# 1  Grammar

$$
\begin{aligned}
e ::=\ &x && \textit{expressions}\\
|\ &r\\
|\ &\texttt{new}_\sigma\ x \Rightarrow \overline{\sigma = e}\\
|\ &\texttt{new}_d\ x \Rightarrow \overline{d = e}\\
|\ &e.m(e)\\
|\ &e.\pi
\end{aligned}
$$

$$
\begin{aligned}
\tau ::=\ &\{\bar{\sigma}\} && \textit{types}\\
|\ &\{\bar{r}\}\\
|\ &\{\bar{d}\}\\
|\ &\{\bar{d}\ \texttt{captures}\ \varepsilon\}
\end{aligned}
$$

$$
\sigma ::= d\ \texttt{with}\ \varepsilon \qquad \textit{labeled decls.}
$$

$$
d ::= \texttt{def}\ m(x : \tau) : \tau\ \textit{unlabeled decls.}
$$

**Notes:**

- $\sigma$ 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, $\varepsilon$ is an upper-bound on the effects captured by $\{\bar{d}\}$.

# 2  Semantics

## 2.1  Static Semantics

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

**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 $\varepsilon$-ValidImpl$_d$ can be applied. **We are assuming the T-rules on their own are sound**.
- In C-NewObj, $\Gamma'$ is intended to be some subcontext of the current $\Gamma$. The object is labelled as capturing the effects in $\Gamma'$ (exact definition in the next section).
- In C-NewObj we must add $\text{effects}(\tau_2)$ to the static effects of the object, because the method body will have access to the resources captured by $\tau_2$ (the type of the argument passed into the method).
- A good choice of $\Gamma'$ would be $\Gamma$ restricted to the free variables in the object definition.
- The purpose of C-Inference is to ascribe static effects to unannotated portions of code (for instance, the body of an unlabeled method).
- As a useful convention we'll often use $\varepsilon_c$ to denote the output of the $\text{effects}$ function.

## 2.2 `effects` Function

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

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

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

**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 $\varepsilon_c$) by a potentially different context.

## 2.3 Dynamic Semantics

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

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

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

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

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

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

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

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

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

**Notes:**

- E-MethCall2$_d$ and E-MethCall2$_\sigma$ are really doing the same thing, but one applies to labeled objects (the $\sigma$ 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 $y \neq z$
- $[e'/z]r = r$
- $[e'/z](e_1.m(e_2)) = ([e'/z]e_1).m([e'/z]e_2)$
- $[e'/z](e_1.\pi) = ([e'/z]e_1).\pi$
- $[e'/z](\texttt{new}_d\ x \Rightarrow \overline{d = e}) = \texttt{new}_\sigma\ x \Rightarrow \overline{\sigma = [e'/z]e}$, if $z \neq x$ and $z \notin \texttt{freevars}(e_i)$
- $[e'/z](\texttt{new}_\sigma\ x \Rightarrow \overline{\sigma = e}) = \texttt{new}_\sigma\ x \Rightarrow \overline{\sigma = [e'/z]e}$, 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 $\Gamma \vdash e : \{\bar{r}\}$ with $\varepsilon$, then $e = r$ for some resource $r$.
- If $\Gamma \vdash e : \{\bar{\sigma}\}$ with $\varepsilon$, then $e = \texttt{new}_\sigma\ x \Rightarrow \overline{\sigma = e}$.
- If $\Gamma \vdash e : \{\bar{d}\ \texttt{captures}\ \varepsilon_c\}$ 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 $\varepsilon$-RESOURCE, $\varepsilon$-NEWOBJ, and C-NEWOBJ respectively.

□

### Lemma 3.2. (Substitution Lemma)

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

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

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

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

Case. C-INFERENCE.
Then $\Gamma \vdash e : \tau$ with $\texttt{effects}(\Gamma')$, 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 $\Gamma' \vdash [e'/z]e : \tau$ with $\texttt{effects}\ (\Gamma')$ under C-INFERENCE. Because $\Gamma' \subseteq \Gamma$ then $\Gamma \vdash [e'/z]e : \tau$ with $\texttt{effects}\ (\Gamma')$.

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 $\Gamma, x : \{\bar{\sigma}\} \vdash \overline{d = e}$ OK. The only rule with this conclusion is $\varepsilon$-VALIDIMPL$_d$; by inversion on that we know for each $i$ that:
- $d_i = \texttt{def}\ m_i(y : \tau_1) : \tau_2$ with $\varepsilon$
- $\Gamma, y : \tau_1 \vdash e_i : \tau_2$ with $\varepsilon$

If $z$ appears in the body of $e_i$ then $\Gamma, z : \tau \vdash d_i = e_i$ OK by inductive assumption. Then we can use $\varepsilon$-VALIDIMPL$_d$ 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 $\Gamma'$ has been chosen, so $[e'/z]e$ will be ascribed the same type as $e$.

$\square$

## Lemma 3.3. (Monotonicity of `effects`)

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

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

$\square$

## Lemma 3.4. (Use Principle)

| 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, $\varepsilon \subseteq \texttt{effects}(\Gamma)$.

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

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

$\square$

## Lemma 3.5. (Tightening Lemma)

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

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

$\square$

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

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

## Definition 3.6. (`label`)

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

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

**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 $\Gamma$".
- 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 $\varepsilon$ rules.
- `label` is defined on expressions; `label-helper` on declarations. Everywhere other than this section we'll only use `label`.

– Initially it seems like `label` on a `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 `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.

## Lemma 3.7. (Linearity of `label` over `substitution`)

Statement. `label`$([e'/z]e) = [\texttt{label}(e')/z](\texttt{label}(e))$

Proof. Induction on expression $e$.

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

By definition, $\texttt{label}([e'/z](\texttt{new}_\sigma \ x \Rightarrow \overline{\sigma = e}), \Gamma) = \texttt{label}(\texttt{new}_\sigma \ x \Rightarrow \overline{\sigma = [e'/z]e})$, which is $\texttt{new}_\sigma \ x \Rightarrow$ `label-helper`$(\overline{\sigma = [e'/z]e})$. Take any particular $\sigma_i = [e'/z]e_i$. Then `label-helper`$(\sigma_i = [e'/z]e_i)$ is defined as $\sigma_i = \texttt{label}([e'/z]e_i)$. By inductive assumption, this is the same as $\sigma_i = [\texttt{label}(e')/z](\texttt{label}(e_i))$. Therefore, $\texttt{new}_\sigma \ x \Rightarrow \texttt{label-helper}(\overline{\sigma = [e'/z]e})$ is the same as $\texttt{new}_\sigma \ x \Rightarrow \overline{\sigma = [\texttt{label}(e')/z]e}$. By "factoring out" the substitution, this is the same as $[\texttt{label}(e')/z](\texttt{new}_\sigma \ x \Rightarrow \overline{\sigma = e})$

□

## Lemma 3.8. (Runtime Invariance Under `label`)

Statement. If the following are true:

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

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

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

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.

Case. $e = e_1.\pi$.

The only typing rule which applies is $\varepsilon$-OPERCALL, which tells us:
– $\Gamma \vdash e_1 : \{r\} \text{ with } \varepsilon_1$
– $\Gamma \vdash e_1.\pi : \texttt{Unit with } \varepsilon_1 \cup \{r.\pi\}$
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.

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.

<u>Subcase.</u> 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.

<u>Subcase.</u> 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.

<u>Subcase.</u> 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, $\texttt{label}(v_1.m_i(v_2)) = (\texttt{label}(v_1)).m_i(\texttt{label}(v_2)) = \hat{v}_1.m_i(\hat{v}_2)$. 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 $\texttt{label}(\Gamma, e_{body}) = \hat{e}_{body}$. Because $\hat{v}_1$ and $\hat{v}_2$ are values, we can apply the reduction E-METHCALL3$_\sigma$ to $\hat{v}_1.m_i(\hat{v}_2)$; therefore $\hat{v}_1.m_i(\hat{v}_2) \longrightarrow [\hat{v}_1/x, \hat{v}_2/y]\hat{e}_{body} \mid \varnothing$. Lastly, because of the linearity of $\texttt{label}$ over $\texttt{substitution}$, $\texttt{label}(e_B) = \texttt{label}([v_1/x, v_2/y]e_{body}) = [\texttt{label}(v_1)/x, \texttt{label}(v_2)/y](\texttt{label}(e_{body})))$, which is $[\hat{v}_1/x, \hat{v}_2/y]\hat{e}_{body} = \hat{e}_B$ by assumptions.

<u>Subcase.</u> E-METHCALL3$_d$ **TODO**

$\square$

## Theorem 3.9. (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 $\Gamma$ 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 $\Gamma$. (**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 $e_1.\pi$.

<u>Subcase.</u> E-OPERCALL1. Then we know $e_1.\pi \longrightarrow e'_1.\pi \mid \varepsilon$, and $e_1 \rightarrow e'_1 \mid \varepsilon$. Because $\Gamma \vdash e_1 : \{\bar{r}\}$ by assumption of the typing rule, we may apply the inductive assumption. Then $\Gamma \vdash \hat{e}_1 : \{\bar{r}\}\ \texttt{with}\ \hat{\varepsilon}_1$, 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 $\varepsilon$-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 $\varepsilon$-RESOURCE, $\Gamma \vdash r : \{r\} \texttt{ with } \varnothing$ and by $\varepsilon$-OPERCALL, $\Gamma \vdash r.\pi : \texttt{Unit with } \{r.\pi\}$. Since $\hat{e} = r.\pi$, then $\hat{\varepsilon} = r.\pi = \varepsilon$.

$\boxed{\text{Case.}}$ 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 $e_1.m_i(e_2)$.

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

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

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

$\boxed{\text{Case.}}$ T-METHCALL$_d$.
**TODO**

$\square$

## Theorem 3.10. (Refinement Theorem)

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

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

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

$\boxed{\text{Case.}}$ $\varepsilon$-RESOURCE, $\varepsilon$-VAR.
If $e$ is a resource or a variable then $e = \hat{e}$ so the statement is automatically fulfilled.

$\boxed{\text{Case.}}$ $\varepsilon$-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 $\varepsilon$-OPERCALL. Importantly, $\{r.\pi\} \cup \hat{\varepsilon}_1 \subseteq \{r.\pi\} \cup \varepsilon_1$ as claimed.

$\boxed{\text{Case.}}$ $\varepsilon$-METHCALL.
Then $e = e_1.m_i(e_2)$ and we know:
- $\Gamma \vdash e : \tau_3 \texttt{ with } \varepsilon_1 \cup \varepsilon_2 \cup \varepsilon_3$

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

Applying definitions, $\hat{e} = $ `label`$(e_1.m_i(e_2)) = ($`label`$(e_1)).m_i($`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 $\varepsilon$-METHCALL. Importantly, $\hat{\varepsilon}_1 \cup \hat{\varepsilon}_2 \cup \varepsilon_3 \subseteq \varepsilon_1 \cup \varepsilon_2 \cup \varepsilon_3$ as claimed.

---

Case. 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` $\varepsilon_1$
- $\Gamma \vdash e_2 : \tau_2$ `with` $\varepsilon_2$
- $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 $\varepsilon$-METHCALL.

---

Case. C-INFERENCE.

We know:
- $\Gamma' \subseteq \Gamma$
- $\Gamma' \vdash e : \tau$
- $\Gamma \vdash e : \tau$ `with` `effects`$(\Gamma')$

There aren't any judgements of the form $e : \tau$ `with` $\varepsilon$ 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 $\Gamma$ can type $e_1$, we may conclude $\Gamma \vdash e_1 : \{r\}$. By an application of C-INFERENCE choosing the same $\Gamma'$, we know $\Gamma \vdash e_1 : \{r\}$ `with` `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 $ `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` $\varepsilon$ in the antecedent of this rule, so we can't use the induction hypothesis. We also don't know anything about $e$.**

---

Case. $\varepsilon$-NEWOBJ.

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

---

Case. C-NEWOBJ.

Then $e = $ `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`$(\Gamma')$ `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 $\varepsilon$-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` $\varepsilon_3$. Applying definitions, $\hat{e} = $ `label`$($`new`$_\sigma$ $x \Rightarrow \overline{\sigma = e}) = $ `new`$_d$ $x \Rightarrow$ `label-helper`$(\overline{d = e})$. Then for each $i$, `label-helper`(`def` $m(y : \tau_2) : \tau_3 = e) = $ `def` $m(y : \tau_2) : \tau_3$ `with` `effects`$(\Gamma \cap$ `freevars`$(e_i)) = $ `label`$(e_i)$. Let $\hat{e}_i = $ `label`$(e_i)$. By inductive assumption, $\Gamma \vdash \hat{e}_i : \tau_3$ `with` $\hat{\varepsilon}_3$. This was for any $i$, so if $\sigma_i$ is the labeled version of $d_i$ then $\Gamma \vdash \overline{\sigma_i = \text{label}(e_i)}$ `OK`. Finally we can apply $\varepsilon$-NEWOBJ to the labeled object $\overline{d_i = \text{label}(e_i)}$, which gives the judgement $\Gamma \vdash \hat{e} : \{\bar{d}\}$ `with` $\varnothing$.

□

## Theorem 3.11. (Soundness Theorem)

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

Proof. Let $\hat{e}_A = \mathtt{label}(e_A)$. By Refinement Theorem, $\Gamma \vdash \hat{e}_A : \hat{\tau}_A$ `with` $\hat{\varepsilon}_A$, where $\tau_A = \hat{\tau}_A$ and $\hat{\varepsilon}_A \subseteq \varepsilon_A$. By Invariance Of Runtime Under `label`, we know $\hat{e}_A \longrightarrow \hat{e}_B \mid \varepsilon$. Since $\hat{e}_A$ is a fully-labeled program, by Soundness of $\varepsilon$ rules, we know $\Gamma \vdash \hat{e}_B : \hat{\tau}_A$ `with` $\hat{\varepsilon}_B$ where $\varepsilon \subseteq \hat{\varepsilon}_A$. By Runtime Invariance Under `label`, $\mathtt{label}(e_B) = \hat{e}_B$. By Refinement Theorem, $\Gamma \vdash \hat{e}_B : \tau_B$ `with` $\hat{\varepsilon}_B$.

Putting this all together we know $\tau_A = \hat{\tau}_A = \tau_B$ and $\varepsilon \subseteq \hat{\varepsilon}_A \subseteq \varepsilon_A$. □