Combining Algebraic Effects with Continuations

Combining algebraic effects with continuations

Martin Hyland,[1] Paul Blain Levy,[2] Gordon Plotkin and John Power[3,⋆]

[1] Department of Mathematics, University of Cambridge, Cambridge CB3 0WB, England. Email: M.Hyland@dpmms.cam.ac.uk
[2] School of Computer Science, University of Birmingham, Birmingham B15 2TT, England. Email: P.B.Levy@cs.bham.ac.uk
[3] Laboratory for the Foundations of Computer Science, School of Informatics, University of Edinburgh, King's Buildings, Edinburgh EH9 3JZ, Scotland. Email: gdp@inf.ed.ac.uk, ajp@inf.ed.ac.uk.

Abstract. We consider the natural combinations of algebraic computational effects such as side-effects, exceptions, interactive input/output, and nondeterminism with continuations. Continuations are not an algebraic effect, but previously developed combinations of algebraic effects given by sum and tensor extend, with effort, to include commonly used combinations of the various algebraic effects with continuations. Continuations also give rise to a third sort of combination, that given by applying the continuations monad transformer to an algebraic effect. We investigate the extent to which sum and tensor extend from algebraic effects to arbitrary monads, and the extent to which Felleisen et al's $\mathcal{C}$ operator extends from continuations to its combination with algebraic effects. To do all this, we use Dubuc's characterisation of strong monads in terms of enriched large Lawvere theories.

1 Introduction

It is a very great pleasure to contribute to this Festschrift for John Reynolds. Continuations have been an abiding interest of John's, starting with his 1972 paper on definitional interpreters and with his most recent contribution being a historical piece on the origins of continuations [48–52]. Continuations have had several uses in programming languages and their theory. They have provided translations between languages, been used to give semantics for programming languages and have been employed in compilers. They also appear explicitly in programming languages where the programmer is given access to them as 'first-class entities' via control operators such as Rees and Clinger's `call/cc` [46] or Felleisen et al's $\mathcal{C}$ [10, 11]. Two such languages are the untyped language Scheme [1] and the typed language ML, particularly New Jersey Standard ML [8].

In this paper we concern ourselves with continuations in programming languages, in particular, the relation between continuations and other computational effects such as exceptions, side-effects, input/output and nondeterminism.

⋆ This work has been done with the support of EPSRC grant GR/S86372/01, AIST and the Japan Science and Technology Agency; Plotkin acknowledges the support of a Royal Society-Wolfson award.

This work forms part of a research programme to develop a theory of computational effects, following on seminal work of Eugenio Moggi who proposed the use of strong monads T to uniformly model computational effects [33, 34]. He further introduced the *computational λ-calculus* as a basic functional call-by-value language for computational effects, modelled in the Kleisli category of T, and introduced monad transformers as a way of combining effects [6].

One needs to have operations in the language to actually make the effects happen. The research programme focuses on these operations and their interaction as primary, via equational or Lawvere theories. In [40] it was shown how various well-known such monads arise as free algebra monads for natural such theories of the operations; in [18] it was shown how monads for combinations of such effects corresponded to natural combinations of the underlying equational or Lawvere theories, viz the sum or tensor of theories. This yielded a systematic account of monad transformers as well as a corresponding theory of operation transformers. Where possible the work was carried out at a rather general level, that of *enriched* category theory [22], for example considering Lawvere V-theories [43] and their combinations. One benefit of the more general approach is the ability to include domain-theoretic considerations, taking V to be, for example, ω-**Cpo**, the category of ω-cpos and continuous functions. A survey of the work done so far on this programme of an algebraic theory of computational effects appears in [42].

The operations considered are special ones, termed *algebraic operations* [41]. Working in **Set**, these are transformations of the form:

$$\mathrm{op}_X : (TX)^I \longrightarrow (TX)^O$$

subject to a certain naturality requirement. The monads that arise from equational theories of such operations are also special: they have a rank, meaning that for some cardinal κ they preserve κ-filtered colimits, see [2, 23]. Indeed in all natural examples they have finite or countably infinite rank. In our previous work the equivalence between V-monads of, say, countable rank and countable Lawvere V-theories was central to our analysis.

There are also natural operations that are not algebraic, with a notable example being exception handlers. Following a suggestion of Filinski one may call the algebraic operations *constructors* in that they create effects, whereas operations such as exception handlers are *deconstructors* in so far as they analyse what effects have happened. It seems that even when deconstructors are present, the monads are given by the theories on the constructors, as is, for example, the case for exceptions. However, at present no general mathematical account of deconstructors is available.

Moggi could include continuations via the monad R^{R^-}. However this monad does not have a rank. Furthermore, none of the natural operations are algebraic, in fact they do not even have the required form. For example, consider the typed version $\mathcal{C}_\sigma : ((\sigma \to 0) \to 0) \to 0$ of Felleisen et al's control operator, introduced by Griffin [15]. This is modelled by a transformation of the form:

$$\mathcal{C}_X : \neg_T \neg_T X \longrightarrow TX$$

where $\neg_T X =_{\text{def}} T0^X$. As well as not having the right form to be an algebraic operation, $\mathcal{C}_-$ inherently involves contravariance owing to the use of $\neg_T$. It may perhaps best be thought of as a *logical* operation rather than an *algebraic* one, as is, indeed, suggested by the various works linking continuations and classical logic within the Curry-Howard paradigm, e.g., [15, 36]. We choose to focus on $\mathcal{C}$ for reasons which we find persuasive though possibly not definitive: most other control operators are definable from it; the corresponding rule is central in natural deduction treatments of classical logic; and the operator is mathematically both natural and elegant. An operator not definable from $\mathcal{C}$ is *prompt*, but that fits within classical subtractive logic rather than classical implicational logic [3].

In Section 2 we discuss the structure needed to support the combination of the computational λ-calculus, continuations and algebraic effects. It consists of a monad T, a continuations transformation $\mathcal{C}$, and a map of monads $S \to T$ where S is the monad for the algebraic operations at hand. We particularly consider what axioms to impose on the continuations operator. As regards combinations of continuations with other effects one can apply exception and state transformers and there is also the well-known continuations transformer. We examine the three constructions in Sections 3, 4 and 5 respectively. As we shall see, strong axioms on the continuations transformation are not preserved by the sum construction. Also, other than in the case of the continuations transformer, we unfortunately have no universal characterisation of the combinations in terms of axioms on exceptions, or state, and continuations.

A question arise as to the mathematical status of these combinations, and of corresponding extensions of the operations. One problem is that if one wishes to think in terms of Lawvere theories then one needs a notion of *large* Lawvere theory, to correspond to monads without a rank. Fortunately, an enriched notion of such theories has previously been developed by Dubuc [9]. He showed that to give an arbitrary strong monad on a complete and cocomplete cartesian closed category V is equivalent to giving what we call a *large Lawvere V-theory*. That equivalence is less complex than the size-restricted one: given a strong monad T, its large Lawvere V-theory L_T is given by $Kl(T)^{op}$, the opposite of the Kleisli V-category. But the price for that simplicity is that sums and, we conjecture, tensors do not exist in general. We recall and explain the result in Appendix B, and we feel free to use it as convenient: it is often simpler to work in terms of theories and, indeed, that is how we discovered several of our results.

We end the paper in Section 6 with a formula for a typical combination of effects, making clear the elegance and simplicity obtained by our analysis, followed by some discussion of what is missing in our account and what remains to be done.

For the sake of simplicity of exposition, we express our results without referring to enrichment, that is we take $V = \mathbf{Set}$. However, all of our general results extend to the enriched case in a standard way [21]; see [18] for an example. In particular they extend to the important case where $V = \omega\text{-}\mathbf{Cpo}$ where the strong monads are exactly the locally continuous monads often considered in domain theory, i.e., those respecting lubs of ω-chains of morphisms.

2 The Computational λ-Calculus, Continuations and Algebraic Operations

A categorical treatment of the control operation $\mathcal{C}$ has been given recently by Führmann and [14], partly following on previous work of Hofmann [17]. We adapt this a little to the context of the computational λ-calculus, λ_c, treating first-class continuations in terms of suitable basic types and unary function symbols and their rules. We give an account of this calculus in Appendix A, extended with type variables in order to get a smoother treatment of the relation between conditions on the interpretation of the calculus and semantic properties, such as the truth of equations. See [16] for another account of this calculus and [33] for the original source.

To model λ_c by itself in **Set** we assume given a monad T, noting that in **Set** all monads have a unique strength and all monad maps are strong (i.e., they commute with the strengths). The Kleisli exponential $X \Rightarrow_T Y =_{\text{def}} TY^X$ acts functorially as an 'implication' bifunctor $\Rightarrow_T : (\mathbf{Set}_T)^{\text{op}} \times \mathbf{Set}_T \to \mathbf{Set}$; pre- or post-composing with the left adjoint $J : \mathbf{Set} \to \mathbf{Set}_T$ yields bifunctors on **Set** or $\mathbf{Set}_T$, respectively; we may omit J when writing these, if it is clear what is meant from the context.

We next assume given a basic type symbol 0 together with a family of constants $\mathcal{A}_\sigma : 0 \to \sigma$. We interpret 0 by the initial object of **Set**, viz the empty set, and then the $\mathcal{A}_\sigma$ receive a unique interpretation, and Hofmann's equations $\mathcal{A}$-Abs and $\mathcal{A}_0$-Id are satisfied. Using Moggi's logic for λ_c these can be written as, respectively :

$$\Gamma, f : (\sigma \to \tau) \vdash f(\mathcal{A}_\sigma(M)) = \mathcal{A}_\tau(M)$$

for $\Gamma \vdash M : \sigma$ and:

$$\Gamma \vdash \mathcal{A}_0(M) = M$$

for $\Gamma \vdash M : 0$. Our interpretation also validates the following rule, written using Moggi's existence predicate:

$$\frac{\Gamma \vdash M\!\downarrow}{\Gamma \vdash \mathcal{A}_\sigma(M)\!\downarrow}$$

Finally, recall $\neg_T X$ from the Introduction: this is the object part of a 'negation' contravariant functor $\neg_T = - \Rightarrow_T 0 : \mathbf{Set}_T^{\text{op}} \to \mathbf{Set}$; as before, pre- or post-composing with J we obtain functors on **Set** or $\mathbf{Set}_T$, respectively, and we may again omit the J.

The typed continuations operator $\mathcal{C}_\sigma : \neg\neg\sigma \to \sigma$ (where $\neg\sigma =_{\text{def}} \sigma \to 0$) is interpreted using a transformation:

$$\mathcal{C}^T : T0^{T0^-} \longrightarrow T$$

Before considering axioms for this transformation, it will be helpful to see how the continuation monad arises as a standard construction, and to discuss some of its properties.

One has a self-adjoint contravariant functor $R^- : \mathbf{Set}^{\text{op}} \to \mathbf{Set}$ for any given set R; the monad associated to this adjunction is the continuations monad R^{R^-}.

The unit $\eta_X : X \to R^{R^X}$ is given by the formula $\eta_X(x)(\kappa) = \kappa(x)$, and, for any map $f : X \to R^{R^Y}$, its Kleisli extension $f^\sharp : R^{R^X} \to R^{R^Y}$ is given by the formula $f^\sharp(\gamma)(\kappa') = \gamma(x \mapsto f(x)(\kappa'))$. The corresponding large Lawvere theory is equivalent to the full subcategory I_R of **Set** given by exponentials of R, i.e., $I_R(X, Y) = \mathbf{Set}(R^X, R^Y)$.

Proposition 1. *The following are in 1-1 correspondence for any monad T and set R:*

1. *T-algebras $r : TR \longrightarrow R$ on R*
2. *monad maps $d : T \longrightarrow R^{R^-}$*
3. *maps of large Lawvere theories $F : L_T \longrightarrow I_R$*

The correspondences are natural in T and R.

Proof. The correspondences between the first two are in [21, 24]: for any T-algebra $r : TR \to R$, the monad map is given by $d_X(\gamma) = \kappa \in R^X \mapsto r(T(\kappa)(\gamma))$, and given the monad map d the T-algebra map is given by $r(\gamma) = d_R(\gamma)(\mathrm{id}_R)$, and these correspondences are mutual inverses. The correspondences between the last two follow from the above characterisation of the large Lawvere theory of R^{R^-}. In one direction, given d, the component of F at X, Y, viz the map $F_{X,Y} : \mathbf{Set}(Y, TX) \to \mathbf{Set}(R^X, R^Y)$ is given by $F_{X,Y}(f)(\kappa)(y) = d_X(fy)(\kappa)$.

It follows that we have a monad map:

$$d^T : T \to T0^{T0^-} = \neg_T \neg_T$$

given by $d_X(\gamma) = \kappa \in T(0)^X \mapsto \kappa^\sharp(\gamma)$ where $\kappa^\sharp \in T(0)^{T(X)}$ is the Kleisli extension of κ. Note the special case where T is the continuations monad R^{R^-}. Here there is an evident isomorphism $\theta : R \cong T0$ and so T and $T0^{T0^-}$ are isomorphic and d^T is, in fact, the isomorphism. Its inverse provides the standard interpretation of the continuations operator. Explicitly:

$$\mathcal{C}^T_X(F) = \kappa \in R^X \mapsto \theta^{-1}(F(\theta \circ \kappa))$$

With this interpretation in mind, returning to the question of how to choose axioms for $\mathcal{C}$, it is natural to seek axioms for the transformation $\mathcal{C}^T$ in terms of its relation to d^T. One evident such property is that $\mathcal{C}^T$ is a retract, meaning a left inverse to d^T. This is equivalent to Hofmann's equation $\mathcal{C}$-App holding, that:

$$\Gamma \vdash \mathcal{C}_\sigma(\lambda\kappa.\sigma \to 0.\kappa M) = M$$

for $\Gamma \vdash M : \sigma$; the implication from the rule's holding to the semantic property makes evident use of the availability of type variables. Another evident such property is that $\mathcal{C}^T$ is right inverse to d^T; this is equivalent to Führmann and Thielecke's equation $\mathcal{C}$-Delay holding, that:

$$F : \neg\neg\sigma \vdash \lambda\kappa.\kappa(\mathcal{C}_\sigma F) = F$$

Yet another property is that $\mathcal{C}^T$ is a natural transformation in $\mathbf{Set}_T$ from $\neg_T\neg_T$ to Id, i.e., the following diagram commutes in $\mathbf{Set}_T$ for all X,T and $f : X \to TY$:

$$\begin{array}{ccc} \neg_T\neg_T X & \xrightarrow{\mathcal{C}_X} & X \\ \downarrow{\scriptstyle \neg_T\neg_T f} & & \downarrow{\scriptstyle f} \\ \neg_T\neg_T Y & \xrightarrow[\mathcal{C}_Y]{} & Y \end{array}$$

This property is equivalent to Hofmann's equation $\mathcal{C}$-NAT holding, that:

$$\Gamma, f : \sigma \to \tau \vdash f(\mathcal{C}_\sigma(M)) = \mathcal{C}_\tau(\lambda\kappa : \tau \to 0.M(\lambda x : \sigma.\kappa(fx)))$$

for $\Gamma \vdash M : \neg\neg\sigma$.

The final property we consider is that $\mathcal{C}$ is a map of monads. Taking this apart, naturality in **Set** is equivalent to the following restricted version $\mathcal{C}$-NAT$_t$ of $\mathcal{C}$-NAT holding:

$$\frac{\Gamma, x : \sigma \vdash V \downarrow}{\Gamma \vdash V(\mathcal{C}_\sigma(M)) = \mathcal{C}_\tau(\lambda\kappa : \tau \to 0.M(\lambda x : \sigma.\kappa(Vx)))}$$

for $\Gamma \vdash V : \sigma \to \tau$. Preservation of multiplication is equivalent to the following equation $\mathcal{C}$-KLEISLI holding:

$$f : \sigma \to \neg\neg\tau, u : \neg\neg\sigma \vdash \mathcal{C}_\tau(\lambda k : \neg\tau.\, u(\lambda x : \sigma.\, fxk)) = \mathcal{C}_\tau(f(\mathcal{C}_\sigma(u))$$

Preservation of the unit corresponds to the following equation $\mathcal{C}$-UNIT holding:

$$x : \sigma \vdash \mathcal{C}_\sigma(\lambda\kappa.\sigma \to 0.\kappa x) = x$$

but that is just an instance of $\mathcal{C}$-APP.

Some interesting relationships then hold between these properties:

Proposition 2. *Suppose that $\mathcal{C}^T$ is left inverse to d^T. Then the following are equivalent:*

1. *$\mathcal{C}^T$ is right inverse to d^T.*
2. *$\mathcal{C}^T$ is a natural transformation in $\mathbf{Set}_T$*
3. *$\mathcal{C}^T$ is a monad map.*

Proof. That the first two are equivalent under the given assumptions is essentially given in [14], though the statement and proof there are syntactic. For the implication from the first to the second, one checks that the following diagram commutes in **Set**:

$$\begin{array}{ccc} TX & \xrightarrow{d_X} & \neg_T\neg_T X \\ \downarrow{\scriptstyle f^\sharp} & & \downarrow{\scriptstyle \neg_T\neg_T f} \\ TY & \xrightarrow[d_Y]{} & \neg_T\neg_T Y \end{array}$$

But then, as d^T and $\mathcal{C}^T$ are, by assumption, inverses, the conclusion follows.

For the converse implication first note that $\neg_T\neg_T 0 \cong T0$ in **Set**, with the inverse to d^T_0 being $F \mapsto F(\eta^T_0)$. So as $\mathcal{C}^T_0$ is left inverse to d^T_0, we have that $\mathcal{C}^T_0 = F \mapsto F(\eta^T_0)$. Then, taking $Y = 0$ in the naturality diagram, and any $f : X \to T0$, one calculates that, for $F \in \neg_T\neg_T X$, $f^\sharp(\mathcal{C}(F)) = F(f)$.

For the third property, if two transformations are mutually inverse and one is a monad map then so is the other. Conversely, fix a map $f : X \to TY$ to show naturality in $\mathbf{Set}_T$. We have to show, in **Set**, that $\mu^T_Y \circ (Tf) \circ \mathcal{C}^T_X = \mathcal{C}^T_Y \circ \neg_T\neg_T f$. We have:

$$\begin{aligned}
\mu^T_Y \circ (Tf) \circ \mathcal{C}^T_X &= \mu^T_Y \circ \mathcal{C}^T_{TY} \circ R^{R^f} && \text{(by naturality of } \mathcal{C}^T \text{ in } \mathbf{Set}) \\
&= \mu^T_Y \circ \mathcal{C}^T_{TY} \circ R^{R^{\mathcal{C}^T_Y}} \circ R^{R^{d^T_Y}} \circ R^{R^f} && (d^T \text{ is right inverse to } \mathcal{C}^T) \\
&= \mathcal{C}^T_Y \circ \mu_{R^{R^Y}} \circ R^{R^{d^T_Y}} \circ R^{R^f} && \text{(preservation of the multiplication)} \\
&= \mathcal{C}^T_Y \circ \neg_T\neg_T f
\end{aligned}$$

with the last line following from the equation $\mu_{R^{R^Y}} \circ R^{R^{d^T_Y}} \circ R^{R^f} = \neg_T\neg_T f$, whose verification is a straightforward calculation.

That said, what axioms should one assume on the continuations operator? To model continuations without any other effects, one simply assumes that $\mathcal{C}^T$ is a two-sided inverse to d^T, following [14]. Note that this trivially characterises T as being (isomorphic to) the continuations monad $T0^{T0^-}$. This can perhaps be regarded as a natural characterisation of the continuations monad via properties of $\mathcal{C}$, and roughly analogous to our previous characterisations of other monads by equational means. Unfortunately there is a big difference between the two situations: the other characterisations are *stable* in that when the effects are combined with other effects, the resulting monads can be characterized in terms of the equations of each collection of operations separately, possibly together with other, new, equations. However, as we shall see, when continuations are combined with exceptions the isomorphism no longer holds.

In general, therefore, we assume only that $\mathcal{C}^T$ is a natural transformation in **Set**, left inverse to d^T. The only claim we make for this axiomatisation is an empirical one, that the axioms are true in all the cases we know, with one exception. The exception is New Jersey Standard ML where the capture/escape mechanism gives modeling difficulties. This is discussed further in the conclusion, and our present view is that we prefer not to include this case in our theory but rather to argue against the capture/escape mechanism.

The axiomatisation seems rather weak, and it would be good to test it, perhaps by attempting to determine whether it implies the soundness of operational semantics of continuations and effects such as those of [54, 55]; Proposition 2 shows that some possible strengthenings are too strong as they imply that $\mathcal{C}$ is an isomorphism. In the following, we study both how our assumption and the assumption of invertibility propagate under the various combinations of algebraic effects and continuations.

We now turn to incorporating algebraic operations, and begin with some generalities. For any monad T, an algebraic operation of *arity* (I, O) over T is a

family of maps:

$$\mathrm{op} : (T-)^I \to (T-)^O$$

natural in $\mathbf{Set}_T$. There is a 1-1 correspondence between such algebraic operations and *generic effects* $g : O \to TI$ [41], i.e., morphisms from I to O in the large Lawvere theory of T. The result is an instance of the Yoneda embedding. Explicitly, the correspondence is as follows: given such a g, the corresponding algebraic operation is:

$$(TX)^I \quad = (I \Rightarrow_T X) \xrightarrow{g \Rightarrow_T X} (O \Rightarrow_T X) = \quad (TX)^O$$

and given an algebraic operation op, the corresponding generic effect is:

$$1 \xrightarrow{\ulcorner \eta_I \urcorner} (TI)^I \xrightarrow{\mathrm{op}_I} (TI)^O$$

In the case of monads presented by a finitary or countably infinitary equational theory, every operation symbol in the signature of the theory is associated to an algebraic operation, namely the one which at X is the interpretation of that operation symbol in the free algebra $T(X)$. These algebraic operations therefore obey the equations of the equational theory at every X. There is also a converse: that every algebraic operation is definable from the algebraic operations corresponding to the operation symbols in the signature of the theory.

Let us now consider algebraic operations $\mathrm{op} : (R^{R^-})^I \to (R^{R^-})^O$ of arity (I, O) on the continuations monad R^{R^-}. These are in 1-1 correspondence with maps $g : O \to R^{R^I}$ and so with maps $h : R^I \to R^O$ which can be thought of as operations on R of arity (I, O). Given such an operation h, the corresponding algebraic operation is defined in a pointwise fashion as:

$$\mathrm{op}_X = (R^{R^X})^I \cong (R^I)^{R^X} \xrightarrow{h^{R^X}} (R^O)^{R^X} \cong (R^{R^X})^O$$

If one has a monad map $m : T \to T'$ then, given a generic effect $O \to TI$ for T, one obtains one for T' by composition with m_I. It follows from the above equivalence of generic effects and algebraic operations that, given an algebraic operation op of arity (I, O) over T, one can obtain another, op', of the same arity over T'. One can regard op' as a *lifting* of op in that m acts homomorphically with respect to op and op', meaning that the following diagram commutes for all sets X:

$$\begin{array}{ccc} (TX)^I & \xrightarrow{\mathrm{op}_X} & (TX)^O \\ \Big\downarrow{\scriptstyle (m_X)^I} & & \Big\downarrow{\scriptstyle (m_X)^O} \\ (T'X)^I & \xrightarrow[\mathrm{op}'_X]{} & (T'X)^O \end{array}$$

Conversely, if op, op' are algebraic operations such that the above diagram commutes then op' can be obtained from op by the above process, i.e., the generic

effect associated to op′ is obtained from that associated with op by composition along the monad map. This process of lifting algebraic operations from one monad to another is explained further and exemplified in [18], albeit in a somewhat different mathematical context. In the case where T is presented by a finitary or countably infinitary equational theory, the extensions of the algebraic operations associated to operation symbols in the signature of the equational theory also obey the equations of that theory.

Returning to the structure needed to model continuations with algebraic effects, we assume an additional monad S together with a monad map $m : S \to T$. The new monad is to be thought of as comprising the algebraic operations and their equational relationships, which latter the monad map permits to be lifted to T. It will generally be given by a finite or countable Lawvere theory and one will have an explicit equational presentation.

Any operation over S therefore lifts to one over T and thence, using d^T, to one over $T0^{T0^-}$, where it is pointwise. Furthermore, since C^T is left inverse to d^T, we see, by consideration of the relationship between generic effects and algebraic operations and the above remarks, that it acts homomorphically on the latter two operations.

For example, suppose we are given a binary algebraic operation:

$$\mathrm{op}_X : (SX)^2 \to (SX)^1 \cong SX$$

We introduce an extension of λ_c with expressions of the form $\mathrm{op}(M, N)$, of type σ when both M and N are, and interpret them using the extension of the binary algebraic operation. A more specific example is where S is the monad for nondeterministic choice, given by the theory of semilattices, i.e., with signature a binary operation symbol and with axioms of associativity, commutativity and idempotency; S is then the non-empty finite subsets monad and op is modelled by set-theoretic union. The axioms hold for the extension to T so that, for example, the following general associativity equation holds:

$$(L \,\mathrm{op}\, M) \,\mathrm{op}\, N = L \,\mathrm{op}\, (M \,\mathrm{op}\, N)$$

where L, M, N all have the same type, and we are using infix notation.

The naturality of the algebraic operation op is equivalent to the following commutativity equation for (strict) evaluation contexts:

$$E[M \,\mathrm{op}\, N] = E[M] \,\mathrm{op}\, E[N]$$

(see Appendix A for a brief discussion of evaluation contexts). For example, the fact that $\mathcal{C}^T$ acts homomorphically results in the equation:

$$\mathcal{C}_{\neg\neg\sigma}(M \,\mathrm{op}\, N) = \mathcal{C}_{\neg\neg\sigma}(M) \,\mathrm{op}\, \mathcal{C}_{\neg\neg\sigma}(N)$$

where M, N have type $\neg\neg\sigma$.

When S is presented by a finitary equational theory one can follow the above pattern for all the operation symbols of the signature of that theory. When it is

presented by an infinitary theory, as in, for example, the case of the monad for state, it is natural to seek a finitary syntax involving a bound parameter over a base type denoting the infinitary arity. We do not give more detail here, but some further discussion of this point can be found in [41, 37].

Following the policy stated in the introduction, only the case of **Set** has been considered explicitly, but with the discussion extending to any suitable category V. In fact, for this section, it suffices to assume that V is cartesian closed and has an initial object, i.e., it is a model of intuitionistic logic with conjunction, implication and absurdity. The strong monads are then precisely the enriched ones, and similarly for maps of strong monads.

From the point of view of modelling the λ_c-calculus for a given T the assumptions on V are a little stronger than needed: one normally assumes only finite products, a strong monad and Kleisli exponentials. However we are dealing here with a variety of monads and their combination and we have no theorems to the effect that if Kleisli exponentials exist for each of two monads then they also exist for some combination of them. For that reason the blanket assumption of cartesian closure is convenient.

Considering now the blanket assumption of initiality in V, if we assume instead only initiality in the Kleisli category then both of the equations $\mathcal{A}$-Abs and $\mathcal{A}_O$-Id hold. For the above existence assertion it suffices to assume additionally that 0 is weakly initial in V. However, for the same reasons as before, the blanket assumption is convenient as it does not refer to Kleisli categories.

As regards modelling $\mathcal{C}$ with the weaker assumptions, one can still construct $d^T : T \to \neg_T\neg_T$, Proposition 2 goes through and so do the correspondences between the various assumptions on $\mathcal{C}^T$ and equations or rules in λ_C. However, to treat algebraic operations in an enriched context [41], one employs the normal apparatus of enriched category theory, making assumptions which here include that V is cartesian closed.

3 Sum and exceptions

In this section, we consider the sum of effects. That is one of the natural ways in which to combine algebraic effects, yielding the most commonly used combination of exceptions with any other algebraic effect and the most commonly used combination of interactive input/output with most other algebraic effects. The sum of the interactive input/output monad with the continuations monad does not exist in **Set**, as we shall see; but the sum of the exceptions monad $T_E =_{\text{def}} E + -$ with the continuations monad, and, indeed, any monad, always does. As we shall show in Theorem 2 below, we are in fact able to combine exceptions with any model of continuations. We begin with some general considerations on the existence of sums of monads.

Given monads T and T' on an arbitrary locally small category $\mathbf{A}$, consider the pullback in the category of locally small categories:

$$\begin{array}{ccc} T\text{-}Alg \times_{\mathbf{A}} T'\text{-}Alg & \longrightarrow & T'\text{-}Alg \\ \downarrow & & \downarrow{\scriptstyle U'} \\ T\text{-}Alg & \xrightarrow[U]{} & \mathbf{A} \end{array}$$

Proposition 3. *If $\mathbf{A}$ has all powers, then the sum $T + T'$ of monads exists if and only if the forgetful functor from T-$Alg \times_{\mathbf{A}} T'$-$Alg$ to $\mathbf{A}$ has a left adjoint. And if so, the comparison functor from $(T + T')$-Alg to T-$Alg \times_{\mathbf{A}} T'$-$Alg$ is an isomorphism of categories.*

Proof. Given an object x of $\mathbf{A}$, to give a T-action on x is, by a mild generalisation of Proposition 1, equivalent to giving a monad map from T to $x^{\mathbf{A}(-,x)}$, which has a canonical monad structure. So, if the sum exists, applying the definition of pullback, we are done.

Conversely, the forgetful functor from T-$Alg \times_{\mathbf{A}} T'$-$Alg$ to $\mathbf{A}$ reflects isomorphisms and satisfies Beck's monadicity condition. So, as the forgetful functor has a left adjoint, by Beck's monadicity theorem, it is monadic [4]. A monad is determined uniquely up to equivalence by its category of algebras, so applying the above argument again, i.e., the characterisation of an algebra in terms of monad maps, we are done.

We should mention that the sum of monads agrees with the sum of monads with countable rank as used in [18] to model the sum of algebraic effects. Formally, we can express this as follows:

Proposition 4. *If $\mathbf{A}$ is a locally countably presentable category and T and T' are monads with countable rank on $\mathbf{A}$, then the sum of monads $T + T'$ exists and is of countable rank. Moreover, it is the sum in the category of monads with countable rank.*

Proof. To give a T-action on x is equivalent to giving a monad map from T to a modified form of $x^{\mathbf{A}(-,x)}$ with countable rank [24]. We know, e.g., from [18], that the category of monads with countable rank has sums. So, applying Proposition 3 and a corresponding characterisation of $(T + T')$-Alg for monads of countable rank, we are done.

We now consider some specific sums from the analysis of [18]. For notation, given an endofunctor Σ on a category $\mathbf{A}$, if the forgetful functor from Σ-alg to $\mathbf{A}$ has a left adjoint, we say that the resulting monad is the *free* monad on Σ and write it as Σ^*. Explicitly, Σ^* is $\mu y.(\Sigma y + -)$, the initial algebra of $\Sigma y + -$, with one existing if and only if the other does.

Theorem 1 ([18]). *Let Σ be an endofunctor for which Σ^* exists, and let T be a monad. If $\mu z.T(\Sigma z + x)$ always exists, the sum of monads $\Sigma^* + T$ exists and is given by a canonical monad structure on $\mu z.T(\Sigma z + -)$.*

Theorem 1 includes the example of exceptions, taking $\mathbf{A} = \mathbf{Set}$ and Σ to be the functor given by the constant at E: the sum with T is given by $T(E + -)$; one can also give a direct proof [18, 31]. Note that the sum of monads is not the sum of the underlying functors, i.e., it is not given pointwise.

Let us now consider the combination of exceptions and continuations. As we said above, when there are no additional effects, we take this to be given by the sum of the exceptions monad with the continuations monad, viz $R^{R^{E+-}}$. This monad is isomorphic to $R^{R^E \times R^-}$ and models the combination of exceptions and continuations with the syntax and operational semantics described in [54, 55]. A continuation semantics of this form is called 'double-barrelled' in [56], where such semantics are classified as 'static,' 'dynamic' and 'return': we use the dynamic one here.

When there are additional effects we follow the approach of Section 2 and assume a map of monads $m : S \to T$, with S modelling the additional effects and a natural transformation $\mathcal{C}^T$ left inverse to d^T. Our task is then to find a new model over the monad $T_E + T = T(E + -)$ to incorporate exceptions. We immediately have a map of monads $T_E + m : T_E + S \to T_E + T$, so we then need to find an interpretation:

$$\mathcal{C}^{T_E+T}_X : TE^{TE^X} \to T0^{T0^{E+X}}$$

for the continuations operator for $T_E + T$ (we are identifying $E + 0$ with E). Define $c_X : TE^{TE^X} \to T0^{T0^{E+X}}$ by:

$$c_X(z)[\varepsilon, \kappa] = \varepsilon^\sharp(z(T(\mathrm{inl}_E) \circ \kappa))$$

and then set:

$$\mathcal{C}^{T_E+T}_X = \mathcal{C}^T_{E+X} \circ c_X$$

Theorem 2. *$\mathcal{C}^{T_E+T}$ is a natural transformation left inverse to d^{T_E+T}; it need not be invertible, even if $\mathcal{C}^T$ is.*

Proof. It is clear from the form of the definition that $\mathcal{C}^{T_E+T}$ is natural. The possible lack of invertibility follows by a standard cardinality argument taking the case where T is the continuations monad. That $\mathcal{C}^{T_E+T}$ is left inverse to d^{T_E+T} follows immediately from the equation $c \circ d^{T_E+T} = d^T$ and the assumption that $\mathcal{C}^T_X$ is left inverse to d^T. To prove the equation we calculate:

$$\begin{aligned} c_X(d^{T_E+T}_X(z))[\varepsilon, \kappa] &= \varepsilon^\sharp(d^{T_E+T}_X(z)(T(i_E) \circ \kappa)) \\ &= \varepsilon^\sharp([\eta^T_E, T(i_E) \circ \kappa]^\sharp(z)) \\ &= [\varepsilon^\sharp \circ \eta^T_E, \varepsilon^\sharp \circ T(i_E) \circ \kappa]^\sharp(z)) \\ &= [\varepsilon, \kappa]^\sharp(z) \\ &= d^T_X(z) \end{aligned}$$

As regards syntax for exceptions, following the lines suggested in [41], one can assume: a given base type **exn** of exceptions, with appropriate basic operations and predicates; a family of unary function symbols $\mathbf{raise}_\sigma : 0 \to \sigma$ for raising exceptions (we generally omit the subscript); and terms of the form $\Gamma \vdash \mathbf{handle}(M, (e : \mathbf{exn}.\, N)) : \sigma$, where $\Gamma \vdash M : \sigma$ and $\Gamma, e : \mathbf{exn} \vdash N : \sigma$, for handling them.

As regards semantics, one interprets **exn** by E and **raise** by the algebraic operation $R : T(E + -)^0 \to T(E + -)^E$ inherited from T_E, as described above. Explicitly, and regarding R_X as a function $E \to T(E + X)$, we have that: $R_X = \eta^T_{(E+X)} \circ \mathrm{inl}$.

Next, we seek an operation $H : T(E + -) \times T(E + -)^E \to T(E + -)$ for the interpretation of **handle**, following [13] . To this end, we first define an operation $H' : (E + -) \times T(E + -)^E \to T(E + -)$ where $H'_X(x, \varepsilon) = \eta^T_{E+X}(x)$, for $x \in X$, and where $H'_X(e, \varepsilon) = \varepsilon(e)$, for $e \in E$. Then H_X can be defined via a Kleisli extension by: $H_X(\gamma, \varepsilon) = H'_X(-, \varepsilon)^\dagger(\gamma)$. A theory of deconstructors should surely be able to account for this definition of **handle** by viewing it as an extension of a deconstructor (here, the exception-handling operation) for an effect (here, exceptions alone) to a deconstructor for the combination of that effect with others (here, those modeled by T).

With these definitions, exception handling obeys some natural equations:

$$x : \sigma, g : \mathbf{exn} \to \sigma \vdash \mathbf{handle}(x, (e : \mathbf{exn}.ge)) = x$$

$$e : \mathbf{exn}, g : \mathbf{exn} \to \sigma \vdash \mathbf{handle}(\mathbf{raise}(e), (e' : \mathbf{exn}.ge')) = ge$$

and:

$$\mathbf{handle}(M \,\mathrm{op}\, M', (e : \mathbf{exn}.N)) = \mathbf{handle}(M, (e : \mathbf{exn}.N)) \,\mathrm{op}\, \mathbf{handle}(M', (e : \mathbf{exn}.N))$$

if, for example, op denotes a binary operation inherited from the monad S.

3.1 Existence of Sums

We now consider some questions on the existence of sums of monads and locally continuous monads. Pleasingly, it turns out that Theorem 1 has a converse: if the sum exists, it must be given by the formula.

Theorem 3. *Let Σ be an endofunctor for which Σ^* exists, and let T be a monad. If the sum of monads $\Sigma^* + T$ exists, it is given by a canonical monad structure on $\mu z.T(\Sigma z + -)$.*

Proof. Suppose that the sum of the monads Σ^* and T exists. Then for each x we have the free algebra on x, i.e., the free object a with T-algebra structure $\alpha : Ta \longrightarrow a$ and $s : \Sigma a \longrightarrow a$, and map $x \longrightarrow a$. We can show that a is initial of the form $T(\Sigma a + x) \longrightarrow a$.

For the argument, we can absorb $+x$ into Σ, so it is enough to prove the result for $x = 0$, and thus we can ignore x. The structure $T\Sigma a \longrightarrow a$ is given by

the composite $\alpha(Ts)$. So suppose we have $\beta : T\Sigma b \longrightarrow b$. We need to show that there is a unique $g : a \longrightarrow b$ satisfying the evident coherence property. Note that $T\Sigma b$ has the structure of a free T-algebra and also that of a Σ-algebra, where the Σ-algebra structure is $\eta(\Sigma\beta)$. This gives us a unique map $f : a \longrightarrow T\Sigma b$ satisfying evident properties. One can then check readily that the composite $\beta f : a \longrightarrow b$ is a candidate for g. It remains to prove uniqueness.

Consider the facts that $T\Sigma a$ is a (free) T-algebra and also a Σ-algebra, the map for the latter being $\eta(\Sigma\alpha)(\Sigma Ts)$. It follows that there is a unique map $k : a \longrightarrow T\Sigma a$ satisfying to evident properties as above for f. Now suppose we have any $g : a \longrightarrow b$ that is a map of $T\Sigma$-algebras. Consider $(T\Sigma g)k : a \longrightarrow T\Sigma b$. It readily satisfies the two commuting diagrams for f. So by uniqueness we have $f = (T\Sigma g)k$. It remains to show that $\beta f = g$. Writing f as just given, this reduces readily to showing that $\alpha(Ts)k : a \longrightarrow a$ is the identity. But a is initial, and the composite $\alpha(Ts)k$ is a map of T-algebras as all three components are. Moreover, k is a map of Σ-algebras, and we can check directly that $\alpha(Ts)$ is one too. Thus, by initiality, the composite is the identity as required.

This theorem allows us to see that a sum of monads does not always exist:

Example 1. Let S be the monad on **Set** generated by a single unary operation. If the sum $S + R^{R^-}$ existed, one could solve the isomorphism equation $Z \cong R^{R^{Z+X}}$, but that fails, for evident cardinality reasons, when $R > 1$.

In the same way we see that the sum of the monad $T_{I/O} =_{\text{def}} \mu Y.(Y^I + O \times Y + -)$ for interactive input/output with the continuations monad does not exist either when $R > 1$, unless $I = O = \emptyset$.

Theorem 1 allows us to derive a sufficient condition for the existence of sums.

Proposition 5. *Let T and T' be monads on Set for which the free monad T^* on T qua endofunctor on Set exists, and the sum $T^* + T'$ exists. Then the sum $T + T'$ also exists.*

Proof. First observe that,using freeness of T^* applied to the identity map on T exhibits T as a retract of T^*. So it is a quotient of T^*, and so the corresponding Lawvere theory L_T is a quotient of L_{T^*}. Any relation R that expresses L_T as a quotient of L_{T^*} extends along the coprojection $L_{T^*} \longrightarrow L_{T^*} + L_{T'}$ to a relation on $L_{T^*} + L_{T'}$. Factoring by that using the first part of Proposition 14 yields a sum $L_T + L_{T'}$, and hence a sum $T + T'$.

Sums of locally continuous monads do not in general exist: the above example adapts to ω-**Cpo**, taking$R$ to be, for example, the discrete two-point ω-cpo. However one is more interested in using pointed monads there: a locally continuous monad T is *pointed* if every $T(P)$ has a least element; equivalently if there is a (necessarily unique) monad map $T_L \to T$, where T_L is the lifting monad, which adds a new least element; equivalently if T and $T + T_L$ are isomorphic. We conjecture that the sum of a locally continuous monad with T_L always exists, and that the sum of two locally continuous monads exists if one of them is pointed.

4 Tensor and side-effects

In this section, we extend the notion of tensor of monads with countable rank on **Set**, equivalently countable Lawvere theories, to arbitrary monads on **Set**, equivalently large Lawvere theories. In the setting of monads with countable rank on **Set**, the tensor gave the natural combination of side-effects with most other algebraic effects: the tensor of the side-effects monad $T_S = (S\times -)^S$, see [40], with an arbitrary monad with countable rank T could be characterised by $T(S\times -)^S$. The universal property of the tensor product can be naturally expressed in terms of countable Lawvere theories: given countable Lawvere theories L and L', the tensor, which always exists [18–20] is the countable Lawvere theory $L \otimes L'$ for which there is a canonical equivalence of categories:

$$Mod(L, Mod(L', \mathbf{Set})) \simeq Mod(L \otimes L', \mathbf{Set})$$

where $Mod(L, \mathbf{A})$ denotes the category of countable power preserving functors from L to $\mathbf{A}$.

Although the notion extends, the tensor seems very unlikely to exist for an arbitrary pair of monads. Nevertheless, it follows from Corollary 1 below that the tensor does exist if one of the components is the side-effects monad, and it gives the natural combination of the side-effects monad with the continuations monad R^{R^-}, namely $(R^S)^{(R^S)^-}$, as used in Scheme [1]. The operations associated with an algebraic effect automatically extend to the tensor; with a little effort, one can see that the continuations operation extends too.

Definition 1. *Given large Lawvere theories L and L', the* tensor product *of L and L', if one exists, is the large Lawvere theory $L\otimes L'$ for which $Mod(L\otimes L', \mathbf{Set})$ is coherently equivalent to $Mod(L, Mod(L', \mathbf{Set}))$.*

Theorem 4. *Given large Lawvere theories L and L', the following are equivalent:*

1. *Their tensor $L \otimes L'$ exists.*
2. *The forgetful functor from $Mod(L, Mod(L', \mathbf{Set}))$ to* **Set** *is monadic.*
3. *The forgetful functor from $Mod(L, Mod(L', \mathbf{Set}))$ to* **Set** *has a left adjoint.*

upon which $L \otimes L'$ is the large Lawvere theory corresponding to the monad.

Proof. The equivalence between 1) and 2) follows immediately from the equivalence between large Lawvere theories and monads on *Set* [9]. The implication from 2) to 3) is trivial. It remains to show the converse.

First observe that the forgetful functor

$$U : Mod(L, Mod(L', \mathbf{Set})) \longrightarrow \mathbf{Set}$$

reflects isomorphisms as the maps in $Mod(L, Mod(L', \mathbf{Set}))$ are fully determined by their behaviour on the $(1, 1)$ component by applying powers, and the application of a power is functorial, so preserves invertibility. Now, a model of L in

$Mod(L', \mathbf{Set})$ is exactly a model of L in $\mathbf{Set}$, together with an L'-structure on its underlying object, such that the maps of L are sent to maps of L'-models relative to the induced L'-structure on each power. So, given a U-split coequaliser, say $h : N_{(1,1)} \longrightarrow P$, of a parallel pair $(f, g) : M \longrightarrow N$, one can lift the splittings, not necessarily naturally, to each (L, L')-component. Pointwise, each lifting is a split coequaliser. So one can extend P to have the data for a model of L in $Mod(L', \mathbf{Set})$. The required commutativities of the data hold by construction of the extension and because they hold of N. Similarly, this extension of P is readily seen to be a coequaliser. Thus, Beck's monadicity condition holds, and so the functor U is monadic.

The required left adjoint surely does not exist in general. Moreover, it seems most unlikely that restricting one theory by size will provide it. But it does exist in the particular case of primary interest to us by a mild modification of an argument in [40]. It works as follows.

Proposition 6. *Let L_S be the large Lawvere theory for state. Suppose $\mathbf{A}$ has countable products. Call an object of $Mod(L_S, \mathbf{A})$ an $\mathbf{A}$-model of L_S. Let X be an object of $\mathbf{A}$ for which countable copowers of X exist in $\mathbf{A}$. Then there is a free $\mathbf{A}$-model of L_S on X given by $\prod_S \coprod_S X$.*

Proof. The proof is essentially in [40], subject to the observation that the proof therein is entirely local, only requiring copowers of X without requiring coproducts in general.

Theorem 5. *Let L be an arbitrary large Lawvere theory. Then the forgetful functor from $Mod(L_S, Mod(L, \mathbf{Set}))$ to $\mathbf{Set}$ has a left adjoint which is given by $T_L(S \times -)^S$, yielding the tensor product, qua monad, of L with L_S.*

Proof. Let $\mathbf{A} = Mod(L, \mathbf{Set})$. Then $\mathbf{A}$ has countable products. For each set Y, the category $\mathbf{A}$ has countable copowers of the object $T_L Y$ because countable copowers are preserved by T_L seen as a left adjoint and because $\mathbf{Set}$ has countable copowers. So by Proposition 6 and Theorem 4, we are done.

One can characterise the construction of the tensor product of large Lawvere theories in terms of monads on $\mathbf{Set}$:

Definition 2. *Given monads T and T' on $\mathbf{Set}$, the monad $T \otimes T'$, which we call the* tensor *of T and T' if it exists, is defined by the universal property of having monad maps α and α' from T and T' to $T'' = T \otimes T'$, subject to the commutativity of the following diagram:*

$$\begin{array}{ccc} TX \times T'Y & \xrightarrow{\alpha \otimes \alpha'} & T''X \times T''Y \\ \Big\downarrow{\scriptstyle \alpha \otimes \alpha'} & & \Big\downarrow{\scriptstyle \sigma} \\ T''X \times T''Y & \xrightarrow[\bar{\sigma}]{} & T''(X \times Y) \end{array}$$

where σ and $\bar{\sigma}$ are the two canonical maps induced by the strength of T''.

Proposition 7. *Let L and L' be large Lawvere theories. Then the tensor product $L \otimes L'$ exists if and only if the tensor product of the monads T_L and $T_{L'}$ exists, in which case $T_{L \otimes L'}$ is coherently equivalent to $T_L \otimes T_{L'}$.*

Proof. This follows directly from the correspondence between large Lawvere theories and monads on **Set**: $L = Kl(T)^{op}$, so, taking the component at 1 of the commutativity condition on L and L' yields the coherence condition for $T_L \times T_{L'}$, with the converse given by precomposition.

The coherence condition of the tensor product, expressed in terms of Lawvere theories, is the assertion that the operations of one theory commute with those of the other, and that is the more natural formulation for algebraic effects [18]. There do not seem to be computationally natural operations and equations that generate the continuations monad, so it is unclear to us how best to understand the tensor product of continuations with algebraic effects in general. Thinking of it in terms of monads does not seem to help much.

Proposition 7 allows us to make a formal comparison between the tensor product we use here and that we used for monads with countable rank on **Set**, equivalently countable Lawvere theories, in [18]:

Proposition 8. *Given monads T and T' with countable rank on* **Set***, the tensor product of T and T' seen as monads always exists and agrees with the tensor product of T and T' seen as monads with countable rank.*

Proof. Given any monad T with countable rank on **Set**, the category $Kl(T)^{op}$ is the free category with all small powers on the full subcategory $Kl(T)^{op}_{\aleph_1}$ determined by the countable sets: that it satisfies the freeness property relative to models in **Set** holds because the two theories, one large, the other countable, are equivalent to the same monad; and freeness relative to **Set** is, using representability, equivalent to freeness relative to any category with all small powers, cf. [22]. It follows that the category of small power preserving functors from $Kl(T)^{op}$ to T'-Alg is equivalent to the category of countable power preserving functors from $Kl(T)^{op}_{\aleph_1}$ to T'-Alg. Thus, the two tensor products, which are given by the left adjoints to the forgetful functors to **Set**, agree.

Formulating Theorem 5 in terms of monads, we obtain:

Corollary 1. *If T is an arbitrary monad on* **Set***, the tensor product of T_S with T exists and is given by the monad $T(S \times -)^S$.*

Along the same lines one can prove that the tensor product of the 'read-only' state monad $(-)^S$ with an arbitrary monad T is $(T-)^S$, and, as in [18], for any monoid M the tensor product of $M \times -$ with T is $M \times T(-)$.

In regard to the continuations operation, we have the following result:

Proposition 9. *If d^T, considered as a natural transformation, is invertible (has a left inverse) then $d^{T_S \otimes T}$ is also invertible (has a left inverse).*

Proof. Given a left inverse $\mathcal{C}^T$, consider $\mathcal{C}^{T_S \otimes T} =_{\text{def}} (\mathcal{C}^T_{S\times -})^S$.

As regards tensors and algebraic operations, or generic effects, we note that, analogously with the case for sums, if we assume a monad map $S' \xrightarrow{m} T$ then we obtain another $T_S \otimes S' \xrightarrow{T_S \otimes m} T_S \otimes T$. In particular, for state it is natural to work with generic effects. Suppose, as in [40], that we have a finite set of natural number locations L, so that $S = \mathbb{N}^L$. Then we have generic effects for looking up the value of a location and for updating the contents of one. These are:

$$l : L \to (T_S \otimes T)(\mathbb{N}) = T(S \times \mathbb{N})^S$$

and

$$u : L \times \mathbb{N} \to T(S \times 1)^S$$

and are given by $l(\mathrm{loc})(\sigma) = \eta^T_{S\times\mathbb{N}}(\sigma, \sigma(\mathrm{loc}))$ and $u(\mathrm{loc}, m)(\sigma) = \eta^T_{S\times 1}(\sigma[m/\mathrm{loc}], *)$. Syntactically we employ basic type symbols **loc** and **nat** and unary function symbols $! : \mathbf{loc} \to \mathbf{nat}$ and $:= : \mathbf{loc} \times \mathbf{nat} \to 1$.

4.1 Existence of Tensor Products

We now consider questions on the existence of the tensor product of monads or of locally continuous monads. In general, the tensor product of two arbitrary monads seems not to exist, but we do have a positive result on the existence of tensor products with the continuations monad in **Set**. First we need a generalization of Paré's theorem [4] for the case of the topos **Set**:

Proposition 10. *For any set R with $|R| \geq 2$ the functor $R^- : \mathbf{Set}^{op} \longrightarrow \mathbf{Set}$ is monadic, the monad being given by $R^{(R^-)}$.*

Proof. Since $|R| \geq 2$ it retracts to 2. So, as 2^- is faithful, so reflecting monos and epis, and so isos, R^- is faithful and so reflects isos. Now suppose that we have a reflexive coequalizer diagram in $\mathbf{Set}^{op}$ i.e, a reflexive equalizer diagram in **Set**:

$$A \xrightarrow{\ e\ } B \underset{g}{\overset{f}{\rightrightarrows}} C$$

Then we have a pullback diagram:

$$\begin{array}{ccc} A & \xrightarrow{\ e\ } & B \\ {\scriptstyle e}\downarrow & \lrcorner & \downarrow{\scriptstyle f} \\ B & \xrightarrow[\ g\ ]{} & C \end{array}$$

with all maps monos.

If R is a partial map classifier, i.e., it has a distinguished point, then it admits existential quantification for sets of size less than or equal to 1: i.e., all the maps R^e have a kind of one-sided inverse E_e say (extending, e.g., maps $A \to R$ to maps $B \to R$ using the chosen point), with, e.g., $R^e E_e = id$ and with a Beck-Chevalley condition for the above pullback, i.e., $E_e R^e = R^g E_f$. All that makes the image under R^- of the equalizer diagram a split coequalizer in **Set**. We are now in a position to apply Beck's theorem.

The category $Comod(L, Set)$ of *comodels* of a countable (large) Lawvere theory L in Set is the category of countable (respectively all) co-product-preserving functors from L to Set.

Proposition 11. *Let L be a countable (or large) Lawvere theory. If the forgetful functor $Comod(L, Set) \to Set$ has a right adjoint then the tensor product of L with R^{R^-} exists for any R with $|R| \geq 2$.*

Proof. The categories $Comod(L, Set)^{op}$ and $Mod(L, Set^{op})$ are canonically isomorphic. As the forgetful functor $U : Mod(L, Set^{op}) \to Set^{op}$ has a left adjoint, by Proposition 10 the composite $Mod(L, Set^{op}) \to Set$ of U with R^- also has a left adjoint and so, by Theorem 4, we are done.

With the aid of this proposition, or, rather, its proof, we can find explicit formulas for some tensor products. Consider the free large Lawvere theory with operations of given arities $I_i \to O_i$, for $i \in I$. A comodel of this consists of a set X and functions $O_i \times X \to I_i \times X$, which is the same as a coalgebra of the functor $\prod_{i \in I} (I_i \times -)^{O_i}$. Writing $\nu Y.F(Y)$ for the final co-algebra of a functor F when it exists, for any functor B the cofree B-coalgebra on a set X is $\nu Y.(B(Y) \times X)$, if that exists; if it always exists then $\nu Y.(B(Y) \times -)$ is the free comonad B^* on B. It does always exist in the case at hand, and so, following the proof of the above proposition, the tensor product of T_L and R^{R^-}, for $|R| \geq 2$, is $R^{B^*(R^-)}$ with $B^*(X) = \nu Y.(\prod_{i \in I} (I_i \times Y)^{O_i} \times X)$.

Theorem 6. *The tensor product of any monad with countable rank on* **Set** *with the continuations monad R^{R^-} on* **Set** *exists.*

Proof. The two cases where $|R| < 2$ are simple. Otherwise, as shown in [45], for every countable Lawvere theory L, the forgetful functor from $Comod(L, Set)$ to Set has a right adjoint. So, by the above proposition we are done.

We do not know whether the converse of Proposition 11 holds for large Lawvere theories, but it does suggest that to find a counterexample to the existence of tensor products of large Lawvere theories one should first look for a large Lawvere theory which does not have cofree comodels. We remark that if a large Lawvere theory L has a nullary operation, then its tensor product with continuations exists and is trivial, as it has a unique comodel, the empty one.

Our next proposition relates the existence of tensors to the existence of sums.

Proposition 12. *Let T and T' be monads on Set. Then if their sum exists, so does their tensor product.*

Proof. T and T' correspond to large Lawvere theories L_T and $L_{T'}$. The sum of the former exists if and only if the sum of the latter exists; likewise for the tensor. But the tensor $L_T \otimes L_{T'}$ is given by quotienting the sum by the commutativity equations. And by by Part 1 of Proposition 14, such a quotient always exists.

We do not know any version of Paré's theorem for ω-**Cpo** and, as regards general statements on the existence of tensor products of two locally continuous monads,we only have a conjecture: that they always exist, provided one of them is pointed.

5 The continuations monad transformer

Suppose we have a monad S and wish to obtain a model for continuations over it. Proposition1 tells us that given an S-algebra $SA \to A$, we obtain a map of monads $S \to \mathcal{C}(S) =_{\text{def}} A^{A^-}$; we also know, from the discussion after this proposition, that $d_{\mathcal{C}(S)}$ is an isomorphism. This defines the *continuations monad transformer*, parameterised on an S-algebra [28]. Algebraic operations are defined pointwise on it from the corresponding operations on the algebra A. The usual continuations monad transformer [6, 5] is a little less general, being parameterised on a set R; it is a special case of ours, taking $A = S(R)$. Note that the continuations monad transformer is a parameterised unary construction, whereas sum and tensor are binary constructions, applied to continuations and some other effect.

The continuations monad transformer applies naturally to nondeterminism and I/O, e.g., taking the finite non-empty powerset monad $\mathcal{F}^+$ to $(\mathcal{F}^+ R)^{(\mathcal{F}^+ R)^-}$. Another interesting example is the application of the transformer to exceptions, taking $E + -$ to $(E + R)^{(E+R)^-}$; here the operation for raising exceptions is defined pointwise by $R_X(i) = \lambda e.\lambda\kappa.\text{inl}(e)$; operationally this amounts to ignoring the continuation and raising the exception at top level. This monad was, implicitly, used in [8] to model run-time errors, there taking $E = 1$. When working over ω-**Cpo** the continuations monad transformer also applies naturally to nontermination, now taking the lifting monad $(-)_\perp$ to $R_\perp^{R_\perp^-}$.

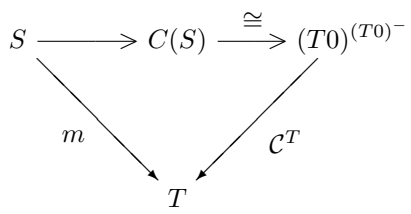
For an example of the more general transformer, consider $(R^S)^{(R^S)^-}$, the tensor product of the side-effects monad T_S with the continuations monad R^{R^-}. This can be alternatively viewed as applying the generalised continuations monad transformer to the side-effects monad T_S, together with the T_S-algebra with structure given canonically on R^S.

The continuations monad transformer seems to be more primitive than the continuations monad: we have $R^{R^-} = C(Id)$, but we do not see any conceptual way to derive $C(-)$ from R^{R^-}. Moreover, it seems there is, in general, no monad map from R^{R^-} to $C(S)$, whereas there is one from T to $C(S)$.

We do not have very substantial results about the continuations transformer, but we can say a little:

Proposition 13. *Given a monad S, and an S-algebra (A, a), the monad A^{A^-} is universal relative to the following structure: a monad T; a monad map $S \xrightarrow{m} T$; a coherent isomorphism $T0 \cong A$; and a monad map $\mathcal{C}^T$ left inverse to d^T.*

Proof. Consider the following diagram in the category of monads:



If one took the diagram, reversed the direction of $\mathcal{C}^T$, and labelled it by d^T, one would have a commutative diagram by assumption. So, since $\mathcal{C}^T$ is a retract of d^T, the above diagram also commutes. Now, given any morphism $f : x \longrightarrow y$ in any category, the colimit of f is, up to coherent isomorphism, given by y. Applying that fact to $S \longrightarrow C(S)$ and the above diagram yields the result.

The various constructions here arise directly in terms of large Lawvere theories, which also allow for easier calculation: the monad S corresponds to the large Lawvere theory L_S; the S-algebra corresponds to a model of L, i.e., a product preserving functor $M : L_S \longrightarrow \mathbf{Set}$; and the continuations transformer is given by the (identity-on-objects, fully-faithful)-factorisation of M. This inherently yields a canonical map of large Lawvere theories from L_S to the theory given by the factorisation. The factorisation also immediately yields Proposition 13, which we expressed in terms of monads.

6 Discussion

Using the constructs we have developed, there is a natural formula for the combination of all of exceptions, side-effects, interactive input/output, (binary) nondeterminism and continuations:

$$T_E + (T_S \otimes \mathcal{C}(T_{I/O} + \mathcal{F}^+))$$

or, explicitly:

$$\left(\overline{R}^{\overline{R}^{S \times (E + -)}}\right)^S$$

where:

$$\overline{R} = \mu Z.\, \mathcal{F}^+(Z^I + O \times Z + R)$$

which we note is linear, having the form $M_E(M_S(\mathcal{C}(M_{I/O}(\mathcal{F}^+))))$ with each M derived from $+$ or $\otimes$ applied to a particular monad. To omit effects, omit the corresponding parts of the formula. We have no independent justification of these proposals, but they are consistent with all the cases we know. A similar formula is available if we pass to ω-**Cpo** and add nontermination.

We have given accounts of the combinations of continuations and other effects in terms of a unary or a binary operation on monads, together with the construction of a continuation operator for the combination. The main thing missing in our account is an understanding of *why* these are the right choices (if indeed they are!). For the combinations of other effects with each other, the choices were justified computationally in terms of the equations involving the sets of operations inherited from each effect [18], but there is nothing of that sort here.

It may be that we are simply taking the wrong approach, as continuations are, as mentioned above, of a logical rather than an algebraic character. For example the computational correlate of implication is the Kleisli exponential and that is justified by its universal property relative to the monad. However we did not succeed in finding such characterisations other than for the continuations monad transformer.

Closely related questions concern finding the right, possibly even complete, axiomatisation of the computational λ-calculus with continuations and the various effects, cf. [17, 54, 55, 27]. Further, investigations of continuations and effects would not be complete without a treatment of operational semantics: see [39] for work on the operational semantics of effects in the context of λ_c, but without continuations, and [54, 55] for operational semantics for combinations of continuations and specific effects. As remarked above, an interesting test of the strength of axioms for continuations is to establish the soundness of an operational semantics. One would also like to understand combinations of continuations with local effects, such as local store or exceptions, as well as with other calling mechanisms.

There are various other computational calculi one might consider when investigating the combination of algebraic effects with continuations. For example, for call-by value there are the call-by-value $\lambda\mu$-calculus [35] and [30], a fine-grained variant of λ_c; for call-by-name there is the call-by-name $\lambda\mu$-calculus [36]; and for the combination there are the $\overline{\lambda}\mu\overline{\mu}$-calculus [7] and the call-by-push-value calculus [28].

In addition to callcc/throw and raise/handle, New Jersey Standard ML provides two additional constructs capture and escape based on the idea of implementing exceptions via a stored handler [53]. Thielecke[1] and Filinski[2] have independently observed that the monad $(R^S)^{(R^S)^-}$, where $S =_{\text{def}} R^E$, can be used to model these constructs. This monad can be viewed either as the tensor product of the read-only state monad with the continuations monad or else as the continuations monad transformer applied to the read-only state monad.

Although d^T is then invertible, we cannot use its inverse to model the $\mathcal{C}$ operator, as that would validate an equation that Laird [27] shows to be broken in the presence of exceptions. Moreover, since the only left inverse of d^T is its inverse, $\mathcal{C}$ cannot even be a left inverse of d^T. This also follows from an observation of Filinski that the capture/escape constructs break $\mathcal{C}$-App.

[1] Unpublished manuscript
[2] Personal communication

Filinski also showed, as mentioned in [29], that capture/escape breaks the equation:

$$M = \mathbf{handle}(M, (e{:}\mathbf{exn}.\,\mathbf{raise}(e)))$$

Indeed it is argued there that, for a monad on **Set** to model the 'basic equations' of exceptions, such as that above, it has to be of the form $T(- + E)$, where E is the set of exceptions. We therefore would argue that capture/escape are incompatible with equations one would naturally wish to hold when reasoning either about continuations or exceptions.

In this paper we have viewed semantics in a standard denotational way, uniformised via the semantics of the computational λ-calculus; the latter is parameterised on a strong monad, following Moggi, and further parameterised to interpret continuations and algebraic effects. As is well known, see, e.g., [12, 13], an alternative would be to give compositional syntactic translations to a target 'metalanguage.' For the computational λ-calculus one could translate to Moggi's computational metalanguage, where a general T is concerned; for the case of the continuations monad one could be more specialised and translate to the ordinary typed λ-calculus, using a continuations transformation. It might be interesting to present the work of this paper in such a style; presumably one would deal with our various combinations of semantics by translations of the chosen metalanguage into itself. The reader will have noted that at various points of the paper we use the λ-calculus informally to give auxiliary definitions; these definitions can generally be be read as being within the computational metalanguage.

Acknowledgments

We would like to thank Andrzej Filinski, Ian Stark and Hayo Thielecke for many useful conversations.

References

1. H. Abelson, R. K. Dybvig, C. T. Haynes, G. J. Rozas, N. I. Adams, D. P. Friedman, E. Kohlbecker, G. L. Steele, D. H. Bartley, R. Halstead, D. Oxley, G. J. Sussman, G. Brooks, C. Hanson, K. M. Pitman & M. Wand, Revised Report on the Algorithmic Language Scheme, Higher-Order Symb. Comput. **11**(1), 7–105, 1998.
2. J. Adámek & J. Rosický, *Locally Presentable and Accessible Categories*, London Mathematical Society Lecture Note Series, **189**, CUP, 1994.
3. Z. M. Ariola, H. Herbelin & Amr Sabry, A Type-Theoretic Foundation of Continuations and Prompts, *Proc. Ninth ACM SIGPLAN Int. Con. on Functional Programming* (eds. C. Okasaki & K. Fisher), 40–53, ACM, 2004.
4. M. Barr & C. Wells, *Toposes, Triples and Theories*, Springer-Verlag, 1985.
5. N. Benton, J. Hughes & E. Moggi, Monads and Effects, *Proc. APPSEM 2000*, LNCS **2395**, 42–122, 2002.
6. P. Cenciarelli & E. Moggi, A Syntactic Approach to Modularity in Denotational Semantics, *Proc. 5th CTCS*, CWI Technical report, 1993.
7. P.-L. Curien & H. Herbelin, The duality of computation, *Proc. 5th. ICFP*, SIGPLAN Notices **35** (9), 233–243, ACM Press, 2000.

8. B. Duba, R. Harper & D. MacQueen, Typing First-Class Continuations in ML, *J. Funct. Program.*, **3**(4), 465–484, 1993.
9. E. J. Dubuc, *Kan Extensions in Enriched Category Theory*, Lecture Notes in Mathematics **145**, Springer-Verlag, 1970.
10. M. Felleisen & D. Friedman, Control Operators, the SECD Machine, and the λ-Calculus, *Formal Description of Programming Concepts III* (ed. M. Wirsing), 193–217, North-Holland, 1986.
11. M. Felleisen, D. P. Friedman, E. E. Kohlbecker & B. F. Duba, Reasoning with Continuations, Proc. 1st. LICS, 131–141, IEEE Press, 1986.
12. A. Filinski, Representing Monads, *Proc. 21st. POPL*, 446–457, 1994.
13. A. Filinski, Representing Layered Monads, *Proc. 26th. POPL*, 175–188, 1999.
14. C. Führmann & H. Thielecke, On the Call-by-Value CPS Transform and its Semantics, *Inf. & Comput.*, **188**(2), 241–283, 2004.
15. T. Griffin, A Formulae-as-Types Notion of Control, *Proc. 17th. POPL*, 47–58, 1990.
16. M. Hasegawa & Y. Kakutani, Axioms for Recursion in Call-by-Value, *Higher-Order Symb. Comput.* **15**(2–3), 235–264, 2002.
17. M. Hofmann, Sound and Complete Axiomatisations of Call-by-Value Control Operators, *Math. Struct. in Comp. Science*, **5**(4), 461–482, 1995.
18. J. M. E. Hyland, G. D. Plotkin & A. J. Power, Combining Effects: Sum and Tensor, Clifford Lectures and the Mathematical Foundations of Programming Semantics, (eds. S. Artemov and M. Mislove), *TCS*, **357**(1–3), 70–99, 2006.
19. J. M. E. Hyland & A. J. Power, Pseudo-Commutative Monads, *Proc. MFPS XVII*, ENTCS **45**, Elsevier, 2001.
20. J. M. E. Hyland & A. J. Power, Pseudo-commutative monads and pseudo-closed 2-categories, *J. Pure Appl. Algebra*, **175**, 141–185, 2002.
21. G. M. Kelly, A Unified Treatment of Transfinite Constructions for Free Algebras, Free Monoids, Colimits, Associated Sheaves, and so on, *Bull. Austral. Math. Soc.*, **22**, 1–83, 1980.
22. G. M. Kelly, *Basic Concepts of Enriched Category Theory*, Cambridge University Press, 1982.
23. G. M. Kelly, Structures Defined by Finite Limits in the Enriched Context I, *Cahiers de Topologie et Géométrie Différentielle*, **23** (1) 3–42, 1982.
24. G. M. Kelly & A. J. Power, Adjunctions whose Counits are Coequalizers, and Presentations of Finitary Enriched Monads, *JPAA*, **89**, 163–179, 1993.
25. R. Kelsey, W. Clinger & J. Rees (eds.), Revised^5 Report on the Algorithmic Language Scheme, *Higher-Order Symb. Comput.*, **11**, 7–105, 1998.
26. A. Kock, Monads on Symmetric Monoidal Closed Categories, *Arch. Math.*, **21**, 1–10, 1970.
27. J. Laird, Exceptions, Continuations and Macro-expressiveness, *Proc. ESOP*, LNCS **2305**, 133-146, Springer-Verlag, 2002.
28. P. B. Levy, *Call-By-Push-Value: A Functional/Imperative Synthesis.* Semantic Structures in Computation, **2**, Springer, 2004.
29. P. B. Levy, Monads and adjunctions for global exceptions, *Proc. 22nd. MFPS*, ENTCS, **158**, 261–287, Elsevier, 2006.
30. P. B. Levy, A. J. Power & H. Thielecke, Modelling environments in call-by-value programming languages, *Inf. & Comp.*, **185**, 182–210, 2003.
31. C. Luth & N. Ghani, Composing Monads Using Coproducts, *Proc. ICFP '02*, 133–144, ACM Press, 2002.
32. E. G. Manes, *Algebraic Theories*, Graduate Texts in Mathematics, **. 26**, Springer-Verlag, 1976.

33. E. Moggi, Computational Lambda-Calculus and Monads, *Proc. 3rd. LICS*, 14–23, IEEE Press, 1989.
34. E. Moggi, Notions of Computation and Monads, *Inf. & Comp.*, **93**(1) 55–92, 1991.
35. C.-H. L. Ong & C. A. Stewart, A Curry-Howard Foundation for Functional Computation with Control, *Proc. 24th. POPL*, 215–227, ACM Press, 1997.
36. M. Parigot, Lambda-Mu-Calculus: An Algorithmic Interpretation of Classical Natural Deduction, *Proc. LPAR 1992* (ed. A. Voronkov), LNCS **624**, 190–201, Springer-Verlag, 1992.
37. G. D. Plotkin, Some Varieties of Equational Logic (Extended Abstract), *Algebra, Meaning, and Computation, Essays Dedicated to Joseph A. Goguen on the Occasion of His 65th Birthday* (eds. K. Futatsugi, J-P. Jouannaud & J. Meseguer), LNCS, **4060**, 150–156, Springer-Verlag, 2006.
38. G. D. Plotkin, *Call-by-Name, Call-by-Value and the λ-Calculus. TCS*, **1**, 125–154, 1975.
39. G. D. Plotkin & A. J. Power, Adequacy for Algebraic Effects, *Proc. FOSSACS 2001* (eds. F. Honsell & M. Miculan), LNCS, **2030**, 1–24, Springer-Verlag, 2001.
40. G. D. Plotkin & A. J. Power, Notions of Computation Determine Monads, *Proc. FOSSACS '02*, LNCS, **2303**, 342–356, Springer-Verlag, 2002.
41. G. D. Plotkin & A. J. Power, Algebraic Operations and Generic Effects, *Applied Categorical Structures*, **11**(1), 69–94, 2003.
42. G. D. Plotkin & A. J. Power, Computational Effects and Operations: an Overview, *Proc. Workshop on Domains VI* (eds. M. Escardó and A. Jung), ENTCS, **73**, 149–163, Elsevier, 2004.
43. A. J. Power, Enriched Lawvere Theories, *Theory and Applications of Categories*, **6**, 83–93, 2000.
44. A. J. Power, Models for the Computational Lambda Calculus, in *Proc. MFCSIT 2000* (eds. T. Hurley, M. Mac an Airchinnigh, M. Schellekens, and A. K. Seda), ENTCS, Vol. 40, Amsterdam: Elsevier, 2001.
45. A. J. Power & O. Shkaravska, From Comodels to Coalgebras: State and Arrays, *Proc. CMCS* (eds. J. Adámek and S. Milius), ENTCS, **106**, 297–314, Elsevier, 2004.
46. J. Rees & W. Clinger, Revised^3 Report on the Algorithmic Language Scheme, *ACM SIGPLAN Notices* **21**(12), 37–79, ACM Press, 1986.
47. J. C. Reynolds, GEDANKEN – A Simple Typeless Language Based on the Principle of Completeness and the Reference Concept, *CACM*, **13** (5), 308–319, 1970.
48. J. C. Reynolds, Definitional Interpreters for Higher-Order Programming Languages, *Proc. 2nd ACM Annual Conference*, 717–740, ACM, 1972 (reprinted in *Higher-Order Symb. Comput.* **11**(4), 363–397, 1998).
49. J. C. Reynolds, On the Relation between Direct and Continuation Semantics, *Proc. 2nd. ICALP* (ed. J. Loeckx), LNCS **14**, 141–156, Springer-Verlag, 1974.
50. J. C. Reynolds, Semantics of the Domain of Flow Diagrams, *JACM* **24**(3), 484–503, 1977.
51. J. C. Reynolds, Definitional Interpreters Revisited, *Higher-Order Symb. Comput.*, **11**(4), 355–361, 1998.
52. J. C. Reynolds, The Discoveries of Continuations, *Lisp and Symbolic Computation*, **6**(3–4), 233–248, 1993.
53. J. C. Reynolds, *Theories of Programming Languages*, Cambridge University Press, 1998.
54. J. G. Riecke & H. Thielecke, Typed Exceptions and Continuations Cannot Macro-Express Each Other, *Proc. 26th ICALP* (eds. J. Wiedermann, P. van Emde Boas & M. Nielsen), LNCS **1644**, 635–644, 1999.

55. H. Thielecke, On Exceptions Versus Continuations in the Presence of State, *Proc. 9th. ESOP* (ed. G. Smolka), LNCS **1782**, 397–411, 2000.
56. H. Thielecke, Comparing Control Constructs by Double-Barrelled CPS, *Higher-Order Symb. Comput.*, **15**, 141–160, 2002.
57. J. M. Wozencraft & A. Evans, *Notes on Programming Linguistics*, M.I.T. Department of Electrical Engineering, 1971.

A The computational λ-calculus

In this appendix, we describe the computational λ-calculus, or λ_c-calculus, extended by type variables, and recall Moggi's notion of λ_c-model [33, 34]. There are several equivalent formulations of the λ_c-calculus. We shall not use the original formulation but a version that is equivalent, modulo the extension by type variables. Our version of the λ_c-calculus has types given by:

$$\sigma ::= B \mid X \mid \sigma \times \sigma \mid 1 \mid \sigma \to \sigma$$

where B ranges over a given set of base types, e.g., int, and X over type variables. We do not assert the existence of a type construction $T\sigma$: this formulation is equivalent to the original one because $T\sigma$ may be defined to be $1 \to \sigma$.

The terms of the λ_c-calculus are given by:

$$M ::= x \mid f(M) \mid MM \mid \lambda x : \sigma.M \mid * \mid (M, M) \mid \pi_{\mathrm{i}}(M)$$

where x is a variable; f ranges over unary function symbols of given closed types $\sigma \to \tau$, such as 0 and succ, of respective types $1 \to$ int and int $\to$ int; and with π_{i} existing for i = 1 or 2. There are evident typing rules for judgements $\Gamma \vdash M : \sigma$, that the term M has type σ in the context Γ (and contexts have the form $\Gamma = x_1 : \sigma_1, \ldots, x_n : \sigma_n$); in particular $*$ is of type 1. This differs from the original formulation of the calculus in that we do not explicitly have a **let** constructor or constructions $[M]$ or $\mu(M)$. The two formulations are equivalent in that **let** $x = M : \sigma$ **in** N may be considered as syntactic sugar for $(\lambda x : \sigma.N)M$ (we may then elide the σ), $[M]$ may be considered as syntactic sugar for $\lambda x : 1.M$ where x is of type 1 and does not occur freely in M, and $\mu(M)$ may be considered as syntactic sugar for $M*$.

The λ_c-calculus has two predicates: existence, denoted by $\downarrow$, and equality, denoted by $=$ (Moggi writes instead $\downarrow_\sigma$ and $\equiv$). The corresponding judgements are $\Gamma \vdash M \downarrow$ and $\Gamma \vdash M = N$. The $\downarrow$ rules may be expressed as saying that $x \downarrow$, $\lambda x : \sigma.M \downarrow$ for all M, $* \downarrow$, if $M \downarrow$ and $N \downarrow$ then (M, N) and if $M \downarrow$ then $\pi_i(M) \downarrow$, that existence is closed under equivalence and substitution by (existing) terms, with given additional rules for the unary function symbols, e.g., succ $M \downarrow$ when $M \downarrow$. We say that a *value* (relative to Γ) is a term V such that $\Gamma \vdash V \downarrow$.

There are three classes of rules for equality. The first class are rules to the effect that it is a congruence, closed under substitution by existing terms (i.e., values); the second class are rules for the unit, product and functional types; and the third class of rules are given rules for the unary function symbols. It follows

from the rules for both predicates that types together with equivalence classes of terms form a category, with a subcategory determined by values.

It is straightforward, using the original formulation of the λ_c-calculus in [33], to spell out the (second class of) inference rules required to make this formulation agree with the original one: one just bears in mind that the models are the same, and uses syntactic sugar as detailed above.

The first class of models for the λ_c-calculus was given by Moggi in [33, 34] with another formulation using *Freyd*-categories being given in [44]. For Moggi, a λ_c-model consists of a category C with finite products, together with a strong monad T on C, such that T has Kleisli exponentials, i.e., for each object x of C, the functor $J(- \times x) : C \longrightarrow C_T$ has a right adjoint, where C_T is the Kleisli category for C and $J : C \longrightarrow C_T$ is the canonical functor.

We assume given an object $[\![b]\!]$ to interpret each basic type b, and then every type σ receives an interpretation as an object $[\![\sigma]\!]$, assuming an interpretation of any type variables occurring in it. We also assume given a map $[\![\sigma]\!] \to T([\![\tau]\!])$ to interpret every unary function symbol f of given type $\sigma \to \tau$. A term of type σ in context Γ is modelled by a map in the Kleisli category for T, i.e., by a map in C from $[\![\Gamma]\!]$ to $T[\![\sigma]\!]$, where $[\![-]\!]$ denotes the semantic construct (and for $\Gamma = x_1 : \sigma_1, \ldots, x_n : \sigma_n$, $[\![\Gamma]\!] = [\![\sigma_1]\!], \times \ldots \times [\![\sigma_n]\!]$), and again assuming an interpretation of any type variables occurring in Γ or σ.

An existence assertion holds if the corresponding map is *total*, meaning that it factors through the unit of T; an equivalence assertion holds if the two corresponding maps are equal. With this all the rules mentioned above automatically hold, except those for the unary function symbols which must be verified separately; indeed Moggi showed that his class of models is complete for the *pure* λ_c-calculus, meaning the one with no rules for the unary function symbols.

The extension of these semantical ideas to the more general situation where C is V-enriched is straightforward, one simply interprets using the underlying ordinary categories and functors. An important example is provided by the case $V = \omega$-**Cpo** where one can interpret the call-by-value recursion operator. Syntactically one assumes a family $Z_{\sigma,\tau} : ((\sigma \to \tau) \times \sigma \to \tau) \times \sigma \to \tau$ of unary function symbols. [3] The fixed-point equation takes the form:

$$F : (\sigma \to \tau) \times \sigma \to \tau, x : \sigma \vdash Z(F, x) = F(\lambda x : \sigma.Z(F, x), x)$$

and one can also give versions of the uniformity and stability axioms of [16] (the authors there employ a slightly different version of the call-by-value recursion operator). The operator can be interpreted in the standard least fixed-point way provided that the locally continuous monad T is *pointed*, meaning here that every $T(P)$ contains a least element; the three axioms then hold.

Evaluation contexts for the λ_c-calculus are defined by the following inductive clauses: $[-]_\sigma$ is an evaluation context, and $f(E)$, EM, VE, (E, M), (V, E) and $\pi_i(E)$ are evaluation contexts for any evaluation context E, term M and value

[3] The name comes from $Z = \lambda f.((\lambda x f.(\lambda z.xxz))(\lambda x f.(\lambda z.xxz)))$ the untyped fixed-point operator for the call-by-value λ-calculus referred to in [38], and due to John Reynolds [47] and Wozencraft and Evans[57].

V. One can type evaluation contexts by adding the rule that $[-]_\sigma : \sigma$. The computational thought behind evaluation contexts is that in a program of the form $E[M]$, where $\Gamma \vdash M : \sigma$, the first computational step arises within M.

Returning to semantics, to each such context $\Gamma \vdash E : \tau$ where the 'hole' in E is $[-]_\sigma$ one can assign a morphism $[\![E]\!] : [\![\Gamma]\!] \times [\![\sigma]\!] \to T([\![\tau]\!])$. One then has that $[\![E[M]]\!] = E^\sharp \circ t_{[\![\Gamma]\!],[\![\sigma]\!]} \circ (\mathrm{id}_{[\![\Gamma]\!]}, [\![M]\!])$, where t is the strength of T and, in this sense, evaluation contexts can be said to be *strict*. In extensions of the λ_c-calculus, e.g., for exception handling, one may employ a larger set of evaluation contexts which are no longer strict, and one may need to pick out a smaller set of evaluation contexts which are.

B Large Lawvere Theories

The notion of monad on **Set** may be proved to be mathematically equivalent to that of *large Lawvere theory*. That result enriches, yielding an equivalence, on an appropriately restricted category V, between strong monads, equivalently enriched monads, and V-enriched large Lawvere theories. The definition of large Lawvere theory or more generally V-enriched large Lawvere theory, is usually more amenable than that of strong monad to the constructions we develop in combining computational effects. So in this appendix we explain the equivalence between the notions of strong monad and large Lawvere V-theory so that we can freely swap between them as convenient.

The abstract work here bears comparison with Section 2 of [18], which exhibits an equivalence between countable Lawvere theories and monads with countable rank. The size distinction is fundamental: the equivalence we describe in this section is considerably less sophisticated, but the issues involved with combining arbitrary monads, equivalently large Lawvere theories, are far more complicated and far less elegant than those that arise under the assumption of countable rank.

For convenience of exposition, we start by considering the base category **Set**.

Definition 3. *A* large Lawvere theory *is given by a locally small category L with small products, together with a strict product preserving identity-on-objects functor $I : \mathbf{Set}^{op} \longrightarrow L$.*

Note that, in this context, strictly preserving all small products is equivalent to strictly preserving all powers. One typically denotes a large Lawvere theory by L, leaving the data for the functor I implicit. Large Lawvere theories form the objects of a category, for which a map from L to L' is a functor (that necessarily strictly preserves products) from L to L' commuting with I and I'. Thus we have a category of large Lawvere theories.

Definition 4. *A* model *of a large Lawvere theory L in any locally small category* **A** *with products is a product preserving functor $M : L \longrightarrow \mathbf{A}$.*

For any large Lawvere theory L and any locally small category **A** with products, we thus have the category $Mod(L, \mathbf{A})$ of models of L in **A**; the maps are

all natural transformations, with the naturality condition implying that they respect the product structure, which in turn implies that the category $Mod(L, \mathbf{A})$ is locally small. There is a canonical forgetful functor $U : Mod(L, \mathbf{A}) \longrightarrow \mathbf{A}$. If it has a left adjoint, this forgetful functor exhibits $Mod(L, \mathbf{A})$ as equivalent to the category T_L-Alg for the induced monad T_L on $\mathbf{A}$.

Restricting to the case that $\mathbf{A} = \mathbf{Set}$, there is a converse: given any monad T on $\mathbf{Set}$, the category $Kl(T)^{op}$ determined by taking the opposite of the Kleisli category of T, is a large Lawvere theory L_T, and the categories $Mod(L_T, \mathbf{Set})$ and T-Alg are canonically equivalent. An enriched version of the following result appears in Dubuc's thesis [9]:

Theorem 7. *The construction sending a large Lawvere theory L to T_L together with that sending a monad T to L_T induce an equivalence of categories between the category of large Lawvere theories and the category of monads on* **Set**. *Moreover, the comparison functor exhibits an equivalence of the categories $Mod(L, \mathbf{Set})$ and T_L-Alg.*

We now consider the generalisation from **Set** to appropriate V, which we take to be complete and cocomplete cartesian closed categories. The notion of locally small category generalises to the notion of V-category; the notion of category that need not be locally small does not immediately generalise to any definition in terms of V. The careful reader may note that, a priori, the category $Mod(L, \mathbf{A})$ we considered above might not be locally small: we shall return to that later.

There are two points that require care in the enrichment the equivalence between monads and theories: one is a size issue, and the other involves the notion of *cotensor*, which yields the appropriate enrichment of the notion of product of copies of a single generator. The notion of cotensor is the most natural enrichment of the notion of a power-object. Given an object X of a V-category $\mathbf{A}$ and given an object A of V, the cotensor X^A satisfies the defining condition that there is an isomorphism in V:

$$\mathbf{A}(Y, X^A) \cong \mathbf{A}(Y, X)^A$$

V-natural in Y.

There is an evident dual notion of *tensor* $A \times X$ satisfying the defining condition:

$$\mathbf{A}(A \times X, Y) \cong \mathbf{A}(X, Y)^A$$

V-natural in Y. In the case $V = \mathbf{Set}$, the tensor is given by $\coprod_A X$, the coproduct of A copies of X. In the case $\mathbf{A} = V$, the tensor $A \times X$ is the product of A and X.

Definition 5. *A* large Lawvere V-theory *is given by a V-category L with cotensors, together with a strict cotensor preserving identity-on-objects V-functor $I : V^{op} \longrightarrow L$. A* model *of L in a V-category $\mathbf{A}$ with cotensors is a cotensor preserving V-functor $M : L \longrightarrow \mathbf{A}$.*

For the size reasons mentioned above, it is not entirely trivial but nonetheless true that if we additionally assume that V has arbitrary intersections, then for any large Lawvere V-theory L and any V-category $\mathbf{A}$ with cotensors, we have a V-category $Mod(L, \mathbf{A})$ of models of L in $\mathbf{A}$. The homobjects are given by all V-natural transformations. That they form an object of V (rather than being too large) is because the V-naturality condition implies that they respect cotensors, and so are determined by the component at 1; the presence of arbitrary intersections allows us to take the intersection of the large family of equalisers that express the preservation of all operations of L.

That said, we can routinely generalise the unenriched case: there is a canonical forgetful V-functor $U : Mod(L, \mathbf{A}) \longrightarrow \mathbf{A}$, and if it has a left V-adjoint, it exhibits $Mod(L, \mathbf{A})$ as V-equivalent to the V-category T_L-Alg for the induced V-monad T_L on $\mathbf{A}$. And for a converse, without needing to assume the presence of intersections in V, if $\mathbf{A} = V$, given a V-monad T on V, the V-category $Kl(T)^{op}$ is a large Lawvere V-theory, with T-Alg canonically equivalent to $Mod(L_T, V)$.

To give a V-enriched V-monad is equivalent to giving a strong monad on V (see [26]). So, in order to make the comparison with Moggi's definition most direct, we express Dubuc's result [9] in terms of strong monads:

Theorem 8 ([9]). *The constructions of T_L from L and of L_T from T induce an equivalence of categories between the category of large Lawvere V-theories and that of strong monads on V. Moreover, the comparison V-functor exhibits an equivalence of the V-categories $Mod(L, V)$ and T_L-Alg.*

We conclude with a proposition specific to **Set**.

Proposition 14. *For any large Lawvere theory L, and for any family of relations $R_{X,Y} \subseteq L(X, Y) \times L(X, Y)$, there is a large Lawvere theory L/R together with a map of large Lawvere theories $L \longrightarrow L/R$ that universally forces any maps related by R to be equal.*

Proof. The proof of the first part is essentially that of Theorem 1.5.45 of [32].

As a consequence we note that the category of monads on **Set** has coequalisers. Unfortunately, we could not prove an analogue of this proposition for ω-**Cpo**.