Unifying Algebraic with Abstract Effects

September 3, 2020

1 Surface language

1.1 syntax

					ε	::=	$\overline{x.E}$	*effects*
e	::=	x	*expressions*		τ	::=	$\{x \Rightarrow \overline{\sigma}\}$	*type*
	|	$\texttt{new}(x \Rightarrow \overline{d})$			σ	::=	$\texttt{def}\ m(x:\tau):\{\varepsilon\}\ \tau$	*decl. types*
	|	$e.m(e)$				|	$\texttt{effect}\ E = \overline{\texttt{o}}$	
	|	$x.op(e)$				|	$\texttt{effect}\ E = \varepsilon$	
	|	$handle\{h\}e$				|	$\texttt{effect}\ E = \varepsilon$	
d	::=	$\texttt{def}\ m(x:\tau):\varepsilon\ \tau = e$	*declarations*		$\texttt{o}$	::=	$\texttt{def}\ op(x:\tau):\tau$	*operation*
	|	$\texttt{effect}\ E = \overline{\texttt{o}}$			$\texttt{h}$	::=	$return\ x \to e$	*handler*
	|	$\texttt{effect}\ E = \varepsilon$				|	$x.op(x) \to e; h$	

Figure 1: Syntax of surface language

The expression of the surface language contains variables, two basic object-oriented expressions: the `new` statement and the method call, the operation call `x.op(e)`, and handled expression `handle{h}e`. Objects are created by `new` statements that contain a variable x representing the current object along with a list of declarations. Declarations come in three kinds: a method declaration, a effect member with operations, and an effect member with subeffects. Method declarations are annotated with a set of effects.

Object types are a collection of declaration types, which include method signatures and the types of effect-member declarations and definitions. Similar to the difference between the modules and their types, effects in an object must always be defined (i.e., has operations or subeffects), whereas effects in object types may or may not have definitions (i.e., be either abstract or concrete).

1.2 Typing Rules

$\boxed{\Gamma \vdash e : \tau}$

$$\frac{x : \tau \in \Gamma}{\Gamma \vdash x : \{\}\ \tau}\ \text{(T-Var)} \qquad \frac{\forall i,\ d_i \in \overline{d},\ \sigma_i \in \overline{\sigma},\ \Gamma,\ x : \{x \Rightarrow \overline{\sigma}\} \vdash d_i : \sigma_i}{\Gamma \vdash \texttt{new}(x \Rightarrow \overline{d}) : \{\}\ \{x \Rightarrow \overline{\sigma}\}}\ \text{(T-New)}$$

$$\frac{\begin{array}{c}\Gamma \vdash e_1 : \{\varepsilon_1\}\{x \Rightarrow \overline{\sigma}\} \quad \texttt{def}\ m(y : \tau_2) : \{\varepsilon_3\}\ \tau_1 \in \overline{\sigma} \\ \Gamma \vdash [e_1/x][e_2/y]\varepsilon_3\ wf \quad \Gamma \vdash e_2 : \{\varepsilon_2\}\ [e_1/x]\tau_2 \quad \varepsilon = \varepsilon_1 \cup \varepsilon_2 \cup [e_1/x][e_2/y]\varepsilon_3\end{array}}{\Gamma \vdash e_1.m(e_2) : \{\varepsilon\}\ [e_1/x][e_2/y]\tau_1}\ \text{(T-Method)}$$

$$\frac{\Gamma \vdash e : \{\varepsilon_1\}\ \tau_1 \quad \Gamma \vdash \tau_1 <: \tau_2 \quad \Gamma \vdash \varepsilon_1 <: \varepsilon_2}{\Gamma \vdash e : \{\varepsilon_2\}\ \tau_2}\ \text{(T-Sub)}$$

$$\frac{\Gamma \vdash x : \{\varepsilon\}\{y \Rightarrow \overline{\sigma}\} \quad \texttt{effect}\ E = \overline{o} \in \overline{\sigma} \quad \texttt{def}\ op(z : \tau_1) : \tau_2 \in \overline{o} \quad \Gamma \vdash e : \{\varepsilon\}\tau_1}{\Gamma \vdash x.op(e) : \{x.E, \varepsilon\}\tau_2}\ \text{(T-Op)}$$

$$\frac{\begin{array}{c}\Gamma \vdash e : \{x.E, \varepsilon\}\ \tau \quad \Gamma \vdash findop(x.E) = \{x_1.op_1(y_1), \ldots, x_n.op_n(y_n)\} \quad \Gamma \vdash x_i.op_i(y_i) : \tau_i \to \tau_i' \\ \Gamma, resume : \{\_ \Rightarrow \texttt{def}\ m(x : \tau_i') : \{\varepsilon\}\ \tau_r\}, y_i : \tau_i \vdash e_i : \{\varepsilon\}\ \tau_r \quad \Gamma, z : \tau \vdash e_r : \{\varepsilon\}\ \tau_r\end{array}}{\Gamma \vdash \texttt{handle}\ \{x_1.op_1(y_1) \to e_1, ..., x_n.op_n(y_n) \to e_n, return\ z \to e_r\} e : \{\varepsilon\}\ \tau_r}\ \text{(T-Handle)}$$

$\boxed{\Gamma \vdash d : \sigma}$

$$\frac{\Gamma,\ x : \tau_1 \vdash e : \{\varepsilon\} \quad \Gamma, x : \tau_1 \vdash \varepsilon wf}{\Gamma \vdash \texttt{def}\ m(x : \tau_1) : \{\varepsilon\}\ \tau_2 = e\ :\ \texttt{def}\ m(x : \tau_1) : \{\varepsilon\}\ \tau_2}\ \text{(DT-Def)}$$

$$\frac{}{\Gamma \vdash \texttt{effect}\ g = \overline{o}\ :\ \texttt{effect}\ g = \overline{o}}\ \text{(DT-Effect-1)} \qquad \frac{\Gamma \vdash \varepsilon\ wf}{\Gamma \vdash \texttt{effect}\ g = \{\varepsilon\}\ :\ \texttt{effect}\ g = \{\varepsilon\}}\ \text{(DT-Effect-2)}$$

Figure 2: type system

Rules T-Var, T-New, T-Method, and T-Sub are standard object oriented typing judgments. In rule T-OP, the type of the operation call is given by its signature, which is defined in its parent object. In rule T-Handle, the effect `x.E` is handled if the handler handles every operation in `x.E`. The expression in each handler clause may contain a `resume` function which serves as the continuation of the operation. The `findop` function is defined in Fig. 5.

1.3 Subtyping

$\boxed{\Gamma \vdash \tau <: \tau'}$

$$\frac{}{\Gamma \vdash \tau <: \tau} \text{ (S-Refl1)} \qquad \frac{\Gamma \vdash \tau_1 <: \tau_2 \quad \Gamma \vdash \tau_2 <: \tau_3}{\Gamma \vdash \tau_1 <: \tau_3} \text{ (S-Trans)}$$

$$\frac{\{x \Rightarrow \sigma_i^{i\in 1..n}\} \textit{ is a permutation of } \{x \Rightarrow \sigma_i'^{i\in 1..n}\}}{\Gamma \vdash \{x \Rightarrow \sigma_i^{i\in 1..n}\} <: \{x \Rightarrow \sigma_i'^{i\in 1..n}\}} \text{ (S-Perm)}$$

$$\frac{}{\Gamma \vdash \{x \Rightarrow \sigma_i^{i\in 1..n+k}\} <: \{x \Rightarrow \sigma_i^{i\in 1..n}\}} \text{ (S-Width)} \quad \frac{\forall i,\ \Gamma,\ x : \{x \Rightarrow \sigma_i^{i\in 1..n}\} \vdash \sigma_i <: \sigma_i'}{\Gamma \vdash \{x \Rightarrow \sigma_i^{i\in 1..n}\} <: \{x \Rightarrow \sigma'^{i\in 1..n}_i\}} \text{ (S-Depth)}$$

$\boxed{\Gamma \vdash \sigma <: \sigma'}$

$$\frac{}{\Gamma \vdash \sigma <: \sigma} \text{ (S-Refl2)} \quad \frac{\Gamma \vdash \tau_1' <: \tau_1 \quad \Gamma \vdash \tau_2 <: \tau_2' \quad \Gamma, x : \tau_1 \vdash \varepsilon_1 <: \varepsilon_2}{\Gamma \vdash \texttt{def}\ m(x : \tau_1) : \{\varepsilon_1\}\ \tau_2 <: \texttt{def}\ m(x : \tau_1') : \{\varepsilon_2\}\ \tau_2'} \text{ (S-Def)}$$

$$\frac{}{\Gamma \vdash \texttt{effect}\ g = \{\varepsilon\} <: \texttt{effect}\ g} \text{ (S-Effect)}$$

$\boxed{\Gamma \vdash \varepsilon <: \varepsilon'}$

$$\frac{\varepsilon_1 \subseteq \varepsilon_2}{\Gamma \vdash \varepsilon_1 <: \varepsilon_2} \text{ (SE-Refl)} \quad \frac{\Gamma \vdash x : \{y \Rightarrow \overline{\sigma}\} \quad \texttt{effect}\ E = \varepsilon'' \in \overline{\sigma} \quad \varepsilon \cup \varepsilon'' <: \varepsilon'}{\Gamma \vdash \varepsilon \cup \{x.E\} <: \varepsilon'} \text{ (SE-Left)}$$

$$\frac{\Gamma \vdash x : \{y \Rightarrow \overline{\sigma}\} \quad \texttt{effect}\ E = \varepsilon'' \in \overline{\sigma} \quad \varepsilon <: \varepsilon' \cup \varepsilon''}{\Gamma \vdash \varepsilon <: \varepsilon' \cup \{x.E\}} \text{ (SE-Right)}$$

Figure 3: Subtyping Rules

The subtyping rules for object types are standard. in rule S-Effect , the effect set declaration is a subtype of an abstract effect declaration. The subtyping rules for effect sets (SE-Refl, SE-Left, SE-Right) states that an effect set is a subeffect of another if it is a subset, or becomes a subeffect by replacing an element in either effect sets according to declarations.

1.4 Abstraction problem

```
type A
  effect E {
    op1() : Unit
  }
module a: A
  effect E {
    op1(): Unit
  }

type B
  effect E
  def handler(Unit -> {this.E} T) : {} T
module b: B
  effect E = {a.E}
  def m() : {this.E} Unit
    a.op1()
  def handler(c: Unit -> {this.E} T) : {} T =
    handle c() with
    | a.op1() -> resume ()

b.handler(
  () => handle b.m() with
          | a.op1() -> ()
 )
```

In the last expression, the method call `b.m()` handled by two nested handlers. Since the effect of `b.m` is abstract, the inner handler should not handle the expression because it handles the concrete effect `a.E`. Instead, the operation in `b.m()` should be handled by the outer handler.

In order to solve this problem, we introduce the language with the `wrap` and `unwrap` constructs to ensure the correctness of dynamic semantics.

2 Language with wrap

2.1 syntax

v	::=	l	*values*				
	$\mid$	$v : \tau \gg \tau$	*coercion*	ε	::=	$\overline{x.E}$	*effects*
	$\mid$	`unwrap` (v)		τ	::=	$\{x \Rightarrow \overline{\sigma}\}$	*type*
e	::=	x	*expressions*	σ	::=	`def` $m(x : \tau) : \{\varepsilon\}\ \tau$	*decl. types*
	$\mid$	v			$\mid$	`effect` $E = \overline{\mathtt{o}}$	
	$\mid$	`new`$(x \Rightarrow \overline{d})$			$\mid$	`effect` $E = \varepsilon$	
	$\mid$	$e.m(e)$			$\mid$	`effect` E	
	$\mid$	$x.op(e)$		o	::=	`def` $op(x : \tau) : \tau$	*operation*
	$\mid$	$handle\{h\}e$		h	::=	$return\ x \to e$	*handler*
	$\mid$	`wrap` $_{l.E}(e)$			$\mid$	$x.op(x) \to e; h$	
	$\mid$	`unwrap` $_{l.E}(e)$		Γ	::=	$\emptyset \mid \Gamma, x : \tau$	*typing context*
	$\mid$	$e : \tau \gg \tau$	*coercion*	μ	::=	$\emptyset \mid \mu, l \mapsto \{x \to \overline{d}\}$	*store*
d	::=	`def` $m(x : \tau) : \{\varepsilon\}\ \tau = e$	*declarations*	Σ	::=	$\emptyset \mid \Sigma, l : \tau$	*store context*
	$\mid$	`effect` $E = \overline{\mathtt{o}}$					
	$\mid$	`effect` $E = \varepsilon$					

Figure 4: Language with wrap

In this language, we add locations

2.2 Static Semantics

$\boxed{\Gamma \vdash e : \tau}$

$$\frac{l : \tau \in \Sigma}{\Gamma \mid \Sigma \vdash l : \{\}\ \tau}\ \text{(T-Loc')} \qquad \frac{x : \tau \in \Gamma \mid \Sigma}{\Gamma \mid \Sigma \vdash x : \{\}\ \tau}\ \text{(T-Var')}$$

$$\frac{\forall i,\ d_i \in \overline{d},\ \sigma_i \in \overline{\sigma},\ \Gamma,\ x : \{x \Rightarrow \overline{\sigma}\} \mid \Sigma \vdash d_i : \sigma_i}{\Gamma \mid \Sigma \vdash \texttt{new}(x \Rightarrow \overline{d}) : \{\}\ \{x \Rightarrow \overline{\sigma}\}}\ \text{(T-New')}$$

$$\frac{\begin{array}{c}\Gamma \mid \Sigma \vdash e_1 : \{\varepsilon_1\}\{x \Rightarrow \overline{\sigma}\} \quad \texttt{def}\ m(y : \tau_2) : \{\varepsilon_3\}\ \tau_1 \in \overline{\sigma} \\ \Gamma \mid \Sigma \vdash [e_1/x][e_2/y]\varepsilon_3\ wf \quad \Gamma \mid \Sigma \vdash e_2 : \{\varepsilon_2\}\ [e_1/x]\tau_2 \quad \varepsilon = \varepsilon_1 \cup \varepsilon_2 \cup [e_1/x][e_2/y]\varepsilon_3\end{array}}{\Gamma \mid \Sigma \vdash e_1.m(e_2) : \{\varepsilon\}\ [e_1/x][e_2/y]\tau_1}\ \text{(T-Method')}$$

$$\frac{\Gamma \mid \Sigma \vdash e : \{\varepsilon_1\}\ \tau_1 \quad \Gamma \mid \Sigma \vdash \tau_1 <: \tau_2 \quad \Gamma \mid \Sigma \vdash \varepsilon_1 <: \varepsilon_2}{\Gamma \mid \Sigma \vdash e : \{\varepsilon_2\}\ \tau_2}\ \text{(T-Sub')}$$

$$\frac{\Gamma \mid \Sigma \vdash x : \{x \Rightarrow \overline{\sigma}\} \quad \texttt{effect}\ E = \overline{o} \in \overline{\sigma} \quad \texttt{def}\ op(y : \tau_1) : \tau_2 \in \overline{o} \quad \Gamma \mid \Sigma \vdash e : \{\varepsilon\}\tau_1}{\Gamma \mid \Sigma \vdash x.op(e) : \{x.E, \varepsilon\}\tau_2}\ \text{(T-op')}$$

$$\frac{\begin{array}{c}\Gamma \mid \Sigma \vdash e : \{x.E, \varepsilon\}\ \tau \quad \Gamma \mid \Sigma \vdash findop(x.E) = \{x_1.op_1(y_1), \ldots, x_n.op_n(y_n)\} \quad \Gamma \mid \Sigma \vdash x_i.op_i(y_n) : \tau_i \to \tau_i' \\ \Gamma, resume : \{\_ \Rightarrow \texttt{def}\ m(z : \tau_i') : \{\varepsilon\}\ \tau_r\}, y_i : \tau_i \mid \Sigma \vdash e_i : \{\varepsilon\}\ \tau_r \quad \Gamma, x : \tau \mid \Sigma \vdash e_r : \{\varepsilon\}\ \tau_r\end{array}}{\Gamma \mid \Sigma \vdash \texttt{handle}\ \{x_1.op_1(y_1) \to e_i, ..., x_n.op_n(y_n) \to e_n, return\ x \to e_r\}e : \{\varepsilon\}\ \tau_r}\ \text{(T-handle')}$$

$$\frac{\Gamma \mid \Sigma \vdash e : \{\varepsilon\}\ \tau_1 \quad \Gamma \mid \Sigma \vdash \tau_1 <: \tau_2}{\Gamma \mid \Sigma \vdash (e : \tau_1 \gg \tau_2) : \{\varepsilon\}\ \tau_2}\ \text{(T-wrap-coercion')}$$

$$\frac{\Gamma \mid \Sigma \vdash e : \{\varepsilon\}\ \tau}{\Gamma \mid \Sigma \vdash \texttt{wrap}\ _{\varepsilon}(e) : \{\varepsilon\}\ \tau}\ \text{(T-wrap')} \quad \frac{\Gamma \mid \Sigma \vdash e : \{\varepsilon\}\ \tau}{\Gamma \mid \Sigma \vdash \texttt{unwrap}\ _{\varepsilon}(e) : \{\varepsilon\}\ \tau}\ \text{(T-unwrap')}$$

$\boxed{\Gamma \vdash d : \sigma}$

$$\frac{\Gamma,\ x : \tau_1 \mid \Sigma \vdash e : \{\varepsilon\} \quad \Gamma, x : \tau_1 \mid \Sigma \vdash \varepsilon\ wf}{\Gamma \mid \Sigma \vdash \texttt{def}\ m(x : \tau_1) : \{\varepsilon\}\ \tau_2 = e\ :\ \texttt{def}\ m(x : \tau_1) : \{\varepsilon\}\ \tau_2}\ \text{(DT-Def')}$$

$$\frac{}{\Gamma \mid \Sigma \vdash \texttt{effect}\ g = \overline{o}\ :\ \texttt{effect}\ g = \overline{o}}\ \text{(DT-Effect-1')} \qquad \frac{\Gamma \mid \Sigma \vdash \varepsilon\ wf}{\Gamma \mid \Sigma \vdash \texttt{effect}\ g = \{\varepsilon\}\ :\ \texttt{effect}\ g = \{\varepsilon\}}\ \text{(DT-Effect-2')}$$

$\boxed{\mu : \Sigma}$

$$\frac{\forall l \in \mu, (l \mapsto \{x \to \overline{d}\} \in \mu \quad l : \{x \to \overline{\sigma}\} \in \Sigma \quad \forall d_i \in \overline{d},\ x : \{x \to \sigma_i\} \mid \Sigma \vdash d_i : \sigma_i)}{\mu : \Sigma}\ \text{(T-store)}$$

$\boxed{\Gamma \mid \Sigma \vdash findop(x.E) = \{...\}}$

$$\frac{\Gamma \mid \Sigma \vdash x : \{y \Rightarrow \overline{d}\} \quad \texttt{effect}\ E = \overline{o}}{\Gamma \mid \Sigma \vdash findop(x.E) = \{x.o \mid o \in \overline{o}\}}\ \text{(T-find-1)}$$

$$\frac{\Gamma \mid \Sigma \vdash x : \{y \Rightarrow \overline{d}\} \quad \texttt{effect}\ E = \{x_1.E_1, \ldots, x_n.E_n\} \quad \forall i \in [1, n], \Gamma \mid \Sigma \vdash findop(x_i.E_i) = s_i}{\Gamma \mid \Sigma \vdash findop(x.E) = s_1 \cup \cdots \cup s_n}\ \text{(T-find-2)}$$

Figure 5: Static Semantics

2.3 Dynamic Semantics

$$
\begin{array}{rcll}
E & ::= & [\,] & \textit{evaluation context} \\
 & | & E.m(e) & \\
 & | & v.m(E) & \\
 & | & E.op(e) & \\
 & | & v.op(E) & \\
 & | & handle\{E\}e & \\
 & | & handle\{h\}E & \\
 & | & \texttt{wrap}\ _{l.F}(E) & \\
 & | & \texttt{unwrap}\ _{l.F}(E) & \\
 & | & E : \tau \gg \tau' &
\end{array}
\qquad
\begin{array}{rcll}
X^{es}_{l.op} & ::= & [\,] & \textit{evaluation context for op} \\
 & | & X^{es}_{l.op}.m(e) & \\
 & | & v.m(X^{es}_{l.op}) & \\
 & | & X^{es}_{l.op}.op(e) & \\
 & | & v.op(X^{es}_{l.op}) & \\
 & | & handle\{h\}(X^{es}_{l.op}) & \textit{if } l.op \to e \notin h \\
 & | & \texttt{wrap}\ _{l'.E}(X^{es}_{l.op}) & \textit{if } l.op \notin l'.E \\
 & | & \texttt{wrap}\ _{l'.E}(X^{es}_{l.op}) & \textit{if } l'.E \in es \\
 & | & \texttt{unwrap}\ _{l'.E}(X^{l'.E::es}_{l.op}) & \\
 & | & X^{es}_{l.op} : \tau \gg \tau' &
\end{array}
$$

Figure 6: Evaluation Contexts

$\boxed{\langle e \mid \mu\rangle \longrightarrow \langle e' \mid \mu'\rangle}$

$$\frac{\langle e \mid \mu\rangle \longrightarrow \langle e' \mid \mu'\rangle}{\langle E[e] \mid \mu\rangle \longrightarrow \langle E[e'] \mid \mu'\rangle}\ \text{(E-Congruence)}$$

$$\frac{l \notin dom(\mu)}{\langle \texttt{new}(x \Rightarrow \overline{d}) \mid \mu\rangle \longrightarrow \langle l \mid \mu, l \mapsto \{x \Rightarrow \overline{d}\}\rangle}\ \text{(E-New)}$$

$$\frac{return\ x \to e \in h}{\langle handle\{h\}(v) \mid \mu\rangle \longrightarrow \langle [v/x]e \mid \mu\rangle}\ \text{(E-Return)}$$

$$\frac{loc(v) = l \quad l.op(y) \to e \in h}{\langle handle\{h\}(X^{es}_{l.op}[v.op(v')]) \mid \mu\rangle \longrightarrow \langle [v'/y][new(\_ \Rightarrow \texttt{def}\ m(x : Unit) : \{\}Unit = handle\{h\}(X^{es}_{l.op}[x]))/resume]e \mid \mu\rangle}\ \text{(E-Handi}$$

$$\frac{l_1 \mapsto \{x \Rightarrow \overline{d}\} \in \mu \quad \texttt{def}\ m(y : \tau_1) : \{\varepsilon\}\ \tau_2 = e \in \overline{d}}{\langle l_1.m(v_2) \mid \mu\rangle \longrightarrow \langle [v_2/y][l_1/x]e \mid \mu\rangle}\ \text{(E-Method)}$$

$$\frac{loc(v_1) = l_1 \quad l_1 \mapsto \{x \Rightarrow \overline{d}\} \in \mu \quad \texttt{def}\ m(y : \tau_1) : \{l_1.E\}\ \tau_2 \in \tau' \quad \texttt{effect}\ E = \{\varepsilon\} \in \tau \quad \texttt{effect}\ E \in \tau'}{\langle (v_1 : \tau \gg \tau').m(v_2) \mid \mu\rangle \longrightarrow \langle \texttt{wrap}\ _{l_1.E}(v_1.m(\texttt{unwrap}\ _{l_1.E}(v_2))) \mid \mu\rangle}\ \text{(E-Method-Wrap)}$$

$$\frac{}{\langle \texttt{unwrap}\ _{l.E}(v_1).m(v_2) \mid \mu\rangle \longrightarrow \langle \texttt{unwrap}\ _{l.E}(v_1.m(v_2)) \mid \mu\rangle}\ \text{(E-Method-Unwrap)}$$

$$\frac{}{\langle \texttt{wrap}\ _{l'.E}(v) \mid \mu\rangle \longrightarrow \langle v \mid \mu\rangle}\ \text{(E-wrap)}$$

$\boxed{loc(v) = l}$

$$
\begin{array}{rl}
loc(l) & = l \\
loc(v : \tau \gg\ \tau) & = loc(v) \\
loc(\texttt{unwrap}\ (v)) & = loc(v)
\end{array}
$$

Figure 7: Wyvern dynamic semantics.

2.4 Translation from surface language

$$\left[\frac{x:\tau\in\Gamma}{\Gamma\vdash x:\{\}\ \tau}\ \text{(T-Var)}\right] = x$$

$$\left[\frac{\forall i,\ d_i\in\overline{d},\ \sigma_i\in\overline{\sigma},\ D_i::\Gamma,x:\{x\Rightarrow\overline{\sigma}\}\vdash d_i:\sigma_i}{\Gamma\vdash \texttt{new}(x\Rightarrow\overline{d}):\{\}\ \{x\Rightarrow\overline{\sigma}\}}\ \text{(T-New)}\right] = \texttt{new}\ (x\Rightarrow\overline{[D]})$$

$$\left[\begin{array}{c} E_1::\Gamma\vdash e_1:\{\varepsilon_1\}\{x\Rightarrow\overline{\sigma}\}\quad \texttt{def}\ m(y:\tau_2):\{\varepsilon_3\}\ \tau_1\in\overline{\sigma} \\ \Gamma\vdash [e_1/x][e_2/y]\varepsilon_3\ wf\quad E_2::\Gamma\vdash e_2:\{\varepsilon_2\}\ [e_1/x]\tau_2\quad \varepsilon=\varepsilon_1\cup\varepsilon_2\cup[e_1/x][e_2/y]\varepsilon_3 \\ \hline \Gamma\vdash e_1.m(e_2):\{\varepsilon\}\ [e_1/x][e_2/y]\tau_1 \end{array}\ \text{(T-Method)}\right] = [E_1].m([E_2])$$

$$\left[\frac{E::\Gamma\vdash e:\{\varepsilon_1\}\ \tau_1\quad S::\Gamma\vdash\tau_1<:\tau_2\quad\Gamma\vdash\varepsilon_1<:\varepsilon_2}{\Gamma\vdash e:\{\varepsilon_2\}\ \tau_2}\ \text{(T-Sub)}\right] = [E]:\tau_1\gg\ \tau_2$$

$$\left[\frac{E_1::\Gamma\vdash e_1:\{\varepsilon\}\{x\Rightarrow\overline{\sigma}\}\quad \texttt{effect}\ E=\overline{o}\in\overline{\sigma}\quad \texttt{def}\ op(x:\tau_1):\tau_2\in\overline{o}\quad E_2::\Gamma\vdash e_2:\{\varepsilon\}\tau_1}{\Gamma\vdash e_1.op(e_2):\{e_1.E,\varepsilon\}\tau_2}\ \text{(T-op)}\right] = [E_1].op[E_2]$$

$$\left[\begin{array}{c} E::\Gamma\vdash e:\{x.E,\varepsilon\}\ \tau\quad \Gamma\vdash findop(x.E)=\{x_1.op_1(y_1),\ldots,x_n.op_n(y_n)\}\quad \Gamma\vdash x_i.op_i:\tau_i\to\tau_i' \\ \Gamma,resume\ :\ \{\_\Rightarrow\ \texttt{def}\ m(z\ :\ \tau_i')\ :\ \{\varepsilon\}\ \tau_r\},y_i\ :\ \tau_i\vdash e_i\ :\ \{\varepsilon\}\ \tau_r\quad \Gamma,y\ :\ \tau\vdash e_r\ :\ \{\varepsilon\}\ \tau_r \\ \hline \Gamma\vdash \texttt{handle}\ \{x_i.op_i(y_i)\to e_i,....,x_n.op_n(y_n)\to e_n,return\ y\to e_r\}e:\{\varepsilon\}\ \tau_r \end{array}\ \text{(T-handle)}\right] = \texttt{handle}\ \{...\}[E]$$

Figure 8: Translation of terms

$$\left[\frac{E::\Gamma,\ x:\tau_1\vdash e:\{\varepsilon\}\ \tau_2\quad \Gamma,\ x:\tau_1\vdash\varepsilon\ wf}{\Gamma\vdash \texttt{def}\ m(x:\tau_1):\{\varepsilon\}\ \tau_2=e\ :\ \texttt{def}\ m(x:\tau_1):\{\varepsilon\}\ \tau_2}\ \text{(DT-Def)}\right] = \texttt{def}\ m(x:\tau_1):\{\varepsilon\}\ \tau_2=[E]$$

$$\left[\frac{}{\Gamma\vdash \texttt{effect}\ g=\overline{o}\ :\ \texttt{effect}\ g=\overline{o}}\ \text{(DT-Effect-1)}\right] = \texttt{effect}\ g=\overline{o}$$

$$\left[\frac{\Gamma\vdash\varepsilon\ wf}{\Gamma\vdash \texttt{effect}\ g=\{\varepsilon\}\ :\ \texttt{effect}\ g}\ \text{(DT-Effect-2)}\right] = \texttt{effect}\ g=\{\varepsilon\}$$

Figure 9: Translation of declarations

2.5 Soundness

Theorem 2.1. *(Progress) If* $\emptyset \mid \Sigma \vdash e : \{\varepsilon\}\ \tau$ *(i.e., e is a closed, well-typed expression), then either*

1. *e is a value*
2. *For all* μ *such that* $\mu : \Sigma$*, there exists* e', μ' *such that* $\langle e \mid \mu \rangle \longrightarrow \langle e' \mid \mu' \rangle$
3. $e = X_{l.op}[v.op(v')]$*, where* $loc(v) = l$

Theorem 2.2. *(Preservation) If* $\Gamma \mid \Sigma \vdash e : \{\varepsilon\}\ \tau$*,* $\mu : \Sigma$*, and* $\langle e \mid \mu \rangle \longrightarrow \langle e' \mid \mu' \rangle$*, then there exists* Σ' *such that* $\Sigma \subseteq \Sigma'$*,* $\mu' : \Sigma'$*, and* $\Gamma \mid \Sigma' \vdash e' : \{\varepsilon\}\ \tau$