1 Grammar

$$
\begin{array}{lll}
\rho & ::= x & \textit{primitives} \\
& \mid \quad r & \\
\\
\tau_\rho & ::= \{r\} & \textit{primitive types}
\end{array}
$$

$$
\begin{array}{lll}
e_u & ::= \rho & \textit{deeply unlabeled progs.} \\
& \mid \quad \texttt{new}_d\ x \Rightarrow \overline{d = e_u} & \\
& \mid \quad e_u.m(e_u) & \\
& \mid \quad e_u.\pi & \\
\\
d & ::= \texttt{def}\ m(y : \tau_u) : \tau_u & e_u\textit{-prog decls.} \\
\\
\tau_u & ::= \{\overline{d}\} & e_u\textit{-prog types} \\
& \mid \quad \{\overline{d}\ \texttt{captures}\ \varepsilon\} & \\
& \mid \quad \tau_\rho &
\end{array}
$$

$$
\begin{array}{lll}
e_l & ::= \rho & \textit{deeply labeled progs.} \\
& \mid \quad \texttt{new}_\sigma\ x \Rightarrow \overline{\sigma = e_l} & \\
& \mid \quad l.m(l) & \\
& \mid \quad l.\pi & \\
\\
\sigma & ::= \texttt{def}\ m(y : \tau_l) : \tau_l\ \texttt{with}\ \varepsilon & e_l\textit{-prog decls.} \\
\\
\tau_l & ::= \{\bar{\sigma}\} & e_l\textit{-prog types} \\
& \mid \quad \tau_\rho &
\end{array}
$$

$$
\begin{array}{lll}
e & ::= \rho & \textit{progs.} \\
& \mid \quad e.m(e) & \\
& \mid \quad e.\pi & \\
& \mid \quad \texttt{new}_d\ x \Rightarrow \overline{d = e_u} & \\
& \mid \quad \texttt{new}_\sigma\ x \Rightarrow \overline{\sigma = e} & \\
\\
\tau & ::= \tau_l & \textit{types} \\
& \mid \quad \tau_u &
\end{array}
$$

Notes:

- e_u programs are *deeply unlabeled* programs: no labels appear in the source code (though label inference may be done by the type system).
- e_l programs are *deeply labeled* programs: everything in the source code is labeled.
- e programs are the general form of a syntactically-correct program. They may contain a mixture of labeled and unlabeled parts. Any unlabeled parts must be deeply unlabeled, but labeled parts need not be deeply labeled. This means you can have unlabeled parts appearing inside labeled parts, but not vice versa.

2 Static Semantics

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

$$\frac{}{\Gamma, x : \tau \vdash x : \tau} \ (\rho\text{-VAR}) \qquad \frac{}{\Gamma, r : \{r\} \vdash r : \{r\}} \ (\rho\text{-RESOURCE})$$

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

$$\frac{}{\Gamma, x : \tau \vdash x : \tau \ \texttt{with} \ \varnothing} \ (\rho\text{-VAR}_\varepsilon) \qquad \frac{}{\Gamma, r : \{r\} \vdash r : \{r\} \ \texttt{with} \ \varnothing} \ (\rho\text{-RESOURCE}_\varepsilon)$$

$\boxed{\Gamma \vdash e_u : \tau_u}$

$$\frac{\Gamma, x : \{\overline{d}\} \vdash \overline{d = e_u} \ \texttt{OK}}{\Gamma \vdash \ \texttt{new}_d \ x \Rightarrow \overline{d = e_u} : \{\overline{d}\}} \ (e_u\text{-NEW}) \qquad \frac{\Gamma \vdash e_u : \{r\}}{\Gamma \vdash e_u.\pi : \texttt{Unit}} \ (e_u\text{-OPERCALL})$$

$$\frac{\Gamma \vdash e_{u,1} : \{\overline{d}\} \qquad \texttt{def} \ m(y : \tau_{u,2}) : \tau_{u,3} \in \{\overline{d}\} \qquad \Gamma \vdash e_{u,2} : \tau_{u,2}}{\Gamma \vdash e_{u,1}.m(e_{u,2}) : \tau_{u,3}} \ (e_u\text{-METHCALL})$$

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

$$\frac{d = \texttt{def} \ m(y : \tau_{u,2}) : \tau_{u,3} \quad \Gamma, y : \tau_{u,2} \vdash e_u : \tau_{u,3}}{\Gamma \vdash d = e_u \ \texttt{OK}} \ (e_u\text{-VALIDIMPL})$$

$\boxed{\Gamma \vdash e_u : \tau_u \ \texttt{with} \ \varepsilon}$

$$\frac{\varepsilon_c = \texttt{effects}(\Gamma') \qquad \Gamma' \subseteq \Gamma \qquad \Gamma' \vdash \ \texttt{new}_d \ x \Rightarrow \overline{d = e_u} : \{\overline{d}\}}{\Gamma \vdash \ \texttt{new}_d \ x \Rightarrow \overline{d = e_u} : \{\overline{d} \ \texttt{captures} \ \varepsilon_c\} \ \texttt{with} \ \varnothing} \ (e_u\text{-NEW}_\varepsilon)$$

$$\frac{\Gamma \vdash e_{u,1} : \{\overline{d} \ \texttt{captures} \ \varepsilon_c\} \ \texttt{with} \ \varepsilon_1 \qquad \Gamma \vdash e_{u,2} : \tau_{u,2} \ \texttt{with} \ \varepsilon_2 \qquad d = \ \texttt{def} \ m(y : \tau_{u,2}) : \tau_{u,3} \in \overline{d}}{\Gamma \vdash e_{u,1}.m(e_{u,2}) : \tau_{l,3} \ \texttt{with} \ \varepsilon_1 \cup \varepsilon_2 \cup \texttt{effects}(\tau_{u,2}) \cup \varepsilon_c} \ (e_u\text{-METHCALL}_\varepsilon)$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Notes:

- A good choice of Γ' for e_u-$\textsc{New}_\varepsilon$ is the intersection of Γ with the free variables in the object.
- By convention we use ε_c to denote the output of the `effects` function.

3 Definition: `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:

1. The function is monotonic; if $\Gamma_1 \subseteq \Gamma_2$, then $\texttt{effects}(\Gamma_1) \subseteq \texttt{effects}(\Gamma_2)$.

4 Dynamic Semantics

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

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

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

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

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

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

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

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

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

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 to unlabeled objects. Same goes for E-MethCall3$_\sigma$ and E-MethCall3$_d$.
- E-MethCall1 can be used for both labeled and unlabeled objects.

5 Definition (`substitution`)

TODO

6 Lemma (Canonical Forms)

Lemma. If u is a value then $\Gamma \vdash u : \tau$ `with` $\varnothing$, and one of the following is true:

1. $u = r$ and $\tau = \texttt{Unit}$.
2. $u = x$ and $x : \tau \in \Gamma$.
3. $u = \texttt{new}_d\ x \Rightarrow \overline{d = u}$ and $\tau = \{\bar{d}\}$

Proof. By inspection.

7 Lemma (Substitution)

Lemma. Suppose the following is true:

1. $\Gamma, z : \tau' \vdash e : \tau$ `with` ε
2. $\Gamma \vdash e' : \tau'$ `with` ε'

Then $\Gamma \vdash [e'/z]e : \tau$ `with` ε.

Proof. TODO (Should be same as the proof in previous grammar, just need to convert everything to new grammar)

8 Definition (`label`)

A program may be converted into a fully-labeled program. This is a function from e-terms to e_l-terms. It is always defined relative to some Γ, which is usually clear from context. The process is well-defined on e if $\Gamma \vdash e : \tau$ `with` ε. Then `label` is defined below.

1. $\texttt{label}(\rho) = \rho$
2. $\texttt{label}(u_1.\pi) = \texttt{label}(u_1).\pi$
3. $\texttt{label}(u_1.m(u_2)) = \texttt{label}(u_1).m(\texttt{label}(u_2))$
4. $\texttt{label}(\texttt{new}_d\ x \Rightarrow \overline{d = u}) = \texttt{new}_\sigma\ x \Rightarrow \underline{\texttt{label-decl}(\overline{d = u})}$
5. $\texttt{label}(\texttt{new}_\sigma\ x \Rightarrow \overline{\sigma = e}) = \texttt{new}_\sigma\ x \Rightarrow \sigma = \texttt{label}(e)$

The helper function `label-decl` works by labeling each declaration with what it captures in the context Γ. We abbreviate this as $\texttt{effects}(\Gamma \cap \texttt{freevars}(e))$. The helper is defined below.

5. $\texttt{label-decl}(d = u) = d$ `with` $\texttt{effects}(\Gamma \cap \texttt{freevars}(e)) = \texttt{label}(u)$

Notes:

- The image of $\texttt{label}(e_u)$ is an e_l-term (proof by induction on definition).
- e_u is a value $\iff$ $\texttt{label}(e_u)$ is a value.
- More rigorously we can define $\varnothing \cap \texttt{freevars}(e)$ as $\varnothing$, and $(\Gamma, x : \tau) \cap \texttt{freevars}(e)$ as $(\{x\} \cap \texttt{freevars}(e)) \cup (\Gamma \cap \texttt{freevars}(e))$.

9 Theorem (`label` and `sub` Commute)

TODO

10 Theorem (Runtime Invariant Under Labeling)

If the following are true:

1. $\Gamma \vdash e_A : \tau_A$ `with` ε_A
2. $e_A \longrightarrow e_B \mid \varepsilon$
3. $\hat{e}_A = \texttt{label}(e_A, \Gamma)$

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

11 Theorem (Refinement)

Theorem. Suppose $\Gamma \vdash u : \tau_A \texttt{ with } \varepsilon_A$. Then $\Gamma \vdash \texttt{label}(u) : \tau_B \texttt{ with } \varepsilon_B$ and:

1. $\tau_B <: \tau_A$
2. $\varepsilon_B \subseteq \varepsilon_A$

Proof. By induction on $\Gamma \vdash u : \tau_A \texttt{ with } \varepsilon_A$.

Case. T-NewInf.
Then the following are known.

5. $u = \texttt{new}_d\ x \Rightarrow \overline{d = u}$
6. $\tau_A = \{\bar{d} \texttt{ captures } \varepsilon_C\}$
7. $\varepsilon_A = \varnothing$
8. $\Gamma' \subseteq \Gamma$
9. $\Gamma', x : \{\bar{d} \texttt{ captures } \varepsilon_C\} \vdash \overline{d = u} \texttt{ OK}$
10. $\varepsilon_C = \texttt{effects}(\Gamma')$

Let $l = \texttt{label}(u)$. Applying the definition of label to (5) we get:

$l = \texttt{label}(u)$
$= \texttt{label}(\texttt{new}_d\ x \Rightarrow \overline{d = u})$
$= \texttt{new}_\sigma\ x \Rightarrow \overline{\texttt{label-decl}(d = u)}$
$= \texttt{new}_\sigma\ x \Rightarrow \overline{d \texttt{ with } \texttt{effects}(\Gamma \cap \texttt{freevars}(u)) = \texttt{label}(u)}$

Fix some method declaration $d_i = u_i$ in the original unlabeled object u. From (9) we know $\Gamma', x : \{\bar{d} \texttt{ captures } \varepsilon_c\} \vdash d_i = u_i \texttt{ OK}$. The only rule with this judgement is T-ValidImpl. By inversion we obtain:

11. $d_i = \texttt{def}\ m(y : \tau_2) : \tau_3$
12. $\Gamma', x : \{\bar{d} \texttt{ captures } \varepsilon_c\}, y : \tau_2 \vdash e : \tau_3$

Case. T-MethCallInf.
hi

12 Theorem (Soundness)

Theorem. Suppose $\Gamma \vdash u_A : \tau_A \texttt{ with } \varepsilon_A$ and $u_A \longrightarrow u_B \mid \varepsilon$. The following are true:

1. $\Gamma \vdash u_B : \tau_B \texttt{ with } \varepsilon_B$
2. $\tau_B <: \tau_A$
3. $\varepsilon_B \cup \varepsilon = \varepsilon_A$

Proof. Without loss of generality assume u_A is not a value. Let $l_A = \texttt{label}(u_A)$. By Theorem 11 (Refinement) we learn:

4. $\Gamma \vdash l_A : \hat{\tau}_A \texttt{ with } \hat{\varepsilon}_A$
5. $\hat{\tau}_A <: \tau_A$
6. $\hat{\varepsilon}_A \subseteq \varepsilon_A$

Because u_A is not a value, neither is l_A. By Theorem 10 the runtime is invariant under labeling, so $l_A \longrightarrow l_B \mid \varepsilon$, where $l_B = \texttt{label}(u_B)$. Because the typing rules for l-terms are sound, $\Gamma \vdash l_B : \hat{\tau}_B \texttt{ with } \hat{\varepsilon}_B$ where:

7. $\hat{\tau}_B <: \hat{\tau}_A$
8. $\hat{\varepsilon}_B \subseteq \hat{\varepsilon}_A$

It's at this point we invoked Refinement again to relate l_B and u_B. However, the judgement for l_B comes from the soundness theorem. We don't know if this is the same judgement guaranteed by the refinement theorem, so it's not right to equate them.

What we need to know: if $\Gamma \vdash \texttt{label}(u_B) : \hat{\tau}_B \texttt{ with } \hat{\varepsilon}_B$, then for *any* judgement $\Gamma \vdash u_B : \tau_B \texttt{ with } \varepsilon_B$, that $\hat{\tau}_B <: \tau_B$ and $\hat{\varepsilon}_B \subseteq \varepsilon_B$ (in general that's not true though, since you can just type $\hat{\tau}_B$ as $\top$ or some other type which isn't useful).