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[3].

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 Tσ.

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[6].

Indexed monads

Indexed monads comprise a functor

\[ T : I \to [C, C] \]

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

The operations of the lax monoidal structure are thus:

\[ \eta_I : I_C \to T1 \quad \mu_{F,G} : TF \circ TG \to T(F \otimes G) \]

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 \(\eta_I\) and \(\mu_{F,G}\) which are analogous to the regular monad laws, but with added indices, e.g., \(\mu_{F,G} \circ (\eta_I)_T G = idTG\).

Example

(Indexed exponent/reader monad) Given the monoid \((P(X), \cup, \emptyset)\) (for some set X), the indexed family of Set endofunctors where \(TXA = X \Rightarrow A\) (with \(\Rightarrow\) denoting exponents) and \(TXF = \lambda k . f \circ k\), k, is an indexed monad with:

\[ \eta_0 a = \lambda x.a \]

\[ \mu_{F,G} k = \lambda x . (k (x - (G - F))) (x - (F - G)) \]

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 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 \(F \in obj(I)\) 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):

\[ (\tau_F)_{A,B} : (A \times TF) \to TF(A \times B) \]

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):

\[ [\Gamma, \lambda x : \sigma + e : \tau] = g : [\Gamma] \times [\sigma] \to TF \tau \]

\[ [\Gamma + \lambda x : \sigma + e : \tau] = \eta_I \circ (\Lambda g) : [\Gamma] \to T1 (\sigma \Rightarrow TF \tau) \]

(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 TF [\tau]\], taking \(obj(I)\) as the effect

\[ [\Gamma, x : \sigma + e : \tau] = g : [\Gamma] \times [\sigma] \to TF \tau \]

\[ [\Gamma + \lambda x : \sigma + e : \tau] = \eta_I \circ (\Lambda g) : [\Gamma] \to T1 (\sigma \Rightarrow TF \tau) \]

(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 \([\Gamma] \vdash e : \tau, F] : [\Gamma] \to TF [\tau]\), taking \(obj(I)\) as the effect

\[ [\Gamma, x : \sigma + e : \tau] = g : [\Gamma] \times [\sigma] \to TF \tau \]

\[ [\Gamma + \lambda x : \sigma + e : \tau] = \eta_I \circ (\Lambda g) : [\Gamma] \to T1 (\sigma \Rightarrow TF \tau) \]

(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 \([\Gamma] \vdash e : \tau, F] : [\Gamma] \to TF [\tau]\), taking \(obj(I)\) as the effect
sets of a traditional effect system, with the strict monoidal structure on \( I \) provided by the effect lattice, with 1 = \( \bot \) and \( \otimes = \sqcup \), and morphisms \( f : X \to Y \) in \( I \) if \( X \sqsubseteq Y \) in the effect lattice. Pleasingly, the usual equational theory for \( \lambda_e \) (such as \( \beta \)-equality for values) follows directly from the strict 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:

\[
\begin{align*}
(\text{sub}) & \quad [\Gamma \vdash e : \tau, F'] = g : [\Gamma'] \to T \rho \tau[\rho] \quad F' \sqsubseteq F' \\
[\Gamma \vdash e : \tau, F] & \quad \Rightarrow [\Gamma \vdash \iota_{X,Y} \circ e : \tau, T[\rho] \tau[\rho]] \Rightarrow T F \tau[\rho]
\end{align*}
\]

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., \( T[\rho] \rho \circ e : \tau, A \to A \) and \( T[\text{write} \rho : \tau] A = A \times \tau \) (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 semantics of effects with traditional effect systems. The indexed (semi-)monoidal comonad \( \iota \) on \( \{\mathcal{C}, \mathcal{C}\} \), with \( \lambda \mathcal{C} \mathcal{C} \mathcal{F} \mathcal{G} \mathcal{G} \mathcal{F} \mathcal{F} \), has been proposed to give a correspondance 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] Nielsen, 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] Petriček, T., Mycroft, A., and Syme, D. Extending Monads with Pattern Matching. In *Proceedings of Haskell Symposium* (2011), Haskell 2011.

[9] Petriček, 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. *AMC Trans. Comput. Logic* 4 (January 2003), 1–32.