The semantic marriage of monads and effects

Extended abstract

Dominic Orchard  Tomas Petricek  Alan Mycroft
Computer Laboratory, University of Cambridge
{firstname.lastname}@cl.cam.ac.uk

Abstract
Wadler and Thiemann unified type-and-effect systems with monadic semantics via a syntactic correspondence and soundness results with respect to an operational semantics. They conjecture that a general, “coherent” denotational semantics can be given to unify effect systems with a monadic-style semantics. We provide such a semantics based on the novel structure of an indexed monad, which we introduce. We redefine the semantics of Moggi’s computational λ-calculus in terms of (strong) indexed monads which gives a one-to-one correspondence between indices of the denotations and the effect annotations of traditional effect systems. Dually, this approach yields indexed comonads which gives a unified semantics and effect system to contextual notions of effect (called coeffects), which we have previously described [9].

Previously, Wadler and Thiemann established a syntactic correspondence between type-and-effect systems and the monadic semantics approach by annotating monadic type constructors with coeffects of denotations have exactly the same structure as the effects of regular monads when ⊗ is a strict monoidal operation of T match the shape of the regular monad operation. Furthermore, the standard associativity and unitality conditions of the lax monoidal functor give coherence conditions to η1 and μF,G which are analogous to the regular monad laws, but with added indices, e.g., μ1,G ◦ (η1)TG = id TG.

Example (Indexed exponent/reader monad) Given the monoid \( (P(X), \cup, \emptyset) \) (for some set \( X \)), the indexed family of Set endofunctions where \( TXA = X \Rightarrow A \) (with \( \Rightarrow \) denoting exponents) and \( TXf = \lambda k, f \circ k \) is an indexed monad with:
\[
\eta0a = \lambda x.a \\
\mu1,Gk = \lambda x. (k(x - (G-F)))(x - (F-G))
\]
where \( x : F \cup G \) and \( k : F \Rightarrow (G \Rightarrow A) \) thus \( k \) takes two arguments, the \( F \)-only subset of \( x \) (written \( x - (G-F) \)) and the \( G \)-only subset of \( x \) (written \( x - (F-G) \)) where \( \neg \) is set difference.

The indexed reader monad models the composition of computations with implicit parameters, where the required implicit parameters of subcomputations are combined in their composition. This provides a more refined model to the notion of implicitly parameterised computations than the traditional reader monad, where implicit parameters are uniform throughout a computation and its subcomputations.

Relating indexed monads and monads Indexed monads collapse to regular monads when \( I \) is a single-object monoidal category. Thus, indexed monads generalise monads.

Note that indexed monads are not indexed families of monads. That is, for all indices \( F \in \text{obj}(I) \) then \( TF \) may not be a monad.

An indexed monadic semantics for \( \lambda \). We extend indexed monads to strong indexed monads, with an indexed strength operation (and analogous laws to usual monadic strength):
\[
(\tau F)_A : (A \times TFB) \Rightarrow TF(A \times B)
\]
We replay Moggi’s categorical semantics for the computational λ-calculus (\( \lambda_c \)) [5], replacing the regular strong monad operations with the analogous operations of an indexed strong monad. This provides an indexed semantics. For example, the semantics of λ-abstraction becomes the following (where we write the parameter to \( T \) as a subscript for notational clarity below):
\[
[\Gamma, x : \sigma \vdash e : \tau] = g : [\Gamma][\sigma \vdash x] \Rightarrow T_F \tau
\]
\[
[\Gamma ; \lambda x : \sigma.e : \sigma \Rightarrow \tau] = \eta_1 \circ (\Lambda g) : [\Gamma] \Rightarrow T_1(\sigma \Rightarrow \tau)
\]
(where for \( g : A \times B \rightarrow C, \Lambda g : A \rightarrow (B \Rightarrow C) \)).

Coherent semantics In this indexed monadic semantics, the indices of denotations have exactly the same structure as the effect annotations of a traditional effect system (with judgments \( \Gamma \vdash e : \tau, F \) for an expression \( e \) with effects \( F \)).

We unify effect systems with indexed monadic semantics, so that \( [\Gamma, e : \tau, F] : [\Gamma] \Rightarrow T_F[\tau] \), taking \( \text{obj}(I) \) as the effect

Indexed monads Indexed monads comprise a functor
\( T : I \rightarrow [\mathcal{C}, \mathcal{C}] \)
\footnote{Note this differs to Johnstone’s notion of indexed monad in the context of topos theory, the indexed monads seen in the work of McBride [4], and parameterised monads by Atkey [1].}
(i.e., an indexed family of endofunctors) where \( I \) is a strict monoidal category \( (I, \otimes, 1) \) and \( T \) is a lax monoidal functor, mapping the strict monoidal structure on \( I \) to the strict monoid of endofunctor composition \( ([\mathcal{C}, \mathcal{C}], \circ, I_C) \).

The operations of the lax monoidal structure are thus:
\[
\eta : 1 \Rightarrow T1 \\
\mu : TF \circ TG \Rightarrow (T(F \otimes G))
\]
sets of a traditional effect system, with the strict monoidal structure
on $\mathcal{I}$ provided by the effect lattice, with $1 = \bot$ and $\otimes = \sqcup$, and
morphisms $f : X \to Y$ in $\mathcal{I}$ iff $X \sqsubseteq Y$ in the effect lattice.

Pleasingly, the usual equational theory for $\lambda$, (such as $\beta$-equality
for values) follows directly from the strong indexed monad axioms.

The morphism mapping of $T$ defines natural transformations
$\iota_{X,Y} : TX \to TY$ when $X \sqsubseteq Y$ which provides a semantics
to sub-effecting:

$$\begin{align*}
\iota_{X,Y} & : TX \to TY & <\text{sub-effecting}> \\
\iota_{X,Y} & = div_{\lambda}: T \Rightarrow T & <\text{product}}
$$

For a particular notion of effect, the indexed strong monad can be
defined such that the propagation of effect annotations in an effect
system maps directly to the semantic properties of effects. For example,
for memory effects the functor can be made more precise
with respect to the effect, e.g. $T\{\text{read } \rho \mapsto \tau\} A = \tau \to A$ and
$T\{\text{write } \rho \mapsto \tau\} A = A \times \tau$ (note: the latter is not itself a monad)

Therefore strong indexed monads neatly unify a (categorical)
semantics of effects with traditional effect systems. The indexed monad
structure arises simply from the standard category theory
construction of lax monoidal functors, where $T$ preserves the strict
monoidal structure of $\mathcal{I}$ in $[\mathcal{C}, \mathcal{C}]$. Crucially, indexed monads are
not an indexed family of monads (contrasting with Wadler and
Thiemann’s original conjecture).

**In context** We argue our approach provides an intermediate so-
ution between the traditional monadic approach (which does not
couple annotations of an effect system to semantics) and algebraic
effect theories (see, e.g., Kammar and Plotkin [3]).

Our approach differs somewhat to Atkey’s parameterised monads,
defined for $T : S \times S^\to \to [\mathcal{C}, \mathcal{C}]$. Our indexed monad structure
has a more systematic derivation, arising from the strict monoidal
preservation of the lax monoidal functor. This technique can be
applied to derive coherent semantic structures/effect system pairs for
other notions of computation.

Effect systems traditionally define effect annotations in terms of
sets with composition via set union [2]. This has the additional
property that combining effect annotations is symmetric (due to
commutativity of union). The more general structure of a monoid
here, also used by Nielson and Nielson [6], provides an opportunity
for generating effect information that records the order of effects.

**Extending the approach to other notions** We apply the same

tecnhique used to derive indexed monads to give richer effect sys-
tems/semantics in two ways.

1. A strict colax monoidal functor $D : \mathcal{I} \to [\mathcal{C}, \mathcal{C}]$ gives rise to the
dual notion of indexed comonads, which we previously
shown to provide the notion of a coeffect system (analysing con-
textual requirements) and a semantics for contextual program
effects [7, 9].

For a monoid $\langle \mathcal{I}, \otimes, 1 \rangle$, indexed comonads have the colax op-
erations:

$$\varepsilon_1 : D1 \to I_c \quad \delta_{F,G} : D(F \otimes G) \to DF \otimes DG$$

Interestingly, indexed comonads seem much more useful than
comonads since they utilize the usual shape preservation prop-
erty of comonads.

**Example** (Indexed partiality comonad) For the boolean con-
junction monoid $\langle \{t, f\}, \wedge, \top \rangle$, the following indexed family of
endofunctors is an indexed comonad:

$$D f A = 1 \quad D t A = A \quad D ff = !A \quad D tf = f$$

with $\varepsilon_A a = a$, $\delta_{t,t} a = a$, and $\delta_{X,Y} 1 = 1$ when $X = f$
and/or $Y = f$. The indexed partiality comonad is essentially

2. Nielson and Nielson defined a more general effect system with a
richer algebraic effect structure, separating the traditional ap-
proach of an effect lattice into operations for sequential com-
position, alternation, and fixed-points [6]. Relatedly on the
semantic side, the structure of a joinad has been proposed to give
the semantics of sequencing, alternation and parallelism in an
effectual language [8], adding additional monoidal structures to
a monad. Similarly to indexed monads, joinads can be gener-
alised to indexed joinads, giving a correspondence between the
richer effect systems of Nielson and Nielson and a joinad-based
semantics. This is future work.

**References**


[4] McBride, C. Functional pearl: Kleisli arrows of outrageous for-


