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

arXiv:1401.5391v1 [cs.PL] 21 Jan 2014

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 the effect sets of the type-and-effect system [10]. They established soundness results between the effect system and an operational semantics, and conjectured a "coherent semantics" of effects and monads in a denotational style. One suggestion was to associate to each effect set σ a different monad T^{σ}.

We take a different approach to a coherent semantics, unifying effect systems with a monadic-style semantics in terms of the novel notion of *indexed monads*, which generalises monads.[1]

Indexed monads Indexed monads comprise a functor

$$\mathsf{T} : \mathcal{I} \to [\mathcal{C}, \mathcal{C}]$$

(*i.e.*, an indexed family of endofunctors) where $\mathcal{I}$ is a strict monoidal category $(\mathcal{I}, \otimes, 1)$ and T is a *lax monoidal functor*, mapping the strict monoidal structure on $\mathcal{I}$ to the strict monoid of endofunctor composition $([\mathcal{C}, \mathcal{C}], \circ, I_C)$.

The operations of the lax monoidal structure are thus:

$$\eta_1 : I_{\mathcal{C}} \dot{\to} \mathsf{T}1 \qquad \mu_{F,G} : \mathsf{T}F \circ \mathsf{T}G \dot{\to} \mathsf{T}(F \otimes G)$$

[1] 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].

These lax monoidal operations of T match the shape of the regular monad operations. Furthermore, the standard associativity and unitality conditions of the lax monoidal functor give coherence conditions to η_1 and $\mu_{F,G}$ which are analogous to the regular monad laws, but with added indices, *e.g.*, $\mu_{1,G} \circ (\eta_1)_{\mathsf{T}G} = id_{\mathsf{T}G}$.

Example (Indexed exponent/reader monad) Given the monoid $(\mathcal{P}(X), \cup, \emptyset)$ (for some set X), the indexed family of $\mathbf{Set}$ endofunctors where $\mathsf{T}XA = X \Rightarrow A$ (with $\Rightarrow$ denoting exponents) and $\mathsf{T}Xf = \lambda k.f \circ k$, is an indexed monad with:

$$\eta_{\emptyset} a = \lambda x.a$$
$$\mu_{F,G} k = \lambda x.(k\,(x - (G - F)))\,(x - (F - G))$$

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 $(-)$ 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 $\mathcal{I}$ 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 $F \in obj(\mathcal{I})$ then $\mathsf{T}F$ may not be a monad.

An indexed monadic semantics for λ_c We extend indexed monads to *strong* indexed monads, with an indexed strength operation (and analogous laws to usual monadic strength):

$$(\tau_F)_{A,B} : (A \times \mathsf{T}FB) \to \mathsf{T}F(A \times B)$$

We replay Moggi's categorical semantics for the computational λ-calculus (λ_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):

$$\frac{[\![\Gamma, x : \sigma \vdash e : \tau]\!] = g : [\![\Gamma]\!] \times [\![\sigma]\!] \to \mathsf{T}_F\, \tau}{[\![\Gamma \vdash \lambda x : \sigma.e : \sigma \to \tau]\!] = \eta_1 \circ (\Lambda g) : [\![\Gamma]\!] \to \mathsf{T}_1\, (\sigma \Rightarrow \mathsf{T}_F\, \tau)}$$

(where for $g : A \times B \to C$, $\Lambda g : A \to (B \Rightarrow C)$).

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 $[\![\Gamma \vdash e : \tau, F]\!] : [\![\Gamma]\!] \to \mathsf{T}_F\, [\![\tau]\!]$, taking $obj(\mathcal{I})$ as the effect 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 λ_c (such as β-equality for values) follows directly from the strong indexed monad axioms.

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

$$\text{(sub)}\frac{[\![\Gamma \vdash e : \tau, F']\!] = g : [\![\Gamma]\!] \to \mathsf{T}_{F'}[\![\tau]\!] \qquad F' \sqsubseteq F}{[\![\Gamma \vdash e : \tau, F]\!] = \iota_{F',F} \circ g : [\![\Gamma]\!] \to \mathsf{T}_F[\![\tau]\!]}$$

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 propagation of effects. For example, for memory effects the functor can be made more *precise* with respect to the effect, *e.g.*, $\mathsf{T}\{\mathsf{read}\,\rho : \tau\}\, A = \tau \to A$ and $\mathsf{T}\{\mathsf{write}\,\rho : \tau\}\, A = A \times \tau$ (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 solution 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 $T : \mathcal{S} \times \mathcal{S}^{\mathsf{op}} \to [\mathcal{C}, \mathcal{C}]$. 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 technique used to derive indexed monads to give richer effect systems/semantics in two ways.

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

   For a monoid $(\mathcal{I}, \otimes, 1)$, indexed comonads have the colax operations:

   $$\varepsilon_1 : \mathsf{D}1 \dot{\to} I_{\mathcal{C}} \qquad \delta_{F,G} : \mathsf{D}(F \otimes G) \dot{\to} \mathsf{D}F \circ \mathsf{D}G$$

   Interestingly, indexed comonads seem much more useful than comonads since they relax the usual *shape preservation* property of comonads.

   Example (Indexed partiality comonad) For the boolean conjunction monoid $(\{\mathbf{f}, \mathbf{t}\}, \wedge, \mathbf{t})$, the following indexed family of endofunctors is an indexed comonad:

   $$\begin{array}{ll} \mathsf{D}\,\mathbf{f}A = 1 & \mathsf{D}\,\mathbf{t}A = A \\ \mathsf{D}\,\mathbf{f}f = !_A & \mathsf{D}\,\mathbf{t}f = f \end{array}$$

   with $\varepsilon_{\mathbf{t}} a = a$, $\delta_{\mathbf{t},\mathbf{t}} a = a$, and $\delta_{X,Y} a = 1$ when $X = \mathbf{f}$ and/or $Y = \mathbf{f}$. The indexed partiality comonad is essentially a dependently-typed partiality construction $\mathsf{D}A = 1 + A$. Note however, that $\mathsf{D}A = 1 + A$ is not a comonad since $\varepsilon : \mathsf{D}A \to A$ is not well defined. The indexed partiality comonad encodes the notion of a partial context/input to a computation, and has been previously shown to give a simple liveness analysis coupled with a semantics that embeds the notion of dead-code elimination [9].

   Coeffect systems differ to effect systems in their treatment of λ-abstraction where, for coeffect judgments $\Gamma ? F \vdash e : \tau$ (meaning expression e has contextual requirements F):

   $$\text{(abs)}\frac{\Gamma, x : \sigma ? F \vee F' \vdash e : \tau}{\Gamma ? F \vdash \lambda x.e : \sigma \xrightarrow{F'} \tau}$$

   Reading this rule top-down, the coeffects of the function body are split between *immediate* contextual requirements and *latent* requirements (written $\xrightarrow{F'}$). Thus, with respect to contextual requirements, λ-abstraction is not "pure" as it is for effects. In an indexed semantics unifying a coeffect system with an indexed comonad, the semantics of λ-abstraction requires the additional structure of an *indexed (semi-)monoidal comonad* with the operation:

   $$(m_{F,G})_{A,B} : \mathsf{D}_F A \times \mathsf{D}_G B \to \mathsf{D}_{F \vee G}(A \times B)$$

   where $\vee$ is an associative binary operation over $\mathcal{I}$.

2. Nielson and Nielson defined a more general effect system with a richer algebraic effect structure, separating the traditional approach of an effect lattice into operations for sequential composition, 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 effectful language [8], adding additional monoidal structures to a monad. Similarly to indexed monads, joinads can be generalised 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

[1] ATKEY, R. Parameterised notions of computation. In *Proceedings of the Workshop on Mathematically Structured Functional Programming* (2006), Cambridge Univ Press.

[2] GIFFORD, D. K., AND LUCASSEN, J. M. Integrating functional and imperative programming. In *Proceedings of Conference on LISP and func. prog.* (1986), LFP '86.

[3] KAMMAR, O., AND PLOTKIN, G. D. Algebraic foundations for effect-dependent optimisations. In *ACM SIGPLAN Notices* (2012), vol. 47, ACM, pp. 349–360.

[4] MCBRIDE, C. Functional pearl: Kleisli arrows of outrageous fortune. *Journal of Functional Programming (to appear).*

[5] MOGGI, E. Computational lambda-calculus and monads. In *Logic in Computer Science, 1989. LICS'89, Proceedings., Fourth Annual Symposium on* (1989), IEEE, pp. 14–23.

[6] NIELSON, F., AND NIELSON, H. Type and effect systems. *Correct System Design* (1999), 114–136.

[7] ORCHARD, D. Programming contextual computations. PhD thesis, University of Cambridge, 2013. (To appear).

[8] PETRICEK, T., MYCROFT, A., AND SYME, D. Extending Monads with Pattern Matching. In *Proceedings of Haskell Symposium* (2011), Haskell 2011.

[9] PETRICEK, T., ORCHARD, D. A., AND MYCROFT, A. Coeffects: Unified static analysis of context-dependence. In *ICALP (2)* (2013), pp. 385–397.

[10] WADLER, P., AND THIEMANN, P. The marriage of effects and monads. *ACM Trans. Comput. Logic 4* (January 2003), 1–32.