Canonical Models for Computational Effects

John Power*

School of Informatics, University of Edinburgh, King's Buildings,
Edinburgh EH9 3JZ, Scotland
ajp@inf.ed.ac.uk

Abstract. Given a signature of basic operations for a computational effect such as side-effects, interactive input/output, or exceptions, we give a unified construction that determines equations that should hold between derived operations of the same arity. We then show how to construct a canonical model for the signature, together with the first-order fragment of the computational λ-calculus, subject to the equations, done at the level of generality of an arbitrary computational effect. We prove a universality theorem that characterises the canonical model, and we recall, from a previous paper, how to extend such models to the full computational λ-calculus. Our leading example is that of side-effects, with occasional reference to interactive input/output, exceptions, and nondeterminism.

1 Introduction

Last year, at the *Typed Lambda Calculus and Applications* conference, I proved that every category theoretic model of what I defined to be the first-order fragment of Moggi's computational λ-calculus can be canonically embedded into a model of the whole calculus, the embedding satisfying an elegant and natural universal property [20]. After my talk, Carolyn Talcott asked what I regarded, and continue to regard, as an excellent question. As I understand her question, she had expected me to discuss computational effects, whereas, as I had pointed out in the talk, the computational λ-calculus does not actually have computational effects in it, but rather is a particularly well-designed calculus to which one can add computational effects. So her question was whether I could say something directly about computational effects in the context of the talk. This paper addresses that question.

Over recent years, there has been a concerted attempt, led by Gordon Plotkin and myself, to develop a unified, elegant theory of computational effects, with both operational and denotational semantics, a logic, and theorems relating them, designed to analyse and reason about call-by-value functional programming languages that extend the simply typed λ-calculus, along the lines of ML [4,5,6,7,13,14,15,16,17]. Our starting point has typically been Eugenio Moggi's computational λ-calculus or λ_c-calculus, which was introduced in [10,

* This work has been done with the support of EPSRC grants GR/N64571, GR/R67842/01, and GR/N23141, and EC grant IST-1999-29082.

I. Walukiewicz (Ed.): FOSSACS 2004, LNCS 2987, pp. 438–452, 2004.

11], with four distinct sound and complete classes of category theoretic models explained in [18], and with further abstract semantic development in [8,19,20, 21]. The ideas surrounding the calculus have been applied extensively by the functional programming community, albeit typically using the computational metalanguage rather than the λ_c-calculus: a recent overview appears as [1].

The heart of Plotkin and my theory of computational effects, and the sense in which it goes beyond the λ_c-calculus, has been the study of operations that may be added as a signature to the λ_c-calculus: for global state, one wants *lookup* and *update*; for interactive input/output, one wants *read* and *write*; for nondeterminism, one wants binary $\vee$; etcetera [14,16]. We have studied signatures of such operations extensively (see [17] for a recent summary), but we have not yet given a unified, elegant account, supported by a body of mathematical theory and by natural computational examples, of what equations should be imposed on the operations, and how to derive canonical models for the λ_c-calculus *together with* the signature and equations determined by the effect at hand. Those issues lie at the heart of Carolyn Talcott's question as I understand it, and this paper is designed to address them.

Consider the example of global state. We have a signature given by basic operations *lookup* and *update*. These basic operations have arities given by $lookup : Val \longrightarrow Loc$ and $update : 1 \longrightarrow Loc \times Val$, where Loc is a finite set of locations and Val is a countable set of values. To decide what equations the operations derived from *lookup* and *update* should satisfy, one natural way to proceed is as follows. First define the set $State$ of states to be the set Val^{Loc} of functions from locations to values. Now model the basic operation *lookup* by the function

$$(State \to State)^{Val} \longrightarrow (State \to State)^{Loc}$$

determined by composition with the function from $Loc \times State$ to $Val \times State$ that, given (loc, σ), "looks up" loc in $\sigma : Loc \to Val$ to determine its value, and by the projection to $State$. And model the basic operation *update* by the function

$$(State \to State) \longrightarrow (State \to State)^{Loc \times Val}$$

determined by composition with the function from $Loc \times Val \times State$ to $State$ that, given (loc, v, σ), "updates" $\sigma : Loc \to Val$ by replacing the value at loc by v. This modelling of *lookup* and *update* automatically generates a model of any operation derived from *lookup* and *update*. Now put two derived operations of the same arity equal if and only if they yield the same function on the appropriate power of $State \to State$. This construction yields exactly the equations that are commonly agreed by the programming community as natural for global state (see [14,15] for more analysis of such equations). For instance, one such equation is given by $lookup_{loc}(update_{loc,v}(x))_v = x$.

This construction can be generalised: starting with any signature of basic operations and any choice of model of the basic operations, one can deduce equations between derived operations. And that construction duly yields the usual equations one expects in such cases as interactive input/output and exceptions, as well as side-effects; nondeterminism involves the additional question

of partiality, as we discuss later. Section 3 of this paper is devoted to giving the generality of the construction and exhibiting more detail especially in the case of global state. In all of the examples we know of computational effects, there are computationally natural choices of such operations and of such models or sets of models.

Now assume we are given a signature of basic operations together with equations that they are to satisfy. We seek to construct canonical models of the λ_c-calculus together with the signature, subject to the equations. Our use of the word "model" here is different to our use of it in the previous paragraph: here, we mean a model of the λ_c-calculus *together with* operations and equations, whereas before, we meant a model of the operations but without modelling the calculus.

The central point of the paper is that a canonical model for the λ_c-calculus together with operations and equations falls out immediately from the mathematics we use to model the operations and equations alone: every signature of operations and equations we consider naturally forms a *countable Lawvere theory* L: this is a category with countable products together with structure that forces it to be generated, in a precise sense, by one object. We observe that the structure of L yields the structure of a $Freyd$-category on L^{op}, and, as we recall in Section 4, a $Freyd$-category is exactly what one needs to model the first-order fragment of the λ_c-calculus. So L^{op} gives us a model of the first-order fragment of the λ_c-calculus, and it inherently yields a model of the signature subject to its equations. In fact, L^{op} does a little more than that: it also canonically models the obvious extension of the first-order fragment of the λ_c-calculus to include sum types and a type of natural numbers; and it is the free model generated by the operations and equations subject to a condition asserting the existence of countable sums. Section 5 is devoted to the details.

Finally, combining Sections 3 with 5, we can now say exactly how the various results relate to last year's paper [20]. In studying computational effects, one invariably has a signature of operations one wishes to model. One also invariably has a natural computational model of the signature. That determines natural equations to place on the derived operations as in Section 3. The operations and equations form a countable Lawvere theory L, and L^{op} is a canonical model of the first-order fragment of the λ_c-calculus together with the signature of operations and its equations (and sum types and a type of natural numbers). That model satisfies a natural universal property detailed in Section 5. The main result of [20] shows how to extend that model to a model of the whole λ_c-calculus (and it can be adapted to preserve the semantics of the sum types and the type of natural numbers), giving a universal property that exhibits its definitiveness. A mild systematic variant of the construction of [20] allows one to recover the standard models on Set listed by Moggi [10,11].

There is one very substantial omission from the above analysis, and that is recursion. The λ_c-calculus does not contain recursion, and nothing we have written here contains it either. One needs to extend both the calculus and the models in order to incorporate it. One way to do that involves replacing the =

predicate of the calculus (see Section 2) by $\leq$, upon which one must provide models (see Section 4) of $\leq$. That seems most elegantly done by use of *enriched* categories and *enriched* Lawvere theories [5,6]. One must extend the analysis even just to model partiality, as we mentioned above in relation to nondeterminism. In fact, there are two distinct ways to extend our analysis, and they are related: one might be seen as equational while the other might be seen as operational; the relationship between them involves the notion of a *discrete* enriched Lawvere theory. We defer the details for later work, but mention that the work here extends elegantly.

This paper is organised as follows. In Section 2, we describe one version of the λ_c-calculus and give a definition of a signature for it. In Section 3, we describe how, given a signature of basic operations, every model induces equations between derived operations. In Section 4, we recall the definition of a closed *Freyd*-category, providing the sound and complete class of models of the λ_c-calculus we need. And in Section 5, we show how every signature of operations and equations generates a canonical model for the first-order fragment of the λ_c-calculus together with those operations and equations.

2 The Computational λ-Calculus and Signatures for It

In this section, we give a succinct formulation of Moggi's computational λ-calculus, or λ_c-calculus, followed by a definition of the notion of signature for the λ_c-calculus: the latter is more subtle than one might at first imagine, as one must make a delicate distinction between constructs that are to be modelled by effect-free terms and constructs that may be modelled by arbitrary terms. The work of this section is largely adapted from that of [17], which lists natural questions to be addressed regarding the λ_c-calculus, including those of this paper. We need to include the section here as, in following sections, we study models for both the λ_c-calculus together with a signature (in Section 5) and for signatures alone (in Section 3).

The syntax for the λ_c-calculus may be taken to be identical to that for the simply typed λ-calculus [18]. So it has type constructors

$$\sigma ::= 1 \mid \sigma_1 \times \sigma_2 \mid \sigma \to \tau$$

and term constructors

$$e ::= * \mid \langle e, e' \rangle \mid \pi_i(e) \mid \lambda x.e \mid e'e \mid x$$

where x ranges over variables, $*$ is of type 1, with π_i existing for $i = 1$ or 2, all subject to the evident typing. The λ_c-calculus has two predicates: an equality predicate exactly as in the simply typed λ-calculus and a unary predicate $(-)\downarrow$ for "definedness" or "effect-freeness". The rules for the latter say $* \downarrow$, $x \downarrow$, $\lambda x.e \downarrow$ for all e, if $e \downarrow$ then $\pi_i(e) \downarrow$, and similarly for $\langle e, e' \rangle$, and that definedness is closed under equality. There are two classes of rules for $=$. The first class say that $=$ is a congruence. And the second class are rules for the basic constructions and for

unit, product and functional types. The rules are closed under substitution of effect-free terms for variables. It follows from the rules for both predicates that types together with equivalence classes of terms in context form a category, with a subcategory determined by effect-free terms. It is straightforward but lengthy, to adapt the formulation of the λ_c-calculus in [12] to list the $=$ rules.

The only aspect of the λ_c-calculus that goes beyond the standard simply typed λ-calculus is the predicate $(-)\downarrow$ together with associated sophistication in the rules for $=$. The λ_c-calculus has typically been treated either as an equational logic or as an higher order intuitionistic logic, both of which were considered in [10,11]. Here, we do the former. We shall later extend the calculus by adding sum types and a type of natural numbers. That is a mild extension, such types existing and being well understood in many call-by-value programming languages. In later work, we shall further extend the calculus by considering the predicate $\leq$ in order to incorporate recursion: as is the case for simply typed λ-calculus, the λ_c-calculus does not contain a mechanism for studying recursion.

We now recall the notion of a signature for the λ_c-calculus [17]. The idea is that each computational effect is generated by a signature of basic operations, subject to equations. A full definition of signature necessarily includes further, less complex, data that we shall describe and for which we shall give an example.

Definition 1. *A* signature *for the λ_c-calculus consists of (base) types, together with typed function symbols, predicate symbols for the programming language, and operation symbols.*

The constant, function, and predicate symbols are to be modelled using effect-free terms in context, while the operation symbols form arbitrary terms that will not in general be effect-free. In this paper, only the operations are of primary concern, so we shall restrict our attention almost exclusively to them after describing our leading example in full.

Example 1. Suppose one wishes to consider an idealised language for the combination of global state with nondeterminism. One might add to the λ_c-calculus a type Nat for natural numbers, function symbols 0, *succ*, and *pred*, for natural numbers, and a predicate symbol $= 0$. Then one adds operation symbols for nondeterminism and global state such as operation symbols $\vee$ for binary nondeterminism and *lookup* and *update* for state. The equational axioms to be added to the λ_c-calculus are those generated by the combination of nondeterminism and global state, as for instance in [14,16]. One can give a systematic account of the combination of nondeterminism and global state providing one already has a system of equations appropriate for each of nondeterminism and global state individually [5,6]. So, in Section 3, we develop a theory for generating equations for individual computational effects, which may then be combined using the results of [5,6].

We mention in passing that we have semantic evidence that suggests how to extend the above-mentioned signature and equations from global state to local state by adding another operation *block* subject to natural axioms [14]: the most

elegant way to achieve that requires further investigation, so although consistent with the work in this paper, it does not seem to provide an example of the work we develop here.

We have many examples of such signatures and associated equations in [5,6, 13,14,15,16]. But we have not had a systematic way to generate the equations for each effect. In principle, we should be able to generate equations from a formalisation of the notion of observation, then asserting that two derived operations should be put equal if they are observationally equivalent.

3 From Operations and a Model to Equations

It is generally clear, given a computational effect, how to choose suitable operations that generate it. For instance, in modelling nondeterminism, one typically starts with binary $\vee$; for global state, one typically chooses *lookup* and *update*; and for interactive input/output, one considers *read* and *write*. It is often less clear what equations to impose as axioms. So we seek a mathematical framework to guide our choice of equations. The λ_c-calculus is an equational theory rather than an inequational one, so we restrict our attention to equational issues here, deferring partiality and recursion for later work using an enriched version of this analysis.

Observe that equations typically hold between derived operations rather than between primitive ones. For instance, to express associativity of $\vee$, one must be able to speak of $(x \vee y) \vee z$, which is given by a derived ternary operation. So, we seek a unified way in which to speak of the derived operations generated by a signature. There are several equivalent ways to do that, and we shall use the notion of *countable Lawvere theory* [5].

Let $\aleph_1$ denote a skeleton of the category of countable sets and all functions between them. So $\aleph_1$ 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. So, in referring to $\aleph_1$, we implicitly make a choice of the structure of its countable coproducts.

Definition 2. *A* countable Lawvere theory *is a small category L with countable products and a strict countable-product preserving identity-on-objects functor $I : \aleph_1^{op} \longrightarrow L$.*

Implicit in the definition is the statement that $\aleph_1^{op}$ and L have the same set of objects. We typically write L for a countable Lawvere theory, with the data given by $I : \aleph_1^{op} \longrightarrow L$ left implicit. Every signature of operations, with arities either natural numbers or $\aleph_0$, freely generates a countable Lawvere theory, a trivial one in the sense that it satisfies no non-trivial equations. The arrows with domain n and codomain 1 in that countable Lawvere theory are exactly the derived n-ary operations generated by the signature; an arrow with domain n and codomain m consists exactly of m derived n-ary operations generated by the signature. And that generalises routinely to $\aleph_0$. Composition of the countable Lawvere theory is a formulation of the notion of substitution.

Example 2. A signature for global state is given by $lookup : Val \longrightarrow Loc$ and $update : 1 \longrightarrow Loc \times Val$, where Loc is a finite set of locations and Val is a countable set of values [14,6]. These freely generate a countable Lawvere theory by identifying the finite set Loc with its cardinality n and by identifying Val with $\aleph_0$, then freely allowing substitutions applied to instances of *lookup* and *update*. So an arrow in the countable Lawvere theory is a word of finite length but possibly infinite breadth (see Example 5 for more detail) of copies of *lookup* and *update*. We shall use this countable Lawvere theory, together with a model of it in Set, to induce natural equations between pairs of such words, yielding a countable Lawvere theory for side-effects: that will include the operations but identify any pairs of words that are equal in the canonical model.

Example 3. A signature for interactive input/output is given by $read : I \longrightarrow 1$ and $write : 1 \longrightarrow O$, typically for countable sets I of O of outputs [14,6]. Again, identifying I and O with $\aleph_0$, these operations freely generate a countable Lawvere theory. In this case, the canonical model does not induce any non-trivial equations between words. So the countable Lawvere theory for interactive input/output is precisely the free theory generated by *read* and *write*.

Exceptions work much as interactive input/output: the countable Lawvere theory is freely generated by an operation $raise : 0 \longrightarrow E$ for a finite or countable set of exceptions E, and the canonical model does not subject it to any non-trivial equations [14,6]. Nondeterminism involves issues of partiality that we do not treat here, but the heart of it is given by the free countable Lawvere theory on a binary operation $\vee$, and the canonical model induces the equations of associativity, commutativity, and idempotence [6]. Of course, one can also consider combinations of such effects.

As mentioned in the examples, the equations that are to hold between derived operations are typically generated by a canonical (observational) model of the signature.

Example 4. Continuing our investigation of global state from Example 2, let $State$ be the set Val^{Loc}. The standard semantics of a command is generally understood to be a state-changing function, i.e., a function of the form

$$State \longrightarrow State$$

So the operations *lookup* and *update* should act on powers of this set. They are generally deemed to act as follows: the operation *lookup* is modelled by the function

$$(State \to State)^{Val} \longrightarrow (State \to State)^{Loc}$$

determined by composition with the function from $Loc \times State$ to $Val \times State$ that, given (loc, σ), "looks up" loc in $\sigma : Loc \to Val$ to determine its value, and is given by the projection to $State$; and the operation *update* is modelled by the function

$$(State \to State) \longrightarrow (State \to State)^{Loc \times Val}$$

determined by composition with the function from $Loc \times Val \times State$ to $State$ that, given (loc, v, σ), "updates" $\sigma : Loc \to Val$ by replacing the value at loc by v. We wish to set a pair of operations generated by *lookup* and *update* equal precisely when they yield the same functions on powers of $State \to State$.

Similar stories can be given for each of our leading examples. For instance, where we considered $State \to State$ to study side-effects, one would usually consider the set $\mu Y.(O \times Y + Y^I + 1)$ in order to study interactive input/output: this set is the free algebra on 1 generated by *read* and *write*. The freeness of the algebra determines canonical behaviour of *read* and *write*. It is routine to verify that such modelling yields no equations between derived operations. Similarly for exceptions. One does obtain non-trivial equations for nondeterminism, and they are the usual ones for idempotence, symmetry and transitivity of $\vee$.

We can describe the constructions we have given for global state and outlined for other examples in a unified way in terms of countable Lawvere theories. The central fact is that one starts with a model of the signature of operations, and one imposes the equations that are equal in that model. We proceed as follows.

Definition 3. *A* model *of a countable Lawvere theory L is a countable-product preserving functor $M : L \longrightarrow Set$.*

It is routine to verify that if L is freely generated by a signature, to give a model of L as we have defined it is equivalent to giving a set X, together with a function $X^\alpha \longrightarrow X$ for each operation of arity α in the signature.

Given a model $M : L \longrightarrow Set$, one can factor it uniquely up to isomorphism as an identity-on-objects full functor followed by a faithful functor, i.e., as

$$L \xrightarrow{\quad m \quad} L_M \xrightarrow{\quad m' \quad} Set$$

where $m : L \longrightarrow L_M$ is an identity-on-objects functor that is surjective on arrows and $m' : L_M \longrightarrow Set$ is a faithful functor.

Proposition 1. *For any model $M : L \longrightarrow Set$ of a countable Lawvere theory L, the category L_M is a countable Lawvere theory, the functor $m : L \longrightarrow L_M$ is a map of countable Lawvere theories, and the functor $m' : L_M \longrightarrow Set$ is a model of L_M.*

Proof. One must check that L_M has countable products, that m strictly preserves countable products, and that m' preserves countable products. One can either check that by direct calculation or deduce it from the fact that the (bijective-on-objects,fully faithful) factorisation system on Cat lifts to the category of small categories with finite products.

The countable Lawvere theory L_M is the construction we seek: if we start with a signature and a model of the signature, the arrows of L_M with codomain 1 are exactly equivalence classes of derived operations generated by the signature, with the equivalence relation given by two derived operations being put equal if they are equal in the model.

This construction yields all of the equations generated observationally between derived operations. For a logic, one would seek a finite presentation of the equations. Such a finite presentation cannot be generated by a notion of observational equivalence alone. But the construction does allow us to check whether a given finite presentation yields all equations that naturally hold observationally. And that sometimes involves delicate logical notions such as that of Hilbert-Post completeness.

Example 5. Continuing our investigation of global state from Examples 2 and 4, the data of Example 4 form the standard model for global state. And Proposition 1 yields the countable Lawvere theory L_S for *update* and *lookup* generated by the standard model. But we have also described a countable Lawvere theory L'_S for global state in terms of operations and equations in [14] (see also [5]) without reference to a model: the operations were *lookup* and *update*, subject to seven equation schema, which, with *lookup* corresponding to the logical symbol l and with *update* corresponding to u, can be expressed syntactically as

1. $l_{loc}(u_{loc,v}(x))_v = x$
2. $l_{loc}(l_{loc}(t_{vv'})_v)_{v'} = l_{loc}(t_{vv})_v$
3. $u_{loc,v}(u_{loc,v'}(x)) = u_{loc,v'}(x)$
4. $u_{loc,v}(l_{loc}(t_{v'})_{v'}) = u_{loc,v}(t_v)$
5. $l_{loc}(l_{loc'}(t_{vv'})_{v'})_v = l_{loc'}(l_{loc}(t_{vv'})_v)_{v'}$ where $loc \neq loc'$
6. $u_{loc,v}(u_{loc',v'}(x)) = u_{loc',v'}(u_{loc,v}(x))$ where $loc \neq loc'$
7. $u_{loc,v}(l_{loc'}(t_{v'})_{v'}) = l_{loc'}(u_{loc,v}(t_{v'}))_{v'}$ where $loc \neq loc'$.

These equations all hold of the standard model, so there is a canonical map of countable Lawvere theories from L'_S to L_S. But L_S is non-trivial as not all parallel pairs of derived operations are equal on $State \to State$, and, as explained in [14], L'_S is Hilbert-Post complete, i.e., to add any further non-trivial equations would force the models all to be trivial. So, as L_S must validate at least as many equations as L'_S does but is non-trivial, L'_S is isomorphic to L_S. Thus L'_S provides an equational characterisation of the countable Lawvere theory generated by *lookup* and *update* subject to the equivalence induced by the standard model in Example 4.

4 Models for the λ_c-Calculus

In this section, we briefly recall the notions of $Freyd$-category and closed $Freyd$-category as used in [20] to model the first-order fragment of the λ_c-calculus and the whole calculus respectively. The λ_c-calculus is a fragment of a call-by-value programming language such as ML or the idealised language FPC (see for instance [3]). For category theoretic models, the key feature is that there are two entities, expressions and values. So the most direct sound and complete class of models involves a pair of categories C_0 and C_1, together with an identity-on-objects inclusion functor $J : C_0 \longrightarrow C_1$, leading to the notion of closed $Freyd$-category. The first sound and complete class of models was given by Moggi

in [11], in which he effectively gave a construction of closed *Freyd*-categories without defining the notion.

In order to define the notions of *Freyd*-category and closed *Freyd*-category, we must recall the definition of symmetric premonoidal category as introduced in [21] and further studied in [19]. A symmetric premonoidal category is a generalisation of the concept of symmetric monoidal category: it is essentially a symmetric monoidal category except that the tensor need only be a functor of two variables and not necessarily be bifunctorial, i.e., given maps $f : X \longrightarrow Y$ and $f' : X' \longrightarrow Y'$, the evident two maps from $X \otimes X'$ to $Y \otimes Y'$ may differ.

There is a general construction that yields symmetric premonoidal categories: given a strong monad T on a symmetric monoidal category C, the Kleisli category $Kl(T)$ for T is always a symmetric premonoidal category, with the functor from C to $Kl(T)$ preserving the symmetric premonoidal structure strictly: of course, a symmetric monoidal category such as C is trivially a symmetric premonoidal category. That construction is fundamental, albeit implicit, in Eugenio Moggi's work on monads as notions of computation [12], as explained in [21].

One requires care in the definition of strict symmetric premonoidal functor, as it involves the notion of a *central* map, such being a map that, in a precise sense, is bifunctorial. But subject to that caveat, we can now define the notions of *Freyd*-category and closed *Freyd*-category.

Definition 4. *A Freyd-category is a category C_0 with finite products, a symmetric premonoidal category C_1, and an identity-on-objects strict symmetric premonoidal functor $J : C_0 \longrightarrow C_1$.*

Definition 5. *A Freyd-category $J : C_0 \longrightarrow C_1$ is* closed *if for every object X of C_0 (equivalently of C_1), the functor*

$$J(- \times X) : C_0 \longrightarrow C_1$$

has a right adjoint $X \to -$.

The following result is proved but only stated implicitly in [21]; it is stated explicitly in [8,20].

Theorem 1. *To give a category C_0 with finite products and a strong monad on it, such that Kleisli exponentials exist, is equivalent to giving a closed Freyd-category $J : C_0 \longrightarrow C_1$.*

This all means that the class of closed *Freyd*-categories provides a sound a complete class of models for the computational λ-calculus, and, as we shall recall later from [20], using a reasonable notion of its first order fragment (including *let* of course), the class of *Freyd*-categories is a sound and complete class of models for its first order fragment.

It is evident how to model types and terms in context in a (faithful) closed *Freyd*-category: the type constructors and contexts are modelled directly by the *Freyd*-structure, an arbitrary term in context is modelled by an arrow in

C_1, the predicate $(-)\downarrow$ is modelled for a term in context by the assertion that the arrow lies in C_0, and $=$ is modelled for two terms in context by the assertion that the two induced arrows are equal. The closed *Freyd*-categories of primary interest are faithful, equivalently the corresponding monad satisfies the "mono requirement". When that is not the case, one needs a little more subtlety in understanding models: the assertion that an arrow of C_1 lies in C_0 involves extra structure, not just a property; such subtlety in modelling a predicate is a standard part of the tradition of categorical logic. If C_0 is a topos, this interpretation canonically extends to intuitionistic predicate logic, cf Kripke-Joyal semantics [9]: see [16] for details. One can model classical logic either by restricting C_0 to be *Set* or by interpreting the predicates using a fibration: the former is given by extending the situation for intuitionistic logic by the observation that its modelling in *Set* is classical; the fibrational view is more complex.

5 Canonical Models for the λ_c-Calculus with Computational Effects

Given any closed *Freyd*-category and any signature of operations for the λ_c-calculus, together with equations between derived operations, one can interpret the operations in the *Freyd*-category, then check whether or not the equations are validated. But here we ask a different question: given operations and equations, can we construct a *canonical* closed *Freyd*-category together with an interpretation of the operations that satisfies the equations? Ideally, such a construction should satisfy a natural universal property.

In fact, if the arities of the operations are all countable (including the possibility of finiteness), as they are in all our leading examples, we can do that, and for what one might reasonably call the first-order fragment of the λ_c-calculus, it is remarkably simple, subject to some thought into exactly what one means by an interpretation of the operations.

Recall from Section 3 that the category $\aleph_1$ has countable coproducts. These are used in the definition of countable Lawvere theory, as the latter is defined to consist of a category L with countable products together with a countable-product preserving functor $I : \aleph_1^{op} \longrightarrow L$. Trivially, to give the countable-product preserving functor I is equivalent to giving a countable-coproduct preserving functor $J : \aleph_1 \longrightarrow L^{op}$. The category $\aleph_1$ not only has countable coproducts but also has finite products: these are given by finite products of countable sets. The category L^{op} generally does not have finite products, and the finite products of $\aleph_1$ are generally not preserved by J. But one can routinely check the following result:

Theorem 2. *For any countable Lawvere theory L, the category L^{op} together with the functor $I^{op} : \aleph_1 \longrightarrow L^{op}$ canonically support the structure of a Freyd-category.*

Proof. Given a countable (possibly finite) set α and given a map in L, say $f : \beta \longrightarrow \gamma$, we must define a map $\alpha \otimes f$ in L from $\alpha \times \beta$ to $\alpha \times \gamma$. The set

$\alpha \times \beta$ is the sum of α-many copies of β, and similarly for $\alpha \times \gamma$. The category L^{op} has countable sums, and countable sums are preserved by I^{op}. So we define $\alpha \otimes f : \alpha \times \beta \longrightarrow \alpha \times \gamma$ to be the sum in L^{op} of α copies of f: the domain and codomain of this sum are as desired because I^{op} preserves countable sums. This determines the rest of the data for a $Freyd$-structure, and it is routine to verify that the $Freyd$-category axioms all hold.

This result suggests a definition of the *first-order fragment* of the λ_c-calculus, yielding a canonical model of the first-order fragment together with operations subject to equational axioms.

By the first-order fragment of the λ_c-calculus, we mean type constructors

$$\sigma ::= 1 \mid \sigma_1 \times \sigma_2$$

and term constructors

$$e ::= * \mid \langle e, e' \rangle \mid \pi_i(e) \mid let\ x = e\ in\ e' \mid x$$

where x ranges over variables, $*$ is of type 1, with π_i existing for $i = 1$ or 2, all subject to the evident typing. We still have the two predicates: $=$ and $(-)\downarrow$ for effect-freeness. The rules for the latter say $* \downarrow$, $x \downarrow$, if $e \downarrow$ then $\pi_i(e) \downarrow$, and similarly for $\langle e, e' \rangle$, and that definedness is closed under equality. The rules for $=$ say that $=$ is a congruence, together with rules for the basic constructions and for unit and product types. The rules are closed under substitution of effect-free terms for variables. It follows from the rules for both predicates that types together with equivalence classes of terms in context form a category, with a subcategory determined by effect-free terms.

The *let* constructor is derivable in the full λ_c-calculus as $(\lambda x.e')e$. The class of $Freyd$-categories provides a sound and complete class of models for the first-order fragment of the λ_c-calculus just as that of closed $Freyd$-categories provides a sound and complete class of models for the full calculus. We can thus deduce the following from Proposition 2:

Corollary 1. *For any countable Lawvere theory L, the category L^{op} together with $I^{op} : \aleph_1 \longrightarrow L^{op}$ is a model of the first-order fragment of the λ_c-calculus.*

We shall call the countable Lawvere theory of the corollary the *canonical* model determined by the computational effect associated with L: we shall next show that the operations can be interpreted canonically in it, and that that interpretation respects the equations. We shall further give a universal property of the construction.

Consider *exactly* what one might mean by an interpretation of the operations of a signature. In previous work, we have investigated three main ways to interpret operations [15]. When considered in the context of a closed $Freyd$-category, all three are equivalent; in the absence of closedness, we can define two of those notions of interpretation, and they are still equivalent to each other. The difficulty for the third notion arises because when S is countable, $S \to (X \times S)$ is uncountable even when $X = 1$ [15]. Here, we focus on the notion that most directly yields our canonicity result. It uses the idea of a *generic effect*.

Definition 6. *Given a signature of typed basic operations and given a semantics for each type, an* interpretation *of an operation of type $\sigma \to \tau$ in a Freyd-category $J : C_0 \longrightarrow C_1$ is a map $M(\tau) \longrightarrow M(\sigma)$ in C_1, where $M(\sigma)$ and $M(\tau)$ are the interpretations of the types σ and τ.*

Example 6. Consider the usual interpretation of side-effects in the Kleisli category $Kl(S \to (- \times S))$ for the monad $S \to (- \times S)$ on Set, where $S = Val^{Loc}$. The operation $lookup : Val \longrightarrow Loc$ is interpreted by the function

$$Loc \longrightarrow (S \to (Val \times S))$$

taking (loc, σ) to (v, σ), where v is given by looking up loc in σ. To give a function from Loc to $(S \to (Val \times S))$ is to give a map in $Kl(S \to (- \times S))$ from Loc to Val. The operation $update : 1 \longrightarrow Loc \times Val$ is interpreted by the function

$$Loc \times Val \longrightarrow (S \to S)$$

sending (loc, v, σ) to the state that updates σ by replacing the value at loc by v; and that is a map in $Kl(S \to (- \times S))$ from $Loc \times Val$ to 1. This way of modelling operations as *generic effects* has proved particularly useful [15,5,6] and is consistent with Example 4 here. If we restrict from the λ_c-calculus to its first-order fragment, we can restrict the interpretation to land in the full sub-$Freyd$-category of $Kl(S \to (- \times S))$ determined by (a skeleton of) countable sets. This latter $Freyd$-category is exactly the canonical $Freyd$-category for global state determined by Corollary 1.

One can similarly use the notion of interpretation as we have defined it here to give canonical interpretations of $\vee$ for nondeterminism, *read* and *write* for interactive input/output, *raise* for exceptions, etcetera [15], all respecting the appropriate equations. One has the following trivial abstract proposition:

Proposition 2. *Every signature of operations of countable (possibly finite) arity has a canonical sound interpretation in the canonical model: an arity α is modelled by the object α, and a basic operation $op : \alpha \longrightarrow \beta$ is modelled by the corresponding map from β to α in L^{op}.*

Moreover, as the category $\aleph_1$ includes the object $\aleph_0$ as a coproduct of countably many copies of 1, we can model all the types, function symbols, and constant symbols in the signature of Example 1 in the canonical model. In particular, the natural numbers Nat is canonically modelled in the canonical model.

The canonical model also suggests a natural definition of what it means for an arbitrary $Freyd$-category to have finite coproducts.

Definition 7. *A $Freyd$-category $J : C_0 \longrightarrow C_1$ has* finite coproducts *if C_0 has and J preserves finite coproducts.*

Proposition 3. *For any closed $Freyd$-category $J : C_0 \longrightarrow C_1$, if C_0 has finite coproducts, so does J.*

This in turn suggests an extension of the first-order fragment of the λ_c-calculus to include sum types: by the first-order fragment of the λ_c-calculus with sum types, we mean type constructors

$$\sigma ::= 1 \mid \sigma_1 \times \sigma_2 \mid 0 \mid \sigma_1 + \sigma_2$$

and term constructors

$$e ::= * \mid \langle e, e' \rangle \mid \pi_i(e) \mid \mathit{let}\ x = e\ \mathit{in}\ e' \mid 0 \mid \mathit{inl}(e) \mid \mathit{inr}(e) \mid \mathit{cases}(e_1, e_2) \mid x$$

subject to evident typing rules and an extension of the rules for the predicates $=$ and $(-)\downarrow$ to make the class of *Freyd*-categories $J : C_0 \longrightarrow C_1$ with finite coproducts a sound and complete class of models.

There is more flexibility here than might first appear. If a cartesian closed category C has finite coproducts, it follows that, for every object X of C, the functor $- \times X : C \longrightarrow C$ preserves them, i.e., product distributes over sum. But if C only has finite products without being closed, $- \times X$ might not preserve finite coproducts. But there is a strong argument that one should insist upon such preservation, yielding the notion of a *distributive* category (see, for instance, [2]). The same issue arises for *Freyd*-categories: it is possible we should ultimately emphasise the (obvious) notion of distributive *Freyd*-category, which in turn would imply further axioms on an extension of the λ_c-calculus to include sum types. Here, we need to define a notion of countable distributivity anyway.

Definition 8. *A Freyd-category $J : C_0 \longrightarrow C_1$ is* countably distributive *if C_0 has and J strictly preserves countable coproducts, and finite products distribute over countable coproducts in C_0.*

It follows immediately from the definition of countable Lawvere theory that if L is a countable Lawvere theory, the *Freyd*-category $I^{op} : \aleph_1 \longrightarrow L^{op}$ is countably distributive. This notion allows us to characterise the canonical model by a universal property.

Theorem 3. *The canonical model is the generic countably distributive Freyd-category, i.e., for any countably distributive Freyd-category $J : C_0 \longrightarrow C_1$ and any sound interpretation of the signature in J that respects the coproduct structure of the arities, there is, up to coherent isomorphism, a unique countable coproduct preserving Freyd-functor from I^{op} to J that respects the interpretations.*

Theorem 3 can now be combined with the work of [20], which shows how to generate a canonical model of the whole λ_c-calculus from a model of its first-order fragment. A variant of the latter construction, involving preservation of countable coproducts, is needed to give a smooth extension of the operations: that must, for space reasons, be deferred, but we mention in passing that it also allows one to recover Moggi's models of all the examples we have investigated here.

References

1. N. Benton, J. Hughes, and E. Moggi, Monads and effects, in *Advanced Lectures form International Summer School on Applied Semantics*, LNCS, Vol. 2395, pp. 42–122, Berlin: Springer-Verlag, 2002.
2. A. Carboni, S. Lack, and R. F. C. Walters, Introduction to extensive and distributive categories, *J. Pure Appl. Algebra*, Vol. 84, pp. 145–158, 1993.
3. M. Fiore, G. D. Plotkin, and D. Turi, Abstract syntax and variable binding, in *Proc. LICS 99*, pp. 214–224, Washington: IEEE Press, 1999.
4. M. Hyland, P. B. Levy, G. D. Plotkin, and A. J. Power, Combining continuations with other effects, submitted.
5. M. Hyland, G. D. Plotkin, and A. J. Power, Combining computational effects: commutativity and sum, in *Proc. IFIP Conf. On Theoretical Computer Science*, pp. 474–484, Kluwer, 2002.
6. M. Hyland, G. D. Plotkin, and A. J. Power, Combining computational effects: sum and tensor, submitted.
7. Y. Kinoshita and A. J. Power, Data Refinement in Call-by-Value Languages, in *Proc. CSL '99*, LNCS, Vol. 1683, pp. 562–576 Berlin: Springer-Verlag, 1999.
8. P. B. Levy, A. J. Power, and H. Thielecke, Modelling environments in call-by-value programming languages, *Information and Computation*, Vol. 185, pp. 182–210, 2003.
9. S. Mac Lane and I. Moerdijk, *Sheaves in Geometry and Logic*, Berlin: Springer-Verlag, 1992.
10. E. Moggi, Computational lambda-calculus and monads, in *Proc. LICS '89*, pp. 14–23, Washington: IEEE Press, 1989.
11. E. Moggi, Notions of computation and monads, *Inf. and Comp.*, Vol. 93, No. 1, pp. 55–92, 1991.
12. E. Moggi, A Semantics for Evaluation Logic, *Fundamenta Informaticae*, Vol. 22, 1995.
13. G. D. Plotkin and A. J. Power, Adequacy for Algebraic Effects, in *Proc. FOSSACS 2001*, LNCS, Vol. 2030, pp. 1–24, Berlin: Springer-Verlag, 2001.
14. G. D. Plotkin and A. J. Power, Notions of Computation Determine Monads, in *Proc. FOSSACS 2002*, LNCS, Vol. 2303, pp. 342–356, Berlin: Springer-Verlag, 2002.
15. G. D. Plotkin and A. J. Power, Algebraic Operations and Generic Effects, in *Proc. MFCSIT 2000, Applied Categorical Structures*, Vol. 11, No. 1, pp. 69–94, 2003.
16. G. D. Plotkin and A. J. Power, Computational Effects and Operations: an Overview, in *Proc. Domains 2002*, ENTCS, Vol. 73, 2002.
17. G. D. Plotkin and A. J. Power, Logic for Computational Effects, in *Proc. International Workshop on Formal Methods 03*, to appear.
18. A. J. Power, Models of the Computational λ-calculus, in *Proc. MFCSIT 2000*, ENTCS, Vol. 40, 2001.
19. A.J. Power, Premonoidal categories as categories with algebraic structure, *Theoretical Computer Science*, Vol. 278, pp. 303–321, 2002.
20. A. J. Power, A Universal Embedding for the Higher Order Structure of Computational Effects, in *Proc. TLCA 2003*, LNCS, Vol. 2701, pp. 301–315.
21. A. J. Power and E. P. Robinson, Premonoidal categories and notions of computation, in *Proc. LDPL 96*, *Math Structures in Computer Science*, Vol. 7, pp. 453–468, 1997.