1 Grammar

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

3 Proofs

Lemma 3.1. (Canonical Forms)

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

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

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

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

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

If z appears in the body of e_i then $\Gamma, z : \tau \vdash d_i = e_i\ \texttt{OK}$ by inductive assumption. Then we can use ε-ValidImpl$_d$ to conclude $\overline{d = [e'/z]e}\ \texttt{OK}$. 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 $\Gamma \vdash e_A : \tau_A \ \texttt{with}\ \varepsilon_A$, 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 `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 $\Gamma \vdash e : \tau \ \texttt{with}\ \varepsilon$ then $\Gamma \cap \texttt{freevars}(e) \vdash e : \tau \ \texttt{with}\ \varepsilon$.

$\boxed{\text{Proof.}}$ 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 $\texttt{label}(\Gamma, e) = \hat{e}$.

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

Notes:

- $\Gamma \cap \texttt{freevars}(e)$ is the set of pairs $x : \tau \in \Gamma$, such that $x \in \texttt{freevars}(e)$.
- $\texttt{label}(e, \Gamma)$ is read as: "the labeling of e in Γ".
- Often the Γ we use is obvious in context; in such cases we write $\texttt{label}(e)$ instead of $\texttt{label}(e, \Gamma)$.
- Beware of confusing notation: there are two types of equality in the above definitions. One is the equality which defines `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 ε 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 $\texttt{new}_\sigma$ 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.
- 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 the type and static effects of the labeled version.

Observations 3.7.

Statement. The following are true.

- $\texttt{label}(e)$ 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 ε_i, the exact same method and label will occur in $\hat{e}$.

Proof. By inspection of the definition of label.

$\square$

Property 3.8. (Commutativity Between label and sub)

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

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 $\texttt{label}([e'/z]e)$ while "right-hand side" refers to $[\texttt{label}(e')/z](\texttt{label}(e))$.

Case. $e = r$.
By definition, $\texttt{label}(r) = r$ 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, $\texttt{label}(x) = x$. $[e'/z]x$ 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, $\texttt{label}([e'/z]x) = \texttt{label}(e')$. On the right-hand side, $[\texttt{label}(e')/z]x = \texttt{label}(e')$.

Case. $e = e_1.\pi$.
On the left-hand side.

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

On the right-hand side.

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

$\boxed{\text{Case.}}$ $e = e_1.m(e_2)$.
On the left-hand side.

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

On the right-hand side.

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

$\boxed{\text{Case.}}$ $e = \texttt{new}_\sigma\ x \Rightarrow \overline{\sigma = e}$.
On the left-hand side.

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

On the right-hand side.

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

$\boxed{\text{Case.}}$ $e = \texttt{new}_d\ x \Rightarrow \overline{d = e}$.
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 $\varepsilon_i = \texttt{effects}(\Gamma \cap \texttt{freevars}(e_i))$, and $\sigma_i = d_i\ \texttt{with}\ \varepsilon_i$ (these are from the definition of $\texttt{label-helper}$).

On the left-hand side.

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

On the right-hand side.

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

□

Lemma 3.9. (Runtime Invariance Under `label`)

$\boxed{\text{Statement.}}$ If the following are true:

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

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

$\boxed{\text{Proof.}}$ 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 Γ, so we'll write $\texttt{label}(e)$ instead of $\texttt{label}(e, \Gamma)$ as a notational short-hand.

$\boxed{\text{Case.}}$ $e = r$, $e = x$, $e = \texttt{new}_\sigma\ x \Rightarrow \overline{\sigma = e}$, $e = \texttt{new}_d\ x \Rightarrow \overline{d = e}$.
Then e is a value and the theorem statement holds automatically.

$\boxed{\text{Case.}}$ $e = e_1.\pi$.
The only typing rule which applies is ε-OperCall, which tells us:
- $\Gamma \vdash e_1 : \{r\}$ $\texttt{with}$ ε_1
- $\Gamma \vdash e_1.\pi : \texttt{Unit with}$ $\varepsilon_1 \cup \{r.\pi\}$

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 $\hat{e}_1' = \texttt{label}(e_1')$. Applying definitions, $\hat{e}_A = \texttt{label}(e_1.\pi) = (\texttt{label}(e_1)).\pi = \hat{e}_1.\pi$. 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, $\hat{e}_B = \texttt{label}(e_1'.\pi) = (\texttt{label}(e_1')).\pi$, which we know to be $\hat{e}_1'.\pi$ by inductive assumption.

Subcase. E-OperCall2. We also know $e_1 = r$ and $r.\pi \longrightarrow \texttt{Unit} \mid \{r.\pi\}$. Applying definitions, $\hat{e}_A = \texttt{label}(r.\pi) = (\texttt{label}(r)).\pi = r.\pi = e_A$. The theorem holds immediately.

$\boxed{\text{Case.}}$ $e = e_1.m_i(e_2)$.
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 $\texttt{label}(e_1') = \hat{e}_1'$. Applying definitions $\hat{e}_A = \texttt{label}(e_1.m_i(e_2)) = (\texttt{label}(e_1)).m_i(\texttt{label}(e_2)) = \hat{e}_1.m_i(\hat{e}_2)$. 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, $\hat{e}_B = \texttt{label}(e_1'.m_i(\hat{e}_2)) = (\texttt{label}(e_1')).m_i(\texttt{label}(e_2))$, 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 $e_1 = v_1 = \texttt{new}_\sigma\ x \Rightarrow \overline{\sigma = e}$, 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 $\texttt{label}(e_2') = \hat{e}_2'$. Applying definitions, $\hat{e}_A = \texttt{label}(v_1.m_i(e_2)) = (\texttt{label}(v_1)).m_i(\texttt{label}(e_2)) = \hat{v}_1.m_i(\hat{e}_2)$. Because $\hat{e}_2 \longrightarrow \hat{e}_2' \mid \varepsilon$, we may apply the reduction E-MethCall$_\sigma$ to obtain $\hat{v}_1.m_i(\hat{e}_2) \longrightarrow \hat{v}_1.m_i(\hat{e}_2')$. Lastly, $\hat{e}_B = \texttt{label}(v_1.m_i(e_2')) = (\texttt{label}(v_1)).m_i(\texttt{label}(e_2'))$, 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 $e_1 = v_1 = \texttt{new}_d\ x \Rightarrow \overline{d = e}$, and we apply the reduction rule E-MethCall$_d$ instead.

Subcase. E-MethCall3$_\sigma$. We also know the following:
- $e_1 = v_1 = \texttt{new}_\sigma\ x \Rightarrow \overline{\sigma = e}$
- $e_2 = v_2$
- $\texttt{def}\ m_i(y : \tau_2) : \tau_3\ \texttt{with}\ \varepsilon_3 = e_{body} \in \{\bar{\sigma}\}$
- $v_1.m_i(v_2) \longrightarrow [v_1/x, v_2/y]e_{body} \mid \varnothing$.

Applying definitions, $\texttt{label}(v_1.m_i(v_2)) = (\texttt{label}(v_1)).m_i(\texttt{label}(v_2)) = \hat{v}_1.m_i(\hat{v}_2)$, where we define $\hat{v}_1 = \texttt{label}(v_1)$ and $\hat{v}_2 = \texttt{label}(v_2)$. 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 $\texttt{label}(e_{body}) = \hat{e}_{body}$. 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 $\hat{v}_1.m_i(\hat{v}_2) \longrightarrow [\hat{v}_1/x, \hat{v}_2/y]\hat{e}_{body} \mid \varnothing$. Because $\texttt{label}$ and $\texttt{sub}$ commute, $\texttt{label}(e_B) = \texttt{label}([v_1/x, v_2/y]e_{body}) = [\texttt{label}(v_1)/x, \texttt{label}(v_2)/y](\texttt{label}(e_{body})))$, 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 $\hat{e}_{body}$.

<u>Subcase.</u> E-MethCall3$_d$. This case is identical to the previous one, except $e_1 = v_1 = \texttt{new}_d\ x \Rightarrow \overline{d = e}$. The same reasoning applies though.

□

Theorem 3.10. (Extension Lemma)

Statement. If $\Gamma \vdash e : \tau$, then one of the following is true:

- $\Gamma \vdash e : \{\bar{d}\}\ \texttt{with}\ \varepsilon$
- $\tau = \{\bar{d}\}$ and $\Gamma \vdash e : \{\bar{d}\ \texttt{captures}\ \varepsilon_C\}\ \texttt{with}\ \varnothing$, for some ε_C.

Proof. By induction the typing judgement.

Case. T-Var, T-Resource, T-New$_\sigma$.
Use ε-Var, ε-Resource, ε-New$_\sigma$ respectively.

Case. T-OperCall.
We have $e = e_1.\pi$ and $\Gamma \vdash e_1 : \{\bar{r}\}$. By inductive assumption, $\Gamma \vdash e_1 : \{\bar{r}\}\ \texttt{with}\ \varepsilon$. Applying ε-OperCall, we get $\Gamma \vdash e_1.\pi : \{\bar{r}\}\ \texttt{with}\ \varepsilon$.

Case. T-New$_d$.
Select $\Gamma' = \Gamma$. Then $\Gamma', x : \{\bar{d}\ \texttt{captures}\ \varepsilon_C\} \vdash \overline{d = e}\ \texttt{OK}$, because we know $\Gamma, x : \{\bar{d}\} \vdash \overline{d = e}\ \texttt{OK}$. By an application of C-NewObj, we get $\Gamma \vdash e : \{\bar{d}\ \texttt{captures}\ \varepsilon_C\}\ \texttt{with}\ \varnothing$, for $\varepsilon_C = \texttt{effects}(\Gamma)$.

Case. T-MethCall$_\sigma$.
Then $e = e_1.m_i(e_2)$. From inversion on the rule we know $\Gamma \vdash e_1 : \{\bar{\sigma}\}$ and $\Gamma \vdash e_2 : \tau_2$. By inductive assumption we know $\Gamma \vdash e_1 : \{\bar{\sigma}\}\ \texttt{with}\ \varepsilon_1$. If $\tau_2 = \{\bar{d}\}$, then applying the inductive hypothesis gives us $\Gamma \vdash e_2 : \tau_2\ \texttt{with}\ \{\bar{d}\ \texttt{captures}\ \varepsilon_C\}$. **However, the types are different and because there's no sub-typing we run into a problem here. If we knew $\{\bar{d}\ \texttt{captures}\ \varepsilon_C\} <: \{\bar{d}\}$ then we'd be fine.**

□

Theorem 3.11. (Extension Theorem)

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

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

Intuition. If Γ can type e without an effect, there is a way to label e with $\hat{\varepsilon}$ which contains the possible runtime effects of e (so $\hat{\varepsilon}$ is an upper-bound). (**Also, $\texttt{effects}(\Gamma)$ is an upper bound on $\hat{\varepsilon}$ but we omit this from the proof (for now) to keep it simple.**)

Proof. Throughout this proof there is only one Γ, so we say $\texttt{label}(e)$ as short-hand for $\texttt{label}(e, \Gamma)$. 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 $\texttt{label}(x) = x$. By assumption that the program is closed under Γ, we can apply ε-Var to conclude $\Gamma \vdash x : \tau\ \texttt{with}\ \varnothing$.

Case. T-Resource.
$e = r$ is a value, and $\texttt{label}(r) = r$. By assumption that the program is closed under Γ, we can apply ε-Resource to conclude $\Gamma \vdash r : \{r\}\ \texttt{with}\ \varnothing$.

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

Fix some i. By assumption, $\Gamma \vdash \sigma_i = e_i\ \texttt{OK}$. By inversion on ε-ValidImpl$_\sigma$, we know $\Gamma, y : \tau_2 \vdash e_i : \tau_3\ \texttt{with}\ \varepsilon_3$. Consider $\hat{e}_i = \texttt{label}(e_i)$. By inductive assumption, $\Gamma, y : \tau_2 \vdash \hat{e}_i : \tau_3\ \texttt{with}\ \hat{\varepsilon}$, and by application of ε-ValidImpl$_\sigma$ we know $\Gamma \vdash \sigma_i = \texttt{label}(e_i)$ OK. (**We're applying inductive assumption to something of the form $\Gamma \vdash e : \tau$ with ε, not $\Gamma \vdash e : \tau$ though.)**

i was arbitrary; therefore $\Gamma \vdash \overline{\sigma = \texttt{label}(e)}$ OK . Therefore $\Gamma \vdash \hat{e} : \{\bar{\sigma}\}$ with $\varnothing$ by ε-NewObj$_\sigma$.

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

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

To type $\hat{e}$ we want to use ε-NewObj; to do that we need to know $\Gamma, x : \{\sigma\} \vdash \overline{\sigma_i = e_i}$ OK, so fix some i. By assumption $\Gamma \vdash d_i = e_i$ OK. By inversion on ε-ValidImpl$_d$ 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 $\Gamma, \{\bar{d}\} \vdash \overline{d_i = e_i}$ OK. By inversion on ε-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 ε-ValidImpl$_\sigma$ we know $\Gamma \vdash \sigma_i = \texttt{label}(e_i)$ OK.

i was arbitrary; therefore $\Gamma \vdash \overline{\sigma = \texttt{label}(e)}$ OK. Therefore $\Gamma \vdash \hat{e} : \{\bar{\sigma}\}$ with $\varnothing$ by ε-NewObj.

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 $\Gamma \vdash \hat{e}_1 : \{\bar{r}\}$ with $\hat{\varepsilon}_1$, where $\varepsilon \subseteq \hat{\varepsilon}_1$ and $\hat{e}_1 = \texttt{label}(e_1)$.

By definition $\hat{e} = \texttt{label}(\Gamma, e) = \texttt{label}(\Gamma, e_1.\pi) = (\texttt{label}(\Gamma, e_1)).\pi = \hat{e}_1.\pi$. Because $\Gamma \vdash \hat{e}_1 : \{\bar{r}\}$ with $\hat{\varepsilon}$ we can aply ε-OperCall and 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, $\hat{\tau} = \texttt{Unit} = \tau$.

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

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, $\hat{e} = \texttt{label}(e_1.m_i(e_2)) = (\texttt{label}(e_1)).m_i(\texttt{label}(e_2)) = \hat{e}_1.m_i(\hat{e}_2)$. Method m_i has the same label ε_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 $\Gamma \vdash \hat{e}_1 : \{\bar{\sigma}\}$ $\texttt{with}$ $\hat{\varepsilon}_1$ and $\Gamma \vdash \hat{e}_2 : \tau_2$ $\texttt{with}$ $\hat{\varepsilon}_2$, where $\varepsilon \subseteq \hat{\varepsilon}_1$.

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

Subcase. E-MethCall2$_\sigma$. Then we know $e_1 = v_1 = \texttt{new}_\sigma\ x \Rightarrow \overline{\sigma = e}$ 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, $\hat{e} = \texttt{label}(e_1.m_i(e_2)) = (\texttt{label}(e_1)).m_i(\texttt{label}(e_2)) = \hat{e}_1.m_i(\hat{e}_2)$. Method m_i has the same label ε_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 $\Gamma \vdash \hat{e}_1 : \{\bar{\sigma}\}$ $\texttt{with}$ $\hat{\varepsilon}_1$ and $\Gamma \vdash \hat{e}_2 : \tau_2$ $\texttt{with}$ $\hat{\varepsilon}_2$, where $\varepsilon \subseteq \hat{\varepsilon}_2$.

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

Subcase. E-MethCall3$_\sigma$. Then we know $e_1 = v_1 = \texttt{new}_\sigma\ x \Rightarrow \overline{\sigma = e}$ and $e_2 = v_2$ is a value. Also, $v_1.m_i(v_2) \longrightarrow [v_1/x, v_2/y]e_{body} \mid \varnothing$. By definition, $\hat{e} = \texttt{label}(v_1.m_i(v_2)) = (\texttt{label}(v_1)).m_i(\texttt{label}(v_2)) = \hat{v}_1.m_i(\hat{v}_2)$.

Applying the inductive assumption to v_1 and v_2, we get $\Gamma \vdash v_1 : \{\bar{\sigma}\}$ $\texttt{with}$ $\hat{\varepsilon}_1$ and $\Gamma \vdash v_2 : \{\bar{\sigma}\}$ $\texttt{with}$ $\hat{\varepsilon}_2$. Because v_1 was already a labeled object, the method m_i has the same label ε_3 in $\hat{v}_1$ as it does in v_1. By an application of ε-MethCall we learn $\Gamma \vdash v_1.m_i(v_2) : \tau_3$ $\texttt{with}$ $\hat{\varepsilon}_1 \cup \hat{\varepsilon}_2 \cup \varepsilon_3$. Since $\varepsilon = \varnothing$ this type contains all the runtime effects.

Case. T-MethCall$_d$.
Then the following are known:

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

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, $\hat{e} = \texttt{label}(e_1.m_i(e_2)) = (\texttt{label}(e_1)).m_i(\texttt{label}(e_2)) = \hat{e}_1.m_i(\hat{e}_2)$. In the object e_1 method m_i has no labeled effects. By definition of $\texttt{label}$, in $\hat{e}_1$ method m_i will have the label $\texttt{effects}(\Gamma) \cap e_i$, where e_i is the body of method m_i. **Not possible to tell if the expression is well-formed at this point i.e. if m_i actually has a method body, so we probably need to add the assumption that e is closed under Γ.**

Because $\Gamma \vdash e_1 : \{\bar{d}\}$ and $\Gamma \vdash e_2 : \tau_2$, by applying the inductive assumption to each we learn $\Gamma \vdash \hat{e}_1 : \{\bar{\sigma}\}$ $\texttt{with}$ $\hat{\varepsilon}_1$ and $\Gamma \vdash \hat{e}_2 : \tau_2$ $\texttt{with}$ $\hat{\varepsilon}_2$ (**first inductive assumption needs justification, and probably an invocation of the extension lemma**), where $\varepsilon \subseteq \hat{\varepsilon}_1$.

Now we can apply ε-MethCall, giving us $\Gamma \vdash \hat{e}_1.m_i(\hat{e}_2) : \tau_3$ $\texttt{with}$ $\hat{\varepsilon}_1 \cup \hat{\varepsilon}_2 \cup \varepsilon_3$, where ε_3 is the label on m_i. This is $\texttt{effects}(\Gamma) \cap e_i$, so this type has the effect-set $\hat{\varepsilon}_1 \cup \hat{\varepsilon}_2 \cup (\texttt{effects}(\Gamma) \cap e_i)$. $\varepsilon \subseteq \hat{\varepsilon}_1$ is an inductive assumption, so this type contains the runtime effects.

$\square$

Theorem 3.12. (Refinement Theorem)

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

- e has the form $\texttt{new}_d\ x \Rightarrow \overline{d = e}$; and $\Gamma \vdash \hat{e} : \overline{d_i \texttt{ with } \varepsilon_i = e_i}$, where
- $\Gamma \vdash \hat{e} : \hat{\tau} \texttt{ with } \hat{\varepsilon}$, where $\hat{\varepsilon} \subseteq \varepsilon$ and $\tau = \hat{\tau}$

Intuition. Labels can only make the static effects more precise; never less precise. **Needs to be edited/proofread so it makes sense with the new Extension theorem.**

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

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

Applying definitions, $\hat{e} = \texttt{label}(e_1.\pi) = (\texttt{label}(e_1)).\pi = \hat{e}_1.\pi$. By inductive assumption, $\Gamma \vdash \hat{e}_1 : \{\bar{r}\} \texttt{ with } \hat{\varepsilon}_1$, 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:
- $\Gamma \vdash e : \tau_3 \texttt{ with } \varepsilon_1 \cup \varepsilon_2 \cup \varepsilon_3$
- $\Gamma \vdash e_1 : \{\bar{\sigma}\} \texttt{ with } \varepsilon_1$
- $\Gamma \vdash e_2 : \tau_2 \texttt{ with } \varepsilon_2$
- $\sigma_i = \texttt{def } m_i(y : \tau_2) : \tau_3 \texttt{ with } \varepsilon_3$

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, $\Gamma \vdash \hat{e}_1 : \{\bar{\sigma}\} \texttt{ with } \hat{\varepsilon}_1$ and $\Gamma \vdash \hat{e}_2 : \tau_2 \texttt{ with } \hat{\varepsilon}_2$, where $\hat{\varepsilon}_1 \subseteq \varepsilon_1$ and $\hat{\varepsilon}_2 \subseteq \varepsilon_2$. Then $\Gamma \vdash \hat{e} : \tau_3 \texttt{ with } \hat{\varepsilon}_1 \cup \hat{\varepsilon}_2 \cup \varepsilon_3$ 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. ε-NewObj.
Then $e = \texttt{new}_\sigma\ x \Rightarrow \overline{\sigma = e}$ and we know:
- $\Gamma \vdash e : \{\bar{\sigma}\} \texttt{ with } \varnothing$
- $\Gamma, x : \{\bar{\sigma}\} \vdash \overline{\sigma = e} \texttt{ OK}$

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

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

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

Case. C-NewObj.
Then $e = \texttt{new}_d\ x \Rightarrow \overline{d = e}$ and we know:
- $\Gamma' \subseteq \Gamma$

- $\varepsilon_c = \texttt{effects}(\Gamma')$ `with` $\varnothing$
- $\Gamma', x : \{\bar{d}\ \texttt{captures}\ \varepsilon_c\} \vdash \overline{d = e}\ \texttt{OK}$

(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 $d_i = \texttt{def}\ m(y : \tau_2) : \tau_3\ \texttt{with}\ \varepsilon_3$. 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, $\texttt{label-helper}(\texttt{def}\ m(y : \tau_2) : \tau_3 = e) = \texttt{def}\ m(y : \tau_2) : \tau_3\ \texttt{with}\ \texttt{effects}(\Gamma \cap \texttt{freevars}(e_i)) = \texttt{label}(e_i)$. Let $\hat{e}_i = \texttt{label}(e_i)$. By inductive assumption, $\Gamma \vdash \hat{e}_i : \tau_3\ \texttt{with}\ \hat{\varepsilon}_3$. This was for any i, so if σ_i is the labeled version of d_i then $\Gamma \vdash \overline{\sigma_i = \texttt{label}(e_i)}\ \texttt{OK}$. Finally we can apply ε-NewObj to the labeled object $\overline{d_i = \texttt{label}(e_i)}$, which gives the judgement $\Gamma \vdash \hat{e} : \{\bar{d}\}\ \texttt{with}\ \varnothing$.

□

Theorem 3.13. (Soundness Theorem)

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

Proof.

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

1. $\Gamma \vdash \hat{e}_A : \hat{\tau}_A\ \texttt{with}\ \hat{\varepsilon}_A$
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. $\texttt{label}(e_B) = \hat{e}_B$

Applying the Refinement theorem to e_B we get:

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

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

8. $\Gamma \vdash \hat{e}_B : \hat{\tau}_A\ \texttt{with}\ \hat{\varepsilon}_B$
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. $\Gamma \vdash \hat{e}_B : \tau_A\ \texttt{with}\ \hat{\varepsilon}_B$

From **1** we know $\hat{e}_A$ has type $\hat{\tau}_A$; from **7** we know $\hat{e}_B$ has type τ_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. $\tau_A = \hat{\tau}_A = \tau_B$

□