Effigy

Henry Blanchette

October 2019

1 Conventions

Notation	Meaning
e	expression
x, y, z	term variable
v	value
h	handler
τ, σ, ρ	type
$\overline{x}$	list of x
ϕ	effect
Φ	set of effects
$\mathsf{H}\ \Phi\ \tau$	Φ-handler yielding τ
$\mathsf{E}\ \Phi\ \tau$	Φ-effect yielding τ

2 Grammar

$$
\begin{array}{rcl}
\textit{program} & ::= & \overline{\textit{declaration}} \ \mid \ \overline{\textit{declaration}}\ \mathsf{Main}\ := \ e\ . \\
\textit{declaration} & ::= & \mathsf{Effect}\ \phi : \tau. \ \mid\ \mathsf{Definition}\ x : \tau := e\ . \\
e & ::= & v \ \mid\ e\ e \ \mid\ x\ :=\ e\ ;\ e \ \mid\ \mathsf{do}\ e\ \mathsf{with}\ e \\
v & := & x \ \mid\ v \in \textit{Primitive-Values} \ \mid\ x \Rightarrow e \ \mid\ \phi\ e \ \mid\ \{\ s\ \} \ \mid\ h \\
s & ::= & e\ ;\ s \ \mid\ x\ \leftarrow\ e\ ;\ s \ \mid\ e \\
h & ::= & \mathsf{handler}\ \{\ \mathsf{effect}\ (x\ x)\ x \Rightarrow e \mid \mathsf{value}\ x \Rightarrow e \mid \mathsf{run}\ x \Rightarrow e\ \} \\
\tau & ::= & \tau \in \textit{Primitive-Types} \ \mid\ \tau \to \tau \ \mid\ \mathsf{H}\ \Phi\ \tau \ \mid\ \mathsf{E}\ \Phi\ \tau \\
x & \in & \textit{Term-Names} \\
\phi & \in & \textit{Effect-Names} \\
\Phi & \in & \mathcal{P}(\textit{Effect-Names}) \\
\textit{Primitive-Values} & := & \{()\} \cup \{\mathsf{true}, \mathsf{false}\} \cup \mathbb{Z}
\end{array}
$$

3 Definitions

3.1 Lifting to Effects

Since $\mathsf{E}\ \Phi\ \tau$ is the type of terms that have effects Φ and result in τ, if $\Phi = \varnothing$ then this $\mathsf{E}\ \Phi\ \tau$ is the type of terms that have no effects and result in τ i.e. just τ. The following bijection reflects this.

$$\mathsf{lift} : \tau \iff \mathsf{E}\ \varnothing\ \tau \qquad \text{(lift to effect)}$$

3.2 Partial Ordering for Effects

The power set of effects, $\mathcal{P}(\textit{Effects})$, has a partial order induced by set inclusion. Define meet, $\wedge$, to be set intersection and join, $\vee$, to be set union. Note that $\varnothing$ is the minimal element and *Effects* is the maximal element.

$\wedge$ and $\vee$ is overloaded to apply to types as well, via the following definitions:

$$
\begin{array}{llll}
\mathsf{E}\ \Phi\ \tau & \wedge & \mathsf{E}\ \Phi'\ \tau' & := \ \mathsf{E}\ (\Phi\ \wedge\ \Phi')\ \tau' \\
\tau & \wedge & \tau' & := \ \mathsf{lift}\tau\ \wedge\ \mathsf{lift}\tau' \\
\\
\mathsf{E}\ \Phi\ \tau & \vee & \mathsf{E}\ \Phi'\ \tau' & := \ \mathsf{E}\ (\Phi\ \vee\ \Phi')\ \tau' \\
\tau & \vee & \tau' & := \ \mathsf{lift}\tau\ \vee\ \mathsf{lift}\tau'
\end{array}
$$

4 Typing

4.1 Functional Foundations

This first set of typing rules reflect a traditional functional-style type system.

$$\frac{x : \tau \in \Gamma}{\Gamma \vdash x : \tau} \qquad \frac{\Gamma, x : \sigma \vdash e : \tau}{\Gamma \vdash x \Rightarrow e : \sigma \to \tau} \qquad \frac{\Gamma \vdash e_1 : \sigma \to \tau \qquad \Gamma \vdash e_2 : \sigma}{\Gamma \vdash e_1\ e_2 : \tau}$$

4.2 Sequences of Effects

This next set of typing rules allow for the introduction of sequenced terms. In the usual pure context, such expressions are unneeded because the sequencing of terms must have no effect on computation. However, with the introduction of effects in mind, sequencing allows for "complex" effects that perform multiple effects in some order, possibly using the results of earlier effectual computations later in the sequence.

$$\frac{\Gamma \vdash e : \tau}{\Gamma \vdash \{\ e\ \} : \tau} \qquad \frac{\Gamma \vdash e_1 : \sigma \qquad \Gamma \vdash e_2 : \tau}{\Gamma \vdash \{\ e_1\ ;\ e_2\ \} : \sigma\ \vee\ \tau} \qquad \frac{\Gamma \vdash \{\ e_1\ ;\ \cdots\ ;\ e_n\ \} : \sigma \qquad \Gamma \vdash e_{n+1} : \tau}{\Gamma \vdash \{\ e_1\ ;\ \cdots\ ;\ e_n\ ;\ e_{n+1}\ \} : \sigma\ \vee\ \tau}$$

4.3 Handlers for Effects

The next rule defines the type for each component of a handler. The rule following it defines the "application" of handlers on effectual computations in order to yield a pure(er) result.

$$\frac{\begin{array}{cc} \phi : \sigma \to \mathsf{E}\ \{\phi\}\ \tau & \Gamma, x : \sigma \vdash e_v : \sigma \to \mathsf{E}\ \{\phi\}\ \rho \\ \Gamma, x : \sigma, k : \tau \to \mathsf{E}\ \{\phi\}\ \rho \vdash e_\phi : \sigma \to \mathsf{E}\ \{\phi\}\ \rho & \Gamma, g : \sigma \to \mathsf{E}\ \{\phi\}\ \rho \vdash e_r : \mathsf{E}\ \{\phi\}\ \rho \end{array}}{\Gamma \vdash \mathsf{handler}\ \{\ \mathsf{effect}\ (\phi\ x)\ k \Rightarrow e_\phi \mid \mathsf{value}\ x \Rightarrow e_v \mid \mathsf{run}\ g \Rightarrow e_r\ \} : \mathsf{H}\ \{\phi\}\ \rho}$$

$$\frac{\Gamma \vdash e : \mathsf{E}\ \Phi\ \rho \qquad \Gamma \vdash h : \mathsf{H}\ \Phi'\ \rho}{\Gamma \vdash \mathsf{do}\ e\ \mathsf{with}\ h : \mathsf{E}\ (\Phi \setminus \Phi')\ \rho}$$

5 Rewriting

Note that a v is a *value* i.e. a completely reduced term i.e. no rewrite rules apply to v.

5.1 Functional Foundations

function application	$(x \Rightarrow e)\ v$	$\twoheadrightarrow$	$[v/x]e$
pure binding	$x\ :=\ e_1\ ;\ e_2$	$\twoheadrightarrow$	$(x\ \Rightarrow\ e_2)\ e_1$

5.2 Sequences of Effects

trivial sequence	$\{\ e\ \}$	$\twoheadrightarrow$	e
effectful binding	$\mathsf{do}\ \{\ x\ \leftarrow\ e_1\ ;\ e_2\ ;\ \cdots\ ;\ e_n\ \}\ \mathsf{with}\ h$	$\twoheadrightarrow$	$x\ :=\ \mathsf{do}\ e_1\ \mathsf{with}\ h\ ;\ \mathsf{do}\ \{\ e_2\ ;\ \cdots\ ;\ e_n\ \}\ \mathsf{with}\ h$

5.3 Handlers for Effects

Define

$$h := \mathsf{handler}\ \{\ \mathsf{effect}\ (\phi\ x)\ k \Rightarrow e_\phi\ \mid\ \mathsf{value}\ x \Rightarrow e_v\ \mid\ \mathsf{run}\ g \Rightarrow e_r\ \}.$$

The following rewrite rules describe how to handled expressions and sequences of expressions into pure terms.

handle effect	$\mathsf{do}\ \phi\ e\ \mathsf{with}\ h$	$\twoheadrightarrow$	$(k\ \Rightarrow\ (x\ \Rightarrow\ e_\phi)\ e_1)\ (g\ \Rightarrow\ e_r)$
handle sequenced effect	$\mathsf{do}\ \{\ \phi\ e_1\ ;\ e_2\ ;\ \cdots\ ;\ e_n\ \}\ \mathsf{with}\ h$	$\twoheadrightarrow$	$(k\ \Rightarrow\ (x\ \Rightarrow\ e_\phi)\ e_1)\ (\mathsf{do}\ \{\ e_2\ ;\ \cdots\ ;\ e_n\ \}\ \mathsf{with}\ h)$
handle value	$\mathsf{do}\ e\ \mathsf{with}\ h$	$\twoheadrightarrow$	$((x \Rightarrow e_v)\ e)\ (g\ \Rightarrow\ e_r)$
handle sequenced value	$\mathsf{do}\ \{\ e_1\ ;\ \cdots\ ;\ e_n\ \}\ \mathsf{with}\ h$	$\twoheadrightarrow$	$((x \Rightarrow e_v)\ e_1)\ (\mathsf{do}\ \{\ e_2\ ;\ \cdots\ ;\ e_n\ \}\ \mathsf{with}\ h)$
handle run	$\mathsf{do}\ e\ \mathsf{with}\ h$	$\twoheadrightarrow$	$(g\ \Rightarrow\ e_r)\ e$