



Countable Lawvere Theories and Computational Effects

John Power^{1,2}

*Laboratory for the Foundations of Computer Science, University of Edinburgh, King's Buildings,
Edinburgh EH9 3JZ, SCOTLAND*

Abstract

Lawvere theories have been one of the two main category theoretic formulations of universal algebra, the other being monads. Monads have appeared extensively over the past fifteen years in the theoretical computer science literature, specifically in connection with computational effects, but Lawvere theories have not. So we define the notion of (countable) Lawvere theory and give a precise statement of its relationship with the notion of monad on the category *Set*. We illustrate with examples arising from the study of computational effects, explaining how the notion of Lawvere theory keeps one closer to computational practice. We then describe constructions that one can make with Lawvere theories, notably sum, tensor, and distributive tensor, reflecting the ways in which the various computational effects are usually combined, thus giving denotational semantics for the combinations.

Keywords: mathematical operational semantics, modularity, timed transition systems, comonads, distributive laws

1 Introduction

Historically, there have been two main category theoretic formulations of universal algebra. The earlier was by Bill Lawvere in his doctoral thesis in 1963 [16]. Nowadays, his central construct is usually called a Lawvere theory, more prosaically a single-sorted finite product theory [1,2]. The notion of Lawvere theory axiomatises the notion of the clone of an equational theory. So every equational theory generates a Lawvere theory, and every Lawvere theory is generated by an infinite class of equational theories, i.e., all those equational theories for which it forms the clone. The notion of equational theory can in turn be given a category-theoretic formulation in terms of the notion of a single-sorted finite product sketch, the notion of sketch having been introduced by Ehresmann [1,2,5].

¹ This work is supported by EPSRC grant GR/586372/01: A Theory of Effects for Programming Languages.

² Email: ajp@inf.ed.ac.uk

The second category-theoretic formulation of universal algebra, which was in terms of monads, arose less directly. There was an extant notion of monad or triple that existed in the study of algebraic topology for reasons distinct from universal algebra. But in the late 1960's, Linton, apparently inspired by a surprising and profound characterisation of monads by Eilenberg and Moore in [6], proved that every equational theory gives rise, somewhat indirectly, to a monad on the category *Set* [1]. With some effort, one can see that the finitary monads on *Set* are precisely those monads that arise [1,14], where finitariness is a size condition. The notion of Lawvere theory can be shown directly to be equivalent to that of a finitary monad (see [27] for an enriched version), the central construction required to prove the equivalence being the Kleisli construction [1,17]. We give the details of a countable version of this in Section 2.

Moving forward to the late 1980's, computer scientists, led by Eugenio Moggi, became enamoured of the notion of monad [19,20,21]. Moggi wanted to unify the study of what he called notions of computation, perhaps better called computational effects, which he presented as a list of imperative features that one might add to an otherwise purely functional programming language. These included features such as exceptions, side-effects, interactive input/output, nondeterminism and probabilistic nondeterminism. He also included partiality and continuations in his list, although they are of a somewhat different nature, for instance because partiality arises from recursion without any imperative behaviour. Here, we shall not address the latter two constructs.

In retrospect, it seems obvious that universal algebra was fundamental to Moggi's idea although that was not clear to him at the time. The various computational effects arise from computationally natural operations, such as *raise* for exceptions, *lookup* and *update* for side-effects, *read* and *write* for interactive input/output, nondeterministic \vee for nondeterminism, and $[0, 1]$ -many binary operations $+_r$ for probabilistic nondeterminism, subject to computationally natural equations; and thus they arise from computationally natural equational theories by Linton's construction. We give the details of the examples in Section 3.

Having noticed that Moggi's monads arise from such computationally natural equational theories, one wonders whether the notion of Lawvere theory might be of any use. That question led to a series of papers, primarily by Martin Hyland, Gordon Plotkin and myself [10,11,22,23,24]. Providing that one is willing to make the routine extension of the notion of Lawvere theory to allow for countable arities, they prove to be particularly helpful. They inherently give rise to the first unified account of the computational operations associated with each computational effect; they allow one to distinguish between different sorts of computational effects; they allow one naturally to abstract from the setting of functional languages to a less context-dependent analysis; and they allow an elegant and natural theory of combining effects, based on constructions that have long appeared in the literature, such as the sum and tensor product of theories [7,29]. The constructions that allow one to combine Lawvere theories are of independent mathematical interest, so we explain our leading examples of such constructions in Section 4. These operations

give mathematical substance to the notion of monad transformer [3,4], replacing the latter notion by more primitive constructs.

Once one has this theory of computational effects, which is independent of functional programming, one can elegantly reincorporate it back into functional programming [28], allowing one a more modular analysis than Moggi had. It works surprisingly simply: the Lawvere theory itself directly gives rise to a canonical model of the relevant effects together with the first-order fragment of Moggi’s computational λ -calculus [19,20,21,28]. We do not give the details here as they would lead us away from our main task, which is to explain Lawvere theories.

There are two caveats to the above analysis. First, Lawvere theories as normally defined are inherently finitary. But because of recursion and hence the leading role of *Nat* in computer science, the main computing interest lies in a mild generalisation from finitariness to a countability condition. But that is routine, the various theorems generalising without fuss. So in our technical development, we shall simply describe the countable version. Second, again because of recursion, computer science ultimately requires base categories such as ωCpo rather than *Set*. That involves enrichment of the notion of Lawvere theory [27]. The enrichment is routine, but to understand the details requires some knowledge of enriched category theory [13], so for simplicity of exposition, we shall not give the enriched version here.

The paper is organised as follows. We recall the definition of a countable Lawvere theory and explain the relationship of the definition with the notion of monad on *Set* in Section 2. We illustrate with examples arising from computational effects in Section 3. And we discuss some of the constructions one can make naturally in terms of countable Lawvere theories in Section 4. There is no substantial new technical content in the paper: it is a distillation of work primarily in [10,11,24,26,27]. For some of the ideas in this paper addressed from a more computational perspective, see [25].

2 Countable Lawvere theories

In this section, we first give the definition of a countable Lawvere theory. We then show how every countable Lawvere theory yields a monad on *Set*. The monads that thus arise are exactly those monads on *Set* that are of countable rank, which is a size condition [13,14]. The correspondence extends to an equivalence of categories between Law_c , the category of countable Lawvere theories, and Mnd_c , the category of monads on *Set* with countable rank [27].

Definition 2.1 Let \aleph_1 denote a skeleton of the category of countable sets and all functions between them.

Spelling out the meaning of the definition, the category \aleph_1 has an object for each natural number n and an object for \aleph_0 . Up to equivalence, \aleph_1 is the free category with countable coproducts on 1: the coproducts are given by cardinal sum. In referring to \aleph_1 , we implicitly make a choice of the structure of its countable

coproducts. Since \aleph_1 has countable coproducts, it is immediate that the opposite category \aleph_1^{op} has countable products.

Definition 2.2 A *countable Lawvere theory* consists of a small category L with countable products and a strict countable-product preserving identity-on-objects functor $I : \aleph_1^{op} \rightarrow L$. A map of countable Lawvere theories from L to L' is a strict countable-product preserving functor from L to L' that commutes with I and I' .

So the objects of any countable Lawvere theory L are exactly the objects of \aleph_1 , and every function between such objects yields a map in L . One often refers to the maps of a countable Lawvere theory as *operations*. Trivially, the definitions of countable Lawvere theory and map between them yield a category Law_c , with composition given by ordinary composition of functors. Note that in the definition of countable Lawvere theory, I need not be an inclusion.

Example 2.3 There is a Lawvere theory $Triv$ that is equivalent to the unit category 1 : its objects are the objects of \aleph_1 , and there is one arrow from any object to any other object. The functor I is the identity-on-objects but is trivial on maps.

That is a useful example for us in constructing counter-examples to natural conjectures. Although trivial, it is important to the structure of the category Law_c as it is the terminal object of Law_c , therefore corresponding to the terminal object of Mnd_c , which is the monad sending every set to 1 . Despite this example, it is generally harmless to pretend that I is faithful, as it is in all examples of primary interest. For most mathematical purposes, one understands a countable Lawvere theory by study of its models.

Definition 2.4 A *model* of a countable Lawvere theory L in any category C with countable products is a countable-product preserving functor $M : L \rightarrow C$.

Definition 2.5 For any countable Lawvere theory L and any category C with countable products, the category $Mod(L, C)$ is defined to have objects given by all models of L in C , with maps given by all natural transformations between them.

The semantic category C of primary interest is Set . So consider a model M of a countable Lawvere theory L in Set . The set $M1$ determines Mn up to coherent isomorphism for every n in L : for M preserves countable products of L , equivalently of \aleph_1^{op} , these are countable coproducts of \aleph_1 , which are given by cardinal sum, and so Mn must be the product of n copies of $M1$. So, to give a model M is equivalent to giving a set $X = M1$ together with, for each map of the form $f : m \rightarrow 1$ in L , a function from X^m to X , subject to the equations given by the composition and product structure of L . This analysis routinely extends to any category C with countable products.

The definition of map in $Mod(L, C)$ is more subtle than it may first appear. One can readily prove that the naturality condition implies that all natural transformations between models respect countable product structure, i.e., for any natural transformation α between models M and N , and for any n in \aleph_1 , the map $\alpha_n : Mn \rightarrow Nn$ is given by the product of n copies of $\alpha_1 : M1 \rightarrow N1$. So

the maps in $Mod(L, C)$, which we defined to be all natural transformations, could equally be defined to be all natural transformations that respect the product structure of L .

The requirement that M preserves projections, which is part of what preservation of products means, determines the the behaviour of M on If for every function f : for projections in L amount to coprojections in \aleph_1 , and every function f is given by a family of coprojections.

The above discussion leads to the notion of a *single-sorted countable-product sketch*, which is a category-theoretic formulation of the notion of equational theory. The usual way in which to obtain countable Lawvere theories is by means of sketches, with the Lawvere theory given freely on the sketch: Barr and Wells' book [2] treats sketches in loving detail, with a leading example given by the sketch for semigroups, i.e., they describe a single-sorted countable-product sketch for which the induced countable Lawvere theory L_{SG} is determined by the property that the category $Mod(L_{SG}, Set)$ is equivalent to the usual category of semigroups. To give a sketch amounts to giving operations and equations, the operations being allowed to be of countable arity, i.e., an equational theory with operations possibly of countable arity. We shall give our leading examples in the next section.

There is a canonical forgetful functor $U_L : Mod(L, C) \rightarrow C$ given by evaluation at the object 1 of L , equally of \aleph_1 . If that forgetful functor has a left adjoint F_L , as it does whenever C is locally countably presentable, it follows from Beck's monadicity theorem [1] that it exhibits $Mod(L, C)$ as equivalent to the category T_L-Alg for the induced monad T_L on C . In particular, Set is locally countably presentable, so every countable Lawvere theory L induces a monad T_L on Set . With only a little more effort, one can prove the following.

Proposition 2.6 *The construction sending a countable Lawvere theory L to the monad T_L determined by the forgetful functor $U_L : Mod(L, Set) \rightarrow Set$ given by evaluation at 1 extends to a functor from Law_c to Mnd .*

One can readily check that for every countable Lawvere theory L , the monad T_L is of countable rank. The issue of rank is something of a distraction for the purposes of this paper: our aim here is to compare the notion of countable Lawvere theory with that of monad, without much concern for exactly what class of monads is obtained beyond the fact of their including all our leading examples. We need to mention rank only in order to make a precise statement of the converse to the construction we have just described, but development of it would detract from our goal. So we leave it with a reference [13], but without further analysis.

For a converse, first observe that for any monad T on Set , the Kleisli category $Kl(T)$ has all coproducts and the canonical functor $I : Set \rightarrow Kl(T)$ preserves them: for the canonical functor I has a right adjoint and is identity-on-objects. So, restricting I to \aleph_1 , which is a skeleton of the full subcategory of Set determined by countable sets, we have (the opposite of) a countable Lawvere theory. With a little more effort, we have the following.

Proposition 2.7 *The construction sending a monad T on Set to the category*

$Kl(T)_{\aleph_1}^{op}$ determined by restricting $Kl(T)$ to the objects of \aleph_1 extends to a functor from Mnd to Law_c .

It is routine to verify that, for any countable Lawvere theory L , the countable Lawvere theory L_{T_L} is isomorphic in Law to L . But the corresponding statement if one began with a monad T on Set is not true because, as mentioned above, the construction T_L yields only monads with countable rank. However, if T has countable rank, the functor from $T\text{-Alg}$ to $Mod(L_T, Set)$ induced by the restriction is an equivalence of categories, and so T is isomorphic to T_{L_T} in Mnd . An enriched, thereby more general, version of the following result appears in [27].

Theorem 2.8 *The construction sending a countable Lawvere theory L to T_L together with that sending a monad T with countable rank to L_T induce an equivalence of categories between the category Law_c of countable Lawvere theories and the category Mnd_c of monads with countable rank on Set . Moreover, the comparison functor exhibits an equivalence between the categories $Mod(L, Set)$ and $T_L\text{-Alg}$.*

We have seen that to give a monad with countable rank on Set is equivalent to giving a countable Lawvere theory. In our motivating class of computational examples, which we explore in Section 3, the countable Lawvere theory changes the emphasis from the assignment to each set X of the set TX of values associated with a computational effect to the study of the operations associated with that computational effect; and that change proves to be fundamental to modelling the commutative combination of effects, to explaining their sum [10,11] and to discussing distributivity.

The work in this section enriches without fuss. The only point that does not enrich routinely is the informal discussion about equational theories: as best we know, there is currently no enriched notion of equational theory corresponding to enriched Lawvere theories. However, as we have outlined, that part of the discussion could be phrased in terms of sketches, for which an enriched account does exist or at least can readily be gleaned from the literature [15].

3 Examples Arising from Computational Effects

In this section, we consider examples of computational effects from the perspective of countable Lawvere theories. These computational effects have, in the past, been studied in terms of monads, as advocated fifteen years ago by Moggi [19,20,21], and they have recently been studied in terms of countable Lawvere theories by Hyland, Plotkin, and myself, notably in [10,11]. The latter approach deals more directly with the various effects in that it takes the operations generating the effects as primitive, whereas the monad approach does not give a unified account of the operations that induce each effect at all.

Example 3.1 The countable Lawvere theory L_E for exceptions is the free countable Lawvere theory generated by an operation *raise* : $0 \rightarrow E$, where E is a countable set of *exceptions*. In terms of operations and equations, this corresponds

to an E -indexed family of nullary operations with no equations. Note our use of the countable set E for the codomain of the operation of the Lawvere theory; strictly speaking we should instead have used the corresponding object of \aleph_1 , namely \aleph_0 ; it is, however, conceptually convenient to allow ourselves such minor liberties.

The monad generated by L_E is $T_E = - + E$. More generally, if C is any category with countable powers and countable coproducts, $Mod(L_E, C)$ is equivalent to the category of algebras for the monad $- + \underline{E}$, where \underline{E} for the E -fold copower of 1, i.e., $\coprod_E 1$.

In the case of side-effects, a sketch, and hence the countable Lawvere theory, is essentially given in [24] and is easy to describe.

Example 3.2 The countable Lawvere theory L_S for side-effects, where $S = Val^{Loc}$, is the free countable Lawvere theory generated by the operations $lookup : Val \rightarrow Loc$ and $update : 1 \rightarrow Loc \times Val$ subject to the seven natural equations listed in [24], four of them specifying interaction equations for $lookup$ and $update$ and three of them specifying commutation equations. Note, as in the case of exceptions, the use of codomains, here Loc and $Loc \times Val$, to handle indexing at the Lawvere theory level. It is shown in [24] that this Lawvere theory corresponds to the side-effects monad. More generally, if C is any category with countable powers and copowers then, slightly generalising the result in [24], $Mod(L_S, C)$ is equivalent to the category of algebras for the monad $(S \times -)^S$ where we write $(S \times -)$ for the S -fold copower $\coprod_S -$, and $(-)^S$ for the S -fold power $\prod_S -$.

For the next example, given any endofunctor F on a category C , let $\mu y.Fy$ denote the initial F -algebra if it exists. Then, for an endofunctor Σ on a category C with binary sums, the free Σ -algebra on an object x is $\mu y.(\Sigma y + x)$, with one existing if and only the other does. These free algebras exist if, for example, C is locally countably presentable and Σ has countable rank.

Example 3.3 The countable Lawvere theory $L_{I/O}$ for interactive input/output is the free countable Lawvere theory generated by operations $read : I \rightarrow 1$ and $write : 1 \rightarrow O$, where I is a countable set of *inputs* and O of *outputs*. The monad for interactive input/output $T_{I/O}(X) = \mu Y.(O \times Y + Y^I + X)$ corresponds to this Lawvere theory: $T_{I/O}(X)$ is the free Σ -algebra on X , where $\Sigma Y = O \times Y + Y^I$ is the *signature* functor determined by the two operations; an algebra for Σ consists of an O -indexed family of unary operations and an I -ary operation. This is also the form of $T_{I/O}$ in the more general situation where it corresponds to $Mod(L_S, C)$ for a locally countably presentable category C .

Example 3.4 The countable Lawvere theory L_N for (binary) nondeterminism is the countable Lawvere theory freely generated by a binary operation $\vee : 2 \rightarrow 1$ subject to equations for associativity, commutativity and idempotence, i.e., the countable Lawvere theory for a semilattice; the corresponding monad on Set is the finite non-empty subset monad \mathcal{F}^+ .

Example 3.5 The countable Lawvere theory L_P for probabilistic nondeterminism is that freely generated by $[0, 1]$ -many binary operations $+_r : 2 \rightarrow 1$ subject to the

equations for associativity, commutativity and idempotence in [8]. The corresponding monad on *Set* is the distributions with finite support monad, \mathcal{D}_f .

The category *Set* is not the category of primary interest in denotational semantics. One is more interested in ωCpo , and variants, in order to model recursion. The relationship between countable Lawvere theories and monads with countable rank generalises without fuss to one between countable *enriched* Lawvere theories and strong monads with countable rank on the category in which the enrichment takes place. For that theory to work, it suffices that category be locally countably presentable as a cartesian closed category. The category ωCpo is an example of such a category. So the work here generalises to include ωCpo [10,11,27].

4 Constructions on Countable Lawvere Theories

The category Law_c has excellent category theoretic structure. We investigate some of its structure in this section. In particular, we investigate three binary operations on the category Law_c that yield denotational semantics for the most common combinations of the computational effects discussed in Section 3. The three operations are given by sum, tensor, and distributivity. Each of these binary operations, modulo size, yields one of Moggi's monad transformers: given any binary operation, if you fix one argument, you immediately have, modulo size, a monad transformer, i.e., a function from the set of monads to itself [3,4]. The binary operations may be applied many times, yielding combinations of multiple effects [10,11]. Most of the work in this section is an abbreviated version of work appearing in [11].

4.1 Sum

The category Law_c has coproducts or sums. In contrast, the category Mnd does not have sums. Sum is the most common way in which the computational effects of Section 3 are combined. The leading examples of the sum of computational effects are given by the combination of exceptions with all the other computational effects we consider: side-effects, interactive input/output and nondeterminism, and by the combination of interactive input/output with all other effects we consider except for side-effects.

The construction of the sum is complicated, especially when attempted in terms of monads. But all our examples of countable Lawvere theories are given freely on equational theories, and in those terms, the sum is easy to describe: one takes all operations of both equational theories, subject to all axioms of both. The complication arises in passing from the induced equational theory to the Lawvere theory freely generated by it, as, in doing so, one may apply the operations of one theory to the operations of the other, yielding a potentially transfinite induction in describing the set of all derived operations.

Care is required. For instance, given Lawvere theories L and L' , there are always maps of Lawvere theories given by coprojections $L \rightarrow L + L'$ and $L' \rightarrow L + L'$. But those coprojection functors need not be faithful. For instance, suppose L was

Triv. Then $L + L'$ is also *Triv*, so the coprojection from L' is trivial.

So we know the sum always exists and is a straightforward, familiar construction on equational theories. But for the purposes of calculation, it is sometimes convenient to have a more explicit construction of the sum qua monad, and that is investigated in detail in [10,11]. Two of the main consequences of the work therein are as follows.

Proposition 4.1 *Given a set E and any countable Lawvere theory L , the sum of the monads $(- + E)$ and T_L exists and is given by the monad $T_L(- + E)$.*

Modulo our usual caveat regarding size, this result explains how the exceptions monad transformer, sending a monad T_L to the composite $T_L(- + E)$, arises: one takes the disjoint union of the two sets of operations and retains the equations for T_L . And this explanation brings with it the theory of coproducts, such as their associativity and commutativity, and their interaction with other operations.

Proposition 4.2 *Let $T_{I/O}$ denote the monad for interactive input/output, i.e., the monad determined by the countable Lawvere theory of Example 3.3, and let L be any countable Lawvere theory. Then we have $(T_L + T_{I/O})x = T_L(\mu y.(O \times T_L y + (T_L y)^I + x))$, or equivalently, $\mu z.T_L(O \times z + z^I + x)$ [4].*

The central fact that allows us to make the above calculations is that, for exceptions and interactive input/output, the monads are generated by operations subject to no equations, and hence by a signature endofunctor [10,11].

4.2 Tensor

We now consider the tensor product $L \otimes L'$ of countable Lawvere theories L and L' [7,29]. The tensor product, which we are about to describe, yields a symmetric monoidal structure on the category Law_c with a universal property that exactly, modulo size, yields the side-effects monad transformer when one takes the tensor product of side-effects qua monad with any other countable Lawvere theory qua monad. This symmetric monoidal structure, with its defining universal property, seems unlikely to extend from Law_c , equivalently Mnd_c , to the whole of the category Mnd , but we do not have a counter-example to prove that.

The category \aleph_1 not only has countable coproducts, but also has finite products, which we denote by $a \times a'$. The object $a \times a'$ may also be seen as the coproduct of a copies of a' . So, given an arbitrary map $f' : a' \rightarrow b'$ in a countable Lawvere theory, it is immediately clear what we mean by the morphism $a \times f' : a \times a' \rightarrow a \times b'$. We define $f \times a'$ by conjugation, and, in the following, we suppress the canonical isomorphisms.

Definition 4.3 Given countable Lawvere theories L and L' , the countable Lawvere theory $L \otimes L'$, called the tensor product of L and L' , is defined by the universal property of having maps of countable Lawvere theories from L and L' to $L \otimes L'$, with commutativity of all operations of L with respect to all operations of L' , i.e., given $f : a \rightarrow b$ in L and $f' : a' \rightarrow b'$ in L' , we demand commutativity of the

diagram

$$\begin{array}{ccc}
 a \times a' & \xrightarrow{a \times f'} & a \times b' \\
 \downarrow f \times a' & & \downarrow f \times b' \\
 b \times a' & \xrightarrow{b \times f'} & b \times b'
 \end{array}$$

The tensor product always exists because it is defined by operations and equations, or equivalently by a sketch [1,2]. Its existence also follows, indeed more profoundly and elegantly, by appeal to the work on pseudo-commutativity in [12].

Proposition 4.4 *The tensor product \otimes extends canonically to a symmetric monoidal structure on the category of countable Lawvere theories.*

A proof for this proposition is elementary. The unit for the tensor product is the initial Lawvere theory, i.e, the theory generated by no operations and no equations. This is the initial object of the category of Lawvere theories, so is also the unit for the sum; and it corresponds to the identity monad.

This result gives some indication of the definitiveness of the tensor product, but not much. What is much less common, and is central to the proof of the main theorem about the combination of side-effects with other effects, and indeed is central to the understanding of what commutativity means, is a characterisation of $L \otimes L'$ in terms of the categories of models of L and L' [10,11].

Theorem 4.5 *For any category C with countable products, there is a coherent equivalence of categories between $Mod(L \otimes L', C)$ and $Mod(L, Mod(L', C))$.*

Corollary 4.6 *Let L_S denote the countable Lawvere theory for side-effects, where $S = Val^{Loc}$, and let L denote any countable Lawvere theory. Then the monad $T_{L_S \otimes L}$ is isomorphic to $(T_L(S \times -))^S$.*

Proof. For any countable Lawvere theory L , the category $Mod(L, Set)$ is complete and cocomplete, so has countable products and countable coproducts. And it is shown in [24] that if C has countable products and countable coproducts, $Mod(L_S, C)$ is equivalent to the category $T-Alg$ for the monad $T- = (S \times -)^S$ on C , using the notation of Example 3.2. By the discussion immediately before Proposition 2.6, the category $Mod(L, Set)$ is equivalent to T_L-Alg . We denote the canonical adjunction by $F_L \dashv U_L : Mod(L, Set) \rightarrow Set$. Right adjoints preserve products, left adjoints preserve coproducts. So the monad $T_{L_S \otimes L}$, which, by Theorem 4.5, is the monad determined by the composite forgetful functor from $T-Alg$ to Set , must be given by $T_{L_S \otimes L}X = U_L(S \times F_L X)^S = (U_L F_L(S \times X))^S = (T_L(S \times X))^S$ as required. □

This result shows that, under the hypotheses of the theorem, our theory of the tensor product of computational effects agrees with Moggi’s definition of the

side-effects monad transformer.

Corollary 4.7 *The side-effects theory for $S = Val^{Loc}$ is the Loc-fold tensor product of the side-effects theory for $S = Val$.*

Proof. By Corollary 4.6, the tensor product of two side effects theories, one for S and the other for S' , is the side effects theory for $S \times S'$. Now use induction and the finiteness of Loc . □

For a final example of the tensor product, let M be a monoid and consider the combination of any countable Lawvere theory L , equivalently T_L , with the monad $M \times -$. There is a canonical distributive law of the monad $M \times -$ over T_L determined by the unique strength $t : M \times T_L - \rightarrow T_L(M \times -)$ of $M \times -$ over T_L . So $T_L(M \times -)$ acquires a canonical monad structure. The following result appears in [11].

Theorem 4.8 *Let L be any countable Lawvere theory, let M be a monoid, and let L_M be the countable Lawvere theory corresponding to the monad $M \times -$. Then $T_{L_M \otimes L}$ is isomorphic to $T_L(M \times -)$.*

Proof. To give a model of L in $(M \times -)$ -Alg is equivalent to giving a model $m : L \rightarrow Set$ of L in Set , together with an M -action $\alpha : M \times m(1) \rightarrow m(1)$ on $m(1)$, such that the corresponding map $\bar{\alpha} : m(1) \rightarrow m(1)^M$ is a map of models. This in turn is equivalent to giving a T_L -algebra (X, β) and an M -action $\alpha : M \times X \rightarrow X$ on X such that $\bar{\alpha} : X \rightarrow X^M$ is a map of T_L -algebras. But that in turn is equivalent to giving a $T_L(M \times -)$ -algebra by generalities about distributive laws of monads [1]. These equivalences are all functorial, yielding an isomorphism from $T_{L_M \otimes L}$ -Alg to $T_L(M \times -)$ -Alg and hence an isomorphism of monads between $T_{L_M \otimes L}$ and $T_L(M \times -)$. □

Corollary 4.9 *The tensor product of $M \times -$ and $M' \times -$ is $(M \times M') \times -$.*

4.3 Distributivity

The third sort of combination of effects that appears in practice is given by distributivity, just like product distributes over sum in a ring. Distributivity of two nondeterministic operations over each other is central to Matthew Hennessy’s modelling of concurrency in [9]. It also applies to the combination of nondeterminism with probabilistic nondeterminism [18]. We shall not spell out here how to describe a distributive combination of countable Lawvere theories, but the idea is clear, broadly similar to the construction of the tensor product.

One wants to take the sum of both theories, then factor by equations that assert distributivity of the operations of one theory over the operations of the other. One can again characterise the category of models of the distributive combination of theories, but the characterisation is quite subtle. If a countable Lawvere theory L is commutative, the category $Mod(L, Set)$ is symmetric monoidal closed; and one can use $Mod(L, Set)$ together with that symmetric monoidal closed structure as a target symmetric monoidal category in which to model the underlying operad $O(L')$ of another theory L' . One can thus speak of the category $Mod(O(L'), Mod(L, Set))$

of models of $O(L')$ in $Mod(L, Set)$. The category $Mod(O(L'), Mod(L, Set))$, subject to taking a canonical pullback that is yet to be fully investigated, is then equivalent to the category of models of the distributive combination in Set . One must do something a little more subtle again if L is not commutative.

Unlike the sum and tensor product, the distributive combination of countable Lawvere theories is not symmetric, but it still yields a monoidal structure on Law_C . It does enrich, but not as directly and easily as do the other constructions.

References

- [1] M. Barr and C. Wells. *Toposes, Triples, and Theories* Grundlehren der math. Wissenschaften 278, Springer-Verlag, 1985.
- [2] M. Barr and C. Wells. *Category Theory for Computing Science* Prentice-Hall, 1990.
- [3] N. Benton, J. Hughes, and E. Moggi. *Monads and Effects* APPSEM '00 Summer School, 2000.
- [4] P. Cenciarelli and E. Moggi. em A syntactic approach to modularity in denotational semantics, CWI Technical Report, 1983.
- [5] C. Ehresmann. Esquisses et types des structures algebriques, *Bul. Inst. Polit. Iasi*, Vol. 14, pp. 1–14, 1968.
- [6] S. Eilenberg and J. C. Moore. Adjoint Functors and Triples, *Illinois J. Math.*, Vol. 9, pp. 381–398, 1965.
- [7] P. J. Freyd. Algebra-valued functors in general and tensor products in particular *Colloq. Math. Wroclaw* Vol. 14, pp. 89–106, 1966.
- [8] R. Heckmann. Probabilistic Domains, in *Proc. CAAP '94*, LNCS, Vol. 136, pp. 21–56, Berlin: Springer-Verlag, 1994.
- [9] M. C. B. Hennessy. *Algebraic Theory of Processes* Cambridge, Massachusetts: MIT Press, 1988.
- [10] J. M. E. H. Hyland, G. D. Plotkin, and A. J. Power. Combining computational effects: commutativity and sum, in *Proc. 2nd IFIP Conf on Theoretical Computer Science* (eds. Ricardo A. Baeza-Yates, Ugo Montanari and Nicola Santoro), pp. 474–484, Kluwer, 2002.
- [11] J. M. E. H. Hyland, G. D. Plotkin, and A. J. Power. Combining effects: sum and tensor, *Theoretical Computer Science* (to appear).
- [12] J. M. E. Hyland and A. J. Power. Pseudo-closed 2-categories and pseudo-commutativities *J. Pure Appl. Algebra*, Vol. 175, pp. 141–185, 2002.
- [13] G. M. Kelly. *Basic Concepts of Enriched Category Theory*, Cambridge: Cambridge University Press, 1982.
- [14] 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.
- [15] Y. Kinoshita, A. J. Power, and M. Takeyama, Sketches, *J. Pure Appl. Algebra*, Vol. 143, pp. 275–291, 1999.
- [16] F. W. Lawvere. *Functorial Semantics of Algebraic Theories*, PhD Thesis, Columbia University, 1963.
- [17] S. Mac Lane. *Categories for the Working Mathematician*, Berlin: Springer-Verlag, 1971.
- [18] M. W. Mislove. Nondeterminism and Probabilistic Choice: Obeying the Laws, in *International Conference on Concurrency Theory*, pp. 350–364, URL:citeseer.nj.nec.com/mislove00nondeterminism.html, 2000.
- [19] E. Moggi. Computational lambda-calculus and monads, in *Proc. LICS '89*, pp. 14–23, Washington: IEEE Press, 1989.
- [20] E. Moggi. *An abstract view of programming languages*, University of Edinburgh, Report ECS-LFCS-90-113, 1989.
- [21] E. Moggi. Notions of computation and monads, *Inf. and Comp.*, Vol. 93, No. 1, pp. 55–92, 1991.

- [22] 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.
- [23] 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.
- [24] G. D. Plotkin and A. J. Power. Notions of computation determine monads, in *Proc. FOSSACS 2002* (eds. M. Nielsen and U. Engberg), LNCS, Vol. 73, Amsterdam: Elsevier, 2002.
- [25] G. D. Plotkin and A. J. Power. Computational effects and operations: an overview, in *Proc. Workshop Domains 2002*, ENTCS, Vol. 2303, pp. 342–356, Berlin: Springer-Verlag, 2002.
- [26] G. D. Plotkin and A. J. Power. Algebraic operations and effects, in *Proc. MFCSIT 2000, Applied Categorical Structures*, Vol. 11, pp. 69–94, 2003.
- [27] A. J. Power. Enriched Lawvere Theories, in *Theory and Applications of Categories*, pp. 83–93, 2000.
- [28] A. J. Power. Canonical Models for Computational Effects, in *Proc. FOSSACS 2004* (ed. I. Walukiewicz), LNCS Vol. 2987, pp. 438–452, Berlin: Springer-Verlag, 2004.
- [29] H. Schubert, *Categories*, Springer-Verlag, 1972.