

June 15, 2016

## 1 Effects

Fix some set of resources  $R$ . A resource is some language primitive that has the authority to directly perform I/O operations. Elements of the set  $R$  are denoted by  $r$ .  $\Pi$  is a fixed set of operations on resources. Its members are denoted  $\pi$ . An effect is a member of the set of pairs  $R \times \Pi$ . A set of effects is denoted by  $\varepsilon$ . In this system we cannot dynamically create resources or resource-operations.

Throughout we refer to the notions of effects and captures. A piece of code  $C$  has the effect  $(r, \pi)$  if operation  $\pi$  is performed on resource  $r$  during execution of  $C$ .  $C$  captures the effect  $(r, \pi)$  if it has the authority to perform operation  $\pi$  on resource  $r$  at some point during its execution.

We use  $r.\pi$  as syntactic sugar for the effect  $(r, \pi)$ . For example, *FileIO.append* instead of  $(FileIO, append)$ .

Types are either resources or structural. Structural types have a set of method declarations. An object of a particular structural type  $\{\bar{\sigma}\}$  can have any of the methods defined by  $\sigma$  invoked on it. The structural type  $\emptyset$  with no methods is called **Unit**.

We assume there are constructions of the familiar types using the basic structural type  $\emptyset$  and method declarations (for example,  $\mathbb{N}$  could be made using  $\emptyset$  and a **successor** function, Peano-style).

Note the distinction between methods (usually denoted  $m$ ) and operations (usually denoted  $\pi$ ). An operation can only be invoked on a resource; resources can only have operations invoked on them. A method can only be invoked on an object; objects can only have methods invoked on them.

We make a simplifying assumption that every method/lambda takes exactly one argument. Invoking some operation  $\pi$  on a resource returns  $\emptyset$ .

## 2 Static Semantics For Fully-Annotated Programs

In this first system every method in the program is explicitly annotated with its set of effects.

### 2.1 Grammar

$$\begin{array}{ll}
 e ::= x & \text{expressions} \\
 \mid r & \\
 \mid \text{new } x \Rightarrow \overline{\sigma} = \overline{e} & \\
 \mid e.m(e) & \\
 \mid e.\pi(e) & \\
 \\
 \tau ::= \{\bar{\sigma}\} \mid \{\bar{r}\} & \text{types} \\
 \\
 \sigma ::= \text{def } m(x : \tau) : \tau \text{ with } \varepsilon \text{ labeled decls.} & \\
 \\
 \Gamma ::= \emptyset & \\
 \mid \Gamma, x : \tau &
 \end{array}$$

#### Notes:

- All declarations ( $\sigma$ -terms) are annotated by what effects they have.
- All methods (and lambda expressions) take exactly one argument. If a method specifies no argument the argument is assumed to be of type **Unit**.
- $\pi$  is the name of an operation from the set  $\Pi$ ;  $m$  is the name of a method.
- The type  $\{\bar{r}\}$  is an (indeterminate) set of resources; there will only be one actual resource at run-time, and it will be one of the resources in the set.

### 2.2 Rules

$$\boxed{\Gamma \vdash e : \tau \text{ with } \varepsilon}$$

$$\frac{}{\Gamma, x : \tau \vdash x : \tau \text{ with } \emptyset} (\varepsilon\text{-VAR}) \qquad \frac{r \in R}{\Gamma, r : \{r\} \vdash r : \{r\} \text{ with } \emptyset} (\varepsilon\text{-RESOURCE})$$

$$\frac{\Gamma, x : \{\bar{\sigma}\} \vdash \overline{\sigma} = \overline{e} \text{ OK}}{\Gamma \vdash \text{new } x \Rightarrow \overline{\sigma} = \overline{e} : \{\bar{\sigma}\} \text{ with } \emptyset} (\varepsilon\text{-NEWOBJ})$$

$$\frac{\Gamma \vdash e_1 : \{\bar{r}\} \text{ with } \varepsilon_1 \quad \Gamma \vdash e_2 : \tau_2 \text{ with } \varepsilon_2 \quad \pi \in \Pi}{\Gamma \vdash e_1.\pi(e_2) : \text{Unit} \text{ with } \{\bar{r}.\pi\} \cup \varepsilon_1 \cup \varepsilon_2} (\varepsilon\text{-OPERCALL})$$

$$\frac{\Gamma \vdash e_1 : \{\bar{\sigma}\} \text{ with } \varepsilon_1 \quad \Gamma \vdash e_2 : \tau_2 \text{ with } \varepsilon_2 \quad \sigma_i = \text{def } m_i(y : \tau_2) : \tau_3 \text{ with } \varepsilon_3}{\Gamma \vdash e_1.m_i(e_2) : \tau_3 \text{ with } \varepsilon_1 \cup \varepsilon_2 \cup \varepsilon_3} (\varepsilon\text{-METHCALL}_\sigma)$$

$$\boxed{\Gamma \vdash \sigma = e \text{ OK}}$$

$$\frac{\Gamma, y : \tau_2 \vdash e : \tau_3 \text{ with } \varepsilon_3 \quad \sigma = \text{def } m(y : \tau_2) : \tau_3 \text{ with } \varepsilon_3}{\Gamma \vdash \sigma = e \text{ OK}} (\varepsilon\text{-VALIDIMPL}_\sigma)$$

#### Notes:

- In  $\varepsilon$ -VAR,  $\varepsilon$ -RESOURCE, and  $\varepsilon$ -NEWOBJ the consequent has an expression typed with no effect; merely possessing a (transitive) capability for an object is not a effect. You must do something with it in order to possibly have an effect.
- $\varepsilon$ -VALIDIMPL says that the return type and effects of the body of a method must agree with what its signature says.

### 3 Static Semantics For Partly-Annotated Programs

In the next system we allow objects which have no effect-annotated methods. If an object has no annotations on its methods, the extra rules below can give a conservative effect inference on what it captures. If an object is fully annotated, the rules from the previous section can be used. There is no inbetween; partially-annotated objects are not allowed.

#### 3.1 Grammar

$$\begin{array}{ll}
 e ::= x & \text{expressions} \\
 \mid r & \\
 \mid \mathbf{new}_\sigma x \Rightarrow \overline{\sigma = e} & \\
 \mid \mathbf{new}_d x \Rightarrow \overline{d = e} & \\
 \mid e.m(e) & \\
 \mid e.\pi(e) & \\
 \\
 \tau ::= \{\bar{\sigma}\} & \text{types} \\
 \mid \{\bar{r}\} & \\
 \mid \{\bar{d}\} & \\
 \mid \{\bar{d} \text{ captures } \varepsilon\} & \\
 \\
 \sigma ::= d \text{ with } \varepsilon & \text{labeled decls.} \\
 \\
 d ::= \mathbf{def } m(x : \tau) : \tau & \text{unlabeled decls.}
 \end{array}$$

#### Notes:

- $\sigma$  denotes a declaration with effect labels;  $d$  a declaration without effect labels.
- $\mathbf{new}_\sigma$  is for creating annotated objects;  $\mathbf{new}_d$  for unannotated objects.
- $\{\bar{\sigma}\}$  is the type of an annotated object.  $\{\bar{d}\}$  is the type of an unannotated object.
- $\{\bar{d} \text{ captures } \varepsilon\}$  is a special kind of type that doesn't appear in source programs but may be assigned by the new rules in this section. Intuitively,  $\varepsilon$  is an upper-bound on the effects captured by  $\{\bar{d}\}$ .

#### 3.2 Rules

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

$$\frac{}{\Gamma, x : \tau \vdash x : \tau} \text{ (T-VAR)} \qquad \frac{r \in R}{\Gamma, r : \{\bar{r}\} \vdash r : \{\bar{r}\}} \text{ (T-RESOURCE)}$$

$$\frac{\Gamma \vdash r : \{\bar{r}\} \quad \Gamma \vdash e : \tau \quad m \in M}{\Gamma \vdash r.\pi(e_1) : \mathbf{Unit}} \text{ (T-METHCALL}_r\text{)}$$

$$\frac{\Gamma \vdash e_1 : \{\bar{\sigma}\}, \mathbf{def } m(y : \tau_2) : \tau_3 \text{ with } \varepsilon_3 \in \{\bar{\sigma}\} \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1.m(e_2) : \tau_3} \text{ (T-METHCALL}_\sigma\text{)}$$

$$\frac{\Gamma \vdash e_1 : \{\bar{d}\}, \mathbf{def } m(y : \tau_2) : \tau_3 \in \{\bar{d}\} \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1.m(e_2) : \tau_3} \text{ (T-METHCALL}_d\text{)}$$

$$\frac{\Gamma \vdash \sigma_i = e_i \text{ OK}}{\Gamma \vdash \mathbf{new}_\sigma x \Rightarrow \overline{\sigma = e} : \{\bar{\sigma}\}} \text{ (T-NEW}_\sigma\text{)}$$

$$\frac{\Gamma \vdash d_i = e_i \text{ OK}}{\Gamma \vdash \mathbf{new}_d x \Rightarrow \overline{d = e} : \{\bar{d}\}} \text{ (T-NEW}_d\text{)}$$

$$\boxed{\Gamma \vdash d = e \text{ OK}}$$

$$\frac{d = \text{def } m(x : \tau_2) : \tau_3 \quad \Gamma \vdash e : \tau_3}{\Gamma \vdash d = e \text{ OK}} (\varepsilon\text{-VALIDIMPL}_d)$$

$$\boxed{\Gamma \vdash e : \tau \text{ with } \varepsilon}$$

$$\frac{\varepsilon_x = \text{effects}(\Gamma') \quad \Gamma' \subseteq \Gamma \quad \Gamma', x : \{\bar{d} \text{ captures } \varepsilon_x\} \vdash \overline{d = e} \text{ OK}}{\Gamma \vdash \text{new}_d x \Rightarrow \overline{d = e} : \{\bar{d} \text{ captures } \varepsilon_x\} \text{ with } \emptyset} (\text{C-NEWOBJ})$$

$$\frac{\Gamma \vdash e_1 : \{\bar{d} \text{ captures } \varepsilon_c\} \text{ with } \varepsilon_1 \quad \Gamma \vdash e_2 : \tau_2 \text{ with } \varepsilon_2 \quad d_i := \text{def } m_i(y : \tau_2) : \tau_3}{\Gamma \vdash e_1.m_i(e_2) : \tau_3 \text{ with } \varepsilon_1 \cup \varepsilon_2 \cup \text{effects}(\tau_2) \cup \varepsilon_c} (\text{C-METHCALL})$$

#### Notes:

- The system includes the rules from the fully-annotated system.
- Rules with the judgement form do standard typing of objects, without any effect analysis. Their sole use is in the  $\varepsilon\text{-ValidImpl}_d$  rule.
- In applying C-NEWOBJ,  $\Gamma$  is the current context. The variable  $\Gamma'$  is some sub-context. A good choice of sub-context is  $\Gamma$  restricted to the free variables in the method-body being typechecked. This tightens the upper-bound to exclude resources never used in the program.
- To perform effect analysis on an unannotated object  $\{\bar{d}\}$ , figure out what it captures using C-NEWOBJ, yielding the type  $\{\bar{d} \text{ captures } \varepsilon\}$ . Then, when a method is called on that object, C-METHCALL can be applied.

### 3.3 Effects Function

The **effects** function returns the set of effects in a particular context.

A method  $m$  can return a resource  $r$  (directly or via some enclosing object). Returning a resource isn't an effect but it means any unannotated program using  $m$  also captures  $r$ . To account for this, when the **effects** function is operating on a type  $\tau$  it must analyse the return type of the method declarations in  $\tau$ . Since the resource might be itself enclosed by an object, we do a recursive analysis.

- $\text{effects}(\emptyset) = \emptyset$
- $\text{effects}(\Gamma, x : \tau) = \text{effects}(\Gamma) \cup \text{effects}(\tau)$
- $\text{effects}(\{\bar{r}\}) = \{(r, \pi) \mid r \in \bar{r}, \pi \in \Pi\}$
- $\text{effects}(\{\bar{\sigma}\}) = \bigcup_{\sigma \in \bar{\sigma}} \text{effects}(\sigma)$
- $\text{effects}(\{\bar{d}\}) = \bigcup_{d \in \bar{d}} \text{effects}(d)$
- $\text{effects}(d \text{ with } \varepsilon) = \varepsilon \cup \text{effects}(d)$
- $\text{effects}(\text{def } m(x : \tau_1) : \tau_2) = \text{effects}(\tau_2)$
- $\text{effects}(\{\bar{d} \text{ captures } \varepsilon\}) = \varepsilon$

#### Notes:

- In the last case, it is not necessary to recurse to sub-declarations; the type  $\{\bar{d} \text{ captures } \varepsilon\}$  can only result from C-NEWOBJ. In that case,  $\varepsilon$  is already an estimate of what the object captures in some context  $\Gamma_0$ ; furthermore, the current  $\Gamma$  may be bigger than  $\Gamma_0$ , in which case re-annotating the object would introduce more effects and lose precision.

## 4 Dynamic Semantics

### 4.1 Terminology

- The runtime effects are what actually happens when the program is executed; the static effect annotations are an estimate of what will happen at runtime (our eventual aim is to show the static annotations are safe).
- If  $e$  is an expression then  $[e_1/x_1]e$  is a new expression, the same as  $e$ , with every free occurrence of  $x_1$  replaced by  $e_1$ .  $[e_1/x_1, \dots, e_n/x_n]e$  is syntactic sugar for repeated one-variable substitution:  $[e_1/x_1] \dots [e_n/x_n]e$ .
- $\emptyset$  is the empty set. The empty type is denoted **Unit**. Its single instance is **unit**.

### 4.2 Grammar

$e ::= x$	<i>expressions</i>	$d ::= \mathbf{def} \ m(x : \tau) : \tau$	<i>unlabeled decls.</i>
$e.m(e)$		$\sigma ::= d \ \mathbf{with} \ \varepsilon$	<i>labeled decls.</i>
$e.\pi(e)$		$\tau ::= \{\bar{\sigma}\}$	<i>types</i>
$v$		$\{\bar{r}\}$	
$v ::= r$	<i>values</i>	$\Gamma ::= \emptyset$	<i>contexts</i>
$\mathbf{new}_\sigma \ x \Rightarrow \overline{\sigma = e}$		$\Gamma, \ x : \tau$	
$\mathbf{new}_d \ x \Rightarrow \overline{d = e}$			

### 4.3 Rules

$$\boxed{e \longrightarrow e \mid \varepsilon}$$

$$\frac{e_1 \longrightarrow e'_1 \mid \varepsilon}{e_1.m(e_2) \longrightarrow e'_1.m(e_2) \mid \varepsilon} \text{ (E-METHCALL1)}$$

$$\frac{v_1 = \mathbf{new}_\sigma \ x \Rightarrow \overline{\sigma = e} \quad e_2 \longrightarrow e'_2 \mid \varepsilon}{v_1.m(e_2) \longrightarrow v_1.m(e'_2) \mid \varepsilon} \text{ (E-METHCALL2}_\sigma\text{)}$$

$$\frac{v_1 = \mathbf{new}_d \ x \Rightarrow \overline{d = e} \quad e_2 \longrightarrow e'_2 \mid \varepsilon}{v_1.m(e_2) \longrightarrow v_1.m(e'_2) \mid \varepsilon} \text{ (E-METHCALL2}_d\text{)}$$

$$\frac{v_1 = \mathbf{new}_\sigma \ x \Rightarrow \overline{\sigma = e} \quad \mathbf{def} \ m(y : \tau_1) : \tau_2 \ \mathbf{with} \ \varepsilon = e \in \overline{\sigma = e}}{v_1.m(v_2) \longrightarrow [v_1/x, v_2/y]e \mid \emptyset} \text{ (E-METHCALL3}_\sigma\text{)}$$

$$\frac{v_1 = \mathbf{new}_d \ x \Rightarrow \overline{d = e} \quad \mathbf{def} \ m(y : \tau_1) : \tau_2 = e \in \overline{d = e}}{v_1.m(v_2) \longrightarrow [v_1/x, v_2/y]e \mid \emptyset} \text{ (E-METHCALL3}_d\text{)}$$

$$\frac{e_1 \longrightarrow e'_1 \mid \varepsilon}{e_1.\pi(e_2) \longrightarrow e'_1.\pi(e_2) \mid \varepsilon} \text{ (E-OPERCALL1)}$$

$$\frac{e_2 \longrightarrow e'_2 \mid \varepsilon}{r.\pi(e_2) \longrightarrow r.\pi(e'_2) \mid \varepsilon} \text{ (E-OPERCALL2)}$$

$$\frac{r \in R \quad \pi \in \Pi}{r.\pi(v) \longrightarrow \mathbf{unit} \mid \{r.\pi\}} \text{ (E-OPERCALL3)}$$

$$\boxed{e \longrightarrow_* e \mid \varepsilon}$$

$$\frac{}{e \longrightarrow_* e \mid \emptyset} \text{ (E-BIGSTEP1)} \qquad \frac{e \longrightarrow e' \mid \varepsilon}{e \longrightarrow_* e' \mid \varepsilon} \text{ (E-BIGSTEP2)}$$

$$\frac{e \longrightarrow_* e' \mid \varepsilon_1 \quad e' \longrightarrow_* e'' \mid \varepsilon_2}{e \longrightarrow_* e'' \mid \varepsilon_1 \cup \varepsilon_2} \text{ (E-BIGSTEP3)}$$

**Notes:**

- A 'big-step' involves *zero* or more applications of a small-step.
- Big-step rules accumulate the run-time effects produced by the individual small-steps.
- The only rule which produces effects is E-OPERCALL3 (the rule for evaluating operations on resources).
- Method calls are evaluated by performing substitution on the body of the method, and evaluating that.

## 5 Theorems

### Lemma 5.1. (Canonical Forms)

**Statement.** Suppose  $e$  is a value. The following are true:

- If  $\Gamma \vdash e : \{\bar{r}\}$  **with**  $\varepsilon$ , then  $\exists r \in R \mid e = r$ .
- If  $\Gamma \vdash e : \{\bar{\sigma}\}$  **with**  $\varepsilon$ , then  $e = \mathbf{new}_\sigma x \Rightarrow \overline{\sigma} \equiv \bar{e}$ .
- If  $\Gamma \vdash e : \{\bar{d} \text{ captures } \varepsilon_1\}$  **with**  $\varepsilon$ , then  $e = \mathbf{new}_d x \Rightarrow \overline{d} = e$ .

Furthermore,  $\varepsilon = \emptyset$  in each case.

**Proof.** These typing judgements each appear exactly once, in the conclusion of different rules. The result follows by inversion of  $\varepsilon$ -RESOURCE,  $\varepsilon$ -NEWOBJ, and C-NEWOBJ respectively.  $\square$

### Lemma 5.2. (Substitution)

**Statement.** If  $\Gamma, z : \tau' \vdash e : \tau$  **with**  $\varepsilon$ , and  $\Gamma \vdash e' : \tau'$  **with**  $\varepsilon'$ , then  $\Gamma \vdash [e'/z]e : \tau$  **with**  $\varepsilon$ .

**Proof.** By structural induction on possible derivations of  $\Gamma \vdash e : \tau_1$  **with**  $\varepsilon_1$ . First, if  $z$  does not appear in  $e$  then  $[e'/z]e = e$ , so the statement holds vacuously. So without loss of generality we assume  $z$  appears somewhere in  $e$  and consider the last rule used in the derivation, and then the location of  $z$ .

**Case.**  $\varepsilon$ -VAR.

Then  $[e'/z]z = e_1$ . By assumption  $\Gamma \vdash e' : \tau$  **with**  $\varepsilon$ , so  $\Gamma \vdash [e'/z]z = e$ .

**Case.**  $\varepsilon$ -RESOURCE.

Then  $\exists r \in R \mid e = r$ , however  $\mathbf{freevars}(r) = \emptyset$ , so the statement holds vacuously.

**Case.**  $\varepsilon$ -NEWOBJ.

Then  $e = \mathbf{new} x \Rightarrow \overline{\sigma} \equiv \bar{e}$ .

If  $z = x$  then the substituted expression is  $e = \mathbf{new} z \Rightarrow \overline{\sigma} \equiv \bar{e}$ . Then  $\Gamma, z : \{\bar{\sigma}\} \vdash \overline{\sigma} \equiv \bar{e}$  OK by the inductive assumption. Then  $[e'/z]e$  types under  $\varepsilon$ -NEWOBJ.

Otherwise  $z$  appears in method body  $e_i$ . By inversion we know  $\Gamma, x : \{\bar{\sigma}\} \vdash \overline{\sigma} \equiv \bar{e}$  OK. The only rule with this conclusion is  $\varepsilon$ -VALIDIMPL $_\sigma$ ; by inversion on that we have:

- $\Gamma, y : \tau_1 \vdash e_i : \tau_2$  **with**  $\varepsilon$
- $\sigma = \mathbf{def} m_i(y : \tau_1) : \tau_2$  **with**  $\varepsilon$

$\Gamma, z : \tau$  can be used to prove  $\sigma_i = e_i$  OK via the inductive assumption. By an application of  $\varepsilon$ -VALIDIMPL $_\sigma$  we may type the substituted object declaration. Then by  $\varepsilon$ -NEWOBJ we type  $[e'/z]e$  to the same as the original.

**Case.**  $\varepsilon$ -OPERCALL.

Then  $e = e_1.\pi(e_2)$ . The variable  $z$  must appear in  $e_1$  or  $e_2$ . By rule inversion we have a sub-derivation for the type of both sub-expressions. If we perform substitution on both, the inductive assumption applies, yielding the same types for  $[e'/z]e_1$  and  $[e'/z]e_2$ . Then  $\varepsilon$ -OPERCALL types the substituted operation call to the same as the original.

**Case.**  $\varepsilon$ -METHCALL $_\sigma$ .

Then  $e = e_1.m_i(e_2)$ . The variable  $z$  must appear in  $e_1$  or  $e - 2$ . By rule inversion we have a sub-derivation for both; we may type the substituted sub-expressions to the same as the original. Then by applying  $\varepsilon$ -METHCALL on the substituted sub-expressions we type  $[e'/z]e_2$  to the same as the original.

**Case.** C-METHCALL.

Same as above case; it's not important that the receiver's declarations are unannotated because substitution



only applies to expressions.

**Case.** C-NEWOBJ.

Same as for the rule  $\varepsilon$ -NEWOBJ; it's not important that the receiver's declarations are unannotated because substitution only applies to expressions.  $\square$

**Corollary.** If  $\Gamma, z_i : \tau'_i \vdash e : \tau$  **with**  $\varepsilon$ , and  $\Gamma \vdash e'_i : \tau'_i$  **with**  $\varepsilon'_i$ , then  $\Gamma \vdash [e'_1/z_1, \dots, e'_n/z_n]e : \tau$  **with**  $\varepsilon$ . This follows by the definition  $[e'_1/z_1, \dots, e'_n/z_n]e = [e'_1] \dots [e'_n]e$  and induction on the length  $n$ .

### Theorem 5.3. (Progress)

**Statement.** If  $\Gamma \vdash e_A : \tau_A$  **with**  $\varepsilon_A$ , either  $e_A$  is a value or a small-step  $e_A \longrightarrow e_B \mid \varepsilon$  can be applied.

**Proof.** By structural induction on possible derivations of  $\Gamma \vdash e_A : \tau_A$  **with**  $\varepsilon_A$ . We consider the last rule used.

**Case.**  $\varepsilon$ -VAR,  $\varepsilon$ -RESOURCE,  $\varepsilon$ -NEWOBJ, C-NEWOBJ.

Then  $e_A$  is a value.

**Case.**  $\varepsilon$ -METHCALL.

Then  $e_A = e_1.m_i(e_2)$  and the following are known:

- $e_A : \tau_3$  **with**  $\varepsilon_1 \cup \varepsilon_2 \cup \varepsilon_3$
- $e_1 : \{\bar{\sigma}\}$  **with**  $\varepsilon_1$
- $e_2 : \tau_2$  **with**  $\varepsilon_2$
- $\sigma_i = \text{def } m_i(y : \tau_2) : \tau_3$  **with**  $\varepsilon_3$

We look at the cases for when  $e_1$  and  $e_2$  are values.

Subcase.  $e_1$  is not a value. The derivation of  $e_A : \tau$  **with**  $\varepsilon_A$  includes the subderivation  $e_1 : \{\bar{\sigma}\}$  **with**  $\varepsilon_1$ . By the inductive hypothesis  $e_1 \longrightarrow e'_1 \mid \varepsilon$ . Then E-METHCALL1 gives the reduction  $e_A \longrightarrow e'_1.m_i(e_2) \mid \varepsilon$ .

Subcase.  $e_2$  is not a value. Without loss of generality,  $e_1 = v_1$  is a value. Also,  $e_2 : \tau_2$  **with**  $\varepsilon_2$  is a subderivation. By inductive hypothesis,  $e_2 \longrightarrow e'_2 \mid \varepsilon$ . Then E-METHCALL2 $_\sigma$  gives the reduction  $e_A \longrightarrow v_1.m_i(e'_2) \mid \varepsilon$ .

Subcase.  $e_1 = v_1$  and  $e_2 = v_2$  are values. By Canonical Forms,  $e_1 = \text{new}_\sigma x \Rightarrow \bar{\sigma} \equiv \bar{e}$ . Also,  $\text{def } m_i(y : \tau_2) : \tau_3$  **with**  $\varepsilon_3 = e_i \in \bar{\sigma} \equiv \bar{e}$ . By the assumption of the typing rule used, the receiver and argument are well-typed for the method  $m_i$ . Then E-METHCALL3 $_\sigma$  gives the reduction  $e_1.m_i(e_2) \longrightarrow [v_1/x, v_2/y]e_i \mid \emptyset$ .

**Case.**  $\varepsilon$ -OPERCALL.

Then  $e_A = e_1.\pi(e_2)$  and the following are known:

- $e_A : \text{Unit with } \{r.\pi\} \cup \varepsilon_1 \cup \varepsilon_2$
- $e_1 : \{\bar{r}\}$  **with**  $\varepsilon_1$
- $e_2 : \tau_2$  **with**  $\varepsilon_2$
- $\pi \in \Pi$

We look at the cases for when  $e_1$  and  $e_2$  are values.

Subcase.  $e_1$  is not a value.  $e_1 : \{\bar{r}\}$  **with**  $\varepsilon_1$  is a subderivation. Applying inductive assumption, we have  $e_1 \longrightarrow e'_1 \mid \varepsilon$ . Then E-OPERCALL1 gives the reduction  $e_1.\pi(e_2) \longrightarrow e'_1.\pi(e_2) \mid \varepsilon$ .

Subcase.  $e_2$  is not a value. Without loss of generality,  $e_1 = v_1$  is a value. Also,  $e_2 : \tau_2$  **with**  $\varepsilon_2$  is a subderivation, so applying inductive assumption we get  $e_2 \longrightarrow e'_2 \mid \varepsilon$ . Then E-OPERCALL2 gives the reduction  $v_1.\pi(e_2) \longrightarrow v_1.\pi(e'_2) \mid \varepsilon$ .

Subcase.  $e_1$  and  $e_2$  are values. By Canonical Forms,  $\exists r \in R \mid e_1 = r$ . Then E-OPERCALL3 gives the reduction  $r.\pi(v_2) \longrightarrow \text{unit} \mid \{r.\pi\}$ .

**Case.** C-METHCALL.

Then  $e_A = e_1.m_i(e_2)$  and the following are known:

- $e_A : \tau_3$  **with**  $\varepsilon_1 \cup \varepsilon_2 \cup \text{effects}(\tau_2) \cup \varepsilon_c$
- $e_1 : \{\bar{d} \text{ captures } \varepsilon_c\}$  **with**  $\varepsilon_1$
- $e_2 : \tau_2$  **with**  $\varepsilon_2$
- $d_i = \text{def } m_i(y : \tau_2) : \tau_3$

We look at the cases for when  $e_1$  and  $e_2$  are values.

Subcase.  $e_1$  is not a value. Also,  $e_1 : \{\bar{d} \text{ captures } \varepsilon_c\}$  **with**  $\varepsilon_1$  is a subderivation. By inductive hypothesis,  $e_1 \longrightarrow e'_1 \mid \varepsilon$ . Then E-METHCALL1 gives the reduction  $e_1.m_i(e_2) \longrightarrow e'_1.m_i(e_2) \mid \varepsilon$ .

Subcase.  $e_2$  is not a value. Without loss of generality,  $e_1 = v_1$  is a value. Also,  $e_2 : \tau_2$  **with**  $\varepsilon_2$  is a subderivation. By inductive hypothesis,  $e_2 \longrightarrow e'_2 \mid \varepsilon$ . Then E-METHCALL2<sub>d</sub> gives the reduction  $v_1.m_i(e_2) \longrightarrow v_1.m_i(e'_2) \mid \varepsilon$ .

Subcase.  $e_1$  and  $e_2$  are values. By Canonical Forms,  $e_1 = \text{new}_d x \Rightarrow \bar{d} = \bar{e}$ . Also,  $\text{def } m_i(y : \tau_2) : \tau_3 = e_i \in \bar{d} = \bar{e}$ . By assumption of the typing rule used, the receiver and argument are well-typed for method  $m_i$ . Then E-METHCALL3<sub>d</sub> gives the reduction  $v_1.m_i(v_2) \longrightarrow [v_1/x, v_2/y]e_i \mid \emptyset$

□

### Theorem 5.4. (Preservation for Mini Calculus)

**Statement.** Work in the mini calculus for fully-annotated programs. Suppose the following hold:

- $\Gamma_A \vdash e_A : \tau_A$  **with**  $\varepsilon_A$
- $e_A \longrightarrow e_B \mid \varepsilon$
- $\Gamma_B \vdash e_B : \tau_B$  **with**  $\varepsilon_B$

Then  $\varepsilon_B \subseteq \varepsilon_A$  and  $\forall r.\pi \in \varepsilon_A \setminus \varepsilon_B \mid r.\pi \in \varepsilon$ . Furthermore, if  $e_A$  is not of the form  $e_1.\pi(e_2)$ , then  $\tau_A = \tau_B$ .

Intuitively: during reduction, effects are only lost from the static information when they are added to the runtime information, so every effect gets “accounted for”. You cannot gain static information after reducing an expression.

**Proof.** By structural induction on the derivation of  $\Gamma_A \vdash e_A : \tau_A$  **with**  $\varepsilon_A$  and then on the reduction rule used.

**Case.**  $\varepsilon$ -RESOURCE,  $\varepsilon$ -VAR,  $\varepsilon$ -NEWOBJ, C-NEWOBJ.

$e_A$  is a value, so no reduction rules can be applied to it. The theorem statement is vacuously satisfied.

**Case.**  $\varepsilon$ -METHCALL <sub>$\sigma$</sub> .

Then  $e_A = e_1.m_i(e_2)$  and the following are true:

- $e_A : \tau$  **with**  $\varepsilon_1 \cup \varepsilon_2 \cup \varepsilon_3$
- $e_1 : \{\bar{\sigma}\}$  **with**  $\varepsilon_1$
- $e_2 : \tau_2$  **with**  $\varepsilon_2$
- $\sigma_i = \text{def } m_i(y : \tau_2) : \tau$  **with**  $\varepsilon_3$

We do a case analysis on the reduction rules applicable to  $e_1.m_i(e_2)$ , for  $m_i$  an annotated method.

Subcase. E-METHCALL1 Then  $e_1 \longrightarrow e'_1 \mid \varepsilon$ . By inductive assumption  $e'_1 : \{\bar{\sigma}\}$  **with**  $\varepsilon'_1$ . Then by  $\varepsilon$ -METHCALL we have  $e_B = e'_1.m_i(e_2) : \tau_3$  **with**  $\varepsilon'_1 \cup \varepsilon_2 \cup \varepsilon_3$ . Then  $\varepsilon_B = \varepsilon'_1 \cup \varepsilon_2 \cup \varepsilon_3$  and  $\varepsilon_B \setminus \varepsilon_A = \varepsilon'_1 \setminus \varepsilon_1$ . Any lost effect info is accounted for by inductive assumption.

Subcase. E-METHCALL2 <sub>$\sigma$</sub>  Then  $e_1 = v_1 = \text{new}_\sigma x \Rightarrow \bar{\sigma} = \bar{e}$ , and  $e_2 \longrightarrow e'_2 \mid \varepsilon$ . By inductive assumption  $e'_2 : \tau_2$  **with**  $\varepsilon_2$ . Then by  $\varepsilon$ -METHCALL we have  $e_B = v_1.m_i(e_2) : \tau_3$  **with**  $\varepsilon_1 \cup \varepsilon'_2 \cup \varepsilon_3$ . Then  $\varepsilon_B = \varepsilon_1 \cup \varepsilon'_2 \cup \varepsilon_3$  and  $\varepsilon_B \setminus \varepsilon_A = \varepsilon'_2 \setminus \varepsilon_2$ . Any lost effect info is accounted for by inductive assumption.

Subcase. E-METHCALL3 <sub>$\sigma$</sub>  Then  $e_1 = v_1 = \text{new}_\sigma \Rightarrow \bar{\sigma} = \bar{e}$ , and  $\text{def } m_i(y : \tau_2) : \tau_3$  **with**  $\varepsilon_3 = e' \in \bar{\sigma} = \bar{e}$ , and  $e_2 = v_2$  is a value, and  $v_1.m_i(v_2) \longrightarrow [v_1/x, v_2/y]e' \mid \emptyset$ .

We already know  $e_1 : \{\bar{\sigma}\}$  **with**  $\varepsilon_1$ . The only rule with this conclusion is  $\varepsilon$ -NEWOBJ. By inversion,  $\bar{\sigma} = \bar{e}$  OK. The only rule with this conclusion is  $\varepsilon$ -VALIDIMPL <sub>$\sigma$</sub> . By inversion,  $e' : \tau_3$  **with**  $\varepsilon_3$ .

Now  $e_B = [v_1/x, v_2/y]e'$ , since the rule E-METHCALL3 was used. We know  $v_1 = e_1$  and  $x$  have the same type, which is  $\{\bar{\sigma}\}$  **with**  $\varepsilon_1$ . We also know  $v_2 = e_2$  and  $y$  have the same type, which is  $\tau_2$  **with**  $\varepsilon_2$ . By the substitution lemma, the type of  $e'$  is preserved under substitution. So  $e_B : \tau_3$  **with**  $\varepsilon_3$ .

Since  $e_1 = v_1$  and  $e_2 = v_2$  are values, by Canonical Forms  $\varepsilon_1 = \varepsilon_2 = \emptyset$ . So  $\varepsilon_A = \varepsilon_3$ . Then  $\varepsilon_B \setminus \varepsilon_A = \emptyset$  and there are no lost effects to account for.

**Case.**  $\varepsilon$ -OPERCALL $_{\sigma}$ .

Then  $e_A = e_1.\pi(e_2) : \mathbf{Unit}$ , and we know:

- $e_A : \{r, \pi\} \cup \varepsilon_1 \cup \varepsilon_2$
- $e_1 : \{\bar{r}\} \text{ with } \varepsilon_1$
- $e_2 : \tau_2 \text{ with } \varepsilon_2$
- $\pi \in \Pi$

There are three reduction rules applicable to terms of the form  $e_1.\pi(e_2)$  for  $\pi$  an operation. We consider each.

Subcase. E-OPERCALL. Then  $e_1 \longrightarrow e'_1 \mid \varepsilon$ . By inductive assumption,  $e'_1 : \{\bar{r}\} \text{ with } \varepsilon'_1$ . From these we can apply  $\varepsilon$ -OPERCALL, giving  $e_B = e'_1.\pi(e_2) : \mathbf{Unit} \text{ with } \{r.\pi\} \cup \varepsilon'_1 \cup \varepsilon_2$ . Then  $\varepsilon_B = \{r.\pi\} \cup \varepsilon'_1 \cup \varepsilon_2$  and  $\varepsilon_A \setminus \varepsilon_B = \varepsilon'_1$ . Any lost effect info from  $\varepsilon'_1$  is accounted for by inductive hypothesis.

Subcase. E-OPERCALL2. Then  $e_1 = r$  for some  $r \in R$  and  $e_2 \longrightarrow e'_2 \mid \varepsilon$ . By inductive assumption  $e'_2 : \tau_2 \text{ with } \varepsilon'_2$ . From these we can apply  $\varepsilon$ -OPERCALL, giving  $e_B = r.\pi(e'_2) \text{ with } \varepsilon$ . Then  $\varepsilon_B = r.\pi \cup \varepsilon_1 \cup \varepsilon'_2$  and  $\varepsilon_A \setminus \varepsilon_B = \varepsilon'_2$ . Any lost effect info from  $\varepsilon'_2$  is accounted for by inductive hypothesis.

Subcase. E-OPERCALL3. Then  $r.\pi(v) \longrightarrow \mathbf{unit} \mid \{r.\pi\}$ . By Canonical Forms,  $\varepsilon_1 = \varepsilon_2 = \emptyset$ , so  $e_A : \mathbf{Unit} \text{ with } \{r.\pi\}$ . By a degenerate case of  $\varepsilon$ -NEWOBJ,  $\varepsilon_B = \mathbf{unit} : \mathbf{Unit} \text{ with } \emptyset$ . Then  $\varepsilon_A \setminus \varepsilon_B = \{r.\pi\}$ . We can see that this is exactly the set of runtime effects  $\varepsilon$ , so we have accounted for the only lost effect.

□

### Theorem 5.5. (Small-Step Soundness For Mini Calculus)

**Statement.** Work in the mini calculus for fully-annotated programs. If  $\Gamma \vdash e_A : \tau_A \text{ with } \varepsilon_A$  and  $e \longrightarrow e_B \mid \varepsilon$ , then  $\varepsilon \subseteq \varepsilon_A$ .

**Proof.** By structural induction on the rule used to derive  $\Gamma \vdash e_A : \tau \text{ with } \varepsilon_A$ , and then on the reduction rule used.

**Case.**  $\varepsilon$ -RESOURCE,  $\varepsilon$ -VAR,  $\varepsilon$ -NEWOBJ.

$e_A$  is a value, so no reduction rules can be applied to it. The theorem statement is vacuously satisfied.

**Case.**  $\varepsilon$ -METHCALL $_{\sigma}$ .

Then  $e_A = e_1.m_i(e_2)$  and the following are known:

- $e_A : \tau_A \text{ with } \varepsilon_1 \cup \varepsilon_2 \cup \varepsilon_3$
- $e_1 : \{\bar{\sigma}\} \text{ with } \varepsilon_1$
- $e_2 : \tau_2 \text{ with } \varepsilon_2$
- $\sigma_i = \mathbf{def } m_i(y : \tau_2) : \tau_3 \text{ with } \varepsilon_3$

Consider the possible reduction rules.

Subcase. E-METHCALL1. Then  $e_1 \longrightarrow e'_1 \mid \varepsilon$ . By inductive assumption,  $\varepsilon \subseteq \varepsilon_1 \subseteq \varepsilon_A$ .

Subcase. E-METHCALL2. Then  $e_2 \longrightarrow e'_2 \mid \varepsilon$ . By inductive assumption,  $\varepsilon \subseteq \varepsilon_2 \subseteq \varepsilon_A$ .

Subcase. E-METHCALL3. Then  $v_1.m_i(v_2) \longrightarrow [v_1/x, v_2/y]e' \mid \emptyset$ . We're done as  $\emptyset \subseteq \varepsilon_A$ .

**Case.**  $\varepsilon$ -OPERCALL.

Then  $e_A = e_1.m_i(e_2)$  and the following are known:

- $e_A : \mathbf{unit} \text{ with } \varepsilon_1 \cup \varepsilon_2 \cup \{r.\pi\}$
- $e_1 : \{\bar{r}\} \text{ with } \varepsilon_1$
- $e_2 : \tau_2 \text{ with } \varepsilon_2$
- $\pi \in \Pi$

Consider the possible reduction rules.

Subcase. E-OPERCALL1 Then  $e_1 \longrightarrow e'_1 \mid \varepsilon$ . By induction assumption  $\varepsilon \subseteq \varepsilon_1 \subseteq \varepsilon_A$ .

Subcase. E-OPERCALL2 Then  $e_2 \longrightarrow e'_2 \mid \varepsilon$ . By induction assumption  $\varepsilon \subseteq \varepsilon_2 \subseteq \varepsilon_A$ .

Subcase. E-OPERCALL3 Then  $\exists r \mid e_1 = r$  and  $e_2 = v_2$  for some value and  $r.\pi_i(v_2) \longrightarrow \mathbf{unit} \mid \{r.\pi\}$ . By Canonical Forms,  $\varepsilon_1 = \varepsilon_2 = \emptyset$ . Then  $\varepsilon_A = \{r.\pi\}$ . We can see this is exactly the set of runtime effects.

□

### Theorem 5.6. (Big-Step Soundness For Mini Calculus)

**Statement.** Work in the mini calculus for fully-annotated programs. If  $\Gamma \vdash e_A : \tau_A$  **with**  $\varepsilon_A$  and  $e \longrightarrow_* e_B \mid \varepsilon$  then  $\varepsilon \subseteq e_A$ .

**Proof.** If the big-step involves zero steps then the theorem is vacuously satisfied. Otherwise the big-step involves more than one step. Induct on the number of steps.

**Base Case.** The length is 1. The theorem holds by Small-Step Soundness For Mini Calculus.

**Inductive Case.** If there is a big-step of length  $n+1$  then by E-BIGSTEP3 it can be decomposed into a big-step of length  $n$ ;  $e_1 \longrightarrow_* e_n \mid \varepsilon_n$ ; and a big-step of length 1;  $e_n \longrightarrow_* e_{n+1} \mid \varepsilon$ . Then  $e_1 \longrightarrow_* e_{n+1} \mid \varepsilon_n \cup \varepsilon$  is the entire big-step.

Let the type derivations be  $\Gamma \vdash e_1 : \tau_1$  **with**  $\varepsilon_1$  and  $\Gamma \vdash e_n$  **with**  $\varepsilon_n$

By inductive assumption on the smaller big-steps, we have that  $\varepsilon \subseteq \varepsilon_n$  and  $\varepsilon_n \subseteq \varepsilon_1$ . Then  $\varepsilon_1 \cup \varepsilon_n \subseteq \varepsilon_1$ .  $\varepsilon_1$  is the static effect information at the start of the big-step, so we're done.

□

### Theorem 5.6. (Extension)

**Statement.** If the following are true:

- $v_1 = \mathbf{new} \ x \Rightarrow d_i = e_i$
- $d_i = \mathbf{def} \ m_i(y : \tau_2) : \tau_3$
- $\Gamma \vdash v_2 : \tau_2$  **with**  $\varepsilon_2$
- $[v_1/x, v_2/y]e_i \longrightarrow_* v \mid \varepsilon$

Then  $\exists \varepsilon_i \mid \varepsilon \subseteq \varepsilon_i \subseteq \mathbf{effects}(\Gamma)$ . Letting  $\sigma_i = d_i$  **with**  $\varepsilon_i$ , then also  $\Gamma \vdash (\mathbf{new} \ x \Rightarrow \sigma_i = e_i) : \{\sigma_i = e_i\}$  **with**  $\emptyset$ .

**Note.** We only consider an object with a single method  $m_i$ . The result holds for objects with multiple methods by extending each method  $m_i$  individually to  $m'_i$  as per this theorem, then using them to collectively build a new annotated object  $v'_1$  containing the methods  $m'_i$ .

**Proof.** Hiya

□

### Theorem 5.7. (Soundness)

**Statement.** If  $\Gamma \vdash e_A : \tau$  **with**  $\varepsilon$  and  $e_A \rightarrow e_B \mid \varepsilon_B$ , then  $\varepsilon_B \subseteq \varepsilon$ .

**Proof.**

□

## 6 OLD OUTDATED STUFF

Please ignore this section, it is for Aaron to clean up at some point.

### Theorem 5.5. (Old Type Preservation)

**Statement.** If  $e_A : \tau$  **with**  $\varepsilon$  and  $e_A \mid \varepsilon_A \longrightarrow e_B \mid \varepsilon_B$ , then  $e_B : \tau$  **with**  $\varepsilon$ .

**Proof.** We first induct on possible derivations of  $e_A : \tau$  **with**  $\varepsilon$ , and then on the rule used to reduce  $e_A \mid \varepsilon_A$  to  $e_B \mid \varepsilon_B$ .

**Case.** C-METHCALL.

Then  $e_A = e_1.m_i(e_2)$  and the following are known:

- $e_A : \tau$  **with**  $\varepsilon_1 \cup \varepsilon_2 \cup \text{effects}(\tau_2) \cup \varepsilon$
- $e_1 : \{\bar{d} \text{ captures } \varepsilon\}$  **with**  $\varepsilon_1$
- $e_2 : \tau_2$  **with**  $\varepsilon_2$
- $d_i = \text{def } m_i(y : \tau_2) : \tau$

We do a case analysis on the reduction rules applicable to  $e_1.m_i(e_2)$ , for  $m_i$  an unannotated method.

Subcase. E-METHCALL1 Then  $e_1.m_i(e_2) \mid \varepsilon_A \longrightarrow e'_1.m_i(e_2) \mid \varepsilon_B$ . By the inductive assumption  $e'_1$  types to the same as  $e_1$ . Then applying C-METHCALL we get  $\text{type}(e_B) = \text{type}(e_A)$ .

Subcase. E-METHCALL2<sub>d</sub> Then  $e_1 = v_1$  some value and  $v_1.m_i(e_2) \mid \varepsilon_A \longrightarrow v_1.m_i(e'_2)$ . By the inductive assumption  $\text{type}(e'_2) = \text{type}(e_2)$ . Then applying C-METHCALL we get  $\text{type}(e_B) = \text{type}(e_A)$ .

Subcase. E-METHCALL3<sub>d</sub> Then  $v_1.m_i(v_2) \mid \varepsilon_A \longrightarrow [v_1/x, v_2/y]e_i \mid \varepsilon_B$ , where  $e_i$  is the body of method  $m_i$ . Now  $e_1$  has the type  $\{\bar{d} \text{ captures } \varepsilon\}$  **with**  $\varepsilon_1$ , and the only rule matching this judgement is C-NEWOBJ. So the premises of that rule, applied to  $e_1$ , must be true.

Firstly this means  $\bar{d} = e$  OK, so  $d_i = e_i$  OK.

**But what happens to the effects?  $e_i$  and  $\tau$  could be anything and there are no rules for effect-checking isolated expressions. Is there a smarter way than proceeding by case analysis on  $e_i$  and  $\tau$ ?**

□

### Theorem 5.6. (Soundness Of Effect Typing).

**Statement.** If  $e_A : \tau$  **with**  $\varepsilon$  and  $e_A \mid \varepsilon_A \longrightarrow e_B \mid \varepsilon_B$ , then  $\varepsilon_B \setminus \varepsilon_A \subseteq \varepsilon$ .

**Proof.** By the Effect Preservation Theorem,  $\varepsilon_A \subseteq \varepsilon_B$ . Now proceed by structural induction on the evaluation rule.

**Case.** E-METHCALL3<sub>d</sub>, E-METHCALL3 <sub>$\sigma$</sub> .

In these rules  $e_A = e_B$ . Then  $e_B \setminus e_A = \emptyset$ , so  $e_B \setminus e_A \subseteq \varepsilon$  is vacuously true.

**Case.** E-OPERCALL1.

Then  $e_1 \mid \varepsilon_A \longrightarrow e'_1 \mid \varepsilon_B$ . Theorem holds by the inductive assumption.

**Case.** E-OPERCALL2.

Then  $\exists r \in R \mid e_1 = r$  and  $r.\pi(e_2) \longrightarrow r.\pi(e'_2)$  and  $e_2 \mid \varepsilon_A \longrightarrow e'_2 \mid \varepsilon_B$ . Theorem holds by the inductive assumption.

**Case.** E-OPERCALL3.

Then  $\varepsilon_B = \varepsilon_A \cup \{r.\pi\}$ , so we have to show  $r.\pi \in \varepsilon$ . From the Atom lemma,  $r : \{r\}$  **with**  $\emptyset$  and  $v : \tau_2$  **with**  $\emptyset$ , for some  $\tau_2$ . By applying  $\varepsilon$ -OPERCALL then  $r.m(\pi) : \{r.\pi\} \cup \emptyset \cup \emptyset$ . So  $\varepsilon = \{r.\pi\}$ .

**Case.** E-METHCALL1.

Then  $e_1 \mid \varepsilon_A \longrightarrow e'_1 \mid \varepsilon_B$ . Theorem holds by the inductive assumption.

**Case.** E-METHCALL2 <sub>$\sigma$</sub> .

Then  $e_1 = v$  some value and  $e_2 \mid \varepsilon_A \longrightarrow e'_2 \mid \varepsilon_B$ . Theorem holds by the inductive assumption.

**Case.** E-METHCALL2<sub>d</sub>.

Then  $e_1 = v = \text{new}_d x \Rightarrow \bar{d} = e$  and  $e_2 \mid \varepsilon \longrightarrow e'_2 \mid \varepsilon_B$ . Theorem holds by inductive assumption.

□