrow \qquad \qquad \downarrow b$$

$$(A^{V})^{V} \xrightarrow{\cong} A^{V \times V} \xrightarrow{A^{\delta}} A^{V}$$

where the horizontal unlabelled map is determined as in the first diagram.

The commutation diagrams are those for global state together with

$$[L, [L, A]] \xrightarrow{[L, b]} [L, A^{V}] \xrightarrow{\cong} [L, A]^{V}$$

$$\downarrow b^{V}$$

$$[L, [L, A]] \qquad (A^{V})^{V}$$

$$\downarrow s$$

$$[L, A^{V}] \xrightarrow{\cong} [L, A]^{V} \xrightarrow{bV} (A^{V})^{V}$$

$$[L, A] \xrightarrow{[L, u]} [L, A^{L \times V}] \xrightarrow{} [L, A]^{L \times V}$$

$$\downarrow b$$

$$\downarrow b^{L \times V}$$

$$\downarrow A^{V} \xrightarrow{u^{V}} (A^{L \times V})^{V} \xrightarrow{\cong} (A^{V})^{L \times V}$$

and

$$[L,A]^{V} \xrightarrow{b^{V}} (A^{V})^{V} \xrightarrow{s} (A^{V})^{V}$$

$$\cong \downarrow \qquad \qquad \downarrow l^{V}$$

$$[L,A^{V}] \qquad \qquad (A^{L})^{V}$$

$$[L,l] \downarrow \qquad \qquad \downarrow \cong$$

$$[L,A^{L}] \longrightarrow [L,A]^{L} \xrightarrow{lL} (A^{V})^{L}$$

One can express the above equations syntactically as

- 1.  $b(v, (u(loc, v', x_{loc})_{loc}) = b(v', x_{loc})$
- 2.  $b(v, l(loc, x_{locv})_{loc}) = b(v, (x_{locv})_{loc})$
- 3.  $b(v,b(v',(x_{loc'loc})_{loc'})_{loc}) = b(v',b(v,(x_{loc'loc})_{loc})_{loc'})$
- 4.  $b(v, u(loc', v', x_{loc})_{loc}) = u(loc', v', b(v, (x_{loc})_{loc}))$
- 5.  $b(v, l(loc', (x_{v'loc})_{v'})_{loc}) = l(loc', (b(v, (x_{v'loc})_{loc}))_v)$

In order to prove that the free algebra is given by the monad for local state, we need to put an algebra structure on TX for arbitrary X in [I, Set]. It is

simplest to express this, using the theory of [17], in terms of generic elements. So we give maps

 $-\ lookup: L \longrightarrow TV$  $-update: L \times V \longrightarrow T1$  $-block: V \longrightarrow TL$ 

with the understanding that  $l:(TX)^V\longrightarrow (TX)^L$  is defined using composition with lookup in the Kleisli category Kl(T), and similarly for update and block. They are defined as follows:

$$lookup_n: n \longrightarrow [Sn, Sn \times V]$$

is defined by  $lookup_n(p,\sigma) = (\sigma,\sigma(p))$ 

$$update_n: n \times V \longrightarrow [Sn, Sn]$$

is defined by  $update_n(p, v, \sigma) = \sigma[v/p]$ , and

$$block_n: V \longrightarrow (\int^{m\epsilon(n/I)} (Sm \times m))^{Sn}$$

is defined by  $block_n(v, \sigma) = ((\sigma, v), 1)\epsilon S(n+1) \times (n+1)$ .

**Proposition 3.** For any object X of [I, Set], the object TX together with the maps l, u and b as defined above, satisfy the commutative diagrams required to give an object of LS([I, Set]).

**Theorem 2.** The forgetful functor  $U: LS([I, Set]) \longrightarrow [I, Set]$  exhibits LS([I, Set])as monadic over [I, Set] with monad T as above.

*Proof.* The proof is essentially the same as that for Theorem 1. The key construction is that given a map  $f: X \longrightarrow A$  where A is an algebra, f extends to an algebra map  $\overline{f}$  given by the composite of  $Tf:TX\longrightarrow TA$  with, on the n-th component,

- a structural map

$$(\int^{m\epsilon(n/I)} (V^m \times Am))^{V^n} \longrightarrow (\int^{m\epsilon(n/I)} (V^{m-n} \times (m \times V)^n \times Am))^{V^n}$$

- $-\text{ a map }(\int^{m\epsilon(n/I)}(V^{m-n}\times(m\times V)^n\times Am))^{V^n}\longrightarrow (\int^{m\epsilon(n/I)}(V^{m-n}\times Am))^{V^n}$
- given by n applications of  $u_m$  a map  $(\int^{m\epsilon(n/I)} (V^{m-n} \times Am))^{V^n} \longrightarrow (An)^{V^n}$  given on the  $(i:n\to m)$ -th component by a composite of applications of  $b_p$  as p varies from n to m-1- a map  $(An)^{V^n} \longrightarrow (An)^{n^n}$  given by n applications of  $l_n$ , and

   a structural map  $(An)^{n^n} \longrightarrow An$ .

The two new interaction axioms are used to prove that  $\overline{f}$  respects b.

# References

- S. O. Anderson and A. J. Power, A Representable Approach to Finite Nondeterminism, Theoret. Comput. Sci., Vol. 177, No. 1, pp. 3–25, 1997.
- N. Benton, J. Hughes, and E. Moggi, Monads and Effects APPSEM '00 Summer School, 2000.
- 3. R. Heckmann, *Probabilistic Domains*, in *Proc. CAAP '94*, LNCS, Vol. 136, pp. 21-56, Berlin: Springer-Verlag, 1994.
- M. C. B. Hennessy and G. D. Plotkin, Full Abstraction for a Simple Parallel Programming Language, in Proc. MFCS '79 (ed. J. Becvar), LNCS, Vol. 74, pp. 108-120, Berlin: Springer-Verlag, 1979.
- C. Jones, Probabilistic Non-Determinism, Ph.D. Thesis, University of Edinburgh, Report ECS-LFCS-90-105, 1990.
- C. Jones and G. D. Plotkin, A Probabilistic Powerdomain of Evaluations, in Proc. LICS '89, pp. 186–195, Washington: IEEE Press, 1989.
- G. M. Kelly, Basic Concepts of Enriched Category Theory, Cambridge: Cambridge University Press, 1982.
- 8. G. M. Kelly and A. J. Power, Adjunctions whose counits are coequalizers, and presentations of finitary enriched monads, *J. Pure Appl. Algebra*, Vol. 89, pp. 163–179, 1993.
- P. B. Levy, Call-by-Push-Value: A Subsuming Paradigm, in Proc. TLCA '99 (ed. J.-Y. Girard), LNCS, Vol. 1581, pp. 228-242, Berlin: Springer-Verlag, 1999.
- 10. E. Moggi, Computational lambda-calculus and monads, in *Proc. LICS '89*, pp. 14–23, Washington: IEEE Press, 1989.
- 11. E. Moggi, An abstract view of programming languages, University of Edinburgh, Report ECS-LFCS-90-113, 1989.
- 12. E. Moggi, Notions of computation and monads, Inf. and Comp., Vol. 93, No. 1, pp. 55–92, 1991.
- 13. P. W. O'Hearn and R. D. Tennent, *Algol-like Languages*, Progress in Theoretical Computer Science, Boston: Birkhauser, 1997.
- G. D. Plotkin, A Powerdomain Construction, SIAM J. Comput. Vol. 5, No. 3, pp. 452–487, 1976.
- 15. G. D. Plotkin, *Domains*, (http://www.dcs.ed.ac.uk/home/gdp/), 1983.
- 16. G. D. Plotkin and A. J. Power, Adequacy for Algebraic Effects, in *Proc. FOSSACS* 2001 (eds. F. Honsell and M. Miculan), LNCS, Vol. 2030, pp. 1–24, Berlin: Springer-Verlag, 2001.
- 17. G. D. Plotkin and A. J. Power, Semantics for Algebraic Operations (extended abstract), in *Proc. MFPS XVII* (eds. S. Brookes and M. Mislove), ENTCS, Vol. 45, Amsterdam: Elsevier, 2001.
- A. J. Power, Enriched Lawvere Theories, in *Theory and Applications of Categories*, pp. 83–93, 2000.