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

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

$$\frac{}{\Gamma,\ x:\tau \vdash x:\tau}\ (\text{T-Var}) \qquad \frac{}{\Gamma,\ r:\{r\} \vdash r:\{r\}}\ (\text{T-Resource}) \qquad \frac{\Gamma \vdash e_1 : \{r\}}{\Gamma \vdash e_1.\pi : \texttt{Unit}}\ (\text{T-OperCall})$$

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

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

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

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

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

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

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

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

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

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

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

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

$$\frac{\begin{array}{c}\varepsilon_c = \texttt{capture}(\Gamma') \qquad \Gamma' \subseteq \Gamma \qquad \Gamma' \vdash e : \{\bar{d}\} \\ \tau \in \texttt{higher-order-args}(\Gamma') \implies \texttt{capture}(\tau) \supseteq \varepsilon_c\end{array}}{\Gamma \vdash \ e : \{\bar{d} \ \texttt{captures}\ \varepsilon_c\} \ \texttt{with}\ \varnothing} \ (\text{C-}\textsc{NewObj})$$

$$\frac{\begin{array}{c}\Gamma \vdash e_1 : \{\bar{d} \ \texttt{captures}\ \varepsilon_c\} \ \texttt{with}\ \varepsilon_1 \qquad \Gamma \vdash e_2 : \tau_2 \ \texttt{with}\ \varepsilon_2 \\ d_i = \ \texttt{def}\ m_i(y : \tau_2) : \tau_3\end{array}}{\Gamma \vdash e_1.m_i(e_2) : \tau_3 \ \texttt{with}\ \varepsilon_1 \cup \varepsilon_2 \cup \varepsilon_c \cup \texttt{capture}(\tau_2)} \ (\text{C-}\textsc{MethCall})$$

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

$$\frac{}{\Gamma \vdash \tau <: \tau}\ (\textsc{St-Reflexive}) \qquad \frac{\Gamma \vdash \tau_1 <: \tau_2 \quad \Gamma \vdash \tau_2 <: \tau_3}{\Gamma \vdash \tau_1 <: \tau_3}\ (\textsc{St-Transitive})$$

$$\frac{\Gamma \vdash e : \tau_1 \quad \Gamma \vdash \tau_1 <: \tau_2}{\Gamma \vdash e : \tau_2}\ (\textsc{St-Subsumption})$$

$$\frac{\Gamma \vdash \{\bar{\sigma}_1\} \text{ is a permutation of } \{\bar{\sigma}_2\}}{\Gamma \vdash \{\bar{\sigma}_1\} <: \{\bar{\sigma}_2\}}\ (\textsc{St-Permutation}_\sigma) \qquad \frac{\Gamma \vdash \{\bar{d}_1\} \text{ is a permutation of } \{\bar{d}_2\}}{\Gamma \vdash \{\bar{d}_1\} <: \{\bar{d}_2\}}\ (\textsc{St-Permutation}_d)$$

$$\frac{\Gamma \vdash \sigma_i <:: \sigma_j}{\Gamma \vdash \{\sigma_i{}^{\,i\in 1..n}\} <: \{\sigma_j{}^{\,j\in 1..n}\}}\ (\textsc{St-Depth}_\sigma) \qquad \frac{\Gamma \vdash d_i <:: d_j}{\Gamma \vdash \{d_i{}^{\,i\in 1..n}\} <: \{d_j{}^{\,j\in 1..n}\}}\ (\textsc{St-Depth}_d)$$

$$\frac{n, k \geq 0}{\Gamma \vdash \{\sigma_i{}^{\,i\in 1..n+k}\} <: \{\sigma_i{}^{\,i\in 1..n}\}}\ (\textsc{St-Width}_\sigma) \qquad \frac{n, k \geq 0}{\Gamma \vdash \{d_i{}^{\,i\in 1..n+k}\} <: \{d_i{}^{\,i\in 1..n}\}}\ (\textsc{St-Width}_d)$$

$\boxed{\Gamma \vdash \sigma <:: \sigma}$

$$\frac{\begin{array}{c}\sigma_i = \texttt{def}\ m_A(y : \tau_1) : \tau_2\ \texttt{with}\ \varepsilon_A \qquad \sigma_j = \texttt{def}\ m_B(y : \tau_1') : \tau_2'\ \texttt{with}\ \varepsilon_B \\ \Gamma \vdash \tau_1' <: \tau_1 \qquad \Gamma \vdash \tau_2 <: \tau_2' \qquad \varepsilon_A \subseteq \varepsilon_B\end{array}}{\Gamma \vdash \sigma_i <:: \sigma_j}\ (\textsc{St-Method}_\sigma)$$

$\boxed{\Gamma \vdash d <:: d}$

$$\frac{\begin{array}{c}d_i = \texttt{def}\ m_A(y : \tau_1) : \tau_2 \qquad d_j = \texttt{def}\ m_B(y : \tau_1') : \tau_2' \\ \Gamma \vdash \tau_1' <: \tau_1 \qquad \Gamma \vdash \tau_2 <: \tau_2'\end{array}}{\Gamma \vdash d_i <:: d_j}\ (\textsc{St-Method}_d)$$

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

$$\frac{\Gamma \vdash \tau_1 <: \tau_2 \qquad \varepsilon_1 \subseteq \varepsilon_2}{\Gamma \vdash \tau_1\ \texttt{with}\ \varepsilon_1 <: \tau_2\ \texttt{with}\ \varepsilon_2}\ (\textsc{St-Subsumption}_\varepsilon)$$

$$\frac{\{\bar{d}_1\} <: \{\bar{d}_2\} \quad \varepsilon_A \subseteq \varepsilon_B \quad \varepsilon_1 \subseteq \varepsilon_2}{\{\bar{d}_1\ \texttt{captures}\ \varepsilon_A\}\ \texttt{with}\ \varepsilon_1 <: \{\bar{d}_2\ \texttt{captures}\ \varepsilon_B\}\ \texttt{with}\ \varepsilon_2}\ (\textsc{St-Summary})$$

$\boxed{\Gamma \vdash \tau\ \texttt{WFT}}$

$$\frac{}{\Gamma, r : \{r\} \vdash \{r\}\ \texttt{WFT}}\ (\text{WFT-Resource}) \qquad \frac{}{\Gamma, x : \tau \vdash \tau\ \texttt{WFT}}\ (\text{WFT-Variable})$$

$$\frac{\begin{array}{c} d_i = \texttt{def}\ m(y : \tau_2) : \tau_3 \\ \Gamma \vdash \tau_2\ \texttt{WFT} \qquad \Gamma \vdash \tau_3\ \texttt{WFT} \end{array}}{\Gamma \vdash \{\bar{d}\}\ \texttt{WFT}}\ (\text{WFT-Obj}_d) \qquad \frac{\begin{array}{c} \sigma_i = \texttt{def}\ m(y : \tau_2) : \tau_3\ \texttt{with}\ \varepsilon \\ \Gamma \vdash \tau_2\ \texttt{WFT} \qquad \Gamma \vdash \tau_3\ \texttt{WFT} \qquad r.\pi \in \varepsilon \implies \Gamma \vdash r\ \texttt{WFT} \end{array}}{\Gamma \vdash \{\bar{\sigma}\}\ \texttt{WFT}}\ (\text{WFT-Obj}_\sigma)$$

$$\frac{\Gamma \vdash \{\bar{d}\}\ \texttt{WFT} \qquad r.\pi \in \varepsilon_c \implies \Gamma \vdash r\ \texttt{WFT}}{\Gamma \vdash \{\bar{d}\ \texttt{captures}\ \varepsilon_c\}\ \texttt{WFT}}\ (\text{WFT-Summary})$$

$\boxed{\Gamma \vdash e\ \texttt{WFE}}$

$$\frac{}{\Gamma, r : \{r\} \vdash r\ \texttt{WFE}}\ (\text{WFE-Resource}) \qquad \frac{}{\Gamma, x : \tau \vdash x\ \texttt{WFE}}\ (\text{WFE-Variable})$$

$$\frac{\Gamma \vdash e\ \texttt{WFE}}{\Gamma \vdash e.\pi\ \texttt{WFE}}\ (\text{WFE-Operation}) \qquad \frac{\Gamma \vdash e_1\ \texttt{WFE} \qquad \Gamma \vdash e_2\ \texttt{WFE}}{\Gamma \vdash e_1.m(e_2)\ \texttt{WFE}}\ (\text{WFE-MethCall})$$

$$\frac{\begin{array}{c} d_i = \texttt{def}\ m(y : \tau_2) : \tau_3 \\ \Gamma \vdash \tau_2\ \texttt{WFT} \qquad \Gamma \vdash \tau_3\ \texttt{WFT} \qquad \Gamma, y : \tau_2 \vdash e_i\ \texttt{WFE} \end{array}}{\Gamma \vdash \texttt{new}_d\ x \Rightarrow \overline{d = e}\ \texttt{WFE}}\ (\text{WFE-Obj}_d)$$

$$\frac{\begin{array}{c} \sigma_i = \texttt{def}\ m(y : \tau_2) : \tau_3\ \texttt{with}\ \varepsilon \qquad r.\pi \in \varepsilon \implies \Gamma \vdash \{r\}\ \texttt{WFT} \\ \Gamma \vdash \tau_2\ \texttt{WFT} \qquad \Gamma \vdash \tau_3\ \texttt{WFT} \qquad \Gamma, y : \tau_2 \vdash e_i\ \texttt{WFE} \end{array}}{\Gamma \vdash \texttt{new}_\sigma\ x \Rightarrow \overline{\sigma = e}\ \texttt{WFE}}\ (\text{WFE-Obj}_\sigma)$$

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**.
- 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).
- A good choice of Γ' would be Γ restricted to the free variables in the object definition.
- C-NewObj: in the premise we need $\texttt{capture}(\tau) \supseteq \varepsilon_c$, for every type of every argument of every visible method. This is to ensure any capabilities passed to that method don't exceed what the type signature says.
- C-MethCall: we must add $\texttt{capture}(\tau_2)$ 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).
- By convention we use ε_c to denote the output of the `capture` function.
- If Γ can prove that an expression is well-formed (WFE) then that means there are no free variables in any types (all types are WFT) and subexpressions are WFE.

2.2 `capture` Function

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

- $\texttt{capture}(\varnothing) = \varnothing$
- $\texttt{capture}(\Gamma, x : \tau) = \texttt{capture}(\Gamma) \cup \texttt{capture}(\tau)$

- $\texttt{capture}(\{r\}) = \{r.\pi \mid \pi \in \Pi\}$
- $\texttt{capture}(\{\bar{\sigma}\}) = \bigcup_{\sigma \in \bar{\sigma}} \ \texttt{capture}(\sigma)$
- $\texttt{capture}(\{\bar{d}\}) = \bigcup_{d \in \bar{d}} \ \texttt{capture}(d)$
- $\texttt{capture}(d \ \texttt{with} \ \varepsilon) = \varepsilon \cup \texttt{capture}(d)$
- $\texttt{capture}(\texttt{def m}(x : \tau_2) : \tau_3) = \texttt{capture}(\tau_2) \cup \texttt{capture}(\tau_3)$
- $\texttt{capture}(\{\bar{d} \ \texttt{captures} \ \varepsilon_c\}) = \varepsilon_c$

Notes:

- 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 (will this matter?).
- `capture` is monotonic: if $\Gamma_1 \subseteq \Gamma_2$ then $\texttt{capture}(\Gamma_1) \subseteq \texttt{capture}(\Gamma_2)$.

2.3 `arg-types` Function

This function examines the declaration of every method which could be (directly) invoked inside a particular Γ. It returns a set of the types of the arguments of those methods.

- $\texttt{arg-types}(\varnothing) = \varnothing$
- $\texttt{arg-types}(\Gamma, x : \tau) = \texttt{arg-types}(\Gamma) \cup \texttt{arg-types}(\tau)$
- $\texttt{arg-types}(\{r\}) = \varnothing$
- $\texttt{arg-types}(\{\bar{\sigma}\}) = \bigcup_{\sigma \in \bar{\sigma}} \ \texttt{arg-types}(\sigma)$
- $\texttt{arg-types}(\{\bar{d}\}) = \bigcup_{d \in \bar{d}} \ \texttt{arg-types}(d)$
- $\texttt{arg-types}(\{\bar{d} \ \texttt{captures} \ \varepsilon_c\}) = \texttt{arg-types}(\{\bar{d}\})$
- $\texttt{arg-types}(d \ \texttt{with} \ \varepsilon) = \texttt{arg-types}(d)$
- $\texttt{arg-types}(\texttt{def} \ m(y : \tau_2) : \tau_3) = \{\tau_2\} \cup \texttt{arg-types}(\tau_3) \cup \texttt{arg-types}(\tau_2)$ (is $\texttt{arg-types}(\tau_2)$ necessary?)

2.4 `higher-order-args` Function

$$\frac{\tau \in \texttt{arg-types}(\Gamma) \quad \texttt{is-higher-order}(\tau)}{\tau \in \texttt{higher-order-args}(\Gamma)} \ (\textsc{HigherOrderArgs})$$

2.5 `is-obj` Predicate

The `is-obj` predicate says whether or not a particular type τ is an object.

$$\frac{}{\texttt{is-obj}(\{\bar{d}\})} \ (\textsc{IsObj}_d) \qquad \frac{}{\texttt{is-obj}(\{\bar{\sigma}\})} \ (\textsc{IsObj}_\sigma) \qquad \frac{}{\texttt{is-obj}(\{\bar{d} \ \texttt{captures} \ \varepsilon_c\})} \ (\textsc{IsObjSummary})$$

2.6 `is-higher-order` Predicate

A type is higher-order if it has a method accepting another object as an argument.

$$\frac{d_i = \texttt{def} \ m(y : \tau_2) : \tau_3 \quad \texttt{is-obj}(\tau_2)}{\texttt{is-higher-order}(\{\bar{d}\})} \ (\textsc{HigherOrder}_d)$$

$$\frac{\sigma_i = \texttt{def} \ m(y : \tau_2) : \tau_3 \ \texttt{with} \ \varepsilon \quad \texttt{is-obj}(\tau_2)}{\texttt{is-higher-order}(\{\bar{\sigma}\})} \ (\textsc{HigherOrder}_\sigma)$$

2.7 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.

2.8 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 $z \notin \texttt{freevars}(e_i)$
- $[e'/z](\texttt{new}_\sigma\ x \Rightarrow \overline{\sigma = e}) = \texttt{new}_\sigma\ x \Rightarrow \overline{\sigma = [e'/z]e}$, 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 ε, then $e = r$ for some resource r.
- If $\Gamma \vdash e : \{\bar{\sigma}\}$ with ε, then $e = \mathtt{new}_\sigma\ x \Rightarrow \overline{\sigma = e}$.
- If $\Gamma \vdash e : \{\bar{d}$ captures $\varepsilon_c\}$ with ε, 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 $\Gamma, z : \tau' \vdash e : \tau$ with ε, and $\Gamma \vdash e' : \tau'$ with ε', then $\Gamma \vdash [e'/z]e : \tau$ with ε.

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

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

□

Lemma 3.3. (Well-Formedness Principle)

Statement. If $\Gamma \vdash \tau$ WFT then $\mathtt{capture}(\tau) \subseteq \mathtt{capture}(\Gamma)$

Proof. By induction on the judgement $\Gamma \vdash \tau$ WFT.

Case. WFT-Resource.
Then $\mathtt{capture}(\tau) = \mathtt{capture}(\{r\}) \subseteq \mathtt{capture}(\Gamma, r : \{r\})$.

Case. WFT-Variable.
Then $\mathtt{capture}(\tau) \subseteq \mathtt{capture}(\Gamma, x : \tau)$

$\boxed{\text{Case.}}$ WFT-OBJ$_d$.
Then for any $\texttt{def}\ m(y : \tau_2) : \tau_3 \in \bar{d}$, we have by inversion $\Gamma \vdash \tau_2$ $\texttt{WFT}$ and $\Gamma \vdash \tau_3$ $\texttt{WFT}$. By inductive assumption, $\texttt{capture}(\tau_2) \cup \texttt{capture}(\tau_3) \subseteq \texttt{capture}(\Gamma)$. But this is $\texttt{capture}(d)$, for an arbitrary $d \in \bar{d}$, so $\texttt{capture}(\{\bar{d}\}) \subseteq \texttt{capture}(\Gamma)$.

$\boxed{\text{Case.}}$ WFT-OBJ$_\sigma$.
Then for any $\texttt{def} m(y : \tau_2) : \tau_3$ $\texttt{with}$ $\varepsilon \in \bar{\sigma}$, we have by inversion $\Gamma \vdash \tau_2$ $\texttt{WFT}$ and $\Gamma \vdash \tau_{\texttt{WFT}}$. By inductive assumption, $\texttt{capture}(\tau_2) \cup \texttt{capture}(\tau_3) \subseteq \texttt{capture}(\Gamma)$. Also by inversion we know that $\Gamma \vdash \{r\}\texttt{WFT}$, for any $r.\pi \in \varepsilon$. By inductive assumption, $\texttt{capture}(\{r\}) \subseteq \texttt{capture}(\Gamma)$. Now $\texttt{capture}(\sigma) = \texttt{capture}(\tau_2) \cup \texttt{capture}(\tau_3) \cup \varepsilon \subseteq \texttt{capture}(\Gamma)$. This is for an arbitrary $\sigma \in \bar{\sigma}$, so $\texttt{capture}(\{\bar{\sigma}\}) \subseteq \texttt{capture}(\Gamma)$.

$\boxed{\text{Case.}}$ WFT-SUMMARY.
By inversion we know that $\texttt{capture}(\{r\}) \subseteq \texttt{capture}(\Gamma)$, for any $r.\pi \in \varepsilon_c$. The union of these is exactly $\texttt{capture}(\{\bar{d}$ $\texttt{captures}$ $\varepsilon_c\})$.

$\square$

Lemma 3.4. (Use Principle)

$\boxed{\text{Statement.}}$ If $\Gamma \vdash e$ $\texttt{WF}$ and $\Gamma \vdash e : \tau$ $\texttt{with}$ ε, then $\varepsilon \subseteq \texttt{capture}(\Gamma)$.

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

$\boxed{\text{Case.}}$ ε-VAR, ε-RESOURCE, ε-NEWOBJ.
Then e is a value. By canonical forms, $\varepsilon = \varnothing \subseteq \texttt{capture}(\Gamma)$.

$\boxed{\text{Case.}}$ C-NEWOBJ.
Then $\Gamma \vdash e : \{\bar{d}$ $\texttt{captures}$ $\varepsilon_c\}$ $\texttt{with}$ $\varnothing$, and $\varnothing \subseteq \texttt{capture}(\Gamma)$.

$\boxed{\text{Case.}}$ C-METHCALL.
Then $e = e_1.m_i(e_2)$. From inversion on C-METHCALL we know $\Gamma \vdash e_1 : \{\bar{d}$ $\texttt{captures}$ $\varepsilon_c\}$ $\texttt{with}$ ε_1 and $\Gamma \vdash e_2 : \tau_2$ $\texttt{with}$ ε_2, where $\varepsilon_c = \texttt{capture}(\Gamma')$ and $\Gamma' \subseteq \Gamma$. The typing rule also gives us $\Gamma \vdash e_1.m_i(e_2) : \varepsilon_1 \cup \varepsilon_2 \cup \varepsilon_c$.

We know $\varepsilon_1 \subseteq \texttt{capture}(\Gamma)$ and $\varepsilon_2 \subseteq \texttt{capture}(\Gamma)$ by inductive assumption on the subexpressions e_1 and e_2. Since $\Gamma' \subseteq \Gamma$ then $\varepsilon_c = \texttt{capture}(\Gamma') \subseteq \texttt{capture}(\Gamma)$.

$\boxed{\text{Case.}}$ ε-OPERCALL.
Then $e = e_1.\pi$ and $\Gamma \vdash e : \texttt{Unit}$ $\texttt{with}$ $\varepsilon_1 \cup \{r.\pi\}$, where $\Gamma \vdash e_1 : \{r\}$ $\texttt{with}$ ε_1. By inductive assumption, $\varepsilon_1 \subseteq \texttt{capture}(\Gamma)$. By well-formedness, $r \in \Gamma$ which implies that $r.\pi \in \texttt{capture}(\Gamma)$.

$\boxed{\text{Case.}}$ ε-METHCALL.
Then $e = e_1.m_i(e_2)$ and $\Gamma \vdash e_1.m_i(e_2) : \varepsilon_1 \cup \varepsilon_2 \cup \varepsilon_3$. By inductive assumption $\varepsilon_1 \cup \varepsilon_2 \subseteq \texttt{capture}(\Gamma)$. To show $\varepsilon_3 \subseteq \texttt{capture}(\Gamma)$ consider inversion on ε-NEWOBJ and then again on ε-VALIDIMPL$_\sigma$. From this we get the subderivation $\Gamma, x : \{\bar{\sigma}\}, y : \tau_2 \vdash e_{body} : \tau_3$ $\texttt{with}$ ε_3. By inductive assumption, $\varepsilon_3 \subseteq \texttt{capture}(\Gamma) \cup \texttt{capture}(\{\bar{\sigma}\}) \cup \texttt{capture}(\tau_2)$.

By definition, $\texttt{capture}(\{\bar{\sigma}\}) = \bigcup_{\sigma \in \bar{\sigma}} \texttt{capture}(\sigma)$. For any particular $\sigma_i = \texttt{def}\ m_i(y : \tau_2) : \tau_3$ $\texttt{with}$ ε_3, we have $\texttt{capture}(\sigma_i) = \texttt{capture}(\tau_2) \cup \texttt{capture}(\tau_3)$.

By inversion on $\Gamma \vdash e$ $\texttt{WFE}$ we know $\Gamma \vdash e_1$ $\texttt{WFE}$. By the well-formedness principle, for any $\tau \in e_1$ we have $\texttt{capture}(\tau) \subseteq \texttt{capture}(\Gamma)$. Then $\texttt{capture}(\tau_2) \cup \texttt{capture}(\tau_3) \subseteq \texttt{capture}(\Gamma)$ This was for an arbitrary σ_i so $\texttt{capture}(\{\bar{\sigma}\}) = \bigcup_{\sigma \in \bar{\sigma}} \texttt{capture}(\sigma) \subseteq \texttt{capture}(\Gamma)$.

$\square$

Theorem 3.5. (Inference Lemma)

Statement. If the following are true:

- $\Gamma \vdash e$ WF
- $\Gamma \vdash e : \tau$
- $\tau \in \texttt{higher-order-args}(\Gamma) \implies \texttt{capture}(\tau) \supseteq \texttt{capture}(\Gamma)$

Then $\Gamma \vdash e : \tau'$ with ε, where $\tau' <: \tau$ and $\varepsilon \subseteq \texttt{effects}(\Gamma)$.

Proof. The proof is by induction on $\Gamma \vdash e$. It suffices to show that $\Gamma \vdash e : \tau'$ with ε in each case. The relation $\varepsilon \subseteq \texttt{effects}(\Gamma)$ holds by the use principle. Furthermore, in every case except T-MethCall$_d$, $\tau' = \tau$. **(NOTE: THIS RELIES ON $\{\bar{d}$ captures $\varepsilon\} <: \{\bar{d}\}$ BEING TRUE)**

Case. T-Var.
You can directly apply ε-Var. Then $\Gamma \vdash x : \tau$ with $\varnothing$.

Case. T-resource.
You can directly apply ε-Resource. Then $\Gamma \vdash r : \{r\}$ with $\varnothing$.

Case. T-OperCall.
Then $e = e_1.\pi$. By inversion $\Gamma \vdash e_1 : \{r\}$ and by inductive hypothesis, $\Gamma \vdash e_1 : \{r\}$ with ε_1. Then by ε-OperCall we know $\Gamma \vdash e_1.\pi : \{r\}$ with $\varepsilon \cup \{r.\pi\}$.

Case. T-MethCall$_\sigma$.
Then $e = e_1.m(e_2)$, where the method m is def $m(y : \tau_2) : \tau_3$ with ε_3. By inversion and inductive hypothesis, $\Gamma \vdash e_1 : \{\bar{\sigma}\}$ with ε_1 and $\Gamma \vdash e_2 : \tau_2$ with ε_2. By applying ε-MethCall we get $\Gamma \vdash e_1.m(e_2) : \tau_3$ with $\varepsilon_1 \cup \varepsilon_2 \cup \varepsilon_3$.

Case. T-MethCall$_d$.
Then $e = e_1.m(e_2)$, where the method m is def $m(y : \tau_2) : \tau_3$. By inversion and inductive hypothesis, $\Gamma \vdash e_1 : \tau_1'$ with ε_1. By inspection, the only rule which could have ascribed this is C-NewObj, from which we learn $\tau_1' = \{\bar{d}$ captures $\varepsilon_c\}$, where $\varepsilon_c = \texttt{capture}(\Gamma')$ and $\Gamma' \subseteq \Gamma$. By inversion and inductive hypothesis again, we have $\Gamma \vdash e_2 : \tau_2$ with ε_2. From an application of C-MethCall, we have $\Gamma \vdash e_1.m(e_2) : \tau_3$ with $\varepsilon_1 \cup \varepsilon_2 \cup \varepsilon_c \cup \texttt{capture}(\tau_2)$.

Case. T-New$_\sigma$.
Then $e = \texttt{new}_\sigma\ x \Rightarrow \overline{\sigma = e}$. By inversion of T-New$_\sigma$, $\Gamma \vdash \overline{\sigma = e}$ OK . This is exactly the premise of ε-NewObj, which gives us the judgement $\Gamma \vdash \texttt{new}_\sigma\ x \Rightarrow \overline{\sigma = e} : \{\bar{\sigma}\}$ with $\varnothing$.

Case. T-New$_d$.
Then $e = \texttt{new}_d\ x \Rightarrow \overline{d = e}$ and $\Gamma \vdash e : \{d\}$. To type e with an effect we shall use C-NewObj, selecting $\Gamma' = \Gamma$. We know $\tau \in \texttt{higher-order-args}(\Gamma) \implies \texttt{capture}(\tau) \supseteq \texttt{capture}(\Gamma)$ from the theorem statement. Therefore we may apply C-NewObj. The result is that $\Gamma \vdash e : \{\bar{d}$ captures $\varepsilon_c\}$ with $\varnothing$, where $\varepsilon_c = \texttt{effects}(\Gamma)$.

□

Theorem 3.6. (Jump Lemma)

Statement. If $\Gamma \vdash e : \{\bar{d}$ captures $\varepsilon_c\}$ with ε, then for a particular $d_{body} = e_{body}$, that is, def $m(y : \tau_2) : \tau_3 = e_{body}$, the following is true.

- $\Gamma, x : \{\bar{d}\}, y : \tau_2 \vdash e_{body} : \tau_3'$ with ε_{body}
- $\tau_3' <: \tau_3$
- $\varepsilon_{body} \subseteq \varepsilon_c \cup \texttt{capture}(\tau_2)$

Proof. From inversion on $\Gamma \vdash e : \{\bar{d}$ captures $\varepsilon_c\}$ with ε we know, for some $\Gamma' \subseteq \Gamma$, that $\Gamma', x : \{\bar{d}$ captures $\varepsilon_c\} \vdash d = e$ OK. Also $\varepsilon_c = \texttt{capture}(\Gamma')$. Fix some particular method def $m(y : \tau_2) : \tau_3 = e_{body}$.

The theorem assumes a typing judgement $\Gamma \vdash \{\bar{d}$ captures $\varepsilon_c\}$ with ε. By inversion on C-NewObj we know $\Gamma \vdash e : \{\bar{d}\}$. By inversion on T-New$_d$ we know $\Gamma, x : \{\bar{d}\} \vdash d = e_{body}$ OK. By inversion on ValidImpl$_d$ we know that $\Gamma, x : \{\bar{d}\}, y : \tau_2 \vdash e_{body} : \tau_3$. For concision, define $\hat{\Gamma}$ as $\Gamma, x : \{\bar{d}\}, y : \tau_2$.

Note that $\texttt{capture}(\hat{\Gamma}) = \texttt{capture}(\Gamma') \cup \texttt{capture}(\tau_2) = \varepsilon_c \cup \texttt{capture}(\tau_2)$. This is because $\varepsilon_c = \texttt{capture}(\Gamma') \subseteq \texttt{capture}(\Gamma)$, by monotonicity.

By the Inference Lemma, $\hat{\Gamma} \vdash e_{body} : \tau_3' \texttt{with } \varepsilon_3$. By the Use Principle, $\varepsilon_3 \subseteq \texttt{effects}(\hat{\Gamma}) = \texttt{effects}(\Gamma') \cup \texttt{capture}(\tau_2)$.

□

Theorem 3.7. (Soundness Theorem)

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

$\boxed{\text{Proof.}}$ By induction on the judgement $\Gamma \vdash e_A : \tau_A \texttt{ with } \varepsilon_A$.

$\boxed{\text{Case.}}$ C-MethCall.
Then $e = e_1.m_i(e_2)$ and we know the following.

1. $\Gamma \vdash e_1 : \{\bar{d} \texttt{ captures } \varepsilon_c\} \texttt{ with } \varepsilon_1$
2. $\Gamma \vdash e_2 : \tau_2 \texttt{ with } \varepsilon_2$
3. $d_i = \texttt{def } m_i(y : \tau_2) : \tau_3$
4. $\Gamma \vdash e_1.m_i(e_2) : \tau_3 \texttt{ with } \varepsilon_1 \cup \varepsilon_2 \cup \texttt{effects } (\tau_2) \cup \varepsilon_c$
5. $\tau \in \texttt{higher-order-args}(\Gamma') \implies \texttt{capture}(\tau) \supseteq \varepsilon_c$

There are three reduction rules which could be applied to $e_1.m_i(e_2)$, where e_1 is a (deeply) unlabelled 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 applying the inductive assumption to e_1, we know that $\Gamma \vdash e_1' : \tau_1' \texttt{ with } \varepsilon_1'$, where $\tau_1' <: \{\bar{d} \texttt{ captures } \varepsilon_c\}$ and $\varepsilon_1' \cup \varepsilon = \varepsilon_1$.

We can typecheck $e_1'.m_i(e_2)$ with E-MethCall. Then $\Gamma \vdash e_1'.m_i(e_2) : \tau_3 \texttt{ with } \varepsilon_1' \cup \varepsilon_2 \cup \texttt{capture}(\tau_2) \cup \varepsilon_c$. $\tau_3 <: \tau_3$ trivially and since $\varepsilon_1' \cup \varepsilon = \varepsilon_1$, we have $\varepsilon \cup \varepsilon_B = \varepsilon \cup \varepsilon_1' \cup \varepsilon_2 \cup \texttt{capture}(\tau_2) \cup \varepsilon_c = \varepsilon_1 \cup \varepsilon_2 \cup \texttt{capture}(\tau_2) \cup \varepsilon_c = \varepsilon_A$.

Subcase. E-MethCall2 Then we know $e_1 = v_1 = \texttt{new}_d\ x \Rightarrow \overline{d = e}$ and $v_1.m_i(e_2) \longrightarrow v_1.m_i(e_2') \mid \varepsilon$, where $e_2 \longrightarrow e_2' \mid \varepsilon$. By applying the inductive assumption to e_2, we know that $\Gamma \vdash e_2' : \tau_2' \texttt{ with } \varepsilon_2'$, where $\tau_2' <: \tau_2$ and $\varepsilon_2' \cup \varepsilon \subseteq \varepsilon_2$.

We can typecheck $v_1.m_i(e_2')$ with E-MethCall. Then $\Gamma \vdash v_1.m_i(e_2') : \tau_3 \texttt{ with } \varepsilon_1 \cup \varepsilon_2' \cup \texttt{capture}(\tau_2) \cup \varepsilon_c$. $\tau_3 <: \tau_3$ trivially and since $\varepsilon_2' \cup \varepsilon \subseteq \varepsilon_2$, we have $\varepsilon \cup \varepsilon_B = \varepsilon \cup \varepsilon_1 \cup \varepsilon_2' \cup \texttt{capture}(\tau_2) \cup \varepsilon_c = \varepsilon_1 \cup \varepsilon_2 \cup \texttt{capture}(\tau_2) \cup \varepsilon_c = \varepsilon_A$.

Subcase. E-MethCall3 Then we know $e_1 = v_1 = \texttt{new}_d\ x \Rightarrow \overline{d = e}$ and $e_2 = v_2$ and $v_1.m_i(v_2) \longrightarrow [v_1/x, v_2/y]e_{body} \mid \varnothing$. Since $\varepsilon = \varnothing$, it is sufficient to type $[v_1/x, v_2/y]e_{body} : \tau_{body} \texttt{ with } \varepsilon_{body}$, where $\tau_{body} <: \tau_A$ and $\varepsilon_{body} \subseteq \varepsilon_A$.

By inversion on (1) we know $\Gamma' \subseteq \Gamma$ and $\Gamma' \vdash e_1 : \{\bar{d}\}$. By inversion on ValidImpl$_d$ we know $\Gamma' \vdash \overline{d_{body} = e_{body}}\ \texttt{OK}$, for every method defined in e_1. Then if method m_i is $d_{body} = e_{body}$, we know $\Gamma', y : \tau_2 \vdash e_{body} : \tau_3$ from inversion on ValidImpl$_d$. By the jump lemma, $\Gamma', y : \tau_2 \vdash e_{body} : \tau_3 \texttt{ with } \varepsilon_B$.

Now in this case, since e_1 and e_2 are values, then $\varepsilon_1 = \varepsilon_2 = \varnothing$ by canonical forms. Then $\varepsilon_A = \varepsilon_c \cup \texttt{capture}(\tau_2) = \texttt{capture}(\Gamma') \cup \texttt{capture}(\tau_2)$. Since $\varepsilon = \varnothing$, we just have to show that $\varepsilon_B \subseteq \varepsilon_A$. The jump lemma also tells us that $\varepsilon_B \subseteq \texttt{capture}(\Gamma', y : \tau_2) = \texttt{capture}(\Gamma') \cup \texttt{capture}(\tau_2) = \varepsilon_A$.

$\boxed{\text{Case.}}$ ε-MethCall.

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

1. $\Gamma \vdash e_1 : \{\bar{\sigma}\}$ `with` ε_1
2. $\Gamma \vdash e_2 : \tau_2$ `with` ε_2
3. $\sigma_i =$ `def` $m_i(y : \tau_2) : \tau_3$ `with` ε_3
4. $\Gamma \vdash e_1.m_i(e_2) : \tau_3$ `with` $\varepsilon_1 \cup \varepsilon_2 \cup \varepsilon_3$

There are three reduction rules which could have been applied, depending on which of e_1 and e_2 are values.

Subcase. E-MethCall1. Then we know $e_1 \longrightarrow e_1' \mid \varepsilon$, and the reduction in the theorem statement is $e_1.m(e_2) \longrightarrow e_1'.m(e_2) \mid \varepsilon$. By inductive assumption, $\Gamma \vdash e_1' : \tau_1'$ `with` ε_1', where $\tau_1' <: \tau_1$ and $\varepsilon_1' \cup \varepsilon = \varepsilon_1$. We may type $e_1'.m_i(e_2)$ using the rule ε-MethCall, from which we get $\Gamma \vdash e_1'.m_i(e_2) : \tau_3'$ `with` $\varepsilon_1' \cup \varepsilon_2 \cup \varepsilon_3$, where $\tau_3' <: \tau_3$. Furthermore, $\varepsilon \cup \varepsilon_B = \varepsilon \cup \varepsilon_1' \cup \varepsilon_2 \cup \varepsilon_3 = \varepsilon_1 \cup \varepsilon_2 \cup \varepsilon_3 = \varepsilon_A$.

Subcase. E-MethCall2. Then we know $e_1 = v_1$ is a value, and $e_2 \longrightarrow e_2' \mid \varepsilon$, and the reduction in the theorem statement is $v_1.m(e_2) \longrightarrow v_1.m(e_2') \mid \varepsilon$. By inductive assumption, $\Gamma \vdash e_2' : \tau_2'$ `with` ε_2', where $\tau_2' <: \tau_2$ and $\varepsilon_2' \cup \varepsilon = \varepsilon_2$. We may type $v_1.m_i(e_2')$ using the rule ε-MethCall, from which we get $\Gamma \vdash v_1.m_i(e_2') : \tau_3' \mid \varepsilon_1 \cup \varepsilon_2' \cup \varepsilon_3$. By canonical forms, since $e_1 = v_1$ is a value, $\varepsilon_1 = \varnothing$. Then $\varepsilon \cup \varepsilon_B = \varepsilon \cup \varepsilon_2' \cup \varepsilon_3 = \varepsilon_2 \cup \varepsilon_3 = \varepsilon_A$.

Subcase. E-MethCall3. Then we know $e_1 = v_1$ and $e_2 = v_2$ are both values and the reduction in the theorem statement is $v_1.m_i(v_2) \longrightarrow [v_1/x, v_2/y]e_{body} \mid \varnothing$. Because $\Gamma \vdash v_1 : \{\bar{\sigma}\}$ `with` $\varnothing$, by inversion on the rule ε-NewObj and then by inversion on the rule ε-ValidImpl$_\sigma$, we know $\Gamma, y : \tau_2 \vdash e_{body} : \tau_3$ `with` ε_3. From the substitution lemma, we have $\Gamma, y : \tau_2 \vdash [v_1/x, v_2/y]e_{body} : \tau_3$ `with` ε_3, so $\tau_A = \tau_B$.

Since v_1 and v_2 are values, by canonical forms $\varepsilon_1 = \varepsilon_2 = \varnothing$, so $\varepsilon_A = \varepsilon_3 = \varnothing \cup \varepsilon_B = \varepsilon \cup \varepsilon_B$.

$\boxed{\text{Case.}}$ ε-OperCall.

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

1. $\Gamma \vdash e_1 : \{r\}$ `with` ε_1
2. $\Gamma \vdash e_1.\pi :$ `Unit with` $\{r.\pi\} \cup \varepsilon_1$

There are two reduction rules which could have been applied, depending on whether e_1 is a value or not.

Subcase. E-OperCall1. Then we know $e_1 \longrightarrow e_1' \mid \varepsilon$, and the reduction in the theorem statement is $e_1.\pi \longrightarrow e_1'.\pi \mid \varepsilon$. By induction assumption, $\Gamma \vdash e_1' : \tau_1'$ `with` ε_1', where $\tau_1' <: \tau_1$ and $\varepsilon_1 \cup \varepsilon = \varepsilon_1'$. We can type $e_1'.\pi$ with the rule ε-OperCall. We get $\Gamma \vdash e_1'.\pi :$ `Unit with` $\varepsilon_1' \cup \{r.\pi\}$. Then $\tau_A = \tau_B =$ `Unit` and $\varepsilon_1' \cup \{r.pi\} \cup \varepsilon \subseteq \varepsilon_1 \cup \{r.\pi\}$, from inductive assumption.

Subcase. E-OperCall2 Then we know $e_1 = v_1 = r$ is a resource, and the reduction in the theorem statement is $r.\pi \longrightarrow$ `unit` $\mid \{r.\pi\}$. By a trivial application of $\varepsilon - NewObj$, we have $\Gamma \vdash$ `unit` : `Unit with` $\varnothing$. Since r is an object, by canonical forms we know $\varepsilon_A = \varnothing \cup \{r.\pi\}$, and $\varepsilon_B \cup \varepsilon = \varnothing \cup \{r.\pi\}$.

$\boxed{\text{Case.}}$ ε-Var, ε-Resource, ε-NewObj, C-NewObj.

Then e_A is a value and cannot be reduced. Soundness holds trivially.

$\square$