Adequacy for Algebraic Effects

Gordon Plotkin and John Power *

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

Abstract. Moggi proposed a monadic account of computational effects. He also presented the computational λ-calculus, λ_c, a core call-by-value functional programming language for effects; the effects are obtained by adding appropriate operations. The question arises as to whether one can give a corresponding treatment of operational semantics. We do this in the case of algebraic effects where the operations are given by a single-sorted algebraic signature, and their semantics is supported by the monad, in a certain sense. We consider call-by-value PCF with—and without—recursion, an extension of λ_c with arithmetic. We prove general adequacy theorems, and illustrate these with two examples: non-determinism and probabilistic nondeterminism.

1 Introduction

Moggi introduced the idea of a general account of computational effects, proposing encapsulating them via monads $T : \mathbf{C} \to \mathbf{C}$; the main idea is that $T(x)$ is the type of computations of elements of x. He also presented the computational λ-calculus λ_c as a core call-by-value functional programming language for effects [21]. The effects themselves are obtained by adding appropriate operations, specified by a signature Σ. Moggi introduced the consideration of these operations in the context of his metalanguage $\mathrm{ML}(\Sigma)$ whose purpose is to give the semantics of programming languages [22, 23], but which is not itself thought of as a programming language.

In our view any complete account of computation should incorporate a treatment of operational semantics; this has been lacking for the monadic view. To progress, one has to deal with the operations as they are the source of the effects. In this paper we give such a treatment in the case of *algebraic* effects where the operations are given by a single-sorted algebraic signature Σ; semantically such an n-ary operation f is taken to denote a family of morphisms

$$f_x : T(x)^n \longrightarrow T(x)$$

parametrically natural with respect to morphisms in the Kleisli category $\mathbf{C}_T$; T is then said to *support* the family f_x. (In [22] only naturality with respect to morphisms in $\mathbf{C}$ is considered; we use the stronger assumption.) Note that

* This work has been done with the support of EPSRC grant GR/M56333: The Structure of Programming Languages: Syntax and Semantics.

there is no assumption that the monads at hand are commutative. For $\mathbf{C} = \mathbf{Set}$, examples are the finite powerset monad and binary choice operations; the monad for probabilistic nondeterminism and probabilistic choice operations; and the monad for printing and the printing operations (these are noncommutative). As will be discussed below, there are natural analogues of these examples in the domain-theoretic context where $\mathbf{C} = \mathbf{Dcppo}$, the category of dcppos and continuous functions. Generally, suppose we are given a category $\mathbf{C}$ with finite products and a finitary equational theory over a signature Σ. Assuming free Σ-algebras exist, let T be the associated monad. Then every operation symbol yields such a family, in an evident way. In the case $\mathbf{C} = \mathbf{Set}$ a converse holds, that every parametrically natural family arises as a composition of such families, as follows, e.g., from a remark in Section 3 below.

On the other hand, for example, the exceptions monad does not support its exception handling operation: only the weaker naturality holds there. This monad is a free algebra functor for an equational theory, viz the one that has a constant for each exception and no equations; however the exception handling operation is not definable: only the exception raising operations are. Other standard monads present further difficulties. So while our account of operational semantics is quite general, it certainly does not cover all cases; it remains to be seen if it can be further extended.

To give an account of operational semantics we need a programming language based on the computational λ-calculus with some basic datatypes and functions in order to permit computation. We take as the test of our account whether a useful general adequacy theorem can be proved. So we consider a call-by-value PCF with algebraic effects, an extension of the computational λ-calculus with operations, arithmetic and recursion (see, e.g., [34, 32] for versions of call-by-value PCF). We begin by treating the sublanguage without recursion. Section 2 presents both a *small step* and a *(collecting) big step* operational semantics; there is also an associated *evaluation function.* Section 3 considers denotational semantics and gives an adequacy theorem. The semantics is given axiomatically in terms of a suitable class of categorical structures appropriately extending the usual monadic view of the computational λ-calculus. This could as well have been based on closed Freyd categories [30], and [2] is a treatment of nondeterminism along such lines. Section 4 considers two examples: nondeterminism and probabilistic nondeterminism.

We consider the full language with recursion in Section 5. Small step semantics is straightforward, but big step semantics presents some difficulties as evaluation naturally yields infinite values since programs may not terminate. We also consider an intermediate *medium step* semantics which is big step as regards effect-free computation and small step as regards effects. For the semantics we assume a suitable order-enrichment [16] in order to give a least fixed-point treatment of recursion. This then yields an adequacy theorem, which is the main result of the paper. One wonders if a more general treatment of recursion is possible within synthetic or axiomatic domain theory, cf. [32]. In Section 6 we revisit the examples, but with recursion now present. Finally, in Section 7 we present some ideas for further progress. While all definitions are given in Section 5, some previous knowledge of domain theory is really assumed; for Section 6 this is essential as both ordinary and probabilistic powerdomains are considered.

Our treatment of operational semantics might well be seen as rather formal and does not immediately specialise to the usual accounts for the examples at hand. In a way this has to be so: it is hard to imagine a theory which yields the natural operational semantics for any possible computational effect. On the other hand we can prove adequacy (with and without recursion) for our formal approach and then easily recover adequacy results for the standard operational semantics of the various examples.

2 PCF without Recursion: Operational Semantics

We begin with the syntax of our language. Its types are given by

$$\sigma \ ::= \ \iota \ \mid \ o \ \mid \ 1 \ \mid \ \sigma \times \sigma \ \mid \ \sigma \to \sigma$$

where ι is the type of the natural numbers and o is that of the booleans. For the terms, we assume we are given a single-sorted algebraic signature Σ and a countably infinite set of variables x; the signature provides a set of operation symbols f and associates an arity $\mathrm{ar}_f \geq 0$ to each. The terms are given by

$$\begin{array}{rcl} M & ::= & \underline{0} \mid \mathrm{succ}(M) \mid \mathrm{zero}(M) \mid \mathrm{pred}(M) \mid \\ & & \mathit{tt} \mid \mathit{ff} \mid \mathbf{if}\ M\ \mathbf{then}\ M\ \mathbf{else}\ M \mid \\ & & * \mid <M, M> \mid \pi_1(M) \mid \pi_2(M) \mid \\ & & x \mid \lambda x : \sigma.M \mid MM \mid \\ & & f(M_1, \ldots, M_n) \end{array}$$

where, in the last clause, f is an operation symbol of arity n. Substitution $M[N/x]$ is defined as usual, and we identify terms up to α-equivalence.

As regards comparison with λ_c, we have fixed a particular set of base types, viz ι and o, and function symbols, viz $\underline{0}$, succ, zero, pred, tt and ff. We also have a conditional construct **if** and the operation symbols f from Σ. We do not have a type constructor $T(\sigma)$ as it may be defined to be $1 \to \sigma$. Nor do we have a **let** constructor or constructions $[M]$ or $\mu(M)$ as we consider $\mathbf{let}\ x : \sigma = M\ \mathbf{in}\ N$ as syntactic sugar for $(\lambda x : \sigma.N)M$ (preferring explicit declaration of types in binding contexts), and $[M]$ as syntactic sugar for $\lambda x : 1.M$, where x is fresh, and $\mu(M)$ as syntactic sugar for $M*$.

The typing rules specify the well formed sequents

$$\Gamma \vdash M : \sigma$$

The rules will all be evident, except perhaps, that for the last construct, which is

$$\frac{\Gamma \vdash M_i : \sigma \ (\text{for } i = 1, n)}{\Gamma \vdash f(M_1, \ldots, M_n) : \sigma}$$

(where $\mathrm{ar}_f = n$). Thus effects can occur at any type. We write $M : \sigma$ for $\vdash M : \sigma$ and say then that the (closed) term M is *well-typed*.

We give both a small step and a collecting big step operational semantics. These are given without using any further information on the operations; it is in that sense that they are purely formal. One might instead have introduced equations corresponding to the intended effects, e.g., semilattice equations for finite nondeterminism, as in [12] and then worked with operational semantics up to provable equality.

The small step semantics proceeds by means of two kinds of transitions between closed terms, an unlabelled one

$$M \to M'$$

and a labelled one

$$M \xrightarrow{f_i} M'$$

where f is an operation symbol of arity $n > 0$ and $1 \leq i \leq n$, and there is also a predicate on closed terms

$$M \downarrow_a$$

for operations a of arity 0 i.e., *constants*. The unlabelled transition relation corresponds to effect-free computation; the labelled one corresponds to an effect, which is mirrored syntactically by "entering" the ith argument place of an operation symbol f of strictly positive arity. Constants yield a kind of exceptional termination.

Transitions terminate at *values*, given by

$$V \ ::= \ \underline{0} \ \mid \ \mathrm{succ}(V) \ \mid \ tt \ \mid \ ff \ \mid \ * \ \mid <V, V> \mid \ \lambda x : \sigma.M$$

where we restrict $\lambda x : \sigma.M$ to be closed. We write $\underline{n}$ for $\mathrm{succ}^n(\underline{0})$.

It is convenient to use Felleisen's idea [7] of specifying transitions via evaluation contexts and redexes. The evaluation contexts here are given by:

$$\begin{aligned} E \ ::= \ & [\,] \mid \mathrm{succ}(E) \mid \mathrm{zero}(E) \mid \mathrm{pred}(E) \mid \mathbf{if}\ E\ \mathbf{then}\ M\ \mathbf{else}\ M \mid \\ & <E, M> \mid <V, E> \mid \ \pi_1(E) \mid \pi_2(E) \mid EM \mid VE \end{aligned}$$

where the terms appearing are restricted to be closed.

Redexes and their transitions are given by:

$$\mathrm{zero}(\underline{0}) \to tt \quad \mathrm{zero}(\underline{n+1}) \to ff$$

$$\mathrm{pred}(\underline{0}) \to \underline{0} \quad \mathrm{pred}(\underline{n+1}) \to \underline{n}$$

$$\mathbf{if}\ tt\ \mathbf{then}\ M\ \mathbf{else}\ N \to M \quad \mathbf{if}\ ff\ \mathbf{then}\ M\ \mathbf{else}\ N \to N$$

$$\pi_1(<V, V'>) \to V \quad \pi_2(<V, V'>) \to V'$$

$$(\lambda x : \sigma.M)V \to M[V/x]$$

$$f(M_1, \ldots, M_n) \xrightarrow{f_i} M_i$$

where, in the last clause, $n = \mathrm{ar}_f > 0$, and $1 \leq i \leq n$, and where we restrict the redexes (the left hand sides) to be closed. Noting that $\rightarrow$ is deterministic here, we will find it useful to write R^+ for the unique N, if any, such that $R \rightarrow N$. For any closed well-typed term one of three mutually exclusive possibilities holds: it is a value; it has the form $E[R]$ for a unique E and R; or it has the form $E[a()]$ for a unique E and a.

We now define the transition relations by the two rules:

$$\frac{R \rightarrow N}{E[R] \rightarrow E[N]}$$

$$\frac{R \xrightarrow{f_i} N}{E[R] \xrightarrow{f_i} E[N]}$$

and the predicate by the rule

$$E[a()] \downarrow_a$$

For any closed well-typed term M which is not a value, exactly one of three mutually exclusive possibilities hold:

- $M \rightarrow N$ for some N; in this case N is determined and of the same type as M.
- $M \xrightarrow{f_i} N_i$ for some f and N_i ($1 \leq i \leq \mathrm{ar}_f$); in this case f and the N_i are determined and the latter are of the same type as M.
- $M \downarrow_a$ for some a; in this case a is determined.

The big step operational semantics

$$M \Rightarrow t$$

evaluates closed terms to *effect values*, which are the terms given by

$$t \ ::= \ V \ \mid \ f(t_1, \ldots, t_n)$$

The idea is that these terms "collect" together all the possible effects of a computation. The big step semantics is then defined by the following rules

$$V \Rightarrow V$$

$$\frac{R \rightarrow N \quad E[N] \Rightarrow t}{E[R] \Rightarrow t}$$

$$\frac{R \xrightarrow{f_i} N_i \quad E[N_i] \Rightarrow t_i \quad (i = 1, n)}{E[R] \Rightarrow f(t_1, \ldots, t_n)} \ \text{(where } n = \mathrm{ar}_f\text{)}$$

$$E[a()] \Rightarrow a()$$

Note that a closed term evaluates to at most one effect value; also if that term is well-typed, so is the effect value and it has the same type as the term. Small step and big step operational semantics can both be presented structurally. Example

$$\frac{M \rightarrow M'}{MN \rightarrow M'N} \qquad \frac{M \xrightarrow{f_i} M'}{MN \xrightarrow{f_i} M'N} \qquad \frac{M \downarrow_a}{MN \downarrow_a}$$

$$\frac{N \rightarrow N'}{VN \rightarrow VN'} \qquad \frac{N \xrightarrow{f_i} N'}{VN \xrightarrow{f_i} VN'} \qquad \frac{N \downarrow_a}{VN \downarrow_a}$$

$$(\lambda x : \sigma.M)V \rightarrow M[V/x]$$

Fig. 1. Small Step Rules for Function Application

small step rules for function application are given in Figure 1. The other rules for small step operational semantics can easily be given in the same (somewhat tedious) style.

The rules for big step semantics are not quite so obvious. First we need *effect contexts*; these are given by

$$\epsilon \ ::= \ [\,] \ | \ f(\epsilon_1, \ldots, \epsilon_n)$$

Any effect term t can be written uniquely in the form $\epsilon[V_1, \ldots, V_k]$. The rules are given in Figure 2. In reading these, if a term M^+ appears in a rule, the rule only applies if M^+ exists.

To connect the two semantics, we associate an effect value $|\,M\,|$ with any closed term M which is *terminating* in the sense that there is no infinite chain of (small step) transitions from M:

$$|\,M\,| = \begin{cases} |\,N\,| & (\text{if } M \rightarrow N) \\ f(|\,N_1\,|, \ldots, |\,N_n\,|) & (\text{if } M \xrightarrow{f_i} N_i, \ \text{for } i = 1, n, \ \text{where } n = \mathrm{ar}_f) \\ a() & (\text{if } M \downarrow_a) \end{cases}$$

Proposition 1. *The following are equivalent for any closed well-typed M and t*

1. *M is terminating, and $|\,M\,| = t$*
2. *$M \Rightarrow t$*

Thus we have two independent characterisations of the *evaluation function* $|\cdot|$ on closed well-typed terms. One could also give a direct recursive definition of this function, but one is then faced with interpreting the recursion and relating this to the above rule-based definitions. While the effort does not seem worthwhile here, it may be so for PCF with recursion, as will be seen.

As the reader may have gathered, there is no possibility of nontermination:

Theorem 1. *Every closed well-typed term terminates.*

Proof. This can be proved by a computability argument. We content ourselves with defining the computability predicates. At the types ι, o and 1 all values are

$$\underline{0} \Rightarrow \underline{0} \qquad \frac{M \Rightarrow \epsilon[V_1, \ldots, V_k]}{\mathrm{succ}(M) \Rightarrow \epsilon[\mathrm{succ}(V_1), \ldots, \mathrm{succ}(V_k)]}$$

$$\frac{M \Rightarrow \epsilon[V_1, \ldots, V_k]}{\mathrm{pred}(M) \Rightarrow \epsilon[\mathrm{pred}(V_1)^+, \ldots, \mathrm{pred}(V_k)^+]} \qquad \frac{M \Rightarrow \epsilon[V_1, \ldots, V_k]}{\mathrm{zero}(M) \Rightarrow \epsilon[\mathrm{zero}(V_1)^+, \ldots, \mathrm{zero}(V_k)^+]}$$

$$\underline{tt} \Rightarrow \underline{tt} \qquad \underline{ff} \Rightarrow \underline{ff}$$

$$\frac{L \Rightarrow \epsilon[V_1, \ldots, V_k] \quad (\mathbf{if}\ V_i\ \mathbf{then}\ M\ \mathbf{else}\ N)^+ \Rightarrow t_i\ (\text{for } i = 1, k)}{\mathbf{if}\ L\ \mathbf{then}\ M\ \mathbf{else}\ N \Rightarrow \epsilon[t_1, \ldots, t_k]}$$

$$* \Rightarrow *$$

$$\frac{M \Rightarrow \epsilon_1[V_1, \ldots, V_k] \quad N \Rightarrow \epsilon_2[W_1, \ldots, W_l]}{<M, N> \Rightarrow \epsilon_1[\epsilon_2[<V_1, W_1>, \ldots, <V_1, W_l>], \ldots, \epsilon_2[<V_k, W_1>, \ldots, <V_k, W_l>]]}$$

$$\frac{M \Rightarrow \epsilon[V_1, \ldots, V_k]}{\pi_i(M) \Rightarrow \epsilon[\pi_i(V_1)^+, \ldots, \pi_i(V_k)^+]}\ (i = 1, 2)$$

$$\lambda x : \sigma.M \Rightarrow \lambda x : \sigma.M$$

$$\frac{M \Rightarrow \epsilon_1[V_1, \ldots, V_k] \quad N \Rightarrow \epsilon_2[W_1, \ldots, W_l] \quad (V_i W_j)^+ \Rightarrow t_{ij}\ (\text{i=1,k; j= 1, l})}{MN \Rightarrow \epsilon_1[\epsilon_2[t_{11}, \ldots, t_{1l}], \ldots, \epsilon_2[t_{k1}, \ldots, t_{kl}]]}$$

$$\frac{M_i \Rightarrow \epsilon_i[V_{i1}, \ldots, V_{ik_i}]\ (i = 1, n)}{f(M_1, \ldots, M_n) \Rightarrow f(\epsilon_1[V_{11}, \ldots, V_{1k_1}], \ldots, \epsilon_n[V_{n1}, \ldots, V_{nk_n}])}$$

Fig. 2. Big Step Operational Semantics.

computable. A value of product type is computable if both its components are. A value $\lambda x : \sigma.M$ is computable if, and only if, $M[V/x]$ is for every computable value $V : \sigma$, where we say that a closed term is computable if, and only if, every transition sequence from it terminates in a computable value.

There is a natural equational theory, including "β-equations" and commutation equations for operation symbols. This establishes judgements of the form

$$\Gamma \vdash M = N : \sigma$$

where it is assumed that $\Gamma \vdash M : \sigma$ and $\Gamma \vdash N : \sigma$. There are evident rules for equality including closure under the term-forming operations. The axioms are given in Figure 3 where they are presented as equations, or equational schemas $M = N$; these should be interpreted as judgements $\Gamma \vdash M = N : \sigma$. The commutation schema for operations is equivalent to a collection of equations for the individual language constructs. It would be better to allow open values and contexts (in a fairly evident sense) but the more restricted version presented here suffices for our purposes. The next proposition makes it easy to verify that the denotational semantics models the operational semantics.

Proposition 2. *Suppose that M is a closed term such that $M : \sigma$. Then if $M \Rightarrow t$ it follows that $\vdash M = t : \sigma$*

$$\begin{array}{c}
\mathrm{zero}(\underline{0}) = tt \qquad \mathrm{zero}(\underline{n+1}) = ff \\
\mathrm{pred}(\underline{0}) = \underline{0} \quad \mathrm{pred}(\underline{n+1}) = \underline{n} \\
\textbf{if } tt \textbf{ then } M \textbf{ else } N = M \;\; \textbf{if } ff \textbf{ then } M \textbf{ else } N = N \\
\pi_1(<V, V'>) = V \quad \pi_2(<V, V'>) = V' \\
(\lambda x : \sigma.M)V = M[V/x] \\
E[f(M_1, \ldots, M_n)] = f(E[M_1], \ldots, E[M_n])
\end{array}$$

Fig. 3. Equations.

3 PCF without Recursion: Adequacy

We begin by defining the categorical structures that provide models of our language, building on the sound and complete class of models for the λ_c-calculus provided by Moggi in [21]. A *model* of λ_c consists of a category $\mathbf{C}$ with finite products, together with a strong monad $< T, \eta, \mu, \mathrm{st} >$ on $\mathbf{C}$, such that T has Kleisli exponentials. The latter means that for each pair of objects x and y, the functor $\mathbf{C}(- \times x, Ty) : \mathbf{C}^{op} \longrightarrow Set$ is representable; in other words, there exists an object $x \Rightarrow y$ and a natural isomorphism

$$(\lambda_T)_z : \mathbf{C}(z \times x, Ty) \cong \mathbf{C}(z, x \Rightarrow y)$$

We write $g^\dagger : z \times T(x) \to T(y)$ for the parametrised lift of $g : z \times x \to T(y)$ to $z \times T(x)$ (and we use the same notation for the ordinary unparametrised lift where $g : x \to T(y)$ as that is essentially the subcase where z is the terminal object).

We need to extend this with structure for the operations, and for arithmetic and booleans. For the former, as already stated, we assume for each n-ary operation f a family

$$f_x : T(x)^n \longrightarrow T(x)$$

parametrically natural with respect to morphisms in the Kleisli category $\mathbf{C}_T$; this means that for every map $g : z \times x \to T(y)$, the diagram

$$\begin{tikzcd} z \times T(x)^n \arrow[r, "(g^\dagger \cdot (z \times \pi_i))_{i=1}^n"] \arrow[d, "z \times f_x"'] & T(y)^n \arrow[d, "f_y"] \\ z \times T(x) \arrow[r, "g^\dagger"'] & T(y) \end{tikzcd}$$

commutes (cf. [27]). Equivalently we can ask that the family be natural with respect to morphisms in the category $\mathbf{C}$ and that it respect the monad multiplication and the strength. Assuming that the n-fold coproduct of 1 with itself exists in $\mathbf{C}$, one can also show that there is a natural 1-1 correspondence between such families and global elements of $T(n)$. Finally, if 1 is a generator then parametric and ordinary naturality (with respect to Kleisli morphisms) coincide.

For arithmetic we assume $\mathbf{C}$ has a natural numbers object

$$1 \xrightarrow{0} \mathbf{N} \xrightarrow{s} \mathbf{N}$$

and for the booleans we assume that the sum $\mathbf{T} =_{\mathrm{def}} 1 + 1$ exists in $\mathbf{C}$. We write $\mathrm{Iter}_x(a, f)$ for the unique morphism from $\mathbf{N}$ to x corresponding to a pair of morphisms $1 \xrightarrow{a} x \xrightarrow{f} x$.

This gives us a $\mathbf{C}$ object as the denotation $[\![\sigma]\!]$ of each type σ, following [21] and taking $[\![\iota]\!] = \mathbf{N}$ and $[\![o]\!] = \mathbf{T}$. The denotation $[\![\Gamma]\!]$ of an environment Γ of the form $x_1 : \sigma_1, \ldots, x_n : \sigma_n$ is then the product of the denotations of the σ_i, as usual, and we now have to find the denotations

$$[\![M]\!] : [\![\Gamma]\!] \to T([\![\sigma]\!])$$

of terms of type σ in the environment Γ. These are defined as in [21] for the λ_c part of our language, once we settle the interpretations of the function symbols $\underline{0}$, succ, zero, pred, *tt* and *ff*. We then have to consider the conditional and the effect operations. For the former, it is enough to specify a morphism in $\mathbf{C}$ of appropriate type. For $\underline{0}$ and succ we take $1 \xrightarrow{0} \mathbf{N}$ and $\mathbf{N} \xrightarrow{s} \mathbf{N}$ respectively; for zero and pred we take $\mathbf{N} \xrightarrow{z} \mathbf{T}$ and $\mathbf{N} \xrightarrow{p} \mathbf{N}$, where $z = \mathrm{Iter}_{\mathbf{T}}(\mathrm{inl}, \mathrm{inr}\circ t)$ and $p = \pi_1 \circ \mathrm{Iter}_{\mathbf{N}\times\mathbf{N}}(<0, 0>, <s\circ\pi_1, \pi_1>)$ respectively; and for *tt* and *ff* we take $1 \xrightarrow{inl} \mathbf{T}$ and $1 \xrightarrow{inr} \mathbf{T}$.

In order to give the semantics of conditionals, we note the isomorphisms

$$\begin{aligned} \mathbf{C}_T(y, z)^2 &\cong \mathbf{C}(1, y \Rightarrow z)^2 \\ &\cong \mathbf{C}(\mathbf{T}, y \Rightarrow z) \\ &\cong \mathbf{C}_T(\mathbf{T} \times y, z) \\ &\cong \mathbf{C}_T(y \times \mathbf{T}, z) \end{aligned}$$

and take $\mathrm{cond}_z : T(z)^2 \times \mathbf{T} \longrightarrow z$ to be the $\mathbf{C}_T$ morphism corresponding to the pair π_1, π_2 of $\mathbf{C}_T$ morphisms from $T(z)^2$ to z. Now, for a term $M$of the form **if** L **then** M **else** N, where $\Gamma \vdash M : \sigma$, we define

$$[\![\mathbf{if}\ L\ \mathbf{then}\ M\ \mathbf{else}\ N]\!] = \mathrm{cond}^{\dagger}_{[\![\sigma]\!]} \circ << [\![M]\!], [\![N]\!] >, [\![L]\!] >$$

Finally, for a term M of type σ of the form $f(M_1, \ldots, M_n)$ we define

$$[\![f(M_1, \ldots, M_n)]\!] = f_{[\![\sigma]\!]} \circ < [\![M_1]\!], \ldots, [\![M_n]\!] >$$

The next two lemmas say that the semantics of a value is effect free (it *exists* in the sense of [21]) and that the above equations are sound for the semantics.

Lemma 1. *Suppose* $V : \sigma$. *Then* $[\![V]\!] : 1 \to [\![\sigma]\!]$ *factors through* $\eta_{[\![\sigma]\!]}$.

Lemma 2. *If* $\Gamma \vdash M = N : \sigma$ *then* $[\![M]\!] = [\![N]\!]$

The naturality condition for operations is used here to establish the soundness of the commutation schema. In fact, only naturality with respect to Kleisli morphisms is used (rather than parametric naturality); the latter would be needed for open contexts.

When reading the following adequacy theorem, recall that, by Theorem 1, all computations terminate.

Theorem 2. Adequacy *Suppose that* $M : \sigma$. *Then* $[\![M]\!] = [\![\,|M|\,]\!]$

Proof. This is an immediate consequence of Proposition 2 and Lemma 2

This result is very much in the spirit of Mezei and Wright [19], and Theorem 4.26 of [8] is a similar result for recursive program schemes. Such results say that the denotational semantics of a program is that of the result of a preliminary symbolic computation. Our result may not seem to the reader to be the expected statement of adequacy, but it does imply that the semantics of a term determines its operational result (at least up to its meaning). Furthermore, as we shall see in the next section, it readily yields the adequacy theorem one would expect in concrete cases.

4 Examples

We take $\mathbf{C}$ to be **Set** in our examples, and consider two monads, one for nondeterminism and the other for probabilistic nondeterminism.

Nondeterminism

Here T is $\mathcal{F}^+$ the nonempty finite powerset functor, equipped with the evident strong monad structure (and recall that every monad on **Set** has a unique strength). We take Σ to have one binary (infix) operator **or**, and $\mathbf{or}_X$ to be binary union. Note that $\mathcal{F}^+(X)$ is the free semilattice over X (meaning the structure with an associative, commutative and absorptive binary operator); this algebraic view of nondeterminism was emphasised in [12]. is $\{0,1\}$.

A small step structural operational semantics can be be defined much like our general one, except that there is no reason to record whether a "left choice" or a "right choice" is made. So the definition of the transition relation $\rightarrow_n$ is exactly as the general one except that one puts

$$M_1\ \mathbf{or} M_2 \rightarrow_n M_i\ \text{(i=1,2)}$$

This $\rightarrow_n$ is then the union of the general $\rightarrow$ and the $\stackrel{\mathbf{or}_i}{\rightarrow}$.

For big step semantics of nondeterminism one normally defines a nondeterministic transition relation between closed terms and values; for example the rule for function application is

$$\frac{M \Rightarrow_n \lambda x:\sigma.M' \quad N \Rightarrow_n V \quad M'[V/x] \Rightarrow_n V'}{MN \Rightarrow_n V'}$$

However, there is another possibility. This is to define a collecting big step transition relation $M \Rightarrow_n u$ between closed terms and nonempty finite sets of values. It can be given a structural definition by very similar rules to those for the general collecting big step semantics, such as

$$\frac{M \Rightarrow_n \{\lambda x:\sigma.M_i\} \quad N \Rightarrow_n \{V_j\} \ \ M_i[V_j/x] \Rightarrow_n u_{ij}\ \text{(for all } i,j)}{MN \Rightarrow_n \bigcup_{i,j} u_{ij}}$$

Its relation to the normal nondeterministic big step transition relation is that for any $M:\sigma$,

$$M \Rightarrow_n u \text{ iff } u = \{V \mid M \Rightarrow_n V\}$$

Now we can make the relationship between operational semantics for nondeterminism and the corresponding case of the general operational semantics for algebraic effects explicit. First to every effect term t assign a nonempty finite set of values $h(t)$ by

$$\begin{aligned} h(V) &= \{V\} \\ h(t\ \mathbf{or}\ t') &= h(t) \cup h(t') \end{aligned}$$

Then one has that for any $M:\sigma$,

$$M \Rightarrow_n u \text{ iff } \exists t. M \Rightarrow t \wedge u = h(t)$$

as will be evident from the form of the rules for the collecting big step transition relation.

One has for any effect term $t:\sigma$ that

$$[\![t]\!](*) = \bigcup_{V \in h(t)} [\![V]\!](*)$$

and with this one can prove an adequacy theorem for nondeterminism using our general adequacy theorem.

Theorem 3. *For any closed term $M : \sigma$,*

$$[\![M]\!](*) = \bigcup_{M \Rightarrow_n V} [\![V]\!](*)$$

Proof.

$$\begin{aligned} [\![M]\!](*) &= [\![\,|M|\,]\!](*) \text{ (by Theorem 2)} \\ &= \bigcup_{V \in h(|M|)} [\![V]\!](*) \text{ (by above remark)} \\ &= \bigcup_{M \Rightarrow_n V} [\![V]\!](*) \text{ (by above remark)} \end{aligned}$$

As stated this theorem is rather abstract because of the higher types. For $\sigma = \iota$ it takes the form that for any closed term $M : \iota$

$$m \in [\![M]\!](*) \text{ iff } M \Rightarrow_n \underline{m}$$

Probabilistic Nondeterminism

Here things are, perhaps, not quite so simple. We take $T(X)$ to be $\mathcal{D}_\omega(X)$ the set of finite probability distributions over X. The unit η sends an element of X to the corresponding point distribution. Every finite distribution can be represented (though not uniquely) as an affine combination $\sum_{i=1,n} p_i\eta(x_i)$ of point distributions (meaning that $p_i \geq 0$ and $\sum_{i=1,n} p_i = 1$). The multiplication is given by:

$$\mu(\sum_{i=1,n} p_i\eta(\nu_i)) = \sum_{i=1,n} p_i\nu_i$$

and the (unique) strength by

$$\mathrm{st}(<x, \sum_{i=1,n} p_i\eta(y_i)>) = \sum_{i=1,n} p_i\eta(<x, y_i>)$$

We take one operation, a binary "fair-coin" probabilistic choice $+$ whose semantics is given by

$$\nu +_X \nu' = 1/2\nu + 1/2\nu'$$

Note the use of infix notation. A point worth noting is that while $\mathcal{D}$ supports this family, $\mathcal{D}_\omega(X)$ is not the free algebra over X corresponding to the equations true of $+_X$ as that only generates binary distributions.

Giving small step operational semantics is a little awkward. One might imagine using a relation $M \xrightarrow{p} N$ where p is 1 if no probabilistic choice is involved and 1/2 otherwise. However consideration of the example $\underline{0} + \underline{0}$ shows that some information needed to find the distribution of final values is lost this way. If one tries a big step semantics $M \stackrel{p}{\Rightarrow} N$ the problem becomes more acute: consider the two terms $(\underline{0} + \underline{1}) + (\underline{1} + \underline{1})$ and $(\underline{1} + \underline{0}) + (\underline{0} + \underline{0})$. One standard solution for the small step semantics is to record enough information on the path taken to resolve any ambiguities; in that case the small step semantics is essentially identical to the general one, where one would have both $M \xrightarrow{+_1} \underline{0}$ and $M \xrightarrow{+_2} \underline{0}$ for $M = \underline{0} + \underline{0}$.

There is also a collecting big step transition relation $M \Rightarrow_p \nu$ where M is closed and ν is a distribution over values. Here is an example rule (we now omit writing η)

$$\frac{M \Rightarrow_p \sum_i p_i(\lambda x : \sigma.M_i) \quad N \Rightarrow_p \sum_j q_j V_j \quad M_i[V_j/x] \Rightarrow_p \nu_{ij} \text{ (for all } i,j\text{)}}{MN \Rightarrow_p \sum_{ij} p_i q_j \nu_{ij}}$$

We can relate this to our general collecting big step operational semantics much as in the case of ordinary nondeterminism, defining a distribution $h(t)$ over values for every effect term t by

$$\begin{aligned} h(V) &= V \\ h(t + t') &= 1/2h(t) + 1/2h(t') \end{aligned}$$

and one has for any $M : \sigma$ that $M \Rightarrow_p \nu$ if, and only if, $\exists t. M \Rightarrow t \wedge \nu = h(t)$. Proceeding as before we now note that for any effect term $t : \sigma$

$$[\![t]\!](*) = \sum p_i [\![V_i]\!](*)$$

where $h(t) = \sum p_i V_i$, which yields an adequacy theorem.

Theorem 4. *Suppose that $M : \sigma$ and set $\sum p_i x_i = [\![M]\!](*)$. Then there are V_i such that $x_i = [\![V_i]\!](*)$ and $M \Rightarrow_p \sum p_i V_i$.*

Again, this takes a clearer form for any terms M of type ι:

$$[\![M]\!](*) = \sum p_i m_i \text{ iff } M \Rightarrow_p \sum p_i \underline{m}_i$$

5 PCF with Recursion

We add recursion to our language by a binding operator:

$$\mathrm{Rec}\,(f : \sigma \to \tau, x : \sigma.M)$$

with the typing rule

$$\frac{\Gamma, f : \sigma \to \tau, x : \sigma \vdash M : \tau}{\Gamma \vdash \mathrm{Rec}\,(f : \sigma \to \tau, x : \sigma.M) : \sigma \to \tau}$$

For the operational semantics, we regard this as a new kind of redex and add the rule

$$\mathrm{Rec}\,(f : \sigma \to \tau, x : \sigma.M) \to \lambda x : \sigma.M[\mathrm{Rec}\,(f : \sigma \to \tau, x : \sigma.M)/f]$$

This yields the small step operational semantics as before, with the analogous definitions of values and evaluation contexts (and with the analogous unique analysis of closed well-typed terms into one of the forms $E[R]$ or $E[a()]$). The possible transitions of a closed well-typed term can again be analysed into one of three mutually exclusive possibilities.

What differs from the previous situation is that terms need not terminate and so the small step semantics yields a tree of possibly infinite depth, branching finitely at labelled transitions but deterministic at unlabelled ones. So it is natural to consider *infinitary* effect values, that is, infinitary Σ-terms. The right tool for these is $CT_\Sigma(X)$, the free continuous Σ-algebra over a set X; it contains both finite partial and total elements as well as infinitary ones. (A *continuous Σ-algebra* is a dcppo equipped with continuous functions of appropriate arity for each operation symbol of Σ; a morphism of such algebras is a strict continuous function preserving the operations; CT_Σ is the left adjoint to the forgetful functor from the category of continuous Σ-algebras to that of sets—see below for the definitions of dcppo, etc.)

We may think of elements of this algebra as finite or infinite Σ-terms, with elements of X acting as extra constants. The finite ones are given by the grammar:

$$t \ ::= \ x \ \mid \ f(t_1, \ldots, t_n) \ \mid \ \Omega$$

with x ranging over X and with least element Ω. Every element t is is the limit of its finite approximants $t^{(k)}$ of level k, defined by

$$\begin{aligned} t^{(0)} &= \Omega \\ t^{(k+1)} &= \begin{cases} \Omega & (\text{if } t = \Omega) \\ x & (\text{if } t = x \in X) \\ f(t_1^{(k)}, \ldots, t_n^{(k)}) & (\text{if } t = f(t_1, \ldots, t_n) \text{ where } n = \mathrm{ar}_f) \end{cases} \end{aligned}$$

We therefore take the (possibly infinitary) effect values of type σ to be the elements of $CT_\Sigma(Val_\sigma)$ where Val_σ is the set of values of type σ, and wish to associate to every term $M : \sigma$ such a value $|\, M\,|$.

To this end we need to "factor out" the $\rightarrow$ moves, which we do by defining a medium step operational semantics for closed terms, by

$$M \Rightarrow V \text{ iff } M \rightarrow^* V$$

$$M \stackrel{f_i}{\Rightarrow} N \text{ iff } \exists L. M \rightarrow^* L \xrightarrow{f_i} N$$

$$M \Downarrow_a \ \text{ iff } \exists L. M \rightarrow^* L \downarrow_a$$

$$M \Uparrow \ \text{ iff there is an infinite sequence } M = M_0 \rightarrow M_1 \rightarrow \ldots \rightarrow M_n \rightarrow \ldots$$

The approximants of level k of $|\, M\,|$ (where $M : \sigma$) are now defined by $|\, M\,|^{(0)} = \Omega$ and

$$|\, M\,|^{(k+1)} = \begin{cases} V & (\text{if } M \Rightarrow V) \\ f(|\, M_1\,|^{(k)}, \ldots, |\, M_n\,|^{(k)}) & (\text{if } M \stackrel{f_i}{\Rightarrow} M_i \text{ for } i = 1, n, \text{ where } n = \mathrm{ar}_f) \\ a() & (\text{if } M \Downarrow_a) \\ \Omega & (\text{if } M \Uparrow) \end{cases}$$

Lemma 3. *Suppose that $M : \sigma$. Then $|M|$ satisfies the following equation.*

$$|M| = \begin{cases} V & (\textit{if } M = V) \\ |N| & (\textit{if } M \to N) \\ f(|N_1|, \ldots, |N_n|) & (\textit{if } M \xrightarrow{f_i} N_i \textit{ for } i = 1, n, \textit{ where } n = ar_f) \\ a() & (\textit{if } M = E[a()]) \end{cases}$$

This lemma can be strengthened to show that $|\cdot|$ is the least such function, under the pointwise ordering; indeed that could be taken as an alternative definition of $|\cdot|$.

For a collecting big step semantics one would naturally wish to give a system of rules defining the relation $M \Rightarrow t$ between closed well-typed terms of type σ and effect values in $CT_\Sigma(Val_\sigma)$ where

$$M \Rightarrow t \text{ iff } t = |M|$$

However it is not immediately clear how to think of a system of finitary rules as generating such a relation, let alone the precise form such rules should take. A related approach would be to define the evaluation function $|\cdot|$ as the least solution to a recursive equation defined by cases following the structure of M. Another idea is to define a nondeterministic relation between closed terms and finite effect values in $CT_\Sigma(Val_\sigma)$ such that the set of such values is directed and has lub the evaluation of the term; one can accomplish this by adding the axiom $M \Rightarrow \Omega$ and the rule

$$\mathrm{Rec}\,(f : \sigma \to \tau, x : \sigma.M) \Rightarrow \lambda x : \sigma.M[\mathrm{Rec}\,(f : \sigma \to \tau, x : \sigma.M)/f]$$

to the other rules; this idea appears in [13]. We do not enter further into these issues here. turn

As before there is a natural equational theory. The axioms are as before (but for the extended language) together with

$$\mathrm{Rec}\,(f : \sigma \to \tau, x : \sigma.M) = \lambda x : \sigma.M[\mathrm{Rec}\,(f : \sigma \to \tau, x : \sigma.M)/f]$$

This goes well with the medium step semantics

Lemma 4. *Suppose $M : \sigma$. Then*

1. *If $M \Rightarrow V$ then $\vdash M = V : \sigma$.*
2. *If $M \overset{f_i}{\Rightarrow} N_i$ for $i = 1, n$ (where $n = ar_f$) then $\vdash M = f(N_1, \ldots, N_n) : \sigma$*
3. *If $M \Downarrow_a$ then $\vdash M = a() : \sigma$*

This lemma is an immediate consequence of the evident corresponding lemma for small step operational semantics.

We now turn to denotational semantics, assuming the same categorical structure as before, but enriched with a suitable ordering structure to accommodate recursion. We use the cartesian closed category **Dcpo** of dcpos and continuous functions, and the closed category **Dcppo** of dcppos and strict continuous functions. (A *dcpo (complete partial order)* is a partial order with lubs of directed sets; a *continuous* function $f : P \to Q$ between such dcpos is a monotone function which preserves the lubs. A *dcppo (complete pointed partial order)* is a dcpo with a least element; a function between such dcppos is *strict* if it preserves the least elements.)

So we assume that $\mathbf{C}$ is **Dcpo**-enriched and so are T, $\mathbf{C}$-products and the Kleisli exponentials. We also assume that $\mathbf{C}_T$ is **Dcppo**-enriched and that the strength is strict, in the sense that, $\mathrm{st}_{x,y} \circ < f, \perp_{z,T(y)} > = \perp_{z,T(x \times y)}$ holds for any $f : z \to x$.

With this we can define $[\![\sigma]\!]$ as before and also the denotations

$$[\![M]\!] : [\![\Gamma]\!] \to T([\![\sigma]\!])$$

of terms $\Gamma \vdash M : \sigma$, except for the recursion construct. For this for a term $\Gamma, f : \sigma \to \tau, x : \sigma \vdash M : \tau$ we set

$$[\![\mathrm{Rec}\,(f : \sigma \to \tau, x : \sigma.M)]\!] = \mathrm{Y}(\lambda g : \mathbf{C}([\![\Gamma]\!], [\![\sigma \to \tau]\!]).\lambda_T([\![M]\!]) \circ < id_{[\![\Gamma]\!]}, g >)$$

where Y is the usual least fixed-point operator, $\mathrm{Y}(G) = \bigvee_{n \geq 0} G^n(\perp)$.

As before, the semantics is sound for the equational theory; combining this with Lemma 4 we obtain

Lemma 5. *Let $M : \sigma$ be a closed term. Then*

1. *If $M \Rightarrow V$ then $[\![M]\!] = [\![V]\!]$*
2. *If $M \overset{f_i}{\Rightarrow} N_i$ for $i = 1, n$ (where $n = ar_f$) then $[\![M]\!] = [\![f(N_1, \ldots, N_n)]\!]$*
3. *If $M \Downarrow_a$ then $[\![M]\!] = [\![a()]\!]$*

Now, since every $V : \sigma$ has a denotation $[\![V]\!]$ in the dcppo $\mathbf{C}_T(1, [\![\sigma]\!])$ and since, for every n-ary operation f, we have an n-ary continuous function on this dcppo induced by $f_{[\![\sigma]\!]}$, there is a unique continuous, strict Σ-homomorphism

$$[\![\ ]\!] : CT_\Sigma(Val_\sigma) \to \mathbf{C}_T(1, [\![\sigma]\!])$$

lifting $[\![\ ]\!] : Val_\sigma \to \mathbf{C}_T(1, [\![\sigma]\!])$. We are now in a position to state the main theorem of the paper:

Theorem 5. Adequacy for recursion. *For any term $M : \sigma$, $[\![M]\!] = [\![|M|]\!]$*

It is straightforward to prove half of this theorem, that $[\![M]\!] \geq [\![|M|]\!]$. That is an immediate consequence of the inequality

$$[\![M]\!] \geq [\![|M|^{(k)}]\!]$$

which can be proved by induction on k using Lemma 5. To prove the other half of the theorem we introduce a new language $\mathcal{A}$ in order to talk about approximants to terms of PCF with recursion. (It would be desirable to find an alternate proof using logical relations, e.g., as in [32].) The language $\mathcal{A}$ is obtained by adding two new families of constructs Ω_σ and $\mathrm{Rec}_n(f : \sigma \to \tau, x : \sigma.M)$ to the language of Section 2 with the typing rules:

$$\Gamma \vdash \Omega_\sigma : \sigma$$

and

$$\frac{\Gamma, f : \sigma \to \tau, x : \sigma \vdash M : \tau}{\Gamma \vdash \mathrm{Rec}_n(f : \sigma \to \tau, x : \sigma.M) : \sigma \to \tau}$$

The redexes and their transitions are defined as in Section 2, but with the addition of:

$$\mathrm{Rec}_{n+1}(f : \sigma \to \tau, x : \sigma.M) \to \lambda x : \sigma.M[\mathrm{Rec}_n(f : \sigma \to \tau, x : \sigma.M)/f]$$

and

$$\mathrm{Rec}_0(f : \sigma \to \tau, x : \sigma.M) \to \lambda x : \sigma.\Omega_\tau$$

Values, evaluation contexts and small step transitions are defined analogously to before. Every closed well-typed term is either a value or else can be analysed uniquely into one of the forms $E[R]$, $E[a()]$ or $E[\Omega_\tau]$; this yields an evident corresponding analysis of the transition possibilities, and we have

Lemma 6. *Every transition sequence terminates, either in a value or in a term of the form $E[\Omega_\tau]$.*

Proof. This can again be proved using a computability argument. The computability predicates are defined as in Theorem 1 except that we say that a closed term is computable iff every transition sequence from it terminates, either in a computable value or else in a term of the form $E[\Omega_\tau]$.

This allows us to define evaluation $|\, M \,|$. For any $M : \sigma$ we define $|\, M \,|$ in $CT_\Sigma(Val_\sigma)$ by

$$|\, M \,| = \begin{cases} V & (\text{if } M = V) \\ |\, N \,| & (\text{if } M \to N) \\ f(|\, N_1 \,|, \ldots, |\, N_n \,|) & (\text{if } M \xrightarrow{f_i} N_i, \text{ for } i = 1, n, \text{ where } n = \mathrm{ar}_f) \\ \Omega_\sigma & (\text{if } M = E[\Omega_\tau]) \end{cases}$$

We again have an equational theory. This is as in Section 2, together with

$$\mathrm{Rec}_{n+1}(f : \sigma \to \tau, x : \sigma.M) = \lambda x : \sigma.M[\mathrm{Rec}_n(f : \sigma \to \tau, x : \sigma.M)/f]$$

$$\mathrm{Rec}_0(f : \sigma \to \tau, x : \sigma.M) = \lambda x : \sigma.\Omega_\tau$$

$$E[\Omega_\tau] = \Omega_\sigma$$

As before $\vdash M = |\, M \,| : \sigma$ (if $M : \sigma$). For the semantics we take the same structure as that for PCF with recursion, defined as for the language of Section 2 and putting for the new constructs

$$[\![\mathrm{Rec}_n(f : \sigma \to \tau, x : \sigma.M)]\!] = \mathrm{Y}^{(n)}(\lambda g : \mathbf{C}([\![\Gamma]\!], [\![\sigma \to \tau]\!]).\lambda_T([\![M]\!]) \circ < id_{[\![\Gamma]\!]}, g >)$$

where $\mathrm{Y}^{(n)}(G) = G^n(\bot)$, and

$$[\![\Omega_\sigma]\!] = \bot$$

The equations are sound for this semantics and so we have that $[\![M]\!] = [\![\,|\, M \,|\,]\!]$, for $M : \sigma$.

With all this in hand we can now turn to the central relation

$$\overline{M} \prec M$$

of *approximation* between $\mathcal{A}$-terms and terms of PCF with recursion. This is the least relation closed under the common term-forming constructs such that $x \prec x$, $\Omega_\sigma \prec M$, and $\mathrm{Rec}_n(f : \sigma \to \tau, x : \sigma.\overline{M}) \prec \mathrm{Rec}(f : \sigma \to \tau, x : \sigma.M)$ if $\overline{M} \prec M$. It is straightforward to prove that this relation is closed under substitution in the sense that if $\overline{M} \prec M$ and $\overline{N} \prec N$ then $\overline{M}[\overline{N}/x] \prec M[N/x]$; it is equally straightforward to show that if $\overline{M} \prec M$ then $[\![\overline{M}]\!] \leq [\![M]\!]$ (where $\Gamma \vdash \overline{M} : \sigma$ and $\Gamma \vdash M : \sigma$).

The *nth-approximant* $M^{(n)}$ of M is obtained by replacing every occurrence of Rec by one of Rec_n. Clearly $M^{(n)} \prec M$ and if $\Gamma \vdash M : \sigma$ then $\Gamma \vdash M^{(n)} : \sigma$; further, for any term $\Gamma \vdash M : \sigma$, $[\![M^{(n)}]\!]$ is increasing and has lub $[\![M]\!]$ (as can be shown by a straightforward induction on n). For the next lemma we extend the definition of the $\prec$ relation to contexts, taking $[\,] \prec [\,]$.

Lemma 7. *Suppose that $\overline{M} \prec M$ where $\overline{M} : \sigma$ and $M : \sigma$. Then:*

1. *If $\overline{M}$ is a value, so is M.*
2. *If $\overline{M}$ has the form $\overline{E}[\overline{R}]$ then M has the form $E[R]$ where $\overline{E} \prec E$ and $\overline{R} \prec R$.*
3. *If $\overline{M}$ has the form $\overline{E}[a()]$ then M has the form $E[a()]$ where $\overline{E} \prec E$.*

Lemma 8. *Suppose that $\overline{R} \prec R$ where $\overline{R} : \sigma$ and $R : \sigma$. Then:*

1. *If $\overline{R} \to \overline{N}$ then, for some $N \succ \overline{N}$, $R \to N$.*
2. *If $\overline{R} \xrightarrow{f_i} \overline{N}_i$, for $i = 1, ar_f$ then for some $N_i \succ \overline{N}_i$, $R \xrightarrow{f_i} N_i$.*

Proposition 3. *Suppose that $\overline{M} \prec M$ where $\overline{M} : \sigma$ and $M : \sigma$. Then we have that $[\![|\,\overline{M}\,|]\!] \leq [\![|\,M\,|]\!]$.*

Proof. The proof proceeds by well-founded induction on the (union of) the transitions from $\overline{M}$ and cases on its form.

Suppose that it is a value. Then so is M, by Lemma 7 and so we have that $|\,\overline{M}\,| = \overline{M}$, $|\,M\,| = M$ and $[\![\overline{M}]\!] \leq [\![M]\!]$ (as $\overline{M} \prec M$). Taking these facts together yields the required conclusion.

Next, suppose that $\overline{M}$ has the form $\overline{E}[\overline{R}]$. Then by Lemma 7, M has the form $E[R]$ where $\overline{E} \prec E$ and $\overline{R} \prec R$. There are two subcases. In the first $\overline{R} \to \overline{N}$ and so, by Lemma 8, for some $N \succ \overline{N}$, $R \to N$. But then we have:

$$\begin{aligned} [\![|\,\overline{M}\,|]\!] &= [\![|\,\overline{E}[\overline{N}]\,|]\!] \text{ (by lemma 3)} \\ &\leq [\![|\,E[N]\,|]\!] \text{ (by induction hypothesis, as } \overline{E}[\overline{N}] \prec E[N]) \\ &= [\![|\,M\,|]\!] \text{ (by lemma 3)} \end{aligned}$$

The second subcase is where $\overline{R} \xrightarrow{f_i} \overline{N}_i$, for $i = 1, \mathrm{ar}_f$, and this is dealt with similarly.

Finally, the case where $\overline{M}$ has the form $\overline{E}[a()]$ is straightforward.

This proposition immediately yields the other half of Theorem 5 as we have

$$\begin{aligned}
[\![M]\!] &= \bigvee_{n\geq 0} [\![M^{(n)}]\!] \\
&= \bigvee_{n\geq 0} [\![\,|\,M^{(n)}\,|\,]\!] \\
&\leq [\![\,|\,M\,|\,]\!] \text{ (by the lemma)}
\end{aligned}$$

6 Examples and Recursion

We now take **C** to be **Dcpo** in our examples, and again consider two monads, for nondeterminism and probabilistic nondeterminism.

Nondeterminism

The original powerdomain construction was not defined for all dcppos; the free continuous algebra approach of [12, 1] yields a usable definition for all dcppos. For dcpos we take $T(P)$ to be the free dcppo over P which is also a continuous semilattice. This can be usefully analysed further. Let $\mathcal{S}(P)$ be the free continuous semilattice over a dcpo P; the existence of such algebras follows from the adjoint functor theorem, as in, e.g., [12, 1]. We write $\cup$ for the semilattice operation. This is a **Dcpo**-enriched monad and so has a unique continuous strength. One can show that if, in fact, P is a dcppo then so is $\mathcal{S}(P)$, and, indeed, it is the free continuous semilattice over P in **Dcppo**; thus it is the powerdomain in the sense of [12]. Now let $P_\bot$ be the free dcppo over a dcpo P. Then one has that $T(P)$ is $\mathcal{S}(P_\bot)$.

For a set X—considered as a flat dcpo—$T(X)$ is easy to describe directly. It consists of all subsets of $X_\bot$ which are either finite or are countable and contain $\bot$. Such sets are ordered by the Egli-Milner ordering $u \leq v$ if, and only if, either 1) $\bot \in X$ and $u \backslash \bot \subset v$ or 2) $\bot \notin X$ and $u = v$. A lub $\bigvee u_k$ of an increasing sequence contains $x \in X$ if, and only if, some u_k does, and $\bot$ if, and only if, every u_k does. Finally, $\cup_X$ is set-theoretic union, and $\eta_X(x)$ is the singleton $\{x\}$.

The small step semantics $\rightarrow_n$ is, as before, the union of the general $\rightarrow$ and the $\stackrel{\cup_i}{\rightarrow}$. We write $M \downarrow_n$ if there is no infinite transition sequence from a term $M : \sigma$; since $\rightarrow_n$ is finitary there is then, by König's lemma, a bound on the length of such transition sequences. Linking up to the evaluation $|\,M\,|$ one finds that $M \rightarrow_n^* V$ if, and only if, V occurs in $|\,M\,|$ (meaning that it occurs in some approximant) and $M \downarrow_n$ if, and only if, $|\,M\,|$ is finite and total (meaning that it contains no occurrence of any Ω_τ).

Here is an adequacy theorem for ground types (ι or o).

Theorem 6. *Let σ be a ground type and suppose that $M : \sigma$. Then $x \in [\![M]\!](*)$ if, and only if, one of the following hold*

1. $M \rightarrow_n^* V$ *and* $x = [\![V]\!](*)$
2. $M \not\downarrow_n$ *and* $x = \bot$

Proof. We have that $[\![M]\!](*) = [\![|\,M\,|]\!](*) = \bigvee_{n\geq 0}[\![|\,M\,|^{(n)}]\!](*)$

Suppose first that $x \neq \perp$. Then $x \in [\![|\,M\,|^{(n)}]\!](*)$ for some n and it follows that $\{x\} = [\![V]\!](*)$ for some V that occurs in $|\,M\,|^{(n)}$. For such a V we have that $M \rightarrow^*_n V$.

Suppose instead that $x = \perp$. Then $\perp$ is in every $[\![|\,M\,|^{(n)}]\!](*)$. Since no $[\![V]\!](*)$ is $\{\perp\}$ at ground types (actually at all types), it follows that no $|\,M\,|^{(n)}$ is total and hence that $M \not\downarrow_n$.

Essentially this result is in [31] and a stronger one for both call-by-name and call-by-value is in [10]; see [3, 11, 9] for work on call-by-name and nondeterminism. One can presumably also prove an adequacy theorem at other types, but the statement is more complex due to the need for closure operators in the set-theoretic representation of powerdomains, and there is some work to be done in extending the theory of bifinite domains to predomains (a dcpo P should be bifinite if, and only if, $P_\perp$ is); see, e.g., [1, 24, 25] for details on powerdomains. However part of the conjectured theorem is simple to state, that at any type σ, if $M : \sigma$ then $\perp \in [\![M]\!](*)$ if, and only if, $M \downarrow_n$. As well as the above *convex* powerdomain, one can also prove appropriate adequacy theorems for the *upper* (Smyth) and *lower* (Hoare) powerdomains.

Probabilistic Nondeterminism

We take T to be $\mathcal{V}$, where $\mathcal{V}(P)$ is the dcpo of all evaluations on P. This functor and its strong monad structure is discussed in [14], and see, e.g., [15]; we just note that $\eta_P(x)$ is the singleton evaluation. The semantics of probabilistic choice $+$ is again an affine combination, this time of evaluations

$$\nu +_X \nu' = 1/2\nu + 1/2\nu'$$

$\mathcal{V}(P)$ is a dcppo which is a continuous Σ-algebra, and for any $f : P \rightarrow \mathcal{V}(Q)$, $f^\dagger$ is a strict, continuous Σ-algebra morphism.

As discussed before we can take the small step operational semantics to be the general one (for this case), and the question is rather one of interpretation. Let us write w to range over words in the alphabet $\{+_1, +_2\}$. We write $M \stackrel{w}{\Rightarrow} N$ to mean that w describes a sequence of medium step transitions from M to N. Now for a term $M : \sigma$ and value $V : \sigma$ set

$$\mathrm{Prob}(M, V) = \sum\{2^{-|w|} \mid M \stackrel{w}{\Rightarrow} V\}$$

Define a function θ from closed terms of type σ to the closed interval $[0, 1]$ by $\theta(M) = \mathrm{Prob}(M, V)$. Then θ obeys the equation

$$\theta(M) = \begin{cases} 1 & (\text{if } M \Rightarrow V) \\ 0 & (\text{if } M \Rightarrow V' \neq V) \\ 1/2\theta(N_1) + 1/2\theta(N_2) & (\text{if } M \stackrel{+_i}{\Rightarrow} N_i \ (i = 1, 2)) \\ 0 & (\text{if } M \not\Downarrow) \end{cases} \tag{1}$$

This equation determines θ uniquely (use the Banach fixed-point theorem with the metric $d(\theta,\theta') = \sup_M |\theta(M) - \theta'(M)|$).

The next lemma relates the small step semantics of a term to its evaluation. Let h be the unique strict continuous Σ-algebra morphism from $CT_\Sigma(Val_\sigma)$ to $\mathcal{V}(Val_\sigma)$. The statement of the following lemma makes implicit use of the fact there there are at most countably infinitely many values of a given type, and that weighted countably infinite sums of evaluations exist (being defined pointwise).

Lemma 9. *Suppose that $M : \sigma$. Then*

$$h(|M|) = \sum_{V:\sigma} Prob(M,V)\eta(V)$$

Proof. It is straightforward to see that $h(|\cdot|)$ obeys the following equation (use the equation given for $|\cdot|$ in the previous section)

$$h(|M|) = \begin{cases} \eta(V') & (\text{if } M \Rightarrow V') \\ 1/2h(|N_1|) + 1/2h(|N_2|) & (\text{if } M \overset{+_i}{\Rightarrow} N_i\ (i=1,2)) \\ \perp & (\text{if } M \not\Rightarrow) \end{cases}$$

It follows that, for any $V : \sigma$, $h(|\cdot|)(V)$ satisfies equation 1, and so as this equation has a unique solution, viz $\mathrm{Prob}(\cdot, V)$, the conclusion follows.

This enables us to prove an adequacy theorem at all types.

Theorem 7. *Suppose that $M : \sigma$. Then*

$$[\![M]\!](*) = \sum_{V:\sigma} Prob(M,V)[\![V]\!](*)$$

Proof. Both $[\![\cdot]\!](*)$ and $([\![\cdot]\!](*))^\dagger \circ h$ are strict continuous Σ-homomorphisms from $CT_\Sigma(Val_\sigma)$ to $\mathcal{V}([\![\sigma]\!])$, and both extend $[\![\cdot]\!](*) : Val_\sigma \to \mathcal{V}([\![\sigma]\!])$. They are therefore equal. We may then calculate:

$$\begin{aligned} [\![M]\!] &= [\![|M|]\!] \text{ (by Theorem 5)} \\ &= ([\![\cdot]\!](*))^\dagger)(h(|M|)) \\ &= ([\![\cdot]\!](*))^\dagger)(\sum_{V:\sigma} \mathrm{Prob}(M,V)\eta(V)) \text{ (by Lemma 9)} \\ &= \sum_{V:\sigma} \mathrm{Prob}(M,V)([\![\cdot]\!](*))^\dagger)(\eta(V)) \\ &= \sum_{V:\sigma} \mathrm{Prob}(M,V)[\![V]\!](*) \end{aligned}$$

An adequacy theorem for FPC with probabilistic choice was already proved in [13] (FPC can be viewed as an extension of our PCF with recursive types); for work on call-by-name and probabilistic nondeterminism see [6].

7 Conclusions

It is interesting to work out other examples. Printing provides one example, where one has a unary operation print_a for each symbol a of an alphabet A, and for **Set** one can take the (noncommutative) monad $T(X) = A^* \times X$, which is, in fact, the free Σ-algebra over X; for **Dcpo** one would also allow the possibility of infinite printing, and use $CT_\Sigma(P)$, the free continuous Σ-algebra over P. Here the general operational semantics is very much the same as what one would write anyway and it is straightforward to read off adequacy results for printing from the general theorems. An example worth some investigation is the combination $\mathcal{F}^+(\mathcal{D}_\omega(X))$ of probabilistic and ordinary nondeterminism; there is natural distributive law $\lambda : \mathcal{D}_\omega(\mathcal{F}^+(X)) \to \mathcal{F}^+(\mathcal{D}_\omega(X))$which makes this a monad; this way to combine the two forms of nondeterminism is used in a domain-theoretic context in [4]—modulo actions—and mentioned in [20] where an interesting idea of restricting to affine sets of evaluations is advocated.

In so far as we are successful with such examples, the question of how to treat other monads and their operations is the more pressing; exceptions, state and continuations all come immediately to mind. Possibly relevant here is the translational approach to defining operations in [5], but adapted to λ_c rather than the metalanguage; the idea would be to recover operational semantics via the translations. Ultimately, we would hope to incorporate the treatment of operational semantics into a modular approach to computational effects, e.g., along the lines of [26, 28, 29].

An obvious question is to consider language variations, such as an extension with recursive types or call-by-name; for the latter it would be preferable to use a framework incorporating both parameter-calling mechanisms, such as Levy's CBPV [18]. More intriguingly, one would wish to reconcile this work with the co-algebraic treatment of operational semantics in [33] with its use of behaviour functors and co-monads contrasting with our use of monads.

References

1. S. Abramsky and A. Jung, Domain Theory, in *Handbook of Logic in Computer Science* (eds. S. Abramsky, D.M. Gabbay and T. S. E. Maibaum), Vol. 3, Semantic Structures, Oxford: Clarendon Press, 1994.
2. S. O. Anderson and A. J. Power, A Representable Approach to Finite Nondeterminism, *Theoret. Comput. Sci.*, Vol. 177, No. 1, pp. 3–25, 1997.
3. E. Astesiano and G. Costa, Nondeterminism and Fully Abstract Models, in Informatique Théorique et Applications, Vol. 14, No. 4, pp. 323–347, 1980.
4. C. Baier and M. Kwiatkowska, *Domain Equations for Probabilistic Processes*, MSCS, Vol. 10, No. 6, pp. 665–717, 2000.
5. P. Cenciarelli and E. Moggi, A Syntactic Approach to Modularity in Denotational Semantics, in *Proc. 5th. Biennial Meeting on Category Theory and Computer Science*, LNCS, Berlin: Springer-Verlag, 1993.
6. V. Danos and R. Harmer, Probabilistic Game Semantics, in *Proc. 15th LICS*, pp. 204–213, Washington: IEEE Press, 2000.

7. M. Felleisen, and D. P. Friedman, Control Operators, the SECD-machine, and the Lambda-Calculus, in *Formal Description of Programming Concepts III* (ed. M. Wirsing), pp. 193–217, Amsterdam: Elsevier, 1986.
8. I. Guessarian, *Algebraic Semantics*, LNCS, Vol. 99, Berlin: Springer-Verlag, 1981.
9. R. Harmer and G. McCusker, A Fully Abstract Game Semantics for Finite Nondeterminism, in *Proc. 14th LICS*, pp. 422–430, Washington: IEEE Press, 1999.
10. M. C. B. Hennessy, The Semantics of Call-By-Value and Call-By-Name in a Nondeterministic Environment, in *SIAM J. Comput.*, Vol. 9, No. 1, pp. 67–84, 1980.
11. M. C. B. Hennessy and E. A. Ashcroft, A Mathematical Semantics for a Nondeterministic Typed Lambda-Calculus, in TCS Vol. 11, pp. 227-245, 1980.
12. M. C. B. Hennessy and G. Plotkin, Full Abstraction for a Simple Parallel Programming Language, in *Proc. 8th MFCS* (ed. J. Beĉvář), Olomouc, Czechoslovakia, LNCS, Vol. 74, pp. 108–120, Berlin: Springer-Verlag, 1979.
13. C. Jones, *Probabilistic Non-Determinism*, Ph.D. Thesis, University of Edinburgh, Report ECS-LFCS-90-105, 1990.
14. C. Jones and G. D. Plotkin, *A Probabilistic Powerdomain of Evaluations*, in *Proc. 4th LICS*, Asilomar, pp. 186–195, Washington: IEEE Press, 1989.
15. A. Jung and R. Tix, The Troublesome Probabilistic Powerdomain, in *Proc. Third COMPROX Workshop*, ENTCS, Vol. 13, Amsterdam: Elsevier, 1998.
16. G. M. Kelly, *Basic Concepts of Enriched Category Theory*, Cambridge: CUP, 1982.
17. Y. Kinoshita and A. J. Power, *Data Refinement for Call by Value Languages*, submitted, 2000.
18. P. B. Levy, Call-by-Push-Value: A Subsuming Paradigm, in *Proc. TLCA '99* (ed. J.-Y. Girard), LNCS, Vol. 1581, pp. 228-242, Berlin: Springer-Verlag, 1999.
19. J. Mezei and J. B. Wright, *Algebraic Automata and Context Free Sets*, in Information and Control, Vol. 11, pp. 3–29, 1967.
20. M. W. Mislove, Nondeterminism and Probabilistic Choice: Obeying the Laws, in *Proc. CONCUR 2000* (ed. C. Palamidessi), LNCS, Vol. 1877, pp. 350–364, Berlin: Springer-Verlag, 2000.
21. E. Moggi, Computational Lambda-Calculus and Monads, in *Proc. LICS '89*, pp. 14–23, Washington: IEEE Press, 1989.
22. E. Moggi, *An Abstract View of Programming Languages*, University of Edinburgh, Report ECS-LFCS-90-113, 1989.
23. E. Moggi, *Notions of Computation and Monads*, Inf. and Comp., Vol. 93, No. 1, pp. 55–92, 1991.
24. G. D. Plotkin, *A Powerdomain Construction*, SIAM J. Comput. Vol. 5, No. 3, pp. 452–487, 1976.
25. G. D. Plotkin, *Domains*, (http://www.dcs.ed.ac.uk/home/gdp/), 1983.
26. A. J. Power, Modularity in Denotational Semantics, in *Proc. MFPS XIII* (eds. S. Brookes and M. Mislove), ENTCS, Vol. 6, Amsterdam: Elsevier, 1997.
27. A. J. Power, Enriched Lawvere Theories, in *Lambek Festschrift* (eds. M. Barr, P. Scott and R. Seely), TAC, Vol. 7, pp. 83–93, 2000.
28. A. J. Power and E. P. Robinson, Modularity and Dyads, in *Proc. MFPS XV* (eds. S. Brookes, A. Jung, M. Mislove and A. Scedrov), ENTCS Vol. 20, Amsterdam: Elsevier, 1999.
29. A. J. Power and G. Rosolini, A Modular Approach to Denotational Semantics, in *Proc. ICALP '98* (eds. K. G. Larsen, S. Skyum and G. Winskel), LNCS, Vol. 1443, pp. 351–362 Berlin: Springer-Verlag, 1998.
30. A. J. Power and H. Thielecke, Closed *Freyd*- and κ-categories, in *Proc. 26th. ICALP* (eds. J. Wiedermann and P. van Emde Boas and M. Nielsen), LNCS, Vol. 1644, pp. 625–634, Berlin: Springer-Verlag, 1999.

31. K. Sieber, Call-by-Value and Nondeterminism, in *Proc. TLCA '93* (eds. M. Bezem and J. F. Groote), LNCS, Vol. 664, pp. 376–390, Berlin: Springer-Verlag, 1993.
32. A. Simpson, Computational Adequacy in an Elementary Topos, in *Proc. CSL '98*, LNCS, Vol. 1584, pp. 323–342, Berlin: Springer-Verlag, 1999.
33. D. Turi and G. D. Plotkin, Towards a Mathematical Operational Semantics, in *Proc. LICS 97*, pp. 268–279, Washington: IEEE Press, 1997.
34. G. Winskel, *The Formal Semantics of Programming Languages*, Cambridge: MIT Press, 1993.