Thesis report

Albert ten Napel

1 Algebraic effects

1.1 Syntax

I took the syntax and semantics from [1]. ε is a single effect, ε^* is a set of effects, *op* is an operation of some effect, O^ε is the set of operations belonging to ε. Each operation has a parameter type and a result type. We presume there is a collection of effects $E := \{\varepsilon_1, ..., \varepsilon_n\}$ (which in a real programming language includes effects like exception, nondeterminism, state and so on). We have a simply typed lambda calculus with unit and boolean values. Values and computations are split ("values are something, computations do something").

$\tau ::=$	(types)
$()$	(unit type)
$Bool$	(type of booleans)
$\tau \to \varepsilon^* \ \tau$	(type of functions)
$\varepsilon \ \tau \Rightarrow \varepsilon^* \ \tau$	(type of handlers)
$\nu ::=$	(values)
x, y, z, k	(variables)
$()$	(unit value)
$true, false$	(booleans)
$\lambda(x : \tau).c$	(abstraction)
$handler \ \{return \ (x : \tau) \to c, op_1(x; k) \to c, ..., op_n(x; k) \to c\}$	(handler)

$c ::=$	(computations)
$\quad \mathit{if}\ \nu\ \mathit{then}\ c\ \mathit{else}\ c$	(if computation)
$\quad \mathit{return}\ \nu$	(return value as computation)
$\quad \nu\ \nu$	(application)
$\quad \mathit{op}(\nu; \lambda(x : \tau).c)$	(operation call)
$\quad x \leftarrow c; c$	(effect sequencing)
$\quad \mathit{with}\ \nu\ \mathit{handle}\ c$	(handle computation)

Note that the operation call also packages a value and a continuation inside of it, having the continuation makes the semantics easier. We can get back the simpler operation calls such as seen in Eff and Koka by defining $\mathit{op} := \lambda(x : \mathit{op}(x; \lambda(y : \mathit{return}\ y).))$. (Pretnar calls these Generic Effects in [1]).

1.2 Typing rules

1.2.1 Judgements

$\mathit{op} : \tau_1 \to \tau_2 \in \varepsilon$ (op is an operation of effect ε with parameter type τ_1 and result type τ_2)
$\Gamma \vdash \nu : \tau$ (value ν has type τ)
$\Gamma \vdash c : \tau\ ; E$ (computation c returns a value of type τ and uses the effects in E)

1.2.2 Value typing rules

The typing rules for values are fairly straightforward.

$$\frac{}{\Gamma, x : \tau \vdash x : \tau} \qquad \frac{}{\Gamma \vdash () : ()} \qquad \frac{}{\Gamma \vdash \mathit{true} : \mathit{Bool}} \qquad \frac{}{\Gamma \vdash \mathit{false} : \mathit{Bool}}$$

$$\frac{\Gamma, x : \tau_1 \vdash c : \tau_2\ ; \varepsilon^*}{\Gamma \vdash \lambda(x : \tau_1).c : \tau_1 \to \varepsilon^*\ \tau_2}$$

For the handler we go over the operations in the handler, check that they are operations of the same effect and get the type of the computation in the cases. We take the union of all the effects that occur in the return and operation cases.

$$\frac{\begin{array}{c}\Gamma, x_r : \tau_1 \vdash c_r : \tau_2 \,;\varepsilon_r^* \\ op_i : \alpha_i \to \beta_i \in \varepsilon \\ \Gamma, x_i : \alpha_i, k_i : \beta_i \to \varepsilon_r^*\ \tau_2 \vdash c_i : \tau_2 \,;\varepsilon_i^*\end{array}}{\begin{array}{l}\Gamma \vdash \mathit{handler}\,\{\mathit{return}\,(x_r : \tau_1) \to c_r, op_1(x_1; k_1) \to c_1, ..., op_n(x_n; k_n) \to c_n\} \\ \quad : \varepsilon\ \tau_1 \Rightarrow (\varepsilon_r^* \cup \bigcup_{i=1}^{n} \varepsilon_i^*)\ \tau_2\end{array}}$$

1.2.3 Computation typing rules

$$\frac{\begin{array}{c}\Gamma \vdash \nu : \mathit{Bool} \\ \Gamma \vdash c_1 : \tau \,;\varepsilon_1^* \\ \Gamma \vdash c_2 : \tau \,;\varepsilon_2^*\end{array}}{\Gamma \vdash \mathit{if}\,\nu\ \mathit{then}\,c_1\ \mathit{else}\,c_2 : \tau \,;\varepsilon_1^* \cup \varepsilon_2^*} \qquad \frac{\Gamma \vdash \nu : \tau}{\Gamma \vdash \mathit{return}\,\nu : \tau \,;\varnothing}$$

$$\frac{\begin{array}{c}\Gamma \vdash \nu_1 : \tau_1 \to \varepsilon^*\ \tau_2 \\ \Gamma \vdash \nu_2 : \tau_1\end{array}}{\Gamma \vdash \nu_1\ \nu_2 : \tau_2 \,;\varepsilon^*} \qquad \frac{\begin{array}{c}op : \tau_1 \to \tau_2 \in \varepsilon \\ \Gamma \vdash \nu : \tau_1 \\ \Gamma, x : \tau_2 \vdash c : \tau_3 \,;\varepsilon^*\end{array}}{\Gamma \vdash op(\nu; \lambda(x : \tau_2).c) : \tau_3 \,;\{\varepsilon\} \cup \varepsilon^*}$$

$$\frac{\begin{array}{c}\Gamma \vdash c_1 : \tau_1 \,;\varepsilon_1^* \\ \Gamma, x : \tau_1 \vdash c_2 : \tau_2 \,;\varepsilon_2^*\end{array}}{\Gamma \vdash (x \leftarrow c_1; c_2) : \tau_2 \,;\varepsilon_1^* \cup \varepsilon_2^*} \qquad \frac{\begin{array}{c}\Gamma \vdash \nu : \varepsilon\ \tau_1 \Rightarrow \varepsilon_1^*\ \tau_2 \\ \Gamma \vdash c : \tau_1 \,;\{\varepsilon\} \cup \varepsilon_2^*\end{array}}{\Gamma \vdash \mathit{with}\,\nu\ \mathit{handle}\,c : \tau_2 \,;\varepsilon_1^* \cup \varepsilon_2^*}$$

In the handle computation notice that we require the computation to have the effect that the handler handles. After we take the union of the effects used in the handler and the remaining effects of the computation. If the handler does not reuse the same effect then the effect is completely handled.

1.3 Small-step operational semantics

Following are the small-step operational semantics of the calculus taken from [1]. With the computations we can always either get to *return* ν or $op(\nu; \lambda(x : \tau).c)$ by floating out the operation call as shown in the 6th rule. This makes the semantics of the handle computation easier since we only have to consider the cases of return and operation calls.

$$\frac{}{\textit{if true then } c_1 \textit{ else } c_2 \rightsquigarrow c_1} \qquad \frac{}{\textit{if false then } c_1 \textit{ else } c_2 \rightsquigarrow c_2}$$

$$\frac{}{(\lambda(x : \tau).c)\ \nu \rightsquigarrow c[\nu/x]} \qquad \frac{c_1 \rightsquigarrow c_1'}{x \leftarrow c_1; c_2 \rightsquigarrow x \leftarrow c_1'; c_2}$$

$$\frac{}{x \leftarrow \textit{return } \nu; c \rightsquigarrow c[\nu/x]} \qquad \frac{}{x \leftarrow op(\nu; \lambda(y : \tau).c_1); c_2 \rightsquigarrow op(\nu; \lambda(y : \tau).x \leftarrow c_1; c_2)}$$

$h := \textit{handler } \{\textit{return } (x_r : \tau) \rightarrow c_r, op_1(x_1; k_1) \rightarrow c_1, ..., op_n(x_n; k_n) \rightarrow c_n\}$, in the following rules:

$$\frac{c \rightsquigarrow c'}{\textit{with } h \textit{ handle } c \rightsquigarrow \textit{with } h \textit{ handle } c'} \qquad \frac{}{\textit{with } h \textit{ handle } (\textit{return } \nu) \rightsquigarrow c[\nu/x]}$$

$$\frac{op_i \in \{op_1, ..., op_n\}}{\textit{with } h \textit{ handle } op_i(\nu; \lambda(x : \tau).c) \rightsquigarrow c_i[\nu/x_i, (\lambda(x : \tau).\textit{with } h \textit{ handle } c)/k_i]}$$

$$\frac{op \notin \{op_1, ..., op_n\}}{\textit{with } h \textit{ handle } op(\nu; \lambda(x : \tau).c) \rightsquigarrow op(\nu; \lambda(x : \tau).\textit{with } h \textit{ handle } c)}$$

2 Algebraic effects with static instances

2.1 Syntax

I looked at [2] to get the rules right, but I simplified the system. We no longer have to mention the effects in the types but just the instances, since every instance belongs to a specific effect and they are all statically known. Handlers in my system only handle operations on a specific instance. For each $\varepsilon \in E$ there is a set of instances $I^{\varepsilon} := \{\iota_1, ..., \iota_n\}$. ι^* is some collection of instances.

Building on top of the previous definitions:

$\tau ::= ...$	(types)
$\quad \mathit{Inst}\ \iota$	(type of instances)
$\quad \tau \to \iota^*\ \tau$	(type of functions)
$\quad \iota\ \tau \Rightarrow \iota^*\ \tau$	(type of handlers)
$\nu ::= ...$	(values)
$\quad \iota$	(instance)
$\quad \mathit{handler}(\nu)\{\mathit{return}\ (x : \tau) \to c, \mathit{op}_1(x; k) \to c, ..., \mathit{op}_n(x; k) \to c\}$	(handler)
$c ::= ...$	(computations)
$\quad \nu \# \mathit{op}(\nu; \lambda(x : \tau).c)$	(operation call)

Now operations are called on a specific instance. Which means we also need instances as values and a type for instances. Handlers now also handle operations on a specific instance only.

2.2 Typing rules

Following are the typing rules for instances and the updated rules for handlers and computations (note that only the rule for operation call really changes). The judgement for computations is now: $\Gamma \vdash c : \tau\ ; \iota^*$, instead of a set of effects we now have a set of instances.

$$\frac{\iota \in I^{\varepsilon}}{\Gamma \vdash \iota : \mathit{Inst}\ \iota}$$

$$\frac{\begin{array}{l}\Gamma \vdash \nu : \mathit{Inst}\ \iota \\ \Gamma, x_r : \tau_1 \vdash c_r : \tau_2 \,;\, \iota_r^* \\ \mathit{op}_i : \alpha_i \to \beta_i \in \varepsilon \\ \Gamma, x_i : \alpha_i, k_i : \beta_i \to \iota_r^*\ \tau_2 \vdash c_i : \tau_2 \,;\, \iota_i^*\end{array}}{\begin{array}{l}\Gamma \vdash \mathit{handler}(\nu)\{\mathit{return}\ (x_r : \tau_1) \to c_r, \mathit{op}_1(x_1; k_1) \to c_1, ..., \mathit{op}_n(x_n; k_n) \to c_n\} \\ \quad : \iota\ \tau_1 \Rightarrow (\iota_r^* \cup \bigcup_{i=1}^n \iota_i^*)\ \tau_2\end{array}}$$

$$\frac{\begin{array}{c}\Gamma \vdash \nu : \mathit{Bool} \\ \Gamma \vdash c_1 : \tau \,;\, \iota_1^* \\ \Gamma \vdash c_2 : \tau \,;\, \iota_2^*\end{array}}{\Gamma \vdash \mathit{if}\ \nu\ \mathit{then}\ c_1\ \mathit{else}\ c_2 : \tau \,;\, \iota_1^* \cup \iota_2^*} \qquad \frac{\Gamma \vdash \nu : \tau}{\Gamma \vdash \mathit{return}\ \nu : \tau \,;\, \varnothing}$$

$$\frac{\begin{array}{c}\Gamma \vdash \nu_1 : \tau_1 \to \iota^*\ \tau_2 \\ \Gamma \vdash \nu_2 : \tau_1\end{array}}{\Gamma \vdash \nu_1\ \nu_2 : \tau_2 \,;\, \iota^*} \qquad \frac{\begin{array}{c}\iota \in I^{\varepsilon} \\ \mathit{op} : \tau_1 \to \tau_2 \in \varepsilon \\ \Gamma \vdash \nu : \tau_1 \\ \Gamma, x : \tau_2 \vdash c : \tau_3 \,;\, \iota^*\end{array}}{\Gamma \vdash \iota \# \mathit{op}(\nu; \lambda(x : \tau_2).c) : \tau_3 \,;\, \{\iota\} \cup \iota^*}$$

$$\frac{\begin{array}{c}\Gamma \vdash c_1 : \tau_1 \,;\, \iota_1^* \\ \Gamma, x : \tau_1 \vdash c_2 : \tau_2 \,;\, \iota_2^*\end{array}}{\Gamma \vdash (x \leftarrow c_1; c_2) : \tau_2 \,;\, \iota_1^* \cup \iota_2^*} \qquad \frac{\begin{array}{c}\Gamma \vdash \nu : \iota\ \tau_1 \Rightarrow \iota_1^*\ \tau_2 \\ \Gamma \vdash c : \tau_1 \,;\, \{\iota\} \cup \iota_2^*\end{array}}{\Gamma \vdash \mathit{with}\ \nu\ \mathit{handle}\ c : \tau_2 \,;\, \iota_1^* \cup \iota_2^*}$$

2.3 Small-step operational semantics

Following are the rules that change.

$$\frac{}{x \leftarrow \iota \# \mathit{op}(\nu; \lambda(y : \tau).c_1); c_2 \rightsquigarrow \iota \# \mathit{op}(\nu; \lambda(y : \tau).x \leftarrow c_1; c_2)}$$

$h := \mathit{handler}(\iota)\{\mathit{return}\ (x_r : \tau) \to c_r, \mathit{op}_1(x_1; k_1) \to c_1, ..., \mathit{op}_n(x_n; k_n) \to c_n\}$, in the following rules:

$$\frac{op_i \in \{op_1, ..., op_n\}}{with\ h\ handle\ \iota \# op_i(\nu; \lambda(x : \tau).c) \rightsquigarrow c_i[\nu/x_i, (\lambda(x : \tau).with\ h\ handle\ c)/k_i]}$$

$$\frac{\iota \neq \iota' \vee op \notin \{op_1, ..., op_n\}}{with\ h\ handle\ \iota' \# op(\nu; \lambda(x : \tau).c) \rightsquigarrow \iota' \# op(\nu; \lambda(x : \tau).with\ h\ handle\ c)}$$

References

[1] Pretnar, Matija. "An introduction to algebraic effects and handlers. invited tutorial paper." Electronic Notes in Theoretical Computer Science 319 (2015): 19-35.

[2] Bauer, Andrej, and Matija Pretnar. "An effect system for algebraic effects and handlers." International Conference on Algebra and Coalgebra in Computer Science. Springer, Berlin, Heidelberg, 2013.