

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:

- σ denotes a declaration with effect labels; d a declaration without effect labels.
- \mathbf{new}_σ 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, ε is an upper-bound on the effects captured by $\{\overline{d}\}$.

2 Semantics

2.1 Static Semantics

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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, Γ' is intended to be some subcontext of the current Γ . The object is labelled as capturing the effects in Γ' (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 τ_2 (the type of the argument passed into the method).
- A good choice of Γ' would be Γ 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 ε_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 ε_c) by a potentially different context.

2.3 Dynamic Semantics

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

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

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

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

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

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

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

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

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

Notes:

- E-METHCALL2_d and E-METHCALL2_σ are really doing the same thing, but one applies to labeled objects (the σ version) and the other on unlabeled objects. Same goes for E-METHCALL3_σ and E-METHCALL3_d.
- 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 $y \neq z$
- $[e'/z]r = r$
- $[e'/z](e_1.m(e_2)) = ([e'/z]e_1).m([e'/z]e_2)$
- $[e'/z](e_1.\pi) = ([e'/z]e_1).\pi$
- $[e'/z](\text{new}_d x \Rightarrow \overline{d=e}) = \text{new}_\sigma x \Rightarrow \overline{\sigma = [e'/z]e}$, 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 ε -RESOURCE, ε -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 ε 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 structure, and their corresponding subexpressions have the same types, then Γ 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 ε -VALIDIMPL_d; 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 ε -VALIDIMPL_d 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 Γ' 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 ε_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 ε 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 Γ , 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 Γ , $\varepsilon \subseteq \mathbf{effects}(\Gamma)$ follows by the definition of $\mathbf{effects}$ for the case of a resource.

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

□

Lemma 3.5. (Tightening Lemma)

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

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 Γ , where the pair (x, τ) is thrown out if $x \notin \mathbf{freevars}(e)$.

Intuition. If you can typecheck e in Γ , you can throw out the parts in Γ not relevant to e and still typecheck it.

Theorem 3.6. (Extension Lemma)

Statement. If the following are true:

- $v_1 = \mathbf{new} \ x \Rightarrow d_i = e_i$
- $d_i = \mathbf{def} \ m_i(y : \tau_2) : \tau_3$
- $\Gamma \vdash d_i = e_i$ OK
- $\Gamma \vdash v_2 : \tau_2$ with ε_2
- $[v_1/x, v_2/y]e_i \longrightarrow_* e'_i \mid \varepsilon$

Then $\exists \varepsilon_i \mid \varepsilon \subseteq \varepsilon_i \subseteq \mathbf{effects}(\Gamma)$. Letting $\sigma_i = d_i$ with ε_i , then also $\sigma_i = e_i$ OK.

Intuition. This lemma says that we can take an unlabeled object v_1 with one method and produce a labeled object with one method, whose static effects contain all of the possible runtime effects. This ε_i is just going to be everything captured by the method body.

Note. In this theorem we only consider objects with a single method m_i . Later we generalise the result to objects with any number of methods.

Proof. First we need to type e_i in Γ with some set ε_i of static effects satisfying the theorem statement. Proceed by induction on the form of e_i .

By the Tightening Lemma, $\Gamma' \vdash d_i = e_i$ OK. The only rule with this conclusion is ε -VALIDIMPL_d. By inversion, $\Gamma', y : \tau_2 \vdash e_i : \tau_3$. Our goal is now to type e_i with an effect ε_i which satisfies the theorem statement. To do that we shall induct on the forms of e_i .

Case. $e_i = r$.

Since $\Gamma \vdash e_i : \tau_3$, if e_i is a resource the only rule that could have been applied is T-RESOURCE. From that rule we may conclude $r \in \Gamma$. Then by application of ε -RESOURCE we know that $\Gamma \vdash e_i : \tau_3$ with \emptyset . Resources are values and cannot be reduced; the multi-step must be zero small steps, hence $\varepsilon = \emptyset = \varepsilon' \subseteq \mathbf{effects}(\Gamma)$.

Case. $e_i = e_1.\pi$.

By induction $\Gamma \vdash e_1 : \tau_1$ with ε_1 .

This lets us apply C-INFERERENCE, with the entirety of Γ' as our subcontext. Then $\Gamma' \vdash e_i : \tau_3$ with **effects** (Γ'). By the Tightening Lemma again, $\Gamma' \vdash v_2 : \tau_2$ with ε_2 . By the Substitution Lemma, $\Gamma' \vdash [v_1/x, v_2/y]e_i : \tau_3$ with **effects** (Γ').

Since $[v_1/x, v_2/y]e_i \longrightarrow_* e'_i \mid \varepsilon$ and Γ' can typecheck the expression before reduction, then $\varepsilon \subseteq \mathbf{effects}(\Gamma')$. Since $\Gamma' \subseteq \Gamma$, by monotonicity, $\varepsilon \subseteq \mathbf{effects}(\Gamma') \subseteq \mathbf{effects}(\Gamma)$. So $\mathbf{effects}(\Gamma')$ is a witness to the ε_i in the statement of the lemma.

Finally, $\Gamma' \vdash \sigma_i = e_i$ under ε_σ . Since $\Gamma' \subseteq \Gamma$, then $\Gamma \vdash \sigma_i = e_i$ also.

□

Lemma 3.7. (Extension Theorem)

Statement. If $v_1 = \mathbf{new} \ x \Rightarrow \overline{d = e}$ and $\forall i$ the following are true:

- $d_i = \mathbf{def} \ m_i(y : \tau_2) : \tau_3$
- $\Gamma \vdash d_i = e_i$ OK
- $\Gamma \vdash v_2 : \tau_2$ with ε_2
- $[v_1/x, v_2/y]e_i \longrightarrow_* e'_i \mid \varepsilon$

Then $\exists \varepsilon_1, \dots, \varepsilon_n \mid \varepsilon \subseteq \varepsilon_i \subseteq \mathbf{effects}(\Gamma)$. Letting $\sigma_i = d_i$ with ε_i then also $\Gamma \vdash \mathbf{new} \ x \Rightarrow \overline{\sigma = e}$ with \emptyset

Intuition. Any unlabeled object can be extended to a labeled object, whose labels contain every possible runtime effect.

Proof. From Single-Method Extension we know that $\forall i \exists \varepsilon_i \mid \varepsilon \subseteq \varepsilon_i \subseteq \mathbf{effects}(\Gamma)$, and that the statement holds for each i . It also tells us that $\sigma_i = e_i$ OK, so $\overline{\sigma = e}$ OK. Then by ε -NEWOBJ we have the claimed typing judgement.

□

Definition 3.8. (label)

The Extension Theorem essentially says that any unlabeled program can be extended to a fully-labeled one, whose labels give a conservative upper-bound on the possible runtime effects. We define a program-transforming function called **label** which does this.

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

Notes:

- The program after labeling will be fully-labeled and contain terms entirely from the grammar for fully-labeled programs. Hence we can appeal to the soundness of that system.
- **label** is defined on expressions; **label-helper** on declarations. This is just for clarity; everywhere other than this section we'll only use **label**.
- Initially it seems like **label** on a **new_σ** 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.
- From here on out we will use \hat{e} to refer to a fully-labeled program. We may sometimes say **labels**(e) = \hat{e} , and from then on refer to the labeled version of e . At least until we've established this transformation preserves types, we'll use $\hat{\tau}$ and $\hat{\varepsilon}$ to refer to the types and effects of \hat{e} .

Theorem 3.9. (Refinement Theorem)

Statement. If $\Gamma \vdash e : \tau$ **with** ε and **label**(e) = \hat{e} , then $\Gamma \vdash \hat{e} : \hat{\tau}$ **with** $\hat{\varepsilon}$, where $\hat{\varepsilon} \subseteq \varepsilon$ and $\tau = \hat{\tau}$.

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

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

Case. ε -RESOURCE, ε -VAR.

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

Case. ε -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** ε_1

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

Case. ε -METHCALL.

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

- $\Gamma \vdash e : \tau_3$ **with** $\varepsilon_1 \cup \varepsilon_2 \cup \varepsilon_3$
- $\Gamma \vdash e_1 : \{\bar{\sigma}\}$ **with** ε_1
- $\Gamma \vdash e_2 : \tau_2$ **with** ε_2
- $\sigma_i = \text{def } m_i(y : \tau_2) : \tau_3$ **with** ε_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, $\Gamma \vdash \hat{e}_1 : \{\bar{\sigma}\}$ **with** $\hat{\varepsilon}_1$ and $\Gamma \vdash \hat{e}_2 : \tau_2$ **with** $\hat{\varepsilon}_2$, where $\hat{\varepsilon}_1 \subseteq \varepsilon_1$ and $\hat{\varepsilon}_2 \subseteq \varepsilon_2$. Then $\Gamma \vdash \hat{e} : \tau_3$ **with** $\hat{\varepsilon}_1 \cup \hat{\varepsilon}_2 \cup \varepsilon_3$ under ε -METHCALL. Importantly, $\hat{\varepsilon}_1 \cup \hat{\varepsilon}_2 \cup \varepsilon_3 \subseteq \varepsilon_1 \cup \varepsilon_2 \cup \varepsilon_3$ as claimed.

Case. C-METHCALL.

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

- $\Gamma \vdash e : \tau_3$ **with** $\varepsilon_1 \cup \varepsilon_2 \cup \varepsilon_3$
- $\Gamma \vdash e_1 : \{\bar{d} \text{ captures } \varepsilon_c\}$ **with** ε_1
- $\Gamma \vdash e_2 : \tau_2$ **with** ε_2
- $d_i = \text{def } m_i(y : \tau_2) : \tau_3$

The reasoning is the same as the above case, but use C-METHCALL instead of ε -METHCALL.

Case. C-INFERENCE.

We know:

- $\Gamma' \subseteq \Gamma$
- $\Gamma' \vdash e : \tau$
- $\Gamma \vdash e : \tau$ **with effects**(Γ')

There aren't any judgements of the form $e : \tau$ **with** ε in the antecedent of this rule so we cannot use the induction hypothesis. We will instead do a case-by-case analysis of the form of e .

Subcase. $e = r$ or $e = x$. Then $e = \hat{e}$ so the statement holds immediately.

Subcase. $e = e_1.\pi$. Then $\hat{e} = (\hat{e}_1).\pi = \hat{e}_1.\pi$. As e_1 is a subexpression of e , and since Γ can type e_1 , we may conclude $\Gamma \vdash e_1 : \{r\}$. By an application of C-INFERENCE choosing the same Γ' , we know $\Gamma \vdash e_1 : \{r\}$ with $\text{effects}(\Gamma')$. By applying the inductive hypothesis to e_1 we know that $\Gamma \vdash \hat{e}_1 : \{r\}$ with $\hat{\varepsilon}_1$, where $\hat{\varepsilon}_1 \subseteq \text{effects}(\Gamma')$. Therefore $\Gamma \vdash \hat{e}_1 : \tau_1$. By an application of T-OPERCALL we know that

This one's kind of interesting. There aren't any judgements of the form $e : \tau$ with ε in the antecedent of this rule, so we can't use the induction hypothesis. We also don't know anything about e .

Case. ε -NEWOBJ.

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

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

For each i , $\sigma_i = e_i$ OK only matches ε -VALIDIMPL $_\sigma$. By inversion on that rule, $\Gamma, y : \tau_2 \vdash e : \tau_3$ with ε_3 and $\sigma_i = \text{def } m_i(y : \tau_2) : \tau_3$ with ε_3 . Applying definitions, $\hat{e} = \text{label}(\text{new}_\sigma x \Rightarrow \overline{\sigma} = \overline{e}) = \text{new}_\sigma x \Rightarrow \text{label-helper}(\overline{\sigma} = \overline{e})$. 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 we get $\Gamma \vdash \hat{e}_i : \tau_3$ with $\hat{\varepsilon}_3$. Then $\Gamma \vdash \sigma_i = \text{label}(e_i)$ OK by ε -VALIDIMPL $_\sigma$. This was for any i , so $\Gamma \vdash \overline{\sigma}_i = \text{label}(e_i)$ OK. Finally we can apply ε -NEWOBJ to the labeled object $\sigma_i = \text{label}(e_i)$, which gives the judgement $\Gamma \vdash \hat{e} : \{\overline{\sigma}\}$ with \emptyset .

Case. C-NEWOBJ.

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

- $\Gamma \vdash e_1.m_i(e_2) : \tau_3$ with $\varepsilon_1 \cup \varepsilon_2 \cup \varepsilon_3$
- $\Gamma' \subseteq \Gamma$
- $\varepsilon_c = \text{effects}(\Gamma')$ with \emptyset
- $\Gamma', x : \{\overline{d} \text{ captures } \varepsilon_c\} \vdash \overline{d} = \overline{e}$ OK

(Similar to above). For each i , $d_i = e_i$ OK only matches ε -VALIDIMPL $_d$. By inversion on that rule, $\Gamma, y : \tau_2 \vdash e : \tau_3$ and $d_i = \text{def } m(y : \tau_2) : \tau_3$ with ε_3 . Applying definitions, $\hat{e} = \text{label}(\text{new}_d x \Rightarrow \overline{d} = \overline{e}) = \text{new}_d x \Rightarrow \text{label-helper}(\overline{d} = \overline{e})$. Then for each i , $\text{label-helper}(\text{def } m(y : \tau_2) : \tau_3 = e) = \text{def } m(y : \tau_2) : \tau_3$ with $\text{effects}(\Gamma \cap \text{freevars}(e_i)) = \text{label}(e_i)$. Let $\hat{e}_i = \text{label}(e_i)$. By inductive assumption, $\Gamma \vdash \hat{e}_i : \tau_3$ with $\hat{\varepsilon}_3$. This was for any i , so if σ_i is the labeled version of d_i then $\Gamma \vdash \sigma_i = \text{label}(e_i)$ OK. Finally we can apply ε -NEWOBJ to the labeled object $\overline{d}_i = \text{label}(e_i)$, which gives the judgement $\Gamma \vdash \hat{e} : \{\overline{d}\}$ with \emptyset . \square

Theorem 3.10. (Soundness Theorem)

Statement. If $\Gamma \vdash e_A : \tau_A$ with ε_A and $e_A \longrightarrow e_B \mid \varepsilon$ then $\Gamma \vdash e_B : \tau_B$ with ε_B , where $\tau_B = \tau_A$ and $\varepsilon \subseteq \varepsilon_A$.

Proof. Induct on the typing judgement for $\Gamma \vdash e_A : \tau_A$ with ε_A and then on the evaluation rule used for $e_A \longrightarrow e_B \mid \varepsilon$. Since we've shown soundness for the rules from the fully-labeled program we only consider the new rules.

Case. C-NEWOBJ.

Then $e_A = \text{new}_d x \Rightarrow \overline{d} = \overline{e}$ is a value. It cannot be reduced; the theorem statement holds immediately.

Case. C-INFERENCE.

Then we know:

- $\Gamma' \subseteq \Gamma$
- $\varepsilon_A = \text{effects}(\Gamma')$
- $\Gamma' \vdash e_A : \tau_A$

Considering the context Γ' we can apply C-INFERENCE again, picking $\Gamma' \subseteq \Gamma'$ as our sub-subcontext. Then $\Gamma' \vdash e_A : \tau$ with $\text{effects}(\Gamma')$. By the Use Principle, $\varepsilon \subseteq \text{effects}(\Gamma')$.

Case. C-METHCALL.

Let $\text{label}(e_A) = \hat{e}_A$. **label** only changes static type information and doesn't affect the runtime semantics of a program, so the same reduction in the theorem statement can be applied to \hat{e}_A . Therefore $\hat{e}_A \longrightarrow e_B \mid \varepsilon$. Since \hat{e}_A is a fully-labeled program we can appeal to the safety of the judgements on those programs. So $\Gamma \vdash e_B : \tau_B$ **with** ε_B and $\varepsilon \subseteq \hat{\varepsilon}_A$ by soundness. By the Refinement Theorem, $\hat{\varepsilon}_A \subseteq \varepsilon_A$, so $\varepsilon \subseteq \varepsilon_A$. □