Variations on Algebra: monadicity and generalisations of equational theories

Edmund Robinson *
Queen Mary, University of London

February 12, 2001

Dedicated to Rod Burstall

Introduction

Rod Burstall gave me my first academic job. This was, he explained, to help his group learn about some of the more advanced aspects of category theory. In fact I spent much of my time learning from them about some of the more advanced, now fundamental, aspects of Computer Science. I've always been grateful to Rod, both for the opportunity, and for his willingness to give me time to learn. So it's appropriate that I offer this essentially tutorial paper to him. Better late than never!

Back in the dawn of time Linton [Lin66] discovered that there was a connection between one-sorted algebraic theories and the categorical notion of monad, or more precisely, monads with rank on the category of sets. From the categorist's point of view this is important because it gives a good ranking of the definitional power of the notion of monad, placing monads at the bottom end of a hierarchy of theories with generic models, with geometric theories and their classifying toposes at the top.

However, monads exist on categories other than Set, and many of the attractive properties of algebras can be proved in great generality of the algebras for a monad. For example the category of algebras for a monad is complete if the monad is on a complete category. There is also an attractive notion of free (in the sense of initial) algebra. Indeed, monads correspond precisely to the theories recoverable from the category of algebras together with the free algebra-forgetful functor adjunction. Partly because of this connection with the notion of adjunction, and partly because of a large collection of examples, the notion of monad has long been one of the basic tools of category theory. The first substantial use in the Computer Science setting was by Moggi [Mog91] who separated the notions of value and computation of a given type, and produced a system in which computations form the free algebra on the values for a monad which encapsulates the computational effects present in the programming language. This idea fed through into the functional programming community where it has led to techniques in structuring compilers for both functional and non-functional languages [CH97, Gor94, LH96, LHJ95, JW93, Wad95, Wad97, Wad98]. Less surprisingly the closely related theory of sketches has been used to give accounts of the construction of datatypes [BW90].

These applications depend on looking at monads on categories other than Set, and for these categories something interesting happens. Equational logic can be soundly interpreted in any category with finite products (indeed, one can stretch even this generality a bit). One-sorted algebraic theories still give monads, but now there are (finitary) monads which do not arise from equational theories, and a natural question to ask is what sort of theory they do come from. The

*While writing this paper the author was partially supported by an SERC/EPSRC Advanced Research Fellowship, EPSRC Research grant GR/L54639, and EU Working Group APPSEM.

answer in many cases turns out to be that the theories are still equational, but that the notion of operation has to be generalised. Instead of operations taking tuples as input and producing a single output, we now have to consider operations which take *structured input* and produce *structured output*. So, in the case of monads on Poset we get theories which have operations which take pairs (x_1, x_2) such that $x_1 \leq x_2$ as input, and where we can say that one term has to be less than or equal to another, giving us more expressive power than simple equations.

The situation becomes still more interesting, and more complex, if we are interested in enriched categories. Here, one can vary both the category over which the monad is taken, and the degree of enrichment, so that it is common, for example, to consider theories monadic over Cat as enriched over any one of Set, Gpd or Cat itself. This gives a range of possibilities which is rarely spelt out. The main aim of this paper is to take some examples and look in detail at the kinds of theories that are monadic or enriched monadic over these categories. However, the general theory itself needs some explanation, and before giving the examples there is an expository account of this material, both in the original Set-based setting, and in the more general enriched version.

This paper, then, falls into three sections. In the first of these there is an account of Linton's theory. Despite the standard nature of these results I have been unable to find a modern accessible account, and accordingly section 1 of the paper contains more detail than I would really like.

Linton's classical results were extended to a wider class of categories. Some relevant early work here is that of Kelly [DK83], and Burroni [Bur81] on monads Graph and Cat. Kelly and Power [KP93] extended the classical results from ordinary to enriched categories and from Set to closed categories enriched over a locally finitely presentable category which are themselves locally finitely presentable as a closed category. (This is not the most general form of the theory. Gordon and Power have generalised it to a class of categories enriched over certain bicategories, see [GP98] for the presentation of monads, and [GP97] for the definition of the class of bicategories). The second part of the paper contains an account of Kelly and Power's work. Here, my main aim has been to communicate (most of) the ideas behind their approach. The account here is quite sketchy, but there are decent references which an interested reader can use to fill in the gaps. The material here should be accessible to a reader familiar with category theory who knows what a monoidal category is. I hope even those who find this part hopelessly obscure will not be put off, because the final (and perhaps most important) part of the paper consists of several concrete examples of the use of the material presented in the second part.

We deal with five different base categories, and in each case explore the implications of choosing different enrichments. We also look at how we might present the generalisations of one-sorted equational theories that arise. Some of this material has been presented elsewhere, some not.

The first example we give concerns the case when the base category is a power of Set. Here, the connection is with many-sorted equational theories, a little-known folklore result. The second example is monadicity over Poset. This corresponds to what appears to be a new form of inequational logic, the chief characteristic of which is the ability to form operations whose domain of definition is subject to some order constraints on the input variables. This gives the logic something of the power of conditional inequational logic, though strictly speaking, the two are incomparable in strength.

The next examples are Graph and Cat. In contrast to the other examples, it is not clear to me how important Graph is as a base of definition in its own right. The category theorist's instinct is to go instead for Cat, but it is possible that Graph may be useful for dealing with the theory of sketches. We have two reasons for including it. One is that many classes of categories are monadic over Graph, and it is useful to give some idea of why this is the case. The other is that it also serves as a counterexample to the principle that the theory is cleanest when it has the maximal degree of enrichment. Cat, on the other hand, is a rich and significant base. Most examples of categories with structure are monadic over Cat, with varying degrees of enrichment. Indeed we only leave the monadic world when we start to talk about fibrations and indexed structures. For concrete examples of categories monadic over Cat, we discuss categories with pullbacks and cartesian closed categories.

The final example is the use of Subset-Cat. This has been used recently by Power to give a treatment of categories in which there are operations functorial on or natural with respect to some particular class of maps. An example is his reworking of the theory of premonoidal categories.

I would like to thank John Power and David Murphy for their constructive comments on an earlier draft of this paper.

1 The classical theory

We begin with a survey of the classical theory, which aims at an analysis of the categories monadic over Set. This material is due originally to Linton [Lin66], and connects monads with one-sorted algebraic theories. We shall not give detailed proofs, since they are not difficult to construct, but we shall aim to communicate the basic ideas.

Definition 1.1 *A* monad T *on a category* C *is given by a functor* $T : \mathsf{C} \to \mathsf{C}$*, together with natural transformations* $\eta : I \to T$ *and* $\mu : T^2 \to T$ *such that* $T(\mu); \mu = \mu_T; \mu$ *and* $T(\eta); \mu = \eta_T; \mu = I_T$*.*

Definition 1.2 *An* algebra *for a monad* T *is given by an object* X *of* C *together with a "structure map"* $\alpha : TX \to X$ *such that* $\eta; \alpha = 1_X$ *and* $T(\alpha); \alpha = \mu; \alpha$*.*
A morphism of algebras $\langle X, \alpha\rangle \to \langle Y, \beta\rangle$ *is given by a map* $\theta : X \to Y$ *such that* $T(\theta); \beta = \alpha; \theta$*.*
Together these form the category $\mathsf{T}\text{-}\mathsf{Alg}$ *of algebras for the monad* T*.*

If we compare these with the definitions of algebras and homomorphisms for a functor T, we see that an algebra for a monad T is given by an algebra for the functor T which satisfies certain compatibility conditions with the extra structure possessed by the monad. A morphism of algebras for the monad, on the other hand, does not have to satisfy extra conditions; it is just a morphism of the underlying T-algebras. This means that the category $\mathsf{T}\text{-}\mathsf{Alg}$ is a full subcategory of the category of algebras for the functor T. As we shall see in a moment, there is a direct analogy here with the algebras for algebraic theories. We shall need one further refinement of the notion of monad, that of a monad with rank.

Definition 1.3 *A monad* T *is said to have* finite rank*, or to be* finitary *if the functor* T *preserves filtered colimits. More generally* T *is said to have rank* α *(where* α *is a regular cardinal) if* T *preserves* α*-filtered colimits. (We recall that a category* D *is said to be* α*-filtered if every family containing less than* α *objects has a co-cone over it, and every family of less than* α *parallel maps can be coequalized. A colimit is said to be* α*-filtered if the diagram of which it is a colimit is an* α*-filtered category).*

We now move on to algebraic theories.

Definition 1.4 *A* signature Σ *for a one-sorted algebraic theory is given by a collection* F_Σ *of function symbols together with a function* $ar : F_\Sigma \to \mathsf{Nat}$ *giving the* arity *of each function.*
An algebraic theory $\mathbf{T}$ *is given by a signature* Σ *together with a collection* E *of equations between terms built up from the functions in the signature and variables.*

As usual, constants are identified with functions of arity zero.

This notion of theory is *finitary* in the sense that the arity of each basic operation is finite (and hence so is the arity of each term of finite depth). There is no requirement that either the set of function symbols or the set of equations be finite. We can generalise this by allowing infinitary operations, and define a theory of rank α to be one in which the basic operations all have arity less than α. The significance of the notion of rank is that a monad of rank α corresponds to an algebraic theory in which the functions have arity at most α.

Definition 1.5 *An* algebra *for the algebraic theory* $\mathbf{T}$ *is given by a set* X*, called the* carrier *of the algebra, together with interpretations for each of the operations in* Σ*. A function symbol* f *of arity* k *must be interpreted by a function* $[\![f]\!] : X^k \to X$*. Given this, a term containing* n *distinct variables gives rise to a function* $X^n \to X$ *defined by induction on the structure of the term. An algebra must also satisfy the equations given in* E *in the sense that equated terms give rise to*

identical functions (with obvious adjustments where the equated terms do not contain exactly the same variables).
A homomorphism of algebras *from an algebra X to an algebra Y is given by a function $\theta : X \to Y$ which commutes with the operations of the algebra: $\theta([\![f]\!]_X(x_1, \ldots, x_k)) = [\![f]\!]_Y(\theta(x_1), \ldots, \theta(x_k))$. This gives rise to a category which we shall call* **T-Alg**.

Now we have the basic definitions, we can explain the connection between algebraic theories and monads. Our first goal is to show that given a (finitary one-sorted) algebraic theory, $\mathbf{T}$, there is a corresponding monad.

We start with the free algebra $T(X)$ on a set X. A concrete way to construct this is to take as carrier a set of equivalence classes of closed terms formed from the function symbols of $\Sigma_{\mathbf{T}}$ and an additional constant symbol for each element of X. The equivalence on this set is provable equality in the equational theory generated by the equations in $E_{\mathbf{T}}$. An alternative is given by thinking of X as a set of variables rather than a set of constants, then we can think of $T(X)$ as being the set of derived operations of arity X for the theory. In either case the operations of the theory are interpreted syntactically: $[\![f]\!]([t_1], \ldots, [t_k]) = [f(t_1, \ldots, t_k)]$, where, as usual, braces denote equivalence class. Proving that this definition is consistent, and verifying the universal property of this algebra is a standard piece of universal algebra. It is immediate from this universal property that the map $X \mapsto T(X)$ extends to a functor $T : \mathsf{Set} \to \mathsf{Set}$.

This functor carries monad structure: the natural transformation η is given by the obvious "inclusion" of X in $T(X)$: $x \mapsto [x]$. The interpretation of μ is almost equally simple. An element of $T^2(X)$ is the equivalence class of a term built up from elements of $T(X)$, so that instead of $t(x_1, \ldots, x_k)$, a typical element of $T^2(X)$ is given by the equivalence class of some $t([t_1], \ldots, [t_k])$. The transformation μ is defined to map $[t([t_1], \ldots, [t_k])]$ to $[t(t_1, \ldots, t_k)]$. This makes sense because substitution of provably equal expressions into the same term results in provably equal terms. With these definitions η and μ are natural transformations and satisfy the conditions of definition 1.1. We shall call this monad T.

The next step is to verify that the two notions of algebra coincide.

Lemma 1.6 *Given a set X, then algebra structures on X for the theory $\mathbf{T}$ are in bijection with algebra structures for the monad T. Moreover, this bijection extends to algebra homomorphisms between algebras on sets X and Y.*

Proof. Suppose we have an algebra structure on X for the theory $\mathbf{T}$. If t is a closed term defined over the extended language which includes constants denoting the elements of X, then the algebra structure on X gives rise to an interpretation of t in X, for which we shall write $[\![t]\!]$. Moreover, if t is provably equal to t', then $[\![t]\!] = [\![t']\!]$. Now $T(X)$ is the free algebra on X, and its elements are precisely equivalence classes of terms of this kind. So $t \mapsto [\![t]\!]$ gives a function $\alpha : T(X) \to X$. We claim that this is an algebra structure for the monad T. This follows since $[\![[x]]\!] = x$ and $[\![[[\![t]\!]]]\!] = [\![[t]]\!]$, which correspond to the two requirements on algebras for a monad from definition 1.2 (n.b. we have abused notation slightly here by identifying an element of X with the constant which denotes it).

Conversely, suppose we have an algebra structure on X for the monad T, given by $\alpha : T(X) \to X$. Then, given a function symbol f, we can define a function $[\![f]\!] : X^k \to X$ by $[\![f]\!](x_1, \ldots, x_k) = \alpha([f(x_1, \ldots, x_k)])$. We can prove by structural induction on terms that this gives rise to an interpretation of terms such that $[\![t]\!] = \alpha([t])$. This clearly equates provably equal terms, and hence satisfies the equations of the theory. Thus we have an algebra structure on X for the theory $\mathbf{T}$.

It is now easy to verify that these two operations are mutually inverse and extend to homomorphisms of algebras. □

Lemma 1.7 *If $\mathbf{T}$ is a finitary algebraic theory, then the monad T arising from it is finitary.*

Proof. We must show that the functor T preserves filtered colimits. Suppose $F : \mathsf{D} \to \mathsf{Set}$ is a filtered colimit diagram. Its colimit Y is calculated by taking the disjoint union of $\{F(d) | d \in \mathsf{D}\}$

and quotienting out by the equivalence relation $x \sim x'$ if $x \in F(d)$, $x' \in F(d')$ and there are $\alpha : d \to d''$, $\beta : d' \to d''$, such that $F(\alpha)(x) = F(\beta)(x')$.

We need to show that $T(Y)$ is the colimit of the functor $F;T$. Clearly there is a cocone over $F;T$ with vertex $T(Y)$, so it suffices to show that every element of $T(Y)$ is in the image of some $T(F(d))$, and that elements of $T(F(d))$ and $T(F(d'))$ get mapped to the same element of $T(Y)$ only if they get mapped to the same element in the colimit.

Now, an element of $T(Y)$ is an equivalence class $[t(y_1, \dots, y_n)]$ for some term with constants from Y. There can only be finitely many of these, $y_1, \dots, y_n$ say. These come from the sets $F(d_1), \dots, F(d_n)$, say. However, since D is filtered, we can find an object d into which all of the d_i map. This means that the y_i could equally well be taken to be equivalent elements in $F(d)$. Hence $[t(y_1, \dots, y_n)]$ comes from an element of $T(F(d))$.

Similarly, suppose $t \in T(F(d))$ and $t' \in T(F(d'))$ are both mapped to the same element in $T(Y)$, then by finding an object of D into which both d and d' map, we can assume that $d = d'$. Now t and t' are both built from finitely many elements of $F(d)$, $x_1, \dots, x_n$, say. So we shall abuse notation and write $t(x_1, \dots, x_n)$ and $t'(x_1, \dots, x_n)$. Let $i_d : F(d) \to Y$ be the insertion of $F(d)$ into the colimit. Then we can derive $T(i_d)(t) = t(i_d(x_1), \dots, i_d(x_n)) = t'(i_d(x_1), \dots, i_d(x_n)) = T(i_d)(t')$ in $T(Y)$. That means that this is provable from the equations for $\mathbf{T}$ and the equalities amongst the $i_d(x_j)$ valid in Y. But since Y is the colimit, and D is filtered, we can find $\alpha : d \to d''$ such that all these equalities hold in $F(d'')$. Hence t and t' are equal in $T(F(d''))$, as required. □

Thus, any finitary algebraic theory gives rise to a finitary monad with the same algebras. Next we show that any finitary monad gives rise to a finitary algebraic theory (the generalisation to monads with rank is straightforward).

First, any set is the colimit of its set of finite subsets (ordered by inclusion), and this colimit is filtered. So, for any set X, $T(X)$ is determined (up to isomorphism) by the values of T on finite sets. Similarly, using naturality, so are η_X and μ_X.

Second, notice that the argument given above for turning theories into monads suggests that $T(X)$ should be the free algebra on X, which we can view as the set of derived operations of arity X in the theory.

We would like to be able to pick out certain operations of particular arities as basic. However, there is obviously no canonical way of doing this in general. Many well-known theories can be presented by operations and equations in more than one irredundant way. For example, in the theory of boolean algebras, each of *and* and *or* is definable from the other and *not*, using the de Morgan laws and double negation. This gives two different irredundant presentations. Moreover, the theory can also be presented using only one operation: either *nand* or *nor*, amongst others.

The solution for this lack of a canonical minimal choice of basic operations is to take the maximal choice. Take all derived operations as basic, and then impose sufficient equations on them to be able to determine how certain operations can be built from others.

Lemma 1.8 *If* T *is a finitary monad on* Set*, then* T *is (up to isomorphism) the monad arising from a one-sorted equational theory* $\mathbf{T}$

Proof. The theory $\mathbf{T}$ has a function symbol f of arity n for each element of $T(n)$ (and for each n). The equations are defined in three groups, corresponding to the functoriality of T, and to η and μ.

- for each $\theta : n \to m$, and each $f \in T(n)$, the equation

$$f(x_1, \dots, x_n) = [T(\theta)(f)](\theta(x_1), \dots, \theta(x_n)).$$

- for each n, and each $i \leq n$, the equation

$$x_i = [\eta_n(i)](x_1, \dots, x_n).$$

- for each $g : n \to T(m)$, and each $f \in T(n)$ the equation

$$f(g_1, \ldots, g_n) = [\mu_m(T(g)(f))](x_1, \ldots, x_m)$$

(writing g_i for $g(i)(x_1, \ldots, x_m)$).

By structural induction, any derived operation for this theory is provably equal to a basic one. Moreover, each $T(n)$ has a natural model structure for the theory which equates n-ary operators only if they are represented by equal elements of $T(n)$. It follows that T is the monad corresponding to the theory is T as required. □

We conclude this section with examples which do not fit into this framework. For example, not every monad has a rank, still less is finitary. One instance is the covariant powerset functor (for which μ is union), though the finite powerset functor does.

Another example is given by the functor: $T(X) = [X \to 2] \to 2$ where $A \to B$ is the set of functions from the set A to the set B, and

$$\begin{array}{rcl} \eta_X(x) & = & \lambda f : X \to 2.\ f(x) \\ \mu_X(\theta) & = & \lambda f : X \to 2. \theta(\eta_{X \to 2}) \end{array}$$

Under a relatively mild set-theoretic hypothesis, that every ordinal is less than a strong inaccessible, a simple counting argument shows that the monad cannot have a rank. For suppose it has rank α, then pick a strong inaccessible κ larger than α. Then $\kappa = \bigvee_{\beta < \kappa} \beta$, and this is κ- and hence α-filtered. However, since κ is strongly inaccessible then for any $\beta < \kappa$, $|T(\beta)| < \kappa$, whereas $|T(\kappa)| > \kappa$, by Cantor's theorem.

This is potentially a significant example, since this is the form used by Moggi and others to give denotational semantics to continuation-passing.

Finally, we note that even a monad without rank can be regarded as generating a theory with a proper class of operations and equations. For example, the powerset monad corresponds to a theory in which there is an operation of arity α for each α giving the subset whose elements are exactly its α arguments (so a subset of cardinality at most α), subject to all the equations for idempotency, permutation, and union. Note that theories arising in this way have only a set of operations of any given arity, and so are dealt with by the theory above unless they have basic operations of unbounded arity.

On the other hand, not every algebraic theory which requires operations of unbounded arity need correspond to a monad. An example of this is the theory of complete boolean algebras. The expression of this requires sup and inf for sets of arbitrary cardinality, and the free such algebra on countably many generators cannot exist because it would be too large to be a set (cf. Johnstone [Joh82], p. 33-34). Hence, we cannot define the functor part of a monad.

2 A more general context

In this section we discuss the generalisation of the classical theory given in section 1. The material in this section is due to Kelly and Power [KP93]. However a slightly less general and more concrete exposition can be found in [Pow90]. The theory generalises in two directions: first we can consider categories monadic over something other than Set (specifically categories monadic over a locally finitely presentable category), and then we can consider an enriched version of this more general theory. Both references consider only the finitary version of the theory, but it seems that they extend trivially to a rank α version.

We begin by recalling the notion of a locally finitely presentable category. Since the reason for the importance of this class of categories is not apparent from the usual abstract definition (2.1.1 below), we give it here in the form of a collection of equivalent conditions, showing how an arbitrary locally finitely presentable category can be viewed as the category of models for a logical theory of a certain form.

An object X of a category C is said to be *finitely presentable* (or *finitary*) if the representable functor $\mathsf{C}(X, \_)$ preserves filtered colimits. We shall write C_{fp} for the full subcategory of C on the finitely presentable objects.

Definition 2.1 ([GU71], et al.) *The following conditions on a category* C *are equivalent:*

1. C *has all small limits and colimits, the category* C_{fp} *is essentially small, and any object in* C *is a filtered colimit of the finitely presentable objects which map into it.*

2. C *is the category of models for an essentially algebraic theory.*

3. C *is equivalent to the category of finite-limit-preserving functors* $\mathsf{D} \to \mathsf{Set}$ *for some small category* D *with finite limits.*

4. C *is equivalent to the category of finite-limit-preserving functors* $\mathsf{C}_{\mathrm{fp}} \to \mathsf{Set}$.

5. C *is the category of models for a finite limit sketch.*

A category satisfying one (and hence all) of these is said to be locally finitely presentable *or* lfp, *for short. (N.B. condition 1 can be replaced by any one of a number of similar but superficially weaker statements).*

The essentially algebraic theories mentioned here are more general than many-sorted algebraic theories. The difference is that an essentially algebraic theory allows operations whose domain is an equationally defined subset of some product of the previously defined domains. The canonical example of this is composition in a category, which is defined only on composable, not arbitrary pairs. An important consequence of this freedom is that we can express conditional equations in the logic, though the extra expressive power of conditionally defined operations goes beyond this. Examples of lfp categories are Set, Set^k, Poset, Graph, and Cat. We shall be considering the categories monadic over each of these, and also considering the effect of various enrichments.

We now turn to the theory of enriched categories.

It is not too misleading, at least initially to think of an enriched category as being a category in which the hom-sets carry some extra structure (and in which that structure is preserved by composition). In traditional mathematical applications, the additional structure is often that of an abelian group, or module, whereas in computer science applications it is more likely to be a partial order. The notion of enriched category is, however, more general, and allows for the "hom-sets" of the enriched category to be objects of some monoidal category, traditionally called $\mathcal{V}$. A surprising (and surprisingly fruitful) example of this more general form is a formulation of the notion of metric space in which the points of space X form the objects of an enriched category X, and the "hom-set" $\mathsf{X}(a, b)$ is the distance from a to b, for details see [Law73]. A more familiar example can be taken from the semantics of type theory. Typically, when giving a categorical semantics to a type theory, the elements of the hom-set $\mathsf{C}(S, T)$ are the closed terms of type $[S \to T]$ (or equivalence classes thereof). However, enriched category theory would allow us to enrich over the category of types, and to take $[S \to T]$ itself as the hom-set. More generally, any cartesian closed category can be regarded as enriched over itself.

Most traditional category theory can be developed in the enriched context, though to get any distance requires some further conditions on the category over which the enrichment is taking place (notably some completeness conditions). For the moment we signal the existence of enriched functors, which are those which preserve the structure on hom-sets, and enriched natural transformations, which as far as we shall be concerned, are given by the data for an ordinary natural transformation, subject to the naturality requirement that the obvious two maps $\mathsf{C}(A, B) \to \mathsf{D}(FA, GB)$ are equal. For details see Kelly [Kel82a].

An enriched monad is given by the obvious generalisation of the notion of monad to enriched category theory: the underlying category, the functor T, and the two natural transformations η and μ must all be enriched, and the equations given by the diagrams in definition 1.1 must still hold. This enables us to define an ordinary category of algebras for the monad. If the enriching category $\mathcal{V}$ has equalizers, then the category of algebras can also be enriched over $\mathcal{V}$: our first example of the need for greater completeness properties on $\mathcal{V}$. The Kelly-Power theory applies in the case that $\mathcal{V}$ is a symmetric monoidal closed category which is locally finitely presentable as a closed category (which is equivalent to demanding that the underlying ordinary category $\mathcal{V}_o$ is lfp, and that the monoidal structure on $\mathcal{V}_o$ restricts to one on its finitely presentable objects, for details see [Kel82a, Kel82b, KP93], but note in particular that the unit I must be finitely presentable). From this point onwards we shall assume that all categories and constructions are enriched over such a $\mathcal{V}$, unless we signal otherwise by the use of the word "ordinary".

However, Kelly and Power take a different, somewhat more abstract view of the notion of monad. Briefly, from a categorist's point of view a monoid is an object M equipped with two maps $1 : 1 \to M$ and $m : M \times M \to M$, such that certain diagrams commute. Now it would not be at all important from the point of view of these diagrams, that $(\ ) \times (\ )$ is cartesian product, were it not for the fact that some implicit use is made of the isomorphisms $1 \times M \cong M \cong M \times 1$ and $(M \times M) \times M \cong M \times (M \times M)$. So if we replace $(\ ) \times (\ )$ by $(\ ) \otimes (\ )$, and 1 by I, and make the use of these isomorphisms explicit, then it makes sense to talk about a monoid in any monoidal category. In particular, if C is *any* category, then, modulo the size problems inherent in forming functor categories, $\mathsf{Cat}(\mathsf{C}, \mathsf{C})$ can be made into a monoidal category by taking I to be the identity functor and $F \otimes G$ to be the composite $F \circ G$. A monoid in this monoidal category is exactly a monad on C.

Now, enriched or not, a finitary functor $T : \mathcal{A} \to \mathcal{A}$ on a locally finitely presentable category is determined by its restriction to the finitely presentable objects of $\mathcal{A}$. Moreover, finitary functors compose to give finitary functors. So the monoidal structure given above for $\mathsf{Cat}(\mathcal{A}, \mathcal{A})$ restricts to the subcategory of finitary functors, which can in turn be identified with $\mathsf{Cat}(\mathcal{A}_{\mathrm{fp}}, \mathcal{A})$.

In the case of Set, the finitely presentable objects are finite sets, in the case of Set^k, they are tuples of sets, all of whose components are finite, and all but finitely many of which are empty. Finally, in the case of Poset, the finitely presentable objects are the finite posets. These objects serve as "arities" for the operators. So, when we consider categories monadic over Set, the basic operations for the corresponding theory will each have arity corresponding to some finite set. Since only the isomorphism class of the set matters, we can regard the arity as the size of this finite set, which is a natural number. Similarly for categories monadic over Set^k, the arity is essentially a k-tuple of natural numbers, all but finitely many of which are zero. This corresponds to the kind of theories involved, which are k-sorted algebras in the sense of equational theories with k basic sorts (for example, the theory of lists has two basic sorts: lists and atoms). Finally, for theories monadic over Poset we have arities which are finite posets. This allows us to use primitive operations which require order constraints on their input. One example is that of bounded sup. This operation takes two elements which are bounded above, and returns their least upper bound. It can be defined algebraically in this sense by giving it three elements as input: the two elements we want to take the sup of, and a common upper bound. Moreover, the basic operations of a given arity share the structure of the underlying category, so when we consider categories monadic over Poset, the basic operations of a given arity will themselves form a poset, when we consider categories monadic over Cat they will form a category, and so on. Thus, the basic operations are given by a function $B : \mathrm{ob}(\mathcal{A}_{\mathrm{fp}}) \to \mathcal{A}$.

There are a number of points worth noting about this definition. The first, and most obvious is that the categorical structure of $\mathcal{A}_{\mathrm{fp}}$ plays no rôle. We are only interested in its set of objects. In particular, there is nothing to force the basic operations of one arity to be related in any way to the basic operations of an isomorphic arity. The second is that we allow, and indeed expect, Bc frequently to be empty. The final point is that instead of talking about a collection of operations structured as, for instance a category, we can think of a single operation whose result is a category.

Now, an algebra structure for a collection of operations given in this way is given by an object A, together with a collection of maps:

$$\beta c : \mathcal{A}(c, A) \otimes Bc \to A.$$

The tensor used here is not that of the monoidal category, but a functor $\mathcal{V} \times \mathcal{A} \to \mathcal{A}$ which coincides in the case $\mathcal{A} = \mathcal{V}$. If we think of objects in $\mathcal{V}$ as sets, and $\mathcal{A}$ as an ordinary category, then $U \otimes A$ is the V-indexed copower of A (the disjoint union of U copies of A). This description generalises in a fairly straightforward way. For example, if $\mathcal{V} = \mathsf{Set}$ and $\mathcal{A} = \mathsf{Poset}$, then $U \otimes A$ is the disjoint union of U copies of A, whereas if $\mathcal{V} = \mathcal{A} = \mathsf{Poset}$, then it is the cartesian product of the posets U and A. In general $U \otimes A$ is a representing object for the functor $[U, \mathcal{A}(A,\ )] : \mathcal{A} \to \mathcal{V}$.

Once we have the basic operations, we can give an account of the operations derived from them. These can be presented as the colimit of an ω-chain:

- $S_0(c) = c$
- $S_{n+1}(c) = c + \sum_{e \in \mathrm{ob}(\mathcal{A}_{\mathrm{fp}})} \mathcal{A}(e, S_n(c)) \otimes Be$

where the map $i_{n+1}(c) : S_n(c) \to S_{n+1}(c)$ is

- $c + \sum_e \mathcal{A}(e, i_n(c)) \otimes Be$

We can think of $S_n(c)$ as the object of derived terms of arity c and depth at most n. Because of the use of the tensor, the derived operations depend on the enriching category $\mathcal{V}$. For examples of this, as well as the dependence of the notion of algebra, see sections 3 and 4. As usual, the derived operations form the initial algebra for the theory.

We must also explain the generalisation of equations. The view taken in [KP93] depends on the identification of monads with monoids, mentioned above.

For any object A of $\mathcal{A}$, there is a finitary functor $\langle A, A\rangle : \mathcal{A} \to \mathcal{A}$, which as usual we can regard as a functor $\mathcal{A}_{\mathrm{fp}} \to \mathcal{A}$ given by

$$\langle A, A\rangle c = \mathcal{A}(c, A) \pitchfork A.$$

Here $\mathcal{A}(c, A)$ is the "set" of c-tuples of elements of A, and $\pitchfork$ is the cotensor (cf. Kelly [Kel82a], p.91 ff.)[1], so that $\langle A, A\rangle c$ is the power of $\mathcal{A}(c, A)$ copies of A, and hence can be thought of as the object of operations on A of arity c. This however only works for a finitary c. For a more general object B, $\langle A, A\rangle B$ has to be thought of as the subobject of the object of B-ary operations which are finitary (i.e. depend only on a finitary part of B).

Now $\langle A, A\rangle$ has a natural monoid structure (in $\mathcal{V}\text{-}\mathsf{Cat}(\mathcal{A}, \mathcal{A})$). A typical element of $\langle A, A\rangle \circ \langle A, A\rangle(c)$ is given by a finitely presentable c', together with a map $\alpha : c' \to \langle A, A\rangle c$, and an element of $\langle A, A\rangle c'$. So we have an operation of arity c' on A, together with a c'-tuple of operations of arity c on A. These compose in the obvious fashion to give an operation of arity c on A, and this gives the multiplication of the monoid.

Now, a collection of basic operations, B as described above also gives rise to a monoid, FB in a similar fashion. Moreover, an algebra structure on A is effectively the same as a monoid homomorphism $FB \to \langle A, A\rangle$. Since the arities of both sides of an equation must be the same, we can regard a collection of equations as giving a collection of operations of the corresponding arities. Moreover, these operations can be interpreted two different ways into the underlying theory, either by taking the derived operation corresponding to the left-hand side of the equation, or the operation corresponding to the right. So, from this point of view, a collection of equations is a collection of operations, E, together with a pair of monoid maps

$$FE \rightrightarrows FB,$$

and an algebra structure on A is a monoid map $FB \to \langle A, A\rangle$ which coequalizes them.

Any theory which can be presented in this fashion has a corresponding monad (the coequalizer of the two maps $FE \to FB$), but conversely, any finitary monad gives rise to a theory presented by operations and equations in this sense. To sum up:

Theorem 2.2 (Kelly-Power) *If $\mathcal{A}$ is a locally finitely presentable category enriched over the locally finitely presentable symmetric monoidal closed category $\mathcal{V}$, and $\mathsf{T} = (T, \eta, \mu)$ is a finitary enriched monad on $\mathcal{A}$, then T admits a presentation by operations and equations.*

[1] If $X \in \mathcal{V}$ and $A \in \mathcal{A}$, then $X \pitchfork A$ is a representing object for the $\mathcal{V}$-functor $\mathcal{V}(X, \mathcal{C}(\ , A))$, i.e. for any $B \in \mathcal{A}$ there is an isomorphism $\mathcal{C}(B, X \pitchfork A) \cong \mathcal{V}(X, \mathcal{C}(B, A))$ $\mathcal{V}$-natural in B

3 Many-sorted algebraic theories

Many traditional mathematical structures are single-sorted, but computer scientists interested in data types are much more interested in many-sorted theories. These can be modelled using the technology of monads, by considering categories monadic over Set^Σ, where Σ is the set of basic sorts for the theory. So in this section we shall be considering theories monadic over Set^k.

In this, as in our other examples, there are at least two possible levels of enrichment. Set^k is an ordinary category (enrichment over Set), and, since it is cartesian closed, it is also enriched over itself. (But the general theory applies only when k is finite: when k is infinite the terminal object is not finitely presentable, and so Set^k is not lfp as a closed category). In either case a collection of basic operations is given by a function

$$B : (\mathsf{Set}^k)_{\mathrm{fp}} \to \mathsf{Set}^k,$$

where, as we remarked above, the finitely presentable objects in Set^k are those k-tuples of finite sets all but finitely many of whose components are empty.

Now, a B-algebra structure on an object A of Set^k is given by a map

$$\alpha_c : \mathsf{Set}^k(c, A) \otimes Bc \to A$$

for each c in $(\mathsf{Set}^k)_{\mathrm{fp}}$.

In the case of enrichment over Set, $\mathsf{Set}^k(c, A)$ is the *set* of morphisms $c \to A$. In other words it is the product of the coordinatewise function spaces:

$$\prod_i [c_i \to A_i].$$

The tensor $S \otimes X$ is the k-tuple whose i'th component is $S \times X_i$. So α_c is given by a k-tuple of maps $\alpha_{c,i} : \mathsf{Set}^k(c, A) \times (Bc)_i \to A_i$.

We can now see how c is functioning as an arity and Bc as a collection of basic operations. For each element f of $(Bc)_i$ we have a function $\alpha_{c,i}(\ \ , f) : \mathsf{Set}^k(c, A) \to A_i$. So, given a c-tuple of elements of A, "f" gives an element of A_i. Here, of course, c-tuple has to be interpreted coordinatewise, so for each j we are given a c_j-tuple of elements of A_j. For example, if $k = 2$ and $c = (3, 1)$, then a c-tuple of elements of A consists of three elements of A_0 and one of A_1, and hence f corresponds to a basic operation with three variables of sort 0 and one of sort 1. So the operations are exactly what we find in many-sorted (or more exactly k-sorted) algebra.

Moreover, if we follow through the recipe given in the previous section, then we find that the elements of $S_n(c)$ are exactly what we would expect as the terms of depth at most n from standard presentations of many-sorted algebra. Finally, the equations are equations.

If, instead, we enrich over Set^k itself, then everything is done coordinatewise. $\mathsf{Set}^k(c, A)$ is no longer a set, but the k-tuple of function spaces whose i'th component is $[c_i \to A_i]$, and the tensor $S \otimes X$ now takes a k-tuple, not a set, on the left, and is the k-tuple $(S_i \times X_i)$. It follows that an algebra structure is given by a k-tuple of maps $\alpha_{c,i} : [c_i \to A_i] \times (Bc)_i \to A_i$. Hence, no basic operation which produces a result of one sort, can depend on arguments of another sort. We are looking at a k-tuple of algebra structures on Set.

So, we can see that the enrichment over Set gives conventional many-sorted algebra, (or rather it gives conventional k-sorted algebra) whereas the enrichment over Set^k simply gives k-tuples of single-sorted algebras. This is reflected in the fact that for a typical k-sorted theory, the free algebra functor, which takes a k-tuple of sets to the k-tuple of the carriers for the free algebra on those sets, has (vacuously) an enrichment over Set, but not an enrichment over Set^k.

4 Poset algebras

This section is concerned with theories monadic over Poset. As before there are two possible enrichments, over Set or over Poset itself. A finitely presentable poset is simply a finite poset. So a collection of basic operations is given by a function $B : \mathrm{ob}(\mathsf{Poset}_{\mathrm{fp}}) \to \mathsf{Poset}$.

First consider enrichment over Set. A B-algebra structure on A is given by a collection of maps:

$$\alpha_c : \mathsf{Poset}(c, A) \otimes Bc \to A$$

Once again we have $\mathsf{Poset}(c, A)$ giving c-tuples of elements of A. In this case a c-tuple is an order-preserving map from the finite poset c to the poset A. $\mathsf{Poset}(c, A)$ is interpreted as a set, and this means that there is no monotonicity requirement on the interpretation of any given function f in Bc. However, there is an order on the elements of Bc, and since $S \otimes X$ is the order-theoretic product of S with the discrete order and X, this means that if $f \leq g$ as operations, then $f(\gamma) \leq g(\gamma)$ for any c-tuple γ in A.

Following this through, there is the same kind of phenomenon in the definition of the derived operations: $f(s) \leq g(t)$ if and only if $f \leq g$ and $s = t$.

This kind of pathology can be avoided by considering the alternative enrichment instead. When we enrich over Poset, $\mathsf{Poset}(c, A)$ has the same elements as before, but is considered as a poset with the pointwise partial order on functions. The tensor, $S \otimes X$, is now the product of two posets, and carries the product partial order. It follows that the interpretation of a typical f in Bc must now be monotone in the usual sense, and also that if $f \leq g$ then the interpretation of f must be pointwise less than or equal to that of g, as before.

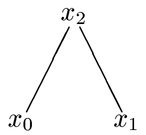
Moreover, when we come to look at the derived operations, we find that $f(s) \leq g(t)$ if and only if $f \leq g$ and $s \leq t$, as we expect.

However, when we come to consider equations, we find that the equations are (essentially) equations between operations. Despite this, we still retain the power of inequational reasoning. The reason is simple. Our basic operations are partially ordered. So we can replace an inequation $s(\vec{x}) \leq t(\vec{x})$ between two derived operations by two new basic operations, f_s and f_t, ordered $f_s \leq f_t$, as in the inequation, together with the two equations $f_s(\vec{x}) = s(\vec{x})$ and $f_t(\vec{x}) = t(\vec{x})$, which identify the new operations with the two sides of the inequation.

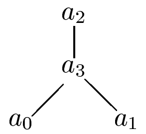
So we are now in the usual world of inequational algebra, except that our arities are posets, not just sets (and also that we cannot handle strict inequality). What this means from the syntactic point of view is that we can define operations which require certain inequalities to hold between their inputs.

For example, if we are interested in ω-CPO, then we want an operation to give us the sup of an ω-chain. This operation is not definable in a finitary theory, since it takes a countable chain as input, but it is definable in a theory of rank ω_1.

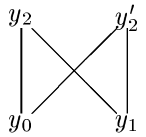
Alternatively, one of the defining characteristics of Scott domains is that they have sups of pairs of elements which are bounded above. This can be handled by an operation whose input is a pair of elements, together with a common upper bound, i.e. a copy of the poset Λ:



and whose output is the least upper bound of the lower pair. Of course we must still specify that the output is this least upper bound. In order to do that, we introduce a new family of basic operations, all of arity Λ, and ordered as



Finally, as arity for derived operations we shall also need the poset



In order to specify that our chosen operation ($a_3(x_0, x_1, x_2)$) is the least upper bound of x_0 and x_1, we impose the equations:

$$\begin{array}{rcl} a_0(x_0, x_1, x_2) & = & x_0 \\ a_1(x_0, x_1, x_2) & = & x_1 \\ a_2(x_0, x_1, x_2) & = & x_2 \\ a_3(y_0, y_1, y_2) & = & a_3(y_0, y_1, y_2'). \end{array}$$

The first three of these imply that $a_3(x_0, x_1, x_2)$ is an upper bound for x_0 and x_1 which is below x_2, and the last implies that it is independent of x_2.

However, this presentation gives a non-standard category of Scott domains. The reason is that algebra homomorphisms must preserve all the operations of the theory. Hence the maps here must preserve bounded joins as well as being continuous.

Other joins and meets can be specified in a similar fashion. Moreover, by combining the techniques of the last section and this, we can deal with many-sorted order-sorted theories by considering theories monadic over Poset^k as a Poset-enriched category.

Thinking more syntactically, having arities which are posets corresponds to having conditionally defined operations: allowing us to specify that the operation f is defined on the set of $(x_0, \ldots, x_n)$ which satisfy a certain family of order constraints (in the Scott domain example it was $\{(x_0, x_1, x_2) | x_0 \leq x_2, x_1 \leq x_2\}$). These constraints, though, have to take the form of some conjunction of constraints $x_i \leq x_j$. What we cannot do is handle constraints of the form $s(\vec{x}) \leq t(\vec{x})$. This puts us in the world of the sketchable [Kel82b, KPT99].

Moreover, given two basic operations of the same arity, i.e. with the same number of inputs and equivalent constraints, we can say that one is pointwise less than or equal to the other.

The derived operations can then be regarded as the derived operations (in the ordinary syntactic sense) that are well-formed, in that it is provable that conditionally defined operators are only applied to terms which satisfy the order constraints in the operator's definition. The basic proof rule here is

$$\frac{f \leq g \quad s_1 \leq t_1, \ldots, s_n \leq t_n}{f(s_1, \ldots, s_n) \leq g(t_1, \ldots, t_n)} \quad \left(\begin{array}{c} f(s_1, \ldots, s_n) \text{ well-defined,} \\ g(t_1, \ldots, t_n) \text{ well-defined} \end{array} \right)$$

Finally, we are allowed equations and (non-strict) inequations between derived operators.

Thus, the theory is more powerful than standard inequational logic. Because of the conditionally defined operations it has some of the power of a conditional inequational logic (it is an extension of the fragment in which premisses can be conjunctions of inequations between variables). However, as a logic it is strictly incomparable with conditional inequational logic as usually formulated, i.e. the logic of Horn clauses where basic propositions are either of the form "$\mathtt{term} = \mathtt{term}$" or of the form "$\mathtt{term} \leq \mathtt{term}$".

We should also make some comment about categories monadic over categories of domains. Where the category of domains is locally presentable, the theory is much the same. This, however, does not cover many examples beyond ω-preCPO and its analogues for larger ordinals. The problem is that even where the objects of the category can be presented as models of a suitable theory, the homomorphisms appropriate are not usually general continuous functions. For example, we get ω-CPO's and *strict* maps, since the bottom element is part of the structure. Similarly, as we have seen, for Scott domains, we would get maps which preserve the sups of bounded sets. Moreover, conditions such as algebraicity or continuity do not have the right logical character to give locally presentable categories. Despite this, the constructions and techniques given here can sometimes be made to apply in these settings.

5 Algèbres graphiques

This section is included mainly for historical reasons, rather than for the intrinsic interest of the category of graphs. However it also serves as a counterexample to a trend developing in previous sections, where it seems that a useful guideline might be to enrich as much as possible, except when trying to do many-sorted algebra. In this case we get a rather cleaner and more useful theory from confining the enrichment to Set.

The pioneering work on Graph was done by A. Burroni [Bur81], who produced a notion of "graphical algebra" (or "algèbre graphique"). Anything which can be characterised as a graphical algebra is monadic over Graph, and in fact conversely, although this does not appear in Burroni's paper. Burroni used this notion to show that all of the common classes of categories (including toposes) were monadic over Graph. However, one has to be slightly careful. The functors between categories must preserve all relevant structure exactly, whereas a category theorist is generally interested in functors which preserve it up to canonical isomorphism, and in the case of toposes, not even that.

Slightly earlier, but independently, Kelly was working on monadicity over Cat, this time with extension to the enriched setting. This is a more subtle and more useful setting, as we shall see in the next section, but for the moment we shall simply note that, in contrast to Burroni, Kelly was able to show that his formulation of algebras completely characterized monadicity over Cat.

Graph is the category of directed graphs with multiple edges. Each graph is given by a set of vertices and a set of edges. Each edge has a domain and a codomain. Morphisms of graphs are given by pairs of functions which respect the domain and codomain operations. Hence Graph is isomorphic to the functor category $\mathsf{Set}^{\rightrightarrows}$. It is also sometimes helpful to think of a graph as a category without composition or identities. Following this analogy, a morphism of graphs is given by the same data as that for a functor, without the requirement that it respect composition or identities. There is also a notion of transformation between morphisms, given by the same data as a natural transformation, but of course the requirement that the naturality squares commute no longer makes sense.

The finitely presentable objects in Graph are simply the finite graphs, and, as usual, these form the arities for operations. So, for each finite graph c, we have a graph Bc of basic operations.

Graph can be enriched over Set and over itself.

If we enrich over Set, then an algebra structure on a graph A consists of giving, for each $f \in Bc$, for each graph morphism $s : c \to A$, a vertex $f(s)$ of A, and for each edge $e : f \to f'$ in Bc, an edge $e(s) : f(s) \to f'(s)$ in A. If we enrich over Graph instead, then we must also consider transformations $s \to s'$. This is not particularly interesting for Graph, in marked contrast to the analogous situation for Cat.

Equations work at the level of vertices, and at the level of edges.

For example, consider the theory of categories. A category is a graph with an associative composition on edges, which has identities. So, we need a graph of shape $\bullet \longrightarrow \bullet$ of operations of arity $\bullet \longrightarrow \bullet \longrightarrow \bullet$ to give the composition and another graph of shape $\bullet \longrightarrow \bullet$ of operations of arity $\bullet$ to give the identities. These are subject to equations which say that the source and target of the composite are as expected, and that the source and target of the identity are the given vertex.

However, there is a problem in expressing the axioms involving composition: the associativity of composition, and the left and right identity axioms. The problem is that we are only allowed to form derived operations when we know that they are well-structured from the "type information" of their definition. In order to know that the left identity law, $\mathrm{id} \circ f = f$, is well formed, we have to know that the target of the identity map is the vertex on which it is an identity. This is given by one of the previous axioms, and not just by the structure of the operations. We cannot form a derived operation which represents the composite $\mathrm{id} \circ f$.

The solution is to define a suitable collection of basic operations whose arity is that of the equation we want to express. Put the equation in terms of these, and then impose further equations which allow us to build the new operations as a composite of our real primitives. In this case we can define a graph of shape

$$\begin{tikzcd} \bullet \arrow[r, "a_0"] \arrow[dr, "a_2"'] & \bullet \arrow[d, "a_1"] \\ & \bullet \end{tikzcd}$$

of operations of arity $\bullet \longrightarrow \bullet$, giving us a composable pair (secretly the original map followed by an identity), and then use composition on that. We then impose the equations $a_0(f) = f$, $a_1(f) = \mathrm{id}(\mathrm{dom}(f))$, $a_2(f) = f$ and $a_2(f) = a_1(f) \circ a_0(f)$. The two equations for a_2 then give us the left identity law.

Analogous solutions exist for the similar problems with the right identity and associativity laws.

There are more problems when it comes to expressing structures such as categories with limits. Here there are two problems, one is that the map into a limit cone is defined only for commuting, not arbitrary, cones, and the other is that we need to be able to express the uniqueness of the factorization map into a limit cone. This is dealt with in detail in [Bur81] and [DK83], and we refer the reader to these papers for details. (N.B. there is also an account in [LS86], which is clear except that it is not sensitive to the difficulty of ensuring that operations have well-defined arities, and which therefore needs reading with care).

6 Algèbres catégoriques

In contrast to previous examples Cat has three useful levels of enrichment. Two of these are of course Set and Cat itself, but the third is Gpd, the category of groupoids. A *groupoid* is a category all of whose morphisms are isomorphisms. Enrichment over Cat corresponds to demanding the functoriality of all operations, enrichment over Set corresponds to not demanding any form of functoriality, and enrichment over Gpd is the half-way house in which we demand functoriality only with respect to isomorphisms. One rôle this level of enrichment plays is quite similar to that played by embedding-projection pairs in domain theory: it enables us to place some kind of parametricity constraint on naturally contravariant operations. But another is that it allows us to cope with homomorphisms which preserve structure up to coherent isomorphism. As we have set things up, our algebras must have designated structure (this corresponds to the commutation of the algebra map with the action of the monad). This does not present a difficulty for most categorists. However, we also insist that homomorphisms preserve the structure exactly, and this does create a problem. However, in a Gpd-enriched category, and with a Gpd-enriched monad we can insert a 2-cell into the definition of homomorphism (subject to coherence conditions corresponding to the identity and multiplication of the monad), and this gives us homomorphisms which preserve the structure up to coherent isomorphism. In fact it gives us a formal definition of what "preserve up to coherent isomorphism" means for this kind of structure. The details of this are, however, beyond the scope of the present paper, and the interested reader is referred to [BKP89]. We should however note that the usual 2-category notation is modified to incorporate the change just mentioned. $\mathsf{T}\text{-}\mathsf{Alg}$ is the category of (strict) algebras and homomorphisms preserving structure up to coherent isomorphism, while the category with which we are concerned here, the category of (strict) algebras and strict homomorphisms, is usually written $\mathsf{T}\text{-}\mathsf{Alg}_s$.

As befits a more complex structure, the arities available in Cat are more complex than they have been up to now. A finitely presentable category is not necessarily finite. A finitely presentable category is one which can be presented as a finite colimit of finite categories. More concretely, it has a finite number of objects, the arrows are generated by composition from a finite number of generating arrows, and the equality between composites is generated by a finite number of equations. Thus a one-object finitely presentable category is a finitely presentable monoid in the usual algebraic sense. In particular the free category on one object and a single (non-identity) endomorphism has countably many maps. Most of our examples will, however, involve only finite categories. In each case, however, given a finitely presentable category c, we shall have a category Bc of basic operations.

As usual, the first thing is to look at the effect of each of the three levels of enrichment on the basic operations, via the tensor product.

For enrichment over Cat, the hom $\mathsf{Cat}(c, A)$ is the category of functors and natural transformations. The tensor, $\mathsf{Cat}(c, A) \otimes Bc$ is the product of categories. This implies that each basic operation, given by an object of the category Bc, is interpreted as a functor $\mathsf{Cat}(c, A) \to A$, and that any morphism between basic operations is interpreted as a natural transformation. For example, if $c = 2$, the discrete category on two objects, and $Bc = 1$, then we have a functor $A^2 \to A$. And if $Bc = \bullet \longrightarrow \bullet$, then we have two functors $A^2 \to A$ corresponding to the two vertices and a natural transformation between them corresponding to the arrow.

For enrichment over Gpd, the hom $\mathsf{Cat}(c, A)_{\mathrm{iso}}$ is the groupoid of functors and natural isomorphisms, which we can regard as a category. The tensor $\mathsf{Cat}(c, A)_{\mathrm{iso}} \otimes Bc$ is then again the product of categories. This implies that each basic operation is interpreted as a functor $\mathsf{Cat}(c, A)_{\mathrm{iso}} \to A$, and hence is functorial with respect to isomorphisms, but not necessarily with respect to all morphisms. Again taking the example $c = 2$ and $Bc = \bullet \longrightarrow \bullet$, we would have two functors $A^2_{\mathrm{iso}} \to A$ (where A_{iso} is the category whose objects are those of A but containing only the isomorphisms), and a natural transformation between them. The components of the natural transformation need not be isomorphisms.

Finally, for enrichment over Set, the hom is structured as a set, which corresponds to the discrete category of functors, and hence there is no functoriality constraint whatever. In our example we would get two binary functions on the objects of A, and a pointwise collection of maps between them.

Life is now somewhat simpler than it is at the level of Graph, since we can express composition information directly in the arities and results of operations.

As a first example of a category with structure monadic over Cat, consider monoidal categories. Here the structure consists of a constant I, given as an operation of arity 0, a functor $\otimes : A \times A \to A$, given as an operation of arity 2, and natural transformations α, λ and ρ. These can be given as morphisms of operations. For example, to give α, we give a category $\alpha_0 \xrightarrow{\alpha_2} \alpha_1$ of operations of arity 3, corresponding to the degree of naturality of α. Thus, for each A, B, C in A we have a map

$$\alpha_0(A, B, C) \xrightarrow{\alpha_2(A,B,C)} \alpha_1(A, B, C)$$

We then impose equations to ensure that the domain and codomain of these transformations are the appropriate functors. For α these are $\alpha_0(A, B, C) = A \otimes (B \otimes C)$ and $\alpha_0(A, B, C) = (A \otimes B) \otimes C$. Finally we need to impose the coherence equations, which can be achieved as equations at the level of morphisms.

In this setting, a morphism of monoidal categories is a functor that preserves all this structure exactly. This is not the usual definition. In order to achieve that definition we would have to study lax morphisms of algebras.

We shall give two further examples of monadicity over Cat: categories with pullbacks, and cartesian closed categories. The technique used to show monadicity is the same in both cases: to make use of the algebraic nature of adjunctions. More precisely, categorical structure is often expressed by means of an adjunction with a previously defined functor. To show that such structure is monadic, introduce basic operations for the adjoint functor and the unit and counit of the adjunction. Then use equations to say that the data for the adjoint functor is in fact functorial, and that the unit and counit are natural transformations, and finally that the triangle identities for an adjunction are valid.

Let's consider pullbacks, an example given in both [Bur81] and [DK83]. Here pullback along a morphism $f : a \to b$ is treated as the right adjoint f^* to the functor $f_* : A/a \to A/b$ given by composition with f. A slight complication is that we need to do this, not for a particular f, but an arbitrary one. So, to give the data for the functor we need an operation of arity

$$\begin{tikzcd} & \bullet \arrow[d] \\ \bullet \arrow[r] & \bullet \end{tikzcd}$$

(the initial data for a pullback square), producing a diagram of shape

$$\begin{tikzcd} \bullet \arrow[d] \\ \bullet \end{tikzcd}$$

(the left-hand side of the pullback square). This gives the effect of the functor on objects. Similarly we need an operation of arity

$$\begin{tikzcd} & \bullet \arrow[d] \\ & \bullet \arrow[d] \\ \bullet \arrow[r] & \bullet \end{tikzcd}$$

producing a diagram of shape

$$\begin{tikzcd} \bullet \arrow[d] \\ \bullet \arrow[d] \\ \bullet \end{tikzcd}$$

(the pullback of a composable pair), giving the action of the functor on morphisms. These are subject to the obvious equations expressing functoriality. There is obviously some redundancy here: we could have defined the object part of the functor from the morphism part. Conversely, if we are enriching over Cat, the morphism part is redundant, since the operation giving the object part is necessarily functorial.

Secondly, we need the data for the unit and counit of the adjunction. The counit is (essentially) the top side of the pullback square. So we need an operation of arity

$$\begin{tikzcd} & \bullet \arrow[d] \\ \bullet \arrow[r] & \bullet \end{tikzcd}$$

producing a diagram of shape

$$\begin{tikzcd} \bullet \arrow[r] & \bullet \end{tikzcd}$$

(again, if we wanted a compact rather than a structured presentation, we could with advantage package this with the object part of the functor into a single operation yielding the entire pullback square). The unit is given by the diagram below:

$$\begin{tikzcd} \bullet \arrow[dr, "\eta"] \arrow[drr, "1"] \arrow[dddr, "h"'] & & \\ & \bullet \arrow[r] \arrow[dd] \arrow[ddr, phantom, "\lrcorner", very near start] & \bullet \arrow[d, "h"] \\ & & \bullet \arrow[d, "f"] \\ & \bullet \arrow[r, "f"] & \bullet \end{tikzcd}$$

so we need an operation of arity

$$\begin{tikzcd} \bullet \arrow[d] & \\ \bullet \arrow[r] & \bullet \end{tikzcd}$$

giving a diagram of shape

$$\begin{tikzcd} \bullet \arrow[dr] & \\ & \bullet \end{tikzcd}$$

Once again, if we are enriching over Cat, as we can in this case, the naturality of both of these is automatic. If not, we have to force it by means of equations. Finally, we need the equations:

$$\epsilon \circ (f_* \eta) = 1 \text{ and } (f^* \epsilon) \circ \eta = 1$$

which are the triangle equalities for the adjunction.

For cartesian closed categories, the first part of the construction, giving binary products and a terminal object can be done in a similar fashion. The interesting part is the function space. The new thing here is that function space is a functor $A^{\mathrm{op}} \times A \to A$, which does not fit into the pattern $\mathsf{Cat}(c, A) \to A$, because of its contravariance in the first argument. This means that cartesian closed categories can not be constructed as the algebras for a Cat-enriched monad. However, we can regard the function space as a functor $\mathsf{Cat}(\mathsf{2}, A)_{\mathrm{iso}} \to A$, where $\mathsf{Cat}(\mathsf{2}, A)_{\mathrm{iso}}$ is the category whose objects are functors $\mathsf{2} \to A$, and whose morphisms are natural isomorphisms. This is the hom in the Gpd-enriched sense, and accordingly it turns out that cartesian closed categories are the algebras for a Gpd-enriched monad.

Note that working in a Gpd-enriched, rather than a Cat-enriched setting has the consequence that we have to be specific about the functoriality and naturality of any constructions. For the function space, we need an operation of arity $\mathsf{2}$ giving the space itself, and for the effect on morphisms, we need an operation of arity

$$\begin{tikzcd} \bullet & \bullet \arrow[d] \\ \bullet \arrow[u] & \bullet \end{tikzcd}$$

giving a result of shape

$$\begin{tikzcd} \bullet \arrow[d] \\ \bullet \end{tikzcd}$$

We also need the unit, an operation taking two objects x and y as arguments and producing a morphism $\eta : y \to [x \to x \times y]$, natural in y, as result, and the counit, again taking two arguments and producing $\epsilon : x \times [x \to y] \to y$, again natural in y as result. Finally, we once again impose the triangle identities.

It will be seen that a large part of the usual constructions of categories are monadic over Cat. Generally they are enriched over Cat when everything is covariant, and enriched over Gpd where there is some contravariance. Enrichment over Cat guarantees the naturality of all constructions, while enrichment over Gpd only guarantees naturality with respect to isomorphisms, a fairly low-level form of parametricity as in Robinson [Rob94]. Moreover, we can treat a functor between categories as a two-sorted algebra. This allows us to handle, for example, cartesian closed functors between cartesian closed categories.

One concept which however seems not to be monadic over any integer power of Cat is the notion of fibration. We do not have any definite proof that it is not monadic, but we can give an intuitive argument to explain why it is unlikely to be. This is that cartesian lifting, which is central to the definition, is only conditionally defined. We recall that a fibration is given by a functor $p : E \to B$ with certain properties. Certain morphisms of E have the property of being cartesian with respect to p, and for p to be a fibration, we require that for all objects e of E, and for all morphisms $f : b \to p(e)$, there is a cartesian morphism with codomain e mapping onto f under p (a cartesian lifting of f). We really require this to be an operation of our theory, but we cannot express its arity. The arities we have available in Cat^2 are pairs of finitely presentable categories. In this case the most appropriate pair is $\langle \mathsf{1}, \cdot \to \cdot \rangle$, but an operation of this arity would take an arbitrary object of E and morphism of B. We have no way of imposing the requirement that the object of E map onto the codomain of the morphism in B. Thus, unfortunately, fibrations live at one level up in a hierarchy of logical structure, the level of "essentially algebraic" structure. For one account of this level of structure, and more examples of categories monadic and not monadic over Cat, we refer the reader to Power [Pow95].

7 Subset-Cat

In recent work [Pow00] Power has been using algebraic structure over rather stranger categories to axiomatise premonoidal categories. Premonoidal categories were introduced in [PR97] as a generalisation of monoidal categories, the essential feature being that $\otimes$ was a bifunctor only with respect to a class of central maps. Both $f \otimes B$ and $A \otimes g$ are always defined, but unless one of f or g is central $(f \otimes B') \circ (A \otimes g)$ is not necessarily equal to $(A' \otimes g) \circ (f \otimes B)$, and $f \otimes g$ is not defined. Another way of putting this is to say that we have functors $h_A(\ ) = A \otimes (\ )$ and $k_B = (\ ) \otimes B$ such that $h_A(B) = k_B(A)$ and h_A is natural in A with respect to central maps. So a premonoidal category is a category with a distinguished class of maps (containing the identities and closed under composition), and some of the structure is natural only with respect to this distinguished subclass. This makes it impossible to axiomatise premonoidal categories using the techniques we have seen so far.

However, a category with a distinguished class of maps containing the identities and closed under composition is exactly a Subset-enriched category. Here, Subset is the category whose objects are sets X equipped with a distinguished subset $Y \subseteq X$ and where a map $(Y \subseteq X) \to (Y' \subseteq X')$ is a function $f : X \to X'$ such that $f(Y) \subseteq Y'$. The monoidal structure is product.

As always, Subset-Cat comes equipped with notions of Subset-functor (a functor which respects the distinguished maps), and Subset-natural transformation (a natural transformation whose components are distinguished maps). These make Subset-Cat into a 2-category, i.e. a Cat-enriched category[2]. In this section we show how to make algebraic over Subset-Cat the notion of a category C equipped with the following structure: a distinguished family of maps containing the identities and closed under composition; and an ob(C)-indexed family of functors $h_A(\ ) : \mathsf{C} \to \mathsf{C}$ all of which preserve distinguished maps and are natural in A with respect to distinguished maps. The presentation will be enriched over Cat. This is the core of Power's presentation of premonoidal categories, and interesting as a technique in its own right.

First some notation. Power writes a Subset-Cat A as $j : \mathsf{C} \to \mathsf{D}$ where D is the ambient category, C the collection of distinguished maps and j the inclusion functor. We will use this, but also use different arrows to distinguish the classes of morphisms: we will reserve the usual solid arrow $a \xrightarrow{f} a'$ for a distinguished morphism (a map from C in Power's notation), and use a dashed arrow $b \overset{g}{\dashrightarrow} b'$ for an arbitrary map (a map from D).This gives us two different two-object single-arrow categories: $\bullet \longrightarrow \bullet$, where the single map is in the distinguished class, and $\bullet \dashrightarrow \bullet$, where it is not.

As arity for the basic operations we take $c = (\bullet \quad \bullet \dashrightarrow \bullet)$. Since we are in the Cat-enriched setting, we are interested in the category of Subset-functors and Subset-natural transformations from c to A. A Subset-functor is given by an object and a D-map: $(a, b \overset{f}{\dashrightarrow} b')$. A Subset-natural transformation is

$$\begin{tikzcd} a \arrow[d] & b \arrow[r, dashed, "f"] \arrow[d] & b' \arrow[d] \\ c & d \arrow[r, dashed, "g"] & d' \end{tikzcd}$$

where all the vertical arrows are C-maps.

Our family of basic operations will be given by a Subset-category: $\bullet \dashrightarrow \bullet$. In this instance the tensor is easy to calculate. Using Power's notation

$$\mathsf{X} \otimes (j : \mathsf{C} \longrightarrow \mathsf{D}) = (\mathsf{X} \times j : \mathsf{X} \times \mathsf{C} \longrightarrow \mathsf{X} \times \mathsf{D})$$

So when $(j : \mathsf{C} \longrightarrow \mathsf{D}) = \bullet \dashrightarrow \bullet$, the only non-identity C-maps in the tensor are pairings of maps in X with identities.

[2]Subset-Cat is cartesian closed. The exponential is the category of all Subset-functors and all natural transformations, with those transformations all of whose components are distinguished as a distinguished subset. We could therefore take it as enriched over itself. That however would be too much for our purposes.

It follows that an interpretation gives

$$\begin{tikzcd}
h_0(a, b \overset{f}{\dashrightarrow} b') \arrow[d, dashed, "h_2(a, b \overset{f}{\dashrightarrow} b')"'] \\
h_1(a, b \overset{f}{\dashrightarrow} b')
\end{tikzcd}$$

This gives the data for the functor $h_a(\ )$. It will be the case that $h_0(a, b \overset{f}{\dashrightarrow} b') = h_a(b)$, $h_1(a, b \overset{f}{\dashrightarrow} b') = h_a(b')$ and $h_2(a, b \overset{f}{\dashrightarrow} b') = h_a(f)$. To ensure this, impose equations to make $h_a(b)$ well-defined (h_0 and h_1 do not depend on f, and $h_0(a, b \overset{f}{\dashrightarrow} b') = h_1(a, b' \overset{g}{\dashrightarrow} b'')$), and h_2 is functorial. Now consider a C-map $a \longrightarrow a'$. Since h is functorial we get a commuting square:

$$\begin{tikzcd}
h_0(a, b \overset{f}{\dashrightarrow} b') \arrow[r] \arrow[d, dashed, "h_2(a, b \overset{f}{\dashrightarrow} b')"'] & h_0(a', b \overset{f}{\dashrightarrow} b') \arrow[d, dashed, "h_2(a', b \overset{f}{\dashrightarrow} b')"] \\
h_1(a, b \overset{f}{\dashrightarrow} b') \arrow[r] & h_1(a, b \overset{f}{\dashrightarrow} b')
\end{tikzcd}$$

Writing this using $h_a(\ )$ we get

$$\begin{tikzcd}
h_a(b) \arrow[r] \arrow[d, dashed, "h_a(f)"'] & h_{a'}(b) \arrow[d, dashed, "h_{a'}(f)"] \\
h_a(b') \arrow[r] & h_{a'}(b')
\end{tikzcd}$$

which expresses the naturality of $h_a(\ )$ in a with respect to C-maps. Since our category of Subset-functors does not allow for D-maps $a \dashrightarrow a'$ we do not get naturality in D-maps in this automatic fashion.

There is also functoriality in $b \overset{f}{\dashrightarrow} b'$ to consider, but this does not add much. Suppose we have a square

$$\begin{tikzcd}
b \arrow[r, "u"] \arrow[d, dashed, "f"'] & c \arrow[d, dashed, "g"] \\
b' \arrow[r, "v"] & c'
\end{tikzcd}$$

then from it we get a square

$$\begin{tikzcd}
h_a(b) \arrow[r] \arrow[d, dashed, "h_a(f)"'] & h_a(c) \arrow[d, dashed, "h_a(g)"] \\
h_a(b') \arrow[r] & h_a(c')
\end{tikzcd}$$

This square commutes if we take the top and bottom maps to be $h_a(u)$ and $h_a(v)$ respectively. We do not need any extra structure, so we impose equations to force this to be the case. In that event the square tells us that $h_a(\ )$ preserves C-maps, which we did not previously know.

The construction just given is hardly elegant, even though we have left out the detail of the equations needed, but the essential point is clear: a single family of operations of unusual arity suffices to give all the essential data, including the naturality of the functors.

The construction of premonoidal categories does not involve much more. Simply repeat the construction for the functors $k_b(a)$, impose an equation for $h_a(b) = k_b(a)$, and add the appropriate natural transformations in C to make C a monoidal category. We saw how to do this in the previous section.

References

[BKP89] R. Blackwell, G.M. Kelly, and A.J. Power. Two-dimensional monad theory. *Journal of Pure and Applied Algebra*, 59:1–41, 1989.

[Bur81] A. Burroni. Algèbres graphiques. *Cahiers de topologie et géométrie différentielle*, 23:249–265, 1981.

[BW90] M. Barr and C. Wells. *Category Theory for Computer Science*. Prentice Hall, 1990.

[CH97] Chih-Ping Chen and Paul Hudak. Rolling your own mutable adt: A connection between linear types and monads. In *24'th Symposium on Principles of Programming Languages*. ACM Press, January 1997.

[DK83] E.J. Dubuc and G.M. Kelly. A presentation of topoi as algebraic relative to categories or graphs. *Journal of Algebra*, 81:420–433, 1983.

[Gor94] A.D. Gordon. *Functional Programming and Input/Output*. Distinguished Dissertations in Computer Science. Cambridge University Press, 1994.

[GP97] R. Gordon and A.J. Power. Enrichment through variation. *Journal of Pure and Applied Algebra*, 120:167–185, 1997.

[GP98] R. Gordon and A.J. Power. Algebraic structure for bicategory enriched categories. *Journal of Pure and Applid Algebra*, 130:119–132, 1998.

[GU71] P. Gabriel and F. Ulmer. *Lokal Präsentierbare Kategorien*, volume 221 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin, 1971.

[Joh82] P.T. Johnstone. *Stone Spaces*, volume 3 of *Cambridge studies in advanced mathematics*. Cambridge University Press, Cambridge, 1982.

[JW93] Simon Peyton Jones and Philip Wadler. Imperative functional programming. In *20'th Symposium on Principles of Programming Languages*, Charlotte, North Carolina, January 1993. ACM Press.

[Kel82a] G.M. Kelly. *Basic Concepts of Enriched Category Theory*, volume 64 of *London Mathematical Society Lecture Note Series*. Cambridge University Press, Cambridge, 1982.

[Kel82b] G.M. Kelly. Structures defined by finite limits in the enriched context, I. *Cahiers de Topologie et Géométrie Différentielle*, 23:3–41, 1982.

[KP93] G.M. Kelly and A.J. Power. Adjunctions whose counits are coequalizers, and presentations of finitary enriched monads. *Journal of Pure and Applied Algebra*, 89:163–179, 1993.

[KPT99] Yoshiki Kinoshita, John Power, and Makoto Takeyama. Sketches. *Journal of Pure and Applied Algebra*, 143:275–291, 1999.

[Law73] F.W. Lawvere. Metric spaces, generalized logic, and closed categories. *Rendiconti del Seminario Matematico e Fisico di Milano*, 43:135–166, 1973.

[LH96] Sheng Liang and Paul Hudak. Modular denotational semantics for compiler construction. In *European Symposium on Programming*, April 1996.

[LHJ95] Sheng Liang, Paul Hudak, and Mark Jones. Monad transformers and modular interpreters. In *22'nd Symposium on Principles of Programming Languages*. ACM Press, January 1995.

[Lin66] F.E.J. Linton. Some aspects of equational categories. In S. Eilenberg, D.K. Harrison, S. MacLane, and H. Röhrl, editors, *Proceedings of the Conference on Categorical Algebra, La Jolla 1965*, pages 84–94. Springer-Verlag, 1966.

[LS86] J. Lambek and P.J. Scott. *Introduction to Higher Order Categorical Logic*, volume 7 of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press, Cambridge, 1986.

[Mog91] E. Moggi. Notions of computation and monads. *Information and Computation*, 93(1):55–92, July 1991.

[Pow90] A.J. Power. An algebraic formulation for data refinement. In M. Main, A. Melton, M. Mislove, and D. Schmidt, editors, *Mathematical Foundations of Programming Semantics, 5th International Conference, Tulane, March/April 1989*, pages 390–401, 1990.

[Pow95] A.J. Power. Why tricategories? *Information and Computation*, 120:251–262, 1995.

[Pow00] J. Power. Premonoidal categories as categories with algebraic structure. *Theoretical Computer Science*, 2000. To appear.

[PR97] John Power and Edmund Robinson. Premonoidal categories and notions of computation. *Mathematical Structures in Computer Science*, 7:453–468, 1997.

[Rob94] Edmund Robinson. Parametricity as isomorphism. *Theoretical Computer Science*, 136(1):163–181, 1994.

[Wad95] Philip Wadler. Monads for functional programming. In J. Jeuring and E. Meijer, editors, *Advanced Functional Programming*, volume 925 of *LNCS*. Springer Verlag, 1995.

[Wad97] Philip Wadler. How to declare an imperative. *ACM Computing Surveys*, 29(3):240–263, September 1997.

[Wad98] Philip Wadler. The marriage of effects and monads. In *International Conference on Functional Programming*, Baltimore, September 1998.