

## 1 Grammar

$e ::= x$	<i>expressions</i>
$r$	
$\mathbf{new}_\sigma x \Rightarrow \overline{\sigma} = \overline{e}$	
$\mathbf{new}_d x \Rightarrow \overline{d} = e$	
$e.m(e)$	
$e.\pi$	
$\tau ::= \{\overline{\sigma}\}$	<i>types</i>
$\{\overline{r}\}$	
$\{\overline{d}\}$	
$\{\overline{d} \text{ captures } \varepsilon\}$	
$\sigma ::= d \text{ with } \varepsilon$	<i>labeled decls.</i>
$d ::= \mathbf{def } m(x : \tau) : \tau$	<i>unlabeled decls.</i>

### 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.
- $\{\overline{\sigma}\}$  is the type of an annotated object.  $\{\overline{d}\}$  is the type of an unannotated object.
- $\{\overline{d} \text{ captures } \varepsilon\}$  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  $\{\overline{d}\}$ .

## 2 Semantics

### 2.1 Static Semantics

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

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

$$\frac{\Gamma \vdash e_1 : \{\overline{r}\}}{\Gamma \vdash e_1.\pi : \mathbf{Unit}} \text{ (T-OPERCALL)}$$

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

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

$$\frac{\Gamma, x : \{\overline{\sigma}\} \vdash \overline{\sigma} = \overline{e} \text{ OK}}{\Gamma \vdash \mathbf{new}_\sigma x \Rightarrow \overline{\sigma} = \overline{e} : \{\overline{\sigma}\}} \text{ (T-NEW}_\sigma\text{)} \qquad \frac{\Gamma, x : \{\overline{d}\} \vdash \overline{d} = e \text{ OK}}{\Gamma \vdash \mathbf{new}_d x \Rightarrow \overline{d} = e : \{\overline{d}\}} \text{ (T-NEW}_d\text{)}$$

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

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

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

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

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

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

$$\frac{\Gamma, x : \{\bar{\sigma}\} \vdash \bar{\sigma} = \bar{e} \text{ OK}}{\Gamma \vdash \text{new}_\sigma x \Rightarrow \bar{\sigma} = \bar{e} : \{\bar{\sigma}\} \text{ with } \emptyset} \quad (\varepsilon\text{-NEWOBJ}) \qquad \frac{\Gamma \vdash e_1 : \{\bar{r}\} \text{ with } \varepsilon_1}{\Gamma \vdash e_1.\pi : \text{Unit with } \{\bar{r}.\pi\} \cup \varepsilon_1} \quad (\varepsilon\text{-OPERCALL})$$

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

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

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

$$\frac{\Gamma' \subseteq \Gamma \quad \Gamma' \vdash e : \tau}{\Gamma \vdash e : \tau \text{ with effects}(\Gamma')} \quad (\text{C-INFERENCE})$$

### Notes:

- This system includes all the rules from the fully-annotated system.
- The T rules do standard typing of objects, without any effect analysis. Their sole purpose is so  $\varepsilon\text{-ValidImpl}_d$  can be applied. **We are assuming the T-rules on their own are sound.**
- In C-NEWOBJ,  $\Gamma'$  is intended to be some subcontext of the current  $\Gamma$ . The object is labelled as capturing the effects in  $\Gamma'$  (exact definition in the next section).
- In C-NEWOBJ we must add  $\text{effects}(\tau_2)$  to the static effects of the object, because the method body will have access to the resources captured by  $\tau_2$  (the type of the argument passed into the method).
- A good choice of  $\Gamma'$  would be  $\Gamma$  restricted to the free variables in the object definition.
- The purpose of C-INFERENCE is to ascribe static effects to unannotated portions of code (for instance, the body of an unlabeled method).
- As a useful convention we'll often use  $\varepsilon_c$  to denote the output of the **effects** function.

## 2.2 effects Function

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

- $\text{effects}(\emptyset) = \emptyset$
- $\text{effects}(\Gamma, x : \tau) = \text{effects}(\Gamma) \cup \text{effects}(\tau)$
- $\text{effects}(\{\bar{r}\}) = \{(r, \pi) \mid r \in \bar{r}, \pi \in \Pi\}$
- $\text{effects}(\{\bar{\sigma}\}) = \bigcup_{\sigma \in \bar{\sigma}} \text{effects}(\sigma)$
- $\text{effects}(\{\bar{d}\}) = \bigcup_{d \in \bar{d}} \text{effects}(d)$

- $\text{effects}(d \text{ with } \varepsilon) = \varepsilon \cup \text{effects}(d)$
- $\text{effects}(\text{def } m(x : \tau_1) : \tau_2) = \text{effects}(\tau_2)$
- $\text{effects}(\{\bar{d} \text{ captures } \varepsilon_c\}) = \varepsilon_c$

**Notes:**

- Since a method can return a capability for a resource  $r$  we need to figure out what the return type of a method captures. This requires a recursive crawl through the definitions and types inside it.
- In the last case we don't want to recurse to sub-declarations because the effects have already been captured previously (this is  $\varepsilon_c$ ) by a potentially different context.

### 2.3 Dynamic Semantics

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

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

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

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

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

$$\frac{e_1 \longrightarrow e'_1 \mid \varepsilon}{e_1.\pi \longrightarrow e'_1.\pi \mid \varepsilon} \text{ (E-OPERCALL1)} \quad \frac{}{r.\pi \longrightarrow \mathbf{unit} \mid \{r.\pi\}} \text{ (E-OPERCALL2)}$$

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

$$\frac{}{e \longrightarrow_* e \mid \emptyset} \text{ (E-MULTISTEP1)} \quad \frac{e \longrightarrow e' \mid \varepsilon}{e \longrightarrow_* e' \mid \varepsilon} \text{ (E-MULTISTEP2)}$$

$$\frac{e \longrightarrow_* e' \mid \varepsilon_1 \quad e' \longrightarrow_* e'' \mid \varepsilon_2}{e \longrightarrow_* e'' \mid \varepsilon_1 \cup \varepsilon_2} \text{ (E-MULTISTEP3)}$$

**Notes:**

- E-METHCALL2<sub>d</sub> and E-METHCALL2<sub>σ</sub> are really doing the same thing, but one applies to labeled objects (the  $\sigma$  version) and the other on unlabeled objects. Same goes for E-METHCALL3<sub>σ</sub> and E-METHCALL3<sub>d</sub>.
- E-METHCALL1 can be used for both labeled and unlabeled objects.

### 2.4 Substitution Function

We extend our Substitution function from the previous system in a straightforward way by adding a new case for unlabeled objects.

- $[e'/z]z = e'$
- $[e'/z]y = y$ , if  $y \neq z$
- $[e'/z]r = r$
- $[e'/z](e_1.m(e_2)) = ([e'/z]e_1).m([e'/z]e_2)$
- $[e'/z](e_1.\pi) = ([e'/z]e_1).\pi$
- $[e'/z](\text{new}_d x \Rightarrow \overline{d=e}) = \text{new}_d x \Rightarrow \overline{\sigma = [e'/z]e}$ , if  $z \neq x$  and  $z \notin \text{freevars}(e_i)$
- $[e'/z](\text{new}_\sigma x \Rightarrow \overline{\sigma=e}) = \text{new}_\sigma x \Rightarrow \overline{\sigma = [e'/z]e}$ , if  $z \neq x$  and  $z \notin \text{freevars}(e_i)$

### 3 Proofs

#### Lemma 3.1. (Canonical Forms)

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

- If  $\Gamma \vdash e : \{\bar{r}\} \text{ with } \varepsilon$ , then  $e = r$  for some resource  $r$ .
- If  $\Gamma \vdash e : \{\bar{\sigma}\} \text{ with } \varepsilon$ , then  $e = \text{new}_\sigma x \Rightarrow \overline{\sigma=e}$ .
- If  $\Gamma \vdash e : \{\bar{d} \text{ captures } \varepsilon_c\} \text{ with } \varepsilon$ , then  $e = \text{new}_d x \Rightarrow \overline{d=e}$ .

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 3.2. (Substitution Lemma)

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

**Intuition** If you substitute  $z$  for something of the same type, the type of the whole expression stays the same after substitution.

**Proof.** We've already proven the lemma by structural induction on the  $\varepsilon$  rules. The new case is defined on a form not in the grammar for the fully-annotated system. So all that remains is to induct on derivations of  $\Gamma \vdash e : \tau \text{ with } \varepsilon$  using the new C rules.

**Case.** C-METHCALL.

Then  $e = e_1.m(e_2)$  and  $[e'/z]e = ([e'/z]e_1).m([e'/z]e_2)$ . By inductive assumption we know that  $e_1$  and  $[e'/z]e_1$  have the same types, and that  $e_2$  and  $[e'/z]e_2$  have the same types. Since  $e$  and  $[e'/z]e$  have the same syntactic struture, and their corresponding subexpressions have the same types, then  $\Gamma$  can use C-METHCALL to type  $[e'/z]e$  the same as  $e$ .

**Case.** C-INFERENCE.

Then  $\Gamma \vdash e : \tau \text{ with effects}(\Gamma')$ , where  $\Gamma' \subseteq \Gamma$ . By inversion  $\Gamma' \vdash e : \tau$ . Applying the inductive hypothesis (and our assumption that the T rules are sound)  $\Gamma' \vdash [e'/z]e : \tau$ . Since  $\Gamma' \subseteq \Gamma$  we have  $\Gamma' \vdash [e'/z]e : \tau \text{ with effects}(\Gamma')$  under C-INFERENCE. Because  $\Gamma' \subseteq \Gamma$  then  $\Gamma \vdash [e'/z]e : \tau \text{ with effects}(\Gamma')$ .

**Case.** C-NEWOBJ.

Then  $e = \text{new}_d x \Rightarrow \overline{d=e}$ .  $z$  appears in some method body  $e_i$ . By inversion we know  $\Gamma, x : \{\bar{\sigma}\} \vdash \overline{d=e}$  OK. The only rule with this conclusion is  $\varepsilon$ -VALIDIMPL<sub>d</sub>; by inversion on that we know for each  $i$  that:

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

If  $z$  appears in the body of  $e_i$  then  $\Gamma, z : \tau \vdash d_i = e_i$  OK by inductive assumption. Then we can use  $\varepsilon$ -VALIDIMPL<sub>d</sub> to conclude  $\overline{d = [e'/z]e}$  OK. This tells us that the types and static effects of all the methods are unchanged under substitution. By choosing the same  $\Gamma' \subseteq \Gamma$  used in the original application of C-NEWOBJ, we can apply C-NEWOBJ to the expression after substitution. The types and static effects the methods are the same, and the same  $\Gamma'$  has been chosen, so  $[e'/z]e$  will be ascribed the same type as  $e$ .

□

### Lemma 3.3. (Monotonicity of effects)

**Statement.** If  $\Gamma_1 \subseteq \Gamma_2$  then  $\mathbf{effects}(\Gamma_1) \subseteq \mathbf{effects}(\Gamma_2)$

**Proof.** Because  $\mathbf{effects}(\Gamma_1)$  is the union of  $\mathbf{effects}(\tau)$ , for every  $(x, \tau) \in \Gamma_1 \subseteq \Gamma_2$ . Then  $\mathbf{effects}(\Gamma_1) \subseteq \mathbf{effects}(\Gamma_2)$ .

□

### Lemma 3.4. (Use Principle)

**Statement.** If  $\Gamma \vdash e_A : \tau_A$  with  $\varepsilon_A$ , and  $e_A \longrightarrow_* e'_A \mid \varepsilon$ , then  $\forall r. \pi \in \varepsilon \mid (r, \{r\}) \in \Gamma$ . Furthermore,  $\varepsilon \subseteq \mathbf{effects}(\Gamma)$ .

**Proof.** The only reduction that can add effects to  $\varepsilon$  is  $r.\pi$ . So at some point, an expression of the form  $r.\pi$  must have been evaluated. In the source program it must have had the form  $e.\pi$ . Since the entire program typechecked under  $\Gamma$ ,  $e$  must have been typed to  $\{r\}$  at some point. Since resources cannot be dynamically created,  $(r, \{r\}) \in \Gamma$ . Since every resource with an operation called upon it is  $\Gamma$ ,  $\varepsilon \subseteq \mathbf{effects}(\Gamma)$  follows by the definition of  $\mathbf{effects}$  for the case of a resource.

**Intuition.** If you typecheck  $e$  with  $\Gamma$ , if an effect can happen on  $r$  when executing  $e$  then  $r$  must be in  $\Gamma$ .

□

### Lemma 3.5. (Tightening Lemma)

**Statement.** If  $\Gamma \vdash e : \tau$  with  $\varepsilon$  then  $\Gamma \cap \mathbf{freevars}(e) \vdash e : \tau$  with  $\varepsilon$ .

**Proof.** The typing judgements operate on the form of  $e$ , so don't consider any variables external to  $e$ .

□

**Note.** We'll use  $\mathbf{freevars}(e) \cap \Gamma$  to mean  $\Gamma$ , where the pair  $(x, \tau)$  is thrown out if  $x \notin \mathbf{freevars}(e)$ .

**Intuition.** If you can typecheck  $e$  in  $\Gamma$ , you can throw out the parts in  $\Gamma$  not relevant to  $e$  and still typecheck it.

### Definition 3.6. (label)

Given a program containing unlabeled parts we can safely label those parts. This process is well-defined if  $\Gamma \vdash e : \tau$ ; then we say the labeling of  $e$  is  $\mathbf{label}(\Gamma, e) = \hat{e}$ .

- $\mathbf{label}(r, \Gamma) = r$
- $\mathbf{label}(x, \Gamma) = x$
- $\mathbf{label}(e_1.m(e_2), \Gamma) = \mathbf{label}(e_1, \Gamma).m(\mathbf{label}(e_2), \Gamma)$
- $\mathbf{label}(e_1.\pi(e_2), \Gamma) = \mathbf{label}(e_1, \Gamma).\pi(\mathbf{label}(e_2), \Gamma)$
- $\mathbf{label}(\mathbf{new}_\sigma x \Rightarrow \overline{\sigma = e}, \Gamma) = \mathbf{new}_\sigma x \Rightarrow \mathbf{label\_helper}(\overline{\sigma = e}, \Gamma)$
- $\mathbf{label}(\mathbf{new}_d x \Rightarrow \overline{d = e}, \Gamma) = \mathbf{new}_d x \Rightarrow \mathbf{label\_helper}(\overline{d = e}, \Gamma)$
- $\mathbf{label\_helper}(\sigma = e, \Gamma) = \sigma = \mathbf{label}(e, \Gamma)$
- $\mathbf{label\_helper}(\mathbf{def } m(y : \tau_2) : \tau_3 = e, \Gamma) = \mathbf{def } m(y : \tau_2) : \tau_3 \text{ with } \mathbf{effects}(\Gamma \cap \mathbf{freevars}(e)) = \mathbf{label}(e, \Gamma)$

#### Notes:

- $\Gamma \cap \mathbf{freevars}(e)$  is the set of pairs  $x : \tau \in \Gamma$ , such that  $x \in \mathbf{freevars}(e)$ .
- $\mathbf{label}(e, \Gamma)$  is read as: “the labeling of  $e$  in  $\Gamma$ ”.
- Often the  $\Gamma$  we use is obvious in context; in such cases we write  $\mathbf{label}(e)$  instead of  $\mathbf{label}(e, \Gamma)$ .
- Beware of confusing notation: there are two types of equality in the above definitions. One is the equality which defines  $\mathbf{label}$ , and the other is the equality  $\sigma = e$  of declarations in the programming language.
- The program after labeling will be fully-labeled, so typing it will be sound under the  $\varepsilon$  rules.

- `label` is defined on expressions; `label-helper` on declarations. Everywhere other than this section we'll only use `label`.
- Initially it seems like `label` on a `newσ` object should just be the identity function; but the body of the methods of such an object may instantiate unlabeled objects and/or call methods on unlabeled objects, so we must recursively label those.
- We may sometimes say `labels(e) = ê`, and from then on refer to the labeled version of  $e$  as  $\hat{e}$ . We'll use  $\hat{\tau}$  and  $\hat{\varepsilon}$  to refer to the type and static effects of the labeled version.

### Observations 3.7.

**Statement.** The following are true.

- `label(e)` is a value if and only if  $e$  is a value.
- If  $e$  has type  $\{\bar{\sigma}\}$ , then for any method  $m_i \in \{\bar{\sigma}\}$  with a label  $\varepsilon_i$ , the exact same method and label will occur in  $\hat{e}$ .

**Proof.** By inspection of the definition of `label`.

□

### Property 3.8. (Commutativity Between `label` and `sub`)

**Statement.** Fix  $\Gamma$  and define `label(e) = label(e, Γ)`. Then `label([e'/z]e) = [label(e')/z](label(e))`

**Intuition.** If perform substitution and labeling on an expression, the order in which you do things doesn't matter.

**Proof.** Induction on the form of  $e$ . In each case, “left-hand side” refers to `label([e'/z]e)` while “right-hand side” refers to `[label(e')/z](label(e))`.

**Case.**  $e = r$ .

By definition, `label(r) = r` and  $[e'/z]r = r$ , for any  $e'$ . Both sides are equivalent to  $r$  because `sub` and `label` act like the identity function.

**Case.**  $e = x$ .

By definition, `label(x) = x`.  $[e'/z]x$  has two definitions, depending on if  $x = z$ ; consider each case.

**Subcase.**  $x \neq z$ . Then  $[e'/z]x = x$ . Both sides are equivalent to  $x$  because `sub` and `label` act like the identity function.

**Subcase.**  $x = z$ . Then  $[e'/z]x = z$ . On the left-hand side, `label([e'/z]x) = label(e')`. On the right-hand side, `[label(e')/z]x = label(e')`.

**Case.**  $e = e_1.\pi$ .

On the left-hand side.

$$\begin{aligned}
 & \text{label}([e'/z](e_1.\pi)) \\
 &= \text{label}([e'/z]e_1).\pi && \text{(definition of sub)} \\
 &= (\text{label}([e'/z]e_1)).\pi && \text{(definition of label)} \\
 &= ([\text{label}(e')/z](\text{label}(e_1))).\pi && \text{(inductive assumption on } e_1)
 \end{aligned}$$

On the right-hand side.

$$\begin{aligned}
 & [\text{label}(e')/z](\text{label}(e_1.\pi)) \\
 &= [\text{label}(e')/z](\text{label}(e_1).\pi) && \text{(definition of label)} \\
 &= ([\text{label}(e')/z](\text{label}(e_1))).\pi && \text{(definition of sub)}
 \end{aligned}$$

**Case.**  $e = e_1.m(e_2)$ .

On the left-hand side.

$$\begin{aligned}
& \text{label}([e'/z](e_1.m(e_2))) \\
&= \text{label}([e'/z]e_1).m([e'/z]e_2) && \text{(definition of sub)} \\
&= (\text{label}([e'/z]e_1)).m(\text{label}([e'/z]e_2)) && \text{(definition of label)} \\
&= ([\text{label}(e')/z](\text{label}(e_1)).m(\text{label}([e'/z]e_2))) && \text{(inductive assumption on } e_1) \\
&= ([\text{label}(e')/z](\text{label}(e_1)).m([\text{label}(e')/z](\text{label}(e_2)))) && \text{(inductive assumption on } e_2)
\end{aligned}$$

On the right-hand side.

$$\begin{aligned}
& [\text{label}(e')/z](\text{label}(e_1.m(e_2))) \\
&= [\text{label}(e')/z](\text{label}(e_1).m(\text{label}(e_2))) && \text{(definition of label)} \\
&= ([\text{label}(e')/z](\text{label}(e_1)).m([\text{label}(e')/z](\text{label}(e_2)))) && \text{(definition of sub)}
\end{aligned}$$

**Case.**  $e = \text{new}_\sigma x \Rightarrow \overline{\sigma = e}$ .

On the left-hand side.

$$\begin{aligned}
& \text{label}([e'/z](\text{new}_\sigma x \Rightarrow \overline{\sigma_i = e_i})) \\
&= \text{label}(\text{new}_\sigma x \Rightarrow \overline{\sigma_i = [e'/z]e_i}) && \text{(definition of sub)} \\
&= \text{new}_\sigma x \Rightarrow \overline{\text{label-helper}(\sigma_i = [e'/z]e_i)} && \text{(definition of label)} \\
&= \text{new}_\sigma x \Rightarrow \sigma_i = \text{label}([e'/z]e_i) && \text{(definition of label-helper on each } \sigma_i = [e'/z]e_i)
\end{aligned}$$

On the right-hand side.

$$\begin{aligned}
& [\text{label}(e')/z](\text{label}(\text{new}_\sigma x \Rightarrow \overline{\sigma_i = e_i})) \\
&= [\text{label}(e')/z](\text{new}_\sigma x \Rightarrow \overline{\text{label-helper}(\sigma_i = e_i)}) && \text{(definition of label)} \\
&= [\text{label}(e')/z](\text{new}_\sigma x \Rightarrow \sigma_i = \text{label}(e_i)) && \text{(definition of label-helper on each } \sigma_i = e_i) \\
&= \text{new}_\sigma x \Rightarrow \overline{\sigma_i = [\text{label}(e')/z](\text{label}(e_i))} && \text{(definition of sub)} \\
&= \text{new}_\sigma x \Rightarrow \sigma_i = \text{label}([e'/z]e_i) && \text{(inductive assumption on each } e_i)
\end{aligned}$$

**Case.**  $e = \text{new}_d x \Rightarrow \overline{d = e}$ .

The proof of this is quite similar to previous case for labeled objects. The main difference is that when labeling an unlabeled object, each  $d_i = e_i$  turns into a  $\sigma_i = e_i$ . For clarity we will define  $\varepsilon_i = \text{effects}(\Gamma \cap \text{freevars}(e_i))$ , and  $\sigma_i = d_i$  with  $\varepsilon_i$  (these are from the definition of label-helper).

On the left-hand side.

$$\begin{aligned}
& \text{label}([e'/z](\text{new}_d x \Rightarrow \overline{d_i = e_i})) \\
&= \text{label}(\text{new}_d x \Rightarrow \overline{d_i = [e'/z]e_i}) && \text{(definition of sub)} \\
&= \text{new}_d x \Rightarrow \overline{\text{label-helper}(d_i = [e'/z]e_i)} && \text{(definition of label)} \\
&= \text{new}_d x \Rightarrow \overline{d_i \text{ with } \varepsilon_i = \text{label}([e'/z]e_i)} && \text{(definition of label-helper)} \\
&= \text{new}_d x \Rightarrow \sigma_i = \text{label}([e'/z]e_i) && (\sigma_i = d_i \text{ with } \varepsilon_i)
\end{aligned}$$

On the right-hand side.

$$\begin{aligned}
& [\text{label}(e')/z](\text{label}(\text{new}_d x \Rightarrow \overline{d_i = e_i})) \\
&= [\text{label}(e')/z](\text{new}_d x \Rightarrow \overline{\text{label-helper}(d_i = e_i)}) && \text{(definition of label)} \\
&= [\text{label}(e')/z](\text{new}_\sigma x \Rightarrow \overline{d_i \text{ with } \varepsilon_i = \text{label}(e_i)}) && \text{(definition of label-helper on each } d_i = e_i) \\
&= [\text{label}(e')/z](\text{new}_\sigma x \Rightarrow \overline{\sigma_i = \text{label}(e_i)}) && (\sigma_i = d_i \text{ with } \varepsilon_i) \\
&= \text{new}_\sigma x \Rightarrow \overline{\sigma_i = [\text{label}(e')/z](\text{label}(e_i))} && \text{(definition of sub)} \\
&= \text{new}_\sigma x \Rightarrow \sigma_i = \text{label}([e'/z]e_i) && \text{(inductive assumption on each } e_i)
\end{aligned}$$

□

### Lemma 3.9. (Runtime Invariance Under label)

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

- $\Gamma \vdash e_A : \tau_A$  with  $\varepsilon_A$
- $e_A \longrightarrow e_B \mid \varepsilon$
- $\hat{e}_A = \text{label}(e_A, \Gamma)$

Then  $\hat{e}_A \longrightarrow \hat{e}_B \mid \varepsilon$  and  $\hat{e}_B = \mathbf{label}(e_B, \Gamma)$ .

**Proof.** Induct on the form of  $e_A$  and then on the reduction rule  $e_A \longrightarrow e_B \mid \varepsilon$ . Throughout this proof there is only a single context  $\Gamma$ , so we'll write  $\mathbf{label}(e)$  instead of  $\mathbf{label}(e, \Gamma)$  as a notational short-hand.

**Case.**  $e = r, e = x, e = \mathbf{new}_\sigma x \Rightarrow \overline{\sigma} = \overline{e}, e = \mathbf{new}_d x \Rightarrow \overline{d} = \overline{e}$ .

Then  $e$  is a value and the theorem statement holds automatically.

**Case.**  $e = e_1.\pi$ .

The only typing rule which applies is  $\varepsilon$ -OPERCALL, which tells us:

- $\Gamma \vdash e_1 : \{r\} \text{ with } \varepsilon_1$
- $\Gamma \vdash e_1.\pi : \mathbf{Unit} \text{ with } \varepsilon_1 \cup \{r.\pi\}$

There are two possible reductions.

**Subcase.** E-OPERCALL1. We also know  $e_1 \longrightarrow e'_1 \mid \varepsilon$ , and  $e_1.\pi \longrightarrow e'_1.\pi \mid \varepsilon$ . By inductive assumption,  $\hat{e}_1 \longrightarrow \hat{e}'_1 \mid \varepsilon$ , and  $\hat{e}'_1 = \mathbf{label}(e'_1)$ . Applying definitions,  $\hat{e}_A = \mathbf{label}(e_1.\pi) = (\mathbf{label}(e_1)).\pi = \hat{e}_1.\pi$ . Because  $\hat{e}_1 \longrightarrow \hat{e}'_1 \mid \varepsilon$ , we may apply the reduction E-OPERCALL1 to obtain  $\hat{e}_1.\pi \longrightarrow \hat{e}'_1.\pi \mid \varepsilon$ . Lastly,  $\hat{e}_B = \mathbf{label}(e'_1.\pi) = (\mathbf{label}(e'_1)).\pi$ , which we know to be  $\hat{e}'_1.\pi$  by inductive assumption.

**Subcase.** E-OPERCALL2. We also know  $e_1 = r$  and  $r.\pi \longrightarrow \mathbf{Unit} \mid \{r.\pi\}$ . Applying definitions,  $\hat{e}_A = \mathbf{label}(r.\pi) = (\mathbf{label}(r)).\pi = r.\pi = e_A$ . The theorem holds immediately.

**Case.**  $e = e_1.m_i(e_2)$ .

There are five possible reductions.

**Subcase.** E-METHCALL1. We also know  $e_1 \longrightarrow e'_1 \mid \varepsilon$  and  $e_1.m_i(e_2) \longrightarrow e'_1.m_i(e_2) \mid \varepsilon$ . By inductive assumption,  $\hat{e}_1 \longrightarrow \hat{e}'_1 \mid \varepsilon$ , and  $\mathbf{label}(e'_1) = \hat{e}'_1$ . Applying definitions  $\hat{e}_A = \mathbf{label}(e_1.m_i(e_2)) = (\mathbf{label}(e_1)).m_i(\mathbf{label}(e_2)) = \hat{e}_1.m_i(\hat{e}_2)$ . Because  $\hat{e}_1 \longrightarrow \hat{e}'_1 \mid \varepsilon$ , we may apply the reduction E-METHCALL1 to obtain  $\hat{e}_1.m_i(\hat{e}_2) \longrightarrow \hat{e}'_1.m_i(\hat{e}_2) \mid \varepsilon$ . Lastly,  $\hat{e}_B = \mathbf{label}(e'_1.m_i(\hat{e}_2)) = (\mathbf{label}(e'_1)).m_i(\mathbf{label}(e_2))$ , which we know to be  $\hat{e}'_1.m_i(\hat{e}_2) = \hat{e}_B$  by assumptions.

**Subcase.** E-METHCALL2 $_\sigma$ . We also know  $e_1 = v_1 = \mathbf{new}_\sigma x \Rightarrow \overline{\sigma} = \overline{e}$ , and  $e_2 \longrightarrow e'_2 \mid \varepsilon$  and  $v_1.m_i(e_2) \longrightarrow v_1.m_i(e'_2) \mid \varepsilon$ . By inductive assumption,  $\hat{e}_2 \longrightarrow \hat{e}'_2 \mid \varepsilon$ , and  $\mathbf{label}(e'_2) = \hat{e}'_2$ . Applying definitions,  $\hat{e}_A = \mathbf{label}(v_1.m_i(e_2)) = (\mathbf{label}(v_1)).m_i(\mathbf{label}(e_2)) = \hat{v}_1.m_i(\hat{e}_2)$ . Because  $\hat{e}_2 \longrightarrow \hat{e}'_2 \mid \varepsilon$ , we may apply the reduction E-METHCALL2 $_\sigma$  to obtain  $\hat{v}_1.m_i(\hat{e}_2) \longrightarrow \hat{v}_1.m_i(\hat{e}'_2)$ . Lastly,  $\hat{e}_B = \mathbf{label}(v_1.m_i(e'_2)) = (\mathbf{label}(v_1)).m_i(\mathbf{label}(e'_2))$ , which we know to be  $\hat{v}_1.m_i(\hat{e}'_2)$  by assumptions.

**Subcase.** E-METHCALL2 $_d$ . Identical to the above subcase, but  $e_1 = v_1 = \mathbf{new}_d x \Rightarrow \overline{d} = \overline{e}$ , and we apply the reduction rule E-METHCALL $_d$  instead.

**Subcase.** E-METHCALL3 $_\sigma$ . We also know the following:

- $e_1 = v_1 = \mathbf{new}_\sigma x \Rightarrow \overline{\sigma} = \overline{e}$
- $e_2 = v_2$
- $\mathbf{def} \ m_i(y : \tau_2) : \tau_3 \text{ with } \varepsilon_3 = e_{body} \in \{\overline{\sigma}\}$
- $v_1.m_i(v_2) \longrightarrow [v_1/x, v_2/y]e_{body} \mid \emptyset$ .

Applying definitions,  $\mathbf{label}(v_1.m_i(v_2)) = (\mathbf{label}(v_1)).m_i(\mathbf{label}(v_2)) = \hat{v}_1.m_i(\hat{v}_2)$ , where we define  $\hat{v}_1 = \mathbf{label}(v_1)$  and  $\hat{v}_2 = \mathbf{label}(v_2)$ . Before labeling, the object  $v_1$  has method  $m_i$  with body  $e_{body}$ . The labeled version,  $\hat{v}_1$ , has method  $m_i$  with body  $\mathbf{label}(e_{body}) = \hat{e}_{body}$ . Because  $v_1$  and  $v_2$  are values, so are  $\hat{v}_1$  and  $\hat{v}_2$ . Therefore we can apply E-METHCALL3 $_\sigma$  to  $\hat{v}_1.m_i(\hat{v}_2)$ , giving us  $\hat{v}_1.m_i(\hat{v}_2) \longrightarrow [\hat{v}_1/x, \hat{v}_2/y]\hat{e}_{body} \mid \emptyset$ . Because  $\mathbf{label}$  and  $\mathbf{sub}$  commute,  $\mathbf{label}(e_B) = \mathbf{label}([v_1/x, v_2/y]e_{body}) = [\mathbf{label}(v_1)/x, \mathbf{label}(v_2)/y](\mathbf{label}(e_{body}))$ , which is  $[\hat{v}_1/x, \hat{v}_2/y]\hat{e}_{body} = \hat{e}_B$ , by how we defined  $\hat{v}_1$ ,  $\hat{v}_2$ , and  $\hat{e}_{body}$ .



Subcase. E-METHCALL3<sub>d</sub>. This case is identical to the previous one, except  $e_1 = v_1 = \mathbf{new}_d x \Rightarrow \overline{d = e}$ . The same reasoning applies though.  $\square$

### Theorem 3.11. (Extension Theorem)

**Statement.** If  $\Gamma \vdash e : \tau$  and  $\hat{e} = \mathbf{label}(e, \Gamma)$  then one of the following is true:

- $e$  is a value, and  $\Gamma \vdash \hat{e} : \hat{\tau}$  **with**  $\hat{\varepsilon}$ , where  $\tau = \hat{\tau}$  and  $\hat{\varepsilon} = \emptyset$ .
- $e$  is an expression, and  $e \longrightarrow e' \mid \varepsilon$ , and  $\Gamma \vdash \hat{e} : \hat{\tau}$  **with**  $\hat{\varepsilon}$ , where  $\hat{\tau} = \tau$  and  $\varepsilon \subseteq \hat{\varepsilon}$ .

**Intuition.** If  $\Gamma$  can type  $e$  without an effect, there is a way to label  $e$  with  $\hat{e}$  which contains the possible runtime effects of  $e$  (so  $\hat{\varepsilon}$  is an upper-bound).

**Todo/Notes**

- **effects( $\Gamma$ ) is an upper-bound on  $\hat{\varepsilon}$ , but we omit this from the proof (for now) to keep it simple.**
- **In the second case of the theorem statement we can use Invariance of Runtime Under label and soundness of fully-annotated programs; to apply that to  $e$  we need to know  $e$  can be typed with an effect.**

**Proof.** Throughout this proof there is only one  $\Gamma$ , so we say  $\mathbf{label}(e)$  as short-hand for  $\mathbf{label}(e, \Gamma)$ . Proceed by induction on  $\Gamma \vdash e : \tau$  and then on the reduction  $e \longrightarrow e' \mid \varepsilon$ .

**Case.** T-VAR.

$e = x$  is a value, and  $\mathbf{label}(x) = x$ . By assumption that the program is closed under  $\Gamma$ , we can apply  $\varepsilon$ -VAR to conclude  $\Gamma \vdash x : \tau$  **with**  $\emptyset$ .

**Case.** T-RESOURCE.

$e = r$  is a value, and  $\mathbf{label}(r) = r$ . By assumption that the program is closed under  $\Gamma$ , we can apply  $\varepsilon$ -RESOURCE to conclude  $\Gamma \vdash r : \{r\}$  **with**  $\emptyset$ .

**Case.** T-NEW <sub>$\sigma$</sub> .

We also know  $e = \mathbf{new}_\sigma x \Rightarrow \overline{\sigma \equiv e}$  and  $\Gamma \vdash \sigma_i = e_i$  OK. By applying the definition of  $\mathbf{label}$ , define  $\hat{e} = \mathbf{label}(\mathbf{new}_\sigma x \Rightarrow \overline{\sigma \equiv e}) = \mathbf{new}_\sigma x \Rightarrow \overline{\sigma = \mathbf{label}(e)}$ . To type  $\hat{e}$  we want to use  $\varepsilon$ -NEWOBJ; to do that we need to know  $\Gamma, x : \{\bar{\sigma}\} \vdash \sigma = \mathbf{label}(e)$  OK.

Fix some  $i$ . By assumption,  $\Gamma \vdash \sigma_i = e_i$  OK. By inversion on  $\varepsilon$ -VALIDIMPL <sub>$\sigma$</sub> , we know  $\Gamma, y : \tau_2 \vdash e_i : \tau_3$  **with**  $\varepsilon_3$ . Consider  $\hat{e}_i = \mathbf{label}(e_i)$ . By inductive assumption,  $\Gamma, y : \tau_2 \vdash \hat{e}_i : \tau_3$  **with**  $\hat{\varepsilon}$ , and by application of  $\varepsilon$ -VALIDIMPL <sub>$\sigma$</sub>  we know  $\Gamma \vdash \sigma_i = \mathbf{label}(e_i)$  OK. **(We're applying inductive assumption to something of the form  $\Gamma \vdash e : \tau$  with  $\varepsilon$ , not  $\Gamma \vdash e : \tau$  though. To do this rigorously we need to know  $\Gamma \vdash e : \tau$  with  $\varepsilon \implies \Gamma \vdash e : \tau$ )**

$i$  was arbitrary; therefore  $\Gamma \vdash \overline{\sigma = \mathbf{label}(e)}$  OK. Therefore  $\Gamma \vdash \hat{e} : \{\bar{\sigma}\}$  **with**  $\emptyset$  by  $\varepsilon$ -NEWOBJ <sub>$\sigma$</sub> .

**Case.** T-NEW <sub>$d$</sub> .

We also know  $e = \mathbf{new}_d x \Rightarrow \overline{d = e}$  and  $\Gamma, \{\bar{d}\} \vdash d_i = e_i$  OK. To simplify things, let  $\varepsilon_i = \mathbf{freevars}(\Gamma) \cap e_i$  (this definition comes from the definition of  $\mathbf{label-helper}$ ) and define  $\hat{e}$  in the following way:

$$\begin{aligned}
 & \mathbf{label}(e) \\
 &= \mathbf{label}(\mathbf{new}_d x \Rightarrow \overline{d = e}) && \text{(definition of } e) \\
 &= \mathbf{new}_\sigma x \Rightarrow \overline{\mathbf{label-helper}(d_i = e_i)} && \text{(definition of } \mathbf{label}) \\
 &= \mathbf{new}_\sigma x \Rightarrow \overline{d_i \text{ with } \varepsilon_i = e_i} && \text{(definition of } \mathbf{label-helper}) \\
 &= \mathbf{new}_\sigma x \Rightarrow \overline{\sigma_i \equiv e_i} && \text{(defining } \sigma = d_i \text{ with } \varepsilon_i)
 \end{aligned}$$

To type  $\hat{e}$  we want to use  $\varepsilon$ -NEWOBJ; to do that we need to know  $\Gamma, x : \{\sigma\} \vdash \overline{\sigma_i \equiv e_i}$  OK, so fix some  $i$ . By assumption  $\Gamma \vdash d_i = e_i$  OK. By inversion on  $\varepsilon$ -VALIDIMPL <sub>$d$</sub>  we know  $\Gamma, y : \tau_2 \vdash e_i : \tau_3$ . By inductive assumption on this,  $\Gamma, y : \tau_2 \vdash \hat{e}_i : \tau_3$  **with**  $\hat{\varepsilon}$ .

Fix some  $i$ . By assumption  $\Gamma, \{\bar{d}\} \vdash \bar{d}_i = e_i$  OK. By inversion on  $\varepsilon\text{-VALIDIMPL}_d$ , we know  $\Gamma, \{\bar{d}\}, y : \tau_2 \vdash e_i : \tau_3$ . By inductive assumption,  $\Gamma, \{\bar{d}\}, y : \tau_2 \vdash \hat{e}_i : \tau_3$  with  $\hat{\varepsilon}$ , and by an application of  $\varepsilon\text{-VALIDIMPL}_\sigma$  we know  $\Gamma \vdash \sigma_i = \text{label}(e_i)$  OK.

$i$  was arbitrary; therefore  $\Gamma \vdash \sigma = \text{label}(e)$  OK. Therefore  $\Gamma \vdash \hat{e} : \{\bar{\sigma}\}$  with  $\emptyset$  by  $\varepsilon\text{-NEWOBJ}$ .

**Case.** T-OPERCALL.

Then the following are known:

- $e = e_1.\pi$
- $\Gamma \vdash e_1 : \{\bar{r}\}$
- $\Gamma \vdash e_1.\pi : \text{Unit}$

There are two reduction rules which could be applied to  $e_1.\pi$ .

Subcase. E-OPERCALL1. Then we know  $e_1.\pi \longrightarrow e'_1.\pi \mid \varepsilon$ , and  $e_1 \rightarrow e'_1 \mid \varepsilon$ . Because  $\Gamma \vdash e_1 : \{\bar{r}\}$  by assumption of the typing rule, we may apply the inductive assumption. Then  $\Gamma \vdash \hat{e}_1 : \{\bar{r}\}$  with  $\hat{\varepsilon}_1$ , where  $\varepsilon \subseteq \hat{\varepsilon}_1$  and  $\hat{e}_1 = \text{label}(e_1)$ .

By definition  $\hat{e} = \text{label}(\Gamma, e) = \text{label}(\Gamma, e_1.\pi) = (\text{label}(\Gamma, e_1)).\pi = \hat{e}_1.\pi$ . Because  $\Gamma \vdash \hat{e}_1 : \{\bar{r}\}$  with  $\hat{\varepsilon}$  we can apply  $\varepsilon\text{-OPERCALL}$  and type  $\hat{e} = \hat{e}_1.\pi$  with the judgement  $\Gamma \vdash \hat{e}_1.\pi : \text{Unit}$  with  $\{r.\pi\} \cup \hat{\varepsilon}_1$ .

$\varepsilon \subseteq \hat{\varepsilon}_1$  is an inductive assumption; so  $\varepsilon \subseteq \hat{\varepsilon}_1 \cup \{r.\pi\} = \hat{\varepsilon}$ . Also,  $\hat{\tau} = \text{Unit} = \tau$ .

Subcase. E-OPERCALL2. Then we know  $e = r.\pi$  and  $r.\pi \longrightarrow \text{Unit} \mid \{r.\pi\}$ . By definition  $\hat{e} = \text{label}(\Gamma, e) = (\text{label}(\Gamma, r)).\pi = r.\pi = e$ , so  $\hat{e} = e$ . Then  $\hat{\tau} = \tau$  automatically. We need only show  $\varepsilon = r.\pi \in \hat{\varepsilon}$ .

By  $\varepsilon\text{-RESOURCE}$ ,  $\Gamma \vdash r : \{r\}$  with  $\emptyset$  and by  $\varepsilon\text{-OPERCALL}$ ,  $\Gamma \vdash r.\pi : \text{Unit}$  with  $\{r.\pi\}$ . Since  $\hat{e} = r.\pi$ , then  $\hat{\varepsilon} = r.\pi = \varepsilon$ .

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

Then the following are known:

- $e = e_1.m_i(e_2)$
- $\Gamma \vdash e_1 : \{\bar{\sigma}\}$
- $\Gamma \vdash e_2 : \tau_2$
- $\Gamma \vdash e_1.m_i(e_2) : \tau_3$
- **def**  $m_i(y : \tau_2) : \tau_3$  with  $\varepsilon_3 \in \{\bar{\sigma}\}$

There are three reduction rules which could be applied to a method call  $e_1.m_i(e_2)$  on a labeled object.

Subcase. E-METHCALL1. Then we know  $e_1 \longrightarrow e'_1 \mid \varepsilon$  and  $e_1.m_i(e_2) \longrightarrow e'_1.m_i(e_2) \mid \varepsilon$ . By definition,  $\hat{e} = \text{label}(e_1.m_i(e_2)) = (\text{label}(e_1)).m_i(\text{label}(e_2)) = \hat{e}_1.m_i(\hat{e}_2)$ . Method  $m_i$  has the same label  $\varepsilon_3$  in  $\hat{e}$  as it does in  $e$ .

Because  $\Gamma \vdash e_1 : \{\bar{\sigma}\}$  and  $\Gamma \vdash e_2 : \tau_2$ , by applying the inductive assumption to each we learn  $\Gamma \vdash \hat{e}_1 : \{\bar{\sigma}\}$  with  $\hat{\varepsilon}_1$  and  $\Gamma \vdash \hat{e}_2 : \tau_2$  with  $\hat{\varepsilon}_2$ , where  $\varepsilon \subseteq \hat{\varepsilon}_1$ .

Putting this all together and using  $\varepsilon\text{-METHCALL}$  we learn  $\Gamma \vdash \hat{e}_1.m_i(\hat{e}_2) : \tau_3$  with  $\hat{\varepsilon}_1 \cup \hat{\varepsilon}_2 \cup \varepsilon_3$ , and since  $\varepsilon \subseteq \hat{\varepsilon}_1$  this type contains the run-time effects.

Subcase. E-METHCALL2 $_\sigma$ . Then we know  $e_1 = v_1 = \text{new}_\sigma x \Rightarrow \bar{\sigma} \equiv \bar{e}$  and  $e_2 \longrightarrow e'_2 \mid \varepsilon$  and  $e_1.m_i(e_2) \longrightarrow e'_1.m_i(e_2) \mid \varepsilon$ . By definition,  $\hat{e} = \text{label}(e_1.m_i(e_2)) = (\text{label}(e_1)).m_i(\text{label}(e_2)) = \hat{e}_1.m_i(\hat{e}_2)$ . Method  $m_i$  has the same label  $\varepsilon_3$  in  $\hat{e}$  as it does in  $e$ .

Because  $\Gamma \vdash v_1 : \{\bar{\sigma}\}$  and  $\Gamma \vdash e_2 : \tau_2$ , by applying the inductive assumption to each we learn  $\Gamma \vdash \hat{e}_1 : \{\bar{\sigma}\}$  with  $\hat{\varepsilon}_1$  and  $\Gamma \vdash \hat{e}_2 : \tau_2$  with  $\hat{\varepsilon}_2$ , where  $\varepsilon \subseteq \hat{\varepsilon}_2$ .

Putting this all together and using  $\varepsilon$ -METHCALL we learn  $\Gamma \vdash \hat{e}_1.m_i(\hat{e}_2) : \tau_3$  **with**  $\hat{\varepsilon}_1 \cup \hat{\varepsilon}_2 \cup \varepsilon_3$ , and since  $\varepsilon \subseteq \hat{\varepsilon}_1$  this type contains the run-time effects.

Subcase. E-METHCALL3 $_{\sigma}$ . Then we know  $e_1 = v_1 = \text{new}_{\sigma} x \Rightarrow \overline{\sigma} = \overline{e}$  and  $e_2 = v_2$  is a value. Also,  $v_1.m_i(v_2) \longrightarrow [v_1/x, v_2/y]e_{\text{body}} \mid \emptyset$ . By definition,  $\hat{e} = \text{label}(v_1.m_i(v_2)) = (\text{label}(v_1)).m_i(\text{label}(v_2)) = \hat{v}_1.m_i(\hat{v}_2)$ .

Applying the inductive assumption to  $v_1$  and  $v_2$ , we get  $\Gamma \vdash v_1 : \{\bar{\sigma}\}$  **with**  $\hat{\varepsilon}_1$  and  $\Gamma \vdash v_2 : \{\bar{\sigma}\}$  **with**  $\hat{\varepsilon}_2$ . Because  $v_1$  was already a labeled object, the method  $m_i$  has the same label  $\varepsilon_3$  in  $\hat{v}_1$  as it does in  $v_1$ . By an application of  $\varepsilon$ -METHCALL we learn  $\Gamma \vdash v_1.m_i(v_2) : \tau_3$  **with**  $\hat{\varepsilon}_1 \cup \hat{\varepsilon}_2 \cup \varepsilon_3$ . Since  $\varepsilon = \emptyset$  this type contains all the runtime effects.

**Case.** T-METHCALL $_d$ .

Then the following are known:

- $e = e_1.m_i(e_2)$
- $\Gamma \vdash e_1 : \{\bar{d}\}$
- $\Gamma \vdash e_2 : \tau_2$
- $\Gamma \vdash e_1.m_i(e_2) : \tau_3$
- **def**  $m_i(y : \tau_2) : \tau_3 \in \{\bar{d}\}$

There are three reduction rules which could be applied to a method call  $e_1.m_i(e_2)$  on an unlabeled object.

Subcase. E-METHCALL1. Then we know  $e_1 \longrightarrow e'_1 \mid \varepsilon$  and  $e_1.m_i(e_2) \longrightarrow e'_1.m_i(e_2) \mid \varepsilon$ . By definition,  $\hat{e} = \text{label}(e_1.m_i(e_2)) = (\text{label}(e_1)).m_i(\text{label}(e_2)) = \hat{e}_1.m_i(\hat{e}_2)$ .

Because  $\Gamma \vdash e_1 : \{\bar{d}\}$  and  $\Gamma \vdash e_2 : \tau_2$ , by applying the inductive assumption to each we learn  $\Gamma \vdash \hat{e}_1 : \{\bar{\sigma}\}$  **with**  $\hat{\varepsilon}_1$  and  $\Gamma \vdash \hat{e}_2 : \tau_2$  **with**  $\hat{\varepsilon}_2$ . **At this point we need to justify that first inductive assumption on  $e_1$ . We need to know  $\hat{e}_1$  still has the method  $m_i$  as part of its type (and that the new label  $\varepsilon_i$  is actually effect-safe)**

**We would want to apply  $\varepsilon$ -MethCall, giving us  $\Gamma \vdash \hat{e}_1.m_i(\hat{e}_2) : \tau_3$  **with**  $\hat{\varepsilon}_1 \cup \hat{\varepsilon}_2 \cup \varepsilon_3$ , where  $\varepsilon_3$  is the label on  $m_i$ . This would be  $\text{effects}(\Gamma) \cap e_i$ , where  $e_i$  is the body of  $m_i$ , so this type has the effect-set  $\hat{\varepsilon}_1 \cup \hat{\varepsilon}_2 \cup (\text{effects}(\Gamma) \cap e_i)$ .  $\varepsilon \subseteq \hat{\varepsilon}_1$  is an inductive assumption, so this type contains the runtime effects.**

Subcase. E-METHCALL3 $_d$ . Then we know  $e_1 = v_1 = \text{new}_d x \Rightarrow \overline{d} = \overline{e}$  and  $e_2 = v_2$  and  $v_1.m_i(v_2) \longrightarrow [v_1/x, v_2/y]e \mid \emptyset$ .

□

### Theorem 3.12. (Refinement Theorem)

**Statement.** If  $\Gamma \vdash e : \tau$  **with**  $\varepsilon$  and  $\text{label}(e) = \hat{e}$ , then one of the following is true:

- $\Gamma \vdash \hat{e} : \hat{\tau}$  **with**  $\hat{\varepsilon}$ , where  $\hat{\varepsilon} \subseteq \varepsilon$  and  $\tau = \hat{\tau}$
- $e$  has the form  $\text{new}_d x \Rightarrow \overline{d} = \overline{e}$  and  $\Gamma \vdash \hat{e} : \overline{d}_i$  **with**  $\varepsilon_i = e_i$ , where  $\varepsilon_i = \text{effects}(\Gamma) \cap e_i$

**Intuition.** Labels can only make the static effects more precise; never less precise.

**Proof.** By induction on the judgement  $\Gamma \vdash e : \tau$  **with**  $\varepsilon$ .

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

If  $e$  is a resource or a variable then  $e = \hat{e}$  so the statement is automatically fulfilled.

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

Then  $e = e_1.\pi$  and we know:

- $\Gamma \vdash e : \text{Unit}$  **with**  $\{r.\pi\} \cup \varepsilon_1$
- $\Gamma \vdash e_1 : \{\bar{r}\}$  **with**  $\varepsilon_1$

Applying definitions,  $\hat{e} = \text{label}(e_1.\pi) = (\text{label}(e_1)).\pi = \hat{e}_1.\pi$ . By inductive assumption,  $\Gamma \vdash \hat{e}_1 : \{\bar{r}\} \text{ with } \hat{\varepsilon}_1$ , where  $\hat{\varepsilon}_1 \subseteq \varepsilon_1$ . Then  $\Gamma \vdash \hat{e} : \text{Unit with } \{r.\pi\} \cup \hat{\varepsilon}_1$  by  $\varepsilon$ -OPERCALL. Importantly,  $\{r.\pi\} \cup \hat{\varepsilon}_1 \subseteq \{r.\pi\} \cup \varepsilon_1$  as claimed.

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

Then  $e = e_1.m_i(e_2)$  and we know:

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

Applying definitions,  $\hat{e} = \text{label}(e_1.m_i(e_2)) = (\text{label}(e_1)).m_i(\text{label}(e_2)) = \hat{e}_1.m_i(\hat{e}_2)$ . By inductive assumption,  $\Gamma \vdash \hat{e}_1 : \{\bar{\sigma}\} \text{ with } \hat{\varepsilon}_1$  and  $\Gamma \vdash \hat{e}_2 : \tau_2 \text{ with } \hat{\varepsilon}_2$ , where  $\hat{\varepsilon}_1 \subseteq \varepsilon_1$  and  $\hat{\varepsilon}_2 \subseteq \varepsilon_2$ . Then  $\Gamma \vdash \hat{e} : \tau_3 \text{ with } \hat{\varepsilon}_1 \cup \hat{\varepsilon}_2 \cup \varepsilon_3$  under  $\varepsilon$ -METHCALL. Importantly,  $\hat{\varepsilon}_1 \cup \hat{\varepsilon}_2 \cup \varepsilon_3 \subseteq \varepsilon_1 \cup \varepsilon_2 \cup \varepsilon_3$  as claimed.

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

Then  $e = \text{new}_\sigma x \Rightarrow \bar{\sigma} = e$  and we know:

- $\Gamma \vdash e : \{\bar{\sigma}\} \text{ with } \emptyset$
- $\Gamma, x : \{\bar{\sigma}\} \vdash \bar{\sigma} = e \text{ OK}$

For each  $i$ ,  $\sigma_i = e_i \text{ OK}$  only matches  $\varepsilon$ -VALIDIMPL $_\sigma$ . By inversion on that rule,  $\Gamma, y : \tau_2 \vdash e_i : \tau_3 \text{ with } \varepsilon_3$  and  $\sigma_i = \text{def } m_i(y : \tau_2) : \tau_3 \text{ with } \varepsilon_3$ . Applying definitions,  $\hat{e} = \text{label}(\text{new}_\sigma x \Rightarrow \bar{\sigma} = e) = \text{new}_\sigma x \Rightarrow \text{label-helper}(\bar{\sigma} = e)$ . Then for each  $i$ ,  $\text{label-helper}(\sigma_i = e_i) = \sigma_i = \text{label}(e_i)$ . Let  $\hat{e}_i = \text{label}(e_i)$ . Applying the inductive assumption to method body  $e_i$  we get  $\Gamma \vdash \hat{e}_i : \tau_3 \text{ with } \hat{\varepsilon}_3$ . Then  $\Gamma \vdash \sigma_i = \text{label}(e_i) \text{ OK}$  by  $\varepsilon$ -VALIDIMPL $_\sigma$ . This was for any  $i$ , so  $\Gamma \vdash \bar{\sigma}_i = \text{label}(e_i) \text{ OK}$ . Finally we can apply  $\varepsilon$ -NEWOBJ to the labeled object  $\text{new}_\sigma x \Rightarrow \bar{\sigma}_i = \text{label}(e_i)$ , which gives the judgement  $\Gamma \vdash \hat{e} : \{\bar{\sigma}\} \text{ with } \emptyset$ .

**Case.** C-METHCALL.

Then  $e = e_1.m_i(e_2)$  and we know:

- $\Gamma \vdash e : \tau_3 \text{ with } \varepsilon_1 \cup \varepsilon_2 \cup \varepsilon_3$
- $\Gamma \vdash e_1 : \{\bar{d} \text{ captures } \varepsilon_c\} \text{ with } \varepsilon_1$
- $\Gamma \vdash e_2 : \tau_2 \text{ with } \varepsilon_2$
- $d_i = \text{def } m_i(y : \tau_2) : \tau_3$

Similar reasoning above. Need to use inductive case on  $e_1$ , but to achieve that must first flesh out the exact theorem statement (base case more important: see C-NewObj below)

**Case.** C-NEWOBJ.

Then  $e = \text{new}_d x \Rightarrow \bar{d} = e$  and we know:

- $\Gamma' \subseteq \Gamma$
- $\varepsilon_c = \text{effects}(\Gamma')$  with  $\emptyset$
- $\Gamma', x : \{\bar{d} \text{ captures } \varepsilon_c\} \vdash \bar{d} = e \text{ OK}$

For each  $i$ ,  $d_i = e_i \text{ OK}$  only matches  $\varepsilon$ -VALIDIMPL $_d$ . By inversion on that rule,  $\Gamma, y : \tau_2 \vdash e : \tau_3$  and  $d_i = \text{def } m(y : \tau_2) : \tau_3 \text{ with } \varepsilon_3$ . Applying definitions,  $\hat{e} = \text{label}(\text{new}_\sigma x \Rightarrow \bar{\sigma} = e) = \text{new}_d x \Rightarrow \text{label-helper}(\bar{d} = e)$ . Then for each  $i$ ,  $\text{label-helper}(\text{def } m(y : \tau_2) : \tau_3 = e) = \text{def } m(y : \tau_2) : \tau_3 \text{ with effects}(\Gamma \cap \text{freevars}(e_i)) = \text{label}(e_i)$ . Let  $\hat{e}_i = \text{label}(e_i)$ .

To type the labeled object need to know  $\Gamma \vdash \sigma_i = e_i \text{ OK}$  in order to apply  $\varepsilon$ -NewObj, where  $\sigma_i$  is the signature on method  $m_i$  after labeling. We only know  $\Gamma \vdash d_i = e_i \text{ OK}$ .

□

### Theorem 3.13. (Soundness Theorem)

**Statement.** If  $\Gamma \vdash e_A : \tau_A \text{ with } \varepsilon_A$  and  $e_A \longrightarrow e_B \mid \varepsilon$  then  $\Gamma \vdash e_B : \tau_B \text{ with } \varepsilon_B$ , where  $\tau_B = \tau_A$  and  $\varepsilon \subseteq \varepsilon_A$ .

**Proof.**

Let  $\hat{e}_A = \text{label}(e_A)$ . By applying the Refinement theorem to  $e_A$  we know the following:

1.  $\Gamma \vdash \hat{e}_A : \hat{\tau}_A$  **with**  $\hat{\varepsilon}_A$
2.  $\tau_A = \hat{\tau}_A$
3.  $\hat{\varepsilon}_A \subseteq \varepsilon_A$

From Invariance of Runtime Under **label** we also know:

5.  $\hat{e}_A \longrightarrow \hat{e}_B \mid \varepsilon$
6. **label**( $e_B$ ) =  $\hat{e}_B$

Applying the Refinement theorem to  $e_B$  we get:

7.  $\Gamma \vdash \hat{e}_B : \tau_B$  **with**  $\hat{\varepsilon}_B$

$\hat{e}_A$  is a fully-labeled program. By the soundness of  $\varepsilon$  rules applied to reduction **5**:

8.  $\Gamma \vdash \hat{e}_B : \hat{\tau}_A$  **with**  $\hat{\varepsilon}_B$
9.  $\varepsilon \subseteq \hat{\varepsilon}_A$

**9** and **3** gives us effect-soundness.

10.  $\varepsilon \subseteq \hat{\varepsilon}_A$

Because of **2**, judgement **7** can be rewritten as:

11.  $\Gamma \vdash \hat{e}_B : \tau_A$  **with**  $\hat{\varepsilon}_B$

From **1** we know  $\hat{e}_A$  has type  $\hat{\tau}_A$ ; from **7** we know  $\hat{e}_B$  has type  $\tau_B$ . By type preservation of fully-labeled programs applied to **5** we know:

12.  $\hat{\tau}_A = \tau_B$

By comparing **2** and **12** we get type-soundness.

12.  $\tau_A = \hat{\tau}_A = \tau_B$

□