# 1 Grammar

Fix some typing context $\Gamma$. We denote the set of methods in $\Gamma$ by $M$ and the set of resources by $R$. A resource has the authority to directly perform I/O operations. Elements on those sets are denoted $m$ and $r$ respectively. An effect is a member of the set of pairs $M \times R$. Intuitively we may read the effect $(m, r)$ as meaning 'the effect on resource $r$ when method $m$ is called'. A set of effects is denoted by $\varepsilon$.

By convention we let $\bar{\sigma}$ denote a (non-empty) sequence of $\sigma$-terms. Similarly for $\bar{d}$ and $\bar{\rho}$.

A type is a set of declarations. The empty set is the `Unit` type.

$$
\begin{array}{lll}
e ::= x & & \textit{expressions} \\
\quad \mid \ \texttt{new}\ x \Rightarrow \overline{\sigma = e} & & \\
\quad \mid \ e.m(e) & & \\
\quad \mid \ r & & \\[1em]
\tau ::= \{\bar{\sigma}\} \mid \{\bar{d}\} \mid \{\bar{r}\} & & \textit{types} \\[1em]
\sigma ::= d\ \texttt{with}\ \varepsilon & & \textit{labeled decls.} \\[1em]
d ::= \texttt{def}\ m(x : \tau) : \tau & & \textit{unlabeled decls.} \\[1em]
\gamma ::= \{\bar{d}\ \texttt{captures}\ \varepsilon\} & & \textit{annotated decls.} \\[1em]
\kappa ::= d = e\ \texttt{OK} & & \textit{well formed decls.} \\
\quad \mid \ \sigma = e\ \texttt{OK} & & \\
\quad \mid \ d\ \texttt{captures}\ \epsilon\ \texttt{OK} & &
\end{array}
$$

**Notes:**

- $\sigma$ denotes a declaration with effect labels. $d$ denotes a declaration without effect labels.
- $\gamma$ and $\kappa$ terms are introduced by the calculus and don't appear in the source program.

# 2 Effect Rules

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

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

$$\frac{\Gamma, x : \{\bar{\sigma}\} \vdash \overline{\sigma = e}\ \texttt{OK}}{\Gamma \vdash \texttt{new}\ x \Rightarrow \overline{\sigma = e} : \{\bar{\sigma}\}\ \texttt{with}\ \varnothing}\ (\varepsilon\text{-}\textsc{NewObj})$$

$\boxed{\Gamma \vdash \kappa}$

$$\frac{\Gamma, x : \tau \vdash e : \tau'\ \texttt{with}\ \varepsilon \quad \sigma = \texttt{def}\ m(x : \tau) : \tau'\ \texttt{with}\ \varepsilon}{\Gamma \vdash \sigma = e\ \texttt{OK}}\ (\varepsilon\text{-}\textsc{ValidImpl}_\sigma)$$

$$\frac{\Gamma, x : \tau \vdash e : \tau' \quad \sigma = \texttt{def}\ m(x : \tau) : \tau'}{\Gamma \vdash d = e\ \texttt{OK}}\ (\varepsilon\text{-}\textsc{ValidImpl}_d)$$

$\boxed{\Gamma \vdash e_1.m(e_2) : \tau \texttt{ with } \varepsilon}$

$$\frac{\Gamma \vdash e_1 : \{\bar{r}\} \texttt{ with } \varepsilon_1 \quad \Gamma \vdash e_2 : \tau_2 \texttt{ with } \varepsilon_2}{\Gamma \vdash e_1.m(e_2) : \{\bar{r}\} \texttt{ with } \{\bar{r}, m\} \cup \varepsilon_1 \cup \varepsilon_2} \ (\varepsilon\text{-}\textsc{MethCallResource})$$

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

**Notes:**

- The $\varepsilon$ judgements are to be applied to portions of the program where the methods are explicitly annotated with their effects.
- The rules $\varepsilon$-Var, $\varepsilon$-Resource, and $\varepsilon$-NewObj have in their antecedents an expression typed with no effect. Merely having an object or resource is not an effect; you must do something with it, like a call a method on it, in order for your program to have effects.
- $\varepsilon$-ValidImpl says that the return type and effects of the body of a method must agree with what its signature says.
- According to $\varepsilon$-MethCallResource, we can call any method on a resource. Doing so returns that same resource.

## 3 Capture Rules (Orange)

$\boxed{\Gamma e : \{\bar{d} \texttt{ captures } \varepsilon\}}$

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

$\boxed{\Gamma \vdash e_1.m(e_2) : \tau \texttt{ with } \varepsilon}$

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

$\boxed{\Gamma \vdash d \texttt{ captures } \varepsilon \texttt{ OK}}$

$$\frac{\varepsilon = effects(\Gamma') \quad \Gamma' \subseteq \Gamma \quad \Gamma' \vdash d = e \texttt{ OK}}{\Gamma \vdash d \texttt{ captures } \varepsilon \texttt{ OK}} \ (\textsc{C-UnlabeledDecl})$$

- The capture judgements are to be applied when the program is not explicitly annotated with their effects. These rules perform a conservative effect analysis.
- The rule C-NewObj takes unannotated methods and labels them using the `captures` keyword. Whereas $d$ `with` $\varepsilon$ means that execution of the method defined by $d$ has the effects $\varepsilon$, $d$ `captures` $\varepsilon$ means that $d$ has the authority to perform the effects $\varepsilon$, though it may not actually do so. We can think of `captures` as an upper bound on the effects of a program, and `with` as a tight upper bound.
- C-MethCall performs a conservative effect analysis by concluding the effects of an expression to be those effects which it captures.
- C-UnlabeledDecl is to be applied to declarations with no effect annotations. The intent is to label $d$ with $\varepsilon$, a conservative set of effects in the body of $d$. The $\Gamma'$ in the rule is meant to be $\Gamma$ restricted to the scope of $d$.

### 3.1 Definition of effects function

The $effects$ function returns the set of effects in a particular typing context. It uses `with` annotations to figure this out. Where there are no `with` annotations it does a conservative effect inference.

- $effects(\cdot) = \varnothing$
- $effects(\{\bar{r}\}) = \{(r, m) \mid r \in \bar{r}, m \in M\}$
- $effects(\{\bar{d}\ \texttt{captures}\ \varepsilon\}) = \varepsilon$
- $effects(d\ \texttt{with}\ \varepsilon) = \varepsilon$
- $effects(\{\bar{\sigma}\}) = \bigcup_{\sigma \in \bar{\sigma}}\ effects(\sigma)$

The case for unlabeled effects is given as an inference rule:

$$\frac{\Gamma \vdash d\ \texttt{captures}\ \varepsilon\ \texttt{OK}}{\Gamma \vdash effects(d) = \varepsilon}\ (\textsc{Effects-UnlabeledDecl})$$