Refinement Types for Algebraic Effects

Danel Ahman and Gordon D. Plotkin

Laboratory for Foundations of Computer Science, University of Edinburgh

We investigate an algebraic treatment of propositional refinement types for languages with computational effects, and develop a refinement-typed fine-grain call-by-value (FGCBV) [5] language for such refinements. Our work stems from an insight that describing computational effects using algebraic theories $\mathcal{T}_{\mathsf{eff}}$ (following Plotkin and Power [6]) allows us to develop a single framework to account for seemingly different effect specifications found in the literature.

The refinement type system

In our system, *refinement types* τ consist of base types $\mathbf{b}$, product types $\tau_1 \times \tau_2$ and function types $\tau_1 \xrightarrow{\psi} \tau_2$. *Effect refinements* ψ are defined as a fragment of the modal μ-calculus:

$$\psi ::= [\,] \mid \langle \mathsf{op} \rangle(\psi_1, \ldots, \psi_n) \mid \bot \mid \psi_1 \vee \psi_n \mid X \mid \mu X.\psi$$

Effect refinements are intended to represent specifications on computations, in terms of sets of (equivalence classes of) algebraic terms, e.g., the *operation modality* $\langle \mathsf{op} \rangle(\psi_1, \ldots, \psi_n)$ describes a set of algebraic terms $\mathsf{op}(t_1, \ldots, t_n)$ which first perform the computational effect op and then, depending on its outcome, continue as a computation t_i satisfying the corresponding ψ_i.

Effect refinements come with a corresponding subtyping relation $\psi_1 \sqsubseteq \psi_2$, built from rules familiar from modal logic, e.g., $\bot$ is the least element. The relation also includes rules describing the interaction between $\bot$ & $\vee$ and $\langle \mathsf{op} \rangle$, stating that operation modalities are multilinear maps.

We also require the subtyping relation to respect the equations in $\mathcal{T}_{\mathsf{eff}}$. Unfortunately, simply translating the axioms of $\mathcal{T}_{\mathsf{eff}}$ to axioms between corresponding effect refinements is not valid in general: problems arise when the axioms of $\mathcal{T}_{\mathsf{eff}}$ include non-linearity. So instead we limit ourselves to only include derivable *semi-linear equations*, i.e., equations satisfying the conditions in the premise of the rule below. For more discussion about when exactly one can soundly extend algebraic operations on algebras to operations on powersets of algebras, see [1].

$$\frac{\vec{x} \vdash t = u \text{ derivable in } \mathcal{T}_{\mathsf{eff}} \qquad t \text{ linear in } \vec{x} \qquad Vars(u) \subseteq Vars(t)}{t^{\bullet}[\vec{\psi}/\vec{x}] \sqsubseteq u^{\bullet}[\vec{\psi}/\vec{x}]}$$

$(-)^{\bullet}$ is a translation induced by sending operations op to a corresponding modalities $\langle \mathsf{op} \rangle$.

The terms in our system consist of both value terms V and producer terms M, as in FGCBV. Well-typed terms are given by judgments $\Gamma \vDash_{\mathsf{v}} V : \tau$ and $\Gamma \vDash_{\mathsf{p}} M : \tau \,!\, \psi$. Here we only present the typing rules that make the interplay between effect refinements and producer terms explicit:

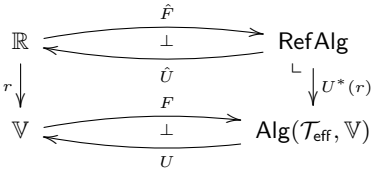
$$\frac{\Gamma \vDash_{\mathsf{v}} V : \tau}{\Gamma \vDash_{\mathsf{p}} \mathsf{ret}\, V : \tau \,!\, [\,]} \qquad \frac{\Gamma \vDash_{\mathsf{p}} M_1 : \tau \,!\, \psi_1 \qquad \ldots \qquad \Gamma \vDash_{\mathsf{p}} M_n : \tau \,!\, \psi_n}{\Gamma \vDash_{\mathsf{p}} \mathsf{op}(M_1, \ldots, M_n) : \tau \,!\, \langle \mathsf{op} \rangle(\psi_1, \ldots, \psi_n)} \qquad \frac{\Gamma \vDash_{\mathsf{p}} M_1 : \tau_1 \,!\, \psi_1 \qquad \Gamma, x : \tau_1 \vDash_{\mathsf{p}} M_2 : \tau_2 \,!\, \psi_2}{\Gamma \vDash_{\mathsf{p}} M_1 \,\mathsf{to}\, x : \tau_1 \,\mathsf{in}\, M_2 : \tau_2 \,!\, \psi_1[\psi_2]}$$

Semantics

We give our system a two-level denotational semantics, based on fibred category theory.

We begin by assuming an adjunction model of FGCBV, i.e., a CCC $\mathbb{V}$ (e.g., Set, or ω-Cpo to also accommodate recursion) and a strong adjunction $F \dashv U : \mathsf{Alg}(\mathcal{T}_{\mathsf{eff}}, \mathbb{V}) \to \mathbb{V}$ between $\mathbb{V}$ and the category of $\mathcal{T}_{\mathsf{eff}}$-algebras in $\mathbb{V}$. To model refinement types we assume a CCC $\mathbb{R}$ and a faithful functor $r : \mathbb{R} \to \mathbb{V}$ such that r strictly preserves the CC structure on $\mathbb{R}$, r is a partially-ordered bifibration, and r has fibre-wise small coproducts that are preserved by reindexing functors.

To model effect refinements, we construct a separate bifibration $U^*(r) : \mathsf{RefAlg} \to \mathsf{Alg}(\mathcal{T}_{\mathsf{eff}}, \mathbb{V})$, by applying change of base to r along U. We interpret effect refinements $\Delta \vdash \psi$ as functors $[\![\psi]\!]_{\mathcal{A}} : \mathsf{RefAlg}_{\mathcal{A}} \times \overrightarrow{\mathsf{RefAlg}_{\mathcal{A}}} \to \mathsf{RefAlg}_{\mathcal{A}}$, separately for each algebra $\mathcal{A}$ from $\mathsf{Alg}(\mathcal{T}_{\mathsf{eff}}, \mathbb{V})$. For closed effect refinements $\vdash \psi$, the interpretations for different algebras extend to a single endofunctor $[\![\psi]\!] : \mathsf{RefAlg} \to \mathsf{RefAlg}$ that preserves the underlying algebras, i.e., $U^*(r) \circ [\![\psi]\!] = U^*(r)$.



$[\![\vdash \tau : \mathsf{Ref}(A)]\!] \in ob(\mathbb{R})$ such that $r([\![\tau]\!]) = [\![A]\!]$

$[\![\Gamma \models_{\mathbb{V}} V : \tau]\!] : [\![\Gamma]\!] \longrightarrow [\![\tau]\!]$

$[\![\Gamma \models_{\mathbb{P}} M : \tau \,!\, \psi]\!] : [\![\Gamma]\!] \longrightarrow (\hat{U} \circ [\![\psi]\!] \circ \hat{F})([\![\tau]\!])$

Notice that the interpretation makes use of the adjunction $\hat{F} \dashv \hat{U}$, induced by the change of base along U [2, Cor. 3.2.5]. Importantly for us, $\hat{F} \dashv \hat{U}$ sits over $F \dashv U$ in an appropriate sense.

Applications

For example, we can represent *effect annotations* ε from type-and-effect systems as suitable fixed point refinements. Following Kammar and Plotkin [3], we can take ε to consist of a set of algebraic operations (the effects permitted to occur in the program) and define the corresponding effect refinement as $\psi_\varepsilon \stackrel{\text{def}}{=} \mu X.\, [\,] \vee \bigvee_{\mathsf{op} \in \varepsilon} \langle \mathsf{op} \rangle (X, \ldots, X)$. In addition, we can equip our system with a relational semantics and validate effect-dependent optimizations, again following [3], based on the algebraic properties determined by effect refinements ψ. However, as our refinements also take the temporal structure of computation into account, we can further validate more involved optimizations, such as dead-code elimination in stateful computation.

Our other examples include *Hoare refinements* corresponding to Hoare types from Hoare Type Theory, and *protocol refinements* for I/O, similar to session types and trace effects.

Related work

The literature contains a range of work on modeling type-and-effect systems by various forms of indexed monad-like structures. Our system is closest to that of Katsumata [4] whose parametric effect monads are indexed by ordered monoids. A fundamental difference with our system is that, rather than taking an abstract indexed-monad view of effect annotations, we obtain both as derived constructs within a wider algebraic theory of effects. In particular, the corresponding parametric effect monad is obtained with the monoid given by closed effect refinements ψ.

Acknowledgments We thank O. Kammar, S. Katsumata, B. Kavanagh, S. Lindley, J. McKinna, A. Simpson, T. Uustalu, and P. Wadler for many useful discussions.

References

[1] N. D. Gautam. The validity of equations of complex algebras. *Arch. Math. Logic*, 3(3-4), 1957.

[2] C. Hermida. *Fibrations, Logical Predicates and Indeterminates*. PhD thesis, U. of Edinburgh, 1993.

[3] O. Kammar and G. D. Plotkin. Algebraic foundations for effect-dependent optimisations. In *Proc. POPL'12*.

[4] S. Katsumata. Parametric effect monads and semantics of effect systems. In *Proc. POPL'14*.

[5] P. B. Levy, J. Power, and H. Thielecke. Modelling environments in call-by-value programming languages. *Inf. & Comp.*, 185(2), 2003.

[6] G. D. Plotkin and J. Power. Notions of computation determine monads. In *Proc. FoSSaCS'02*.