1 Grammar

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

Notes:

- σ denotes a declaration with effect labels; d a declaration without effect labels.
- $\texttt{new}_\sigma$ is for creating annotated objects; $\texttt{new}_d$ for unannotated objects.
- $\{\bar{\sigma}\}$ is the type of an annotated object. $\{\bar{d}\}$ is the type of an unannotated object.
- $\{\bar{d}\ \texttt{captures}\ \varepsilon\}$ 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 $\{\bar{d}\}$.

2 Semantics

2.1 Static Semantics

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

$$\frac{}{\Gamma,\ x : \tau \vdash x : \tau}\ (\text{T-Var}) \qquad \frac{}{\Gamma,\ r : \{\bar{r}\} \vdash r : \{\bar{r}\}}\ (\text{T-Resource})$$

$$\frac{\Gamma \vdash e_1 : \{\bar{r}\}}{\Gamma \vdash e_1.\pi : \texttt{Unit}}\ (\text{T-OperCall})$$

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

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

$$\frac{\Gamma \vdash \sigma_i = e_i\ \texttt{OK}}{\Gamma \vdash\ \texttt{new}_\sigma\ x \Rightarrow \overline{\sigma = e} : \{\bar{\sigma}\}}\ (\text{T-New}_\sigma) \qquad \frac{\Gamma \vdash d_i = e_i\ \texttt{OK}}{\Gamma \vdash\ \texttt{new}_d\ x \Rightarrow \overline{d = e} : \{\bar{d}\}}\ (\text{T-New}_d)$$

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

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

$\boxed{\Gamma \vdash \sigma = e \; \texttt{OK}}$

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

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

$$\frac{}{\Gamma,\ x : \tau \vdash x : \tau \; \texttt{with} \; \varnothing} \; (\varepsilon\text{-}\textsc{Var}) \qquad \frac{}{\Gamma,\ r : \{\bar{r}\} \vdash r : \{\bar{r}\} \; \texttt{with} \; \varnothing} \; (\varepsilon\text{-}\textsc{Resource})$$

$$\frac{\Gamma,\ x : \{\bar{\sigma}\} \vdash \overline{\sigma = e} \; \texttt{OK}}{\Gamma \vdash \texttt{new}_\sigma \; x \Rightarrow \overline{\sigma = e} : \{\bar{\sigma}\} \; \texttt{with} \; \varnothing} \; (\varepsilon\text{-}\textsc{NewObj}) \qquad \frac{\Gamma \vdash e_1 : \{\bar{r}\} \; \texttt{with} \; \varepsilon_1}{\Gamma \vdash e_1.\pi : \texttt{Unit} \; \texttt{with} \; \{\bar{r}.\pi\} \cup \varepsilon_1} \; (\varepsilon\text{-}\textsc{OperCall})$$

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

$$\frac{\varepsilon_c = \texttt{effects}(\Gamma') \quad \Gamma' \subseteq \Gamma \quad \Gamma', x : \{\bar{d} \; \texttt{captures} \; \varepsilon_c\} \vdash \overline{d = e} \; \texttt{OK}}{\Gamma \vdash \; \texttt{new}_d \; x \Rightarrow \overline{d = e} : \{\bar{d} \; \texttt{captures} \; \varepsilon_c\} \; \texttt{with} \; \varnothing} \; (\text{C-}\textsc{NewObj})$$

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

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

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 ε-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 $\texttt{effects}(\tau_2)$ 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.

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

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

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

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

$$\frac{e_1 \longrightarrow e_1' \mid \varepsilon}{e_1.m(e_2) \longrightarrow e_1'.m(e_2) \mid \varepsilon} \text{ (E-MethCall1)}$$

$$\frac{v_1 = \texttt{new}_\sigma\ x \Rightarrow \overline{\sigma = e} \quad e_2 \longrightarrow e_2' \mid \varepsilon}{v_1.m(e_2) \longrightarrow v_1.m(e_2') \mid \varepsilon} \text{ (E-MethCall2}_\sigma\text{)} \qquad \frac{v_1 = \texttt{new}_d\ x \Rightarrow \overline{d = e} \quad e_2 \longrightarrow e_2' \mid \varepsilon}{v_1.m(e_2) \longrightarrow v_1.m(e_2') \mid \varepsilon} \text{ (E-MethCall2}_d\text{)}$$

$$\frac{v_1 = \texttt{new}_\sigma\ x \Rightarrow \overline{\sigma = e} \quad \texttt{def m}(y : \tau_1) : \tau_2 \texttt{ with } \varepsilon = e \in \overline{\sigma = e}}{v_1.m(v_2) \longrightarrow [v_1/x, v_2/y]e \mid \varnothing} \text{ (E-MethCall3}_\sigma\text{)}$$

$$\frac{v_1 = \texttt{new}_d\ x \Rightarrow \overline{d = e} \quad \texttt{def m}(y : \tau_1) : \tau_2 = e \in \overline{d = e}}{v_1.m(v_2) \longrightarrow [v_1/x, v_2/y]e \mid \varnothing} \text{ (E-MethCall3}_d\text{)}$$

$$\frac{e_1 \longrightarrow e_1' \mid \varepsilon}{e_1.\pi \longrightarrow e_1'.\pi \mid \varepsilon} \text{ (E-OperCall1)} \qquad \frac{}{r.\pi \longrightarrow \texttt{unit} \mid \{r.\pi\}} \text{ (E-OperCall2)}$$

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

$$\frac{}{e \longrightarrow_* e \mid \varnothing} \text{ (E-MultiStep1)} \qquad \frac{e \longrightarrow e' \mid \varepsilon}{e \longrightarrow_* e' \mid \varepsilon} \text{ (E-MultiStep2)}$$

$$\frac{e \longrightarrow_* e' \mid \varepsilon_1 \quad e' \longrightarrow_* e'' \mid \varepsilon_2}{e \longrightarrow_* e'' \mid \varepsilon_1 \cup \varepsilon_2} \text{ (E-MultiStep3)}$$

Notes:

- E-MethCall2$_d$ and E-MethCall2$_\sigma$ 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$_\sigma$ 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 $y \neq z$
- $[e'/z]r = r$
- $[e'/z](e_1.m(e_2)) = ([e'/z]e_1).m([e'/z]e_2)$
- $[e'/z](e_1.\pi) = ([e'/z]e_1).\pi$
- $[e'/z](\mathtt{new}_d\ x \Rightarrow \overline{d = e}) = \mathtt{new}_\sigma\ x \Rightarrow \overline{\sigma = [e'/z]e}$, if $z \neq x$ and $z \notin \mathtt{freevars}(e_i)$
- $[e'/z](\mathtt{new}_\sigma\ x \Rightarrow \overline{\sigma = e}) = \mathtt{new}_\sigma\ x \Rightarrow \sigma = [e'/z]e$, if $z \neq x$ and $z \notin \mathtt{freevars}(e_i)$

3 Proofs

Lemma 3.1. (Canonical Forms)

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

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

Furthermore, $\varepsilon = \varnothing$ 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.

□

Lemma 3.2. (Substitution Lemma)

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

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 $\Gamma \vdash e : \tau\ \mathtt{with}\ \varepsilon$ 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 Γ can use C-MethCall to type $[e'/z]e$ the same as e.

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

Case. C-NewObj.
Then $e = \mathtt{new}_d\ x \Rightarrow \overline{d = e}$. z appears in some method body e_i. By inversion we know $\Gamma, x : \{\bar{\sigma}\} \vdash \overline{d = e}\ \mathtt{OK}$. The only rule with this conclusion is ε-ValidImpl$_d$; by inversion on that we know for each i that:
- $d_i = \mathtt{def}\ m_i(y : \tau_1) : \tau_2\ \mathtt{with}\ \varepsilon$
- $\Gamma, y : \tau_1 \vdash e_i : \tau_2\ \mathtt{with}\ \varepsilon$

If z appears in the body of e_i then $\Gamma, z : \tau \vdash d_i = e_i\ \mathtt{OK}$ by inductive assumption. Then we can use ε-ValidImpl$_d$ to conclude $\overline{d = [e'/z]e}\ \mathtt{OK}$. 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)

$\boxed{\text{Statement.}}$ If $\Gamma_1 \subseteq \Gamma_2$ then $\texttt{effects}(\Gamma_1) \subseteq \texttt{effects}(\Gamma_2)$

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

□

Lemma 3.4. (Use Principle)

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

$\boxed{\text{Proof.}}$ 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, $\varepsilon \subseteq \texttt{effects}(\Gamma)$ follows by the definition of $\texttt{effects}$ for the case of a resource.

$\boxed{\text{Intuition.}}$ 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)

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

$\boxed{\text{Proof.}}$ The typing judgements operate on the form of e, so don't consider any variables external to e.

□

$\boxed{\text{Note.}}$ We'll use $\texttt{freevars}(e) \cap \Gamma$ to mean Γ, where the pair (x, τ) is thrown out if $x \notin \texttt{freevars}(e)$.

$\boxed{\text{Intuition.}}$ If you can typecheck e in Γ, you can throw out the parts in Γ 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 $\texttt{label}(\Gamma, e) = \hat{e}$.

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

Notes:

- Beware of confusing notation: there are two types of equality in the above definitions. One is the equality which defines $\texttt{label}$, and the other is the equality $\sigma = e$ of declarations in the programming language.
- 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.
- $\texttt{label}$ is defined on expressions; $\texttt{label-helper}$ on declarations. This is just for clarity; everywhere other than this section we'll only use $\texttt{label}$.

- Initially it seems like `label` on a $\texttt{new}_\sigma$ 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 $\texttt{labels}(e) = \hat{e}$, 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 its type and static effects.

Lemma 3.7. (Extension Lemma)

Statement. If $\Gamma \vdash e : \tau$ and e is a value then $\hat{e} = \texttt{label}(e) = e$ and $\Gamma \vdash e : \tau \texttt{ with } \varepsilon$.

Proof. Consider the different forms of e.

Case. $e = x$.

Case. $e = r$.

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

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

$\square$

Theorem 3.7. (Extension Theorem)

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

Intuition. If Γ can type e without an effect, there is a way to label e with a static effect set $\hat{\varepsilon}$ which contains the possible runtime effects of e (so it is an upper-bound), and no more than what is contained in the environment Γ. (**Also, $\texttt{effects}(\Gamma)$ is an upper bound but we omit this from the proof to keep it as simple as possible.**)

Proof. Proceed by induction on $\Gamma \vdash e : \tau$ and then on the reduction $e \longrightarrow e' \mid \varepsilon$.

Case. T-Var, T-Resource, T-New$_\sigma$, T-New$_d$.
Then e is a value and no reduction can be applied to it. Theorem statement holds immediately.

Case. T-OperCall.
Then the following are known:

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

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 $e_1 \to e_1' \mid \varepsilon$. Because $\Gamma \vdash e_1 : \{\bar{r}\}$ by assumption of the typing rule, we may apply the inductive assumption. Then $\Gamma \vdash \hat{e}_1 : \{\bar{r}\} \texttt{ with } \hat{\varepsilon}_1$, where $\varepsilon \subseteq \hat{\varepsilon}_1$ and $\hat{e}_1 = \texttt{label}(\Gamma, e_1)$.

By definition $\hat{e} = \texttt{label}(\Gamma, e) = \texttt{label}(\Gamma, e_1.\pi) = (\texttt{label}(\Gamma, e_1)).\pi = \hat{e}_1.\pi$. We just established $\Gamma \vdash \hat{e}_1 : \{\bar{r}\} \texttt{ with } \hat{\varepsilon}$, so fulfill the requirements of ε-OperCall and can type $\hat{e} = \hat{e}_1.\pi$ with the judgement $\Gamma \vdash \hat{e}_1.\pi : \texttt{Unit with } \{r.\pi\} \cup \hat{\varepsilon}_1$.

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

<u>Subcase.</u> E-OperCall2. Then we know $e = r.\pi$ and $r.\pi \longrightarrow \texttt{Unit} \mid \{r.\pi\}$. By definition $\hat{e} = \texttt{label}(\Gamma, e) = (\texttt{label}(\Gamma, r)).\pi = r.\pi = e$, so $\hat{e} = e$. Then $\hat{\tau} = \tau$ automatically. We need only show $\varepsilon = r.\pi \in \hat{\varepsilon}$.

By ε-Resource, $\Gamma \vdash r : \{r\}$ with $\varnothing$ and by ε-OperCall, $\Gamma \vdash r.\pi : \texttt{Unit}$ 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 $e_1.m_i(e_2)$.

<u>Subcase.</u> E-OperCall1. Then we know $e_1 \longrightarrow e_1' \mid \varepsilon$ and $e_1.m_i(e_2) \longrightarrow e_1'.m_i(e_2) \mid \varepsilon$. Because $\Gamma \vdash e_1 : \{\bar{\sigma}\}$ by assumption of the typing rule, we may apply the inductive assumption. Then $\Gamma \vdash \hat{e}_1 : \{\bar{\sigma}\}$ with $\hat{\varepsilon}_1$, where $\varepsilon \subseteq \hat{\varepsilon}_1$ and $\hat{e}_1 = \texttt{label}(\Gamma, e_1)$.

By definition, $\hat{e} = \texttt{label}(e_1.m_i(e_2)) = (\texttt{label}(e_1)).m_i(\texttt{label}(e_2)) = \hat{e}_1.m_i(\hat{e}_2)$. We just established $\Gamma \vdash \hat{e}_1 : \{\bar{\sigma}\}$ with $\hat{\varepsilon}_1$.
If e_2 is a value then $\hat{e}_2 = \texttt{label}(e_2) = e_2$.

Can we type $\hat{e}_2$ though?

Case. T-MethCall$_d$.
Hey

□

Theorem 3.9. (Refinement Theorem)

Statement. If $\Gamma \vdash e : \tau$ with ε and $\texttt{label}(e) = \hat{e}$, 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:
- $\Gamma \vdash e : \texttt{Unit}$ with $\{r.\pi\} \cup \varepsilon_1$
- $\Gamma \vdash e_1 : \{\bar{r}\}$ with ε_1

Applying definitions, $\hat{e} = \texttt{label}(e_1.\pi) = (\texttt{label}(e_1)).\pi = \hat{e}_1.\pi$. By inductive assumption, $\Gamma \vdash \hat{e}_1 : \{\bar{r}\}$ with $\hat{\varepsilon}_1$, where $\hat{\varepsilon}_1 \subseteq \varepsilon_1$. Then $\Gamma \vdash \hat{e} : \texttt{Unit}$ 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
- $\sigma_i =$ def $m_i(y : \tau_2) : \tau_3$ with ε_3

Applying definitions, $\hat{e} = \texttt{label}(e_1.m_i(e_2)) = (\texttt{label}(e_1)).m_i(\texttt{label}(e_2)) = \hat{e}_1.m_i(\hat{e}_2)$. 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.

$\boxed{\text{Case.}}$ C-MethCall.
Then $e = e_1.m_i(e_2)$ and we know:
- $\Gamma \vdash e : \tau_3$ with $\varepsilon_1 \cup \varepsilon_2 \cup \varepsilon_3$
- $\Gamma \vdash e_1 : \{\bar{d}$ captures $\varepsilon_c\}$ with ε_1
- $\Gamma \vdash e_2 : \tau_2$ with ε_2
- $d_i =$ 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.

$\boxed{\text{Case.}}$ 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 effects(Γ'). By applying the inductive hypothesis to e_1 we know that $\Gamma \vdash \hat{e}_1 : \{r\}$ with $\hat{\varepsilon}_1$, where $\hat{\varepsilon}_1 \subseteq$ effects(Γ'). 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.

$\boxed{\text{Case.}}$ ε-NewObj.
Then $e = \texttt{new}_\sigma\ x \Rightarrow \overline{\sigma = e}$ and we know:
- $\Gamma \vdash e : \{\bar{\sigma}\}$ with $\varnothing$
- $\Gamma, x : \{\bar{\sigma}\} \vdash \overline{\sigma = e}$ 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 $\sigma_i =$ def $m_i(y : \tau_2) : \tau_3$ with ε_3. Applying definitions, $\hat{e} = \texttt{label}(\texttt{new}_\sigma\ x \Rightarrow \overline{\sigma = e}) = \texttt{new}_\sigma\ x \Rightarrow \texttt{label-helper}(\overline{\sigma = e})$. Then for each i, $\texttt{label-helper}(\sigma_i = e_i) = \sigma_i = \texttt{label}(e_i)$. Let $\hat{e}_i = \texttt{label}(e_i)$. Applying the inductive assumption we get $\Gamma \vdash \hat{e}_i : \tau_3$ with $\hat{\varepsilon}_3$. Then $\Gamma \vdash \sigma_i = \texttt{label}(e_i)$ OK by ε-ValidImpl$_\sigma$. This was for any i, so $\Gamma \vdash \overline{\sigma_i = \texttt{label}(e_i)}$ OK. Finally we can apply ε-NewObj to the labeled object $\overline{\sigma_i = \texttt{label}(e_i)}$, which gives the judgement $\Gamma \vdash \hat{e} : \{\bar{\sigma}\}$ with $\varnothing$.

$\boxed{\text{Case.}}$ C-NewObj.
Then $e = \texttt{new}_d\ x \Rightarrow \overline{d = e}$ and we know:
- $\Gamma \vdash e_1.m_i(e_2) : \tau_3$ with $\varepsilon_1 \cup \varepsilon_2 \cup \varepsilon_3$
- $\Gamma' \subseteq \Gamma$
- $\varepsilon_c =$ effects(Γ') with $\varnothing$
- $\Gamma', x : \{\bar{d}$ captures $\varepsilon_c\} \vdash \overline{d = e}$ 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 $d_i =$ def $m(y : \tau_2) : \tau_3$ with ε_3. Applying definitions, $\hat{e} = \texttt{label}(\texttt{new}_\sigma\ x \Rightarrow \overline{\sigma = e}) = \texttt{new}_d\ x \Rightarrow \texttt{label-helper}(\overline{d = e})$. Then for each i, $\texttt{label-helper}(\texttt{def}\ m(y : \tau_2) : \tau_3 = e) = \texttt{def}\ m(y : \tau_2) : \tau_3$ with $\texttt{effects}(\Gamma \cap \texttt{freevars}(e_i)) = \texttt{label}(e_i)$. Let $\hat{e}_i = \texttt{label}(e_i)$. 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 $\Gamma \vdash \overline{\sigma_i = \texttt{label}(e_i)}$ OK. Finally we can apply ε-NewObj to the labeled object $\overline{d_i = \texttt{label}(e_i)}$, which gives the judgement $\Gamma \vdash \hat{e} : \{\bar{d}\}$ with $\varnothing$.

□

Theorem 3.10. (Soundness Theoerm)

$\boxed{\text{Statement.}}$ 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 $\varepsilon \subseteq \varepsilon_A$.

$\boxed{\text{Proof.}}$ 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.

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

$\boxed{\text{Case.}}$ C-Inference.
Then we know:
- $\Gamma' \subseteq \Gamma$
- $\varepsilon_A = \texttt{effects}(\Gamma')$
- $\Gamma' \vdash e_A : \tau_A$

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` $\texttt{effects}(\Gamma')$. By the Use Principle, $\varepsilon \subseteq \texttt{effects}(\Gamma')$.

$\boxed{\text{Case.}}$ C-MethCall.
Let $\texttt{label}(e_A) = \hat{e}_A$. `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 $\varepsilon \subseteq \varepsilon_A$.

$\square$