Language 42

for more information `L42.is`

Core Language Syntax

$\mathcal{L}$	$::= \mathtt{\{}\ doc\ \mathcal{H}\overline{\mathcal{M}}\ \mathtt{\}}$	library literal
$\mathcal{M}$	$::= C\!:\ doc\ e \mid mh^t \mid mh\ e$	class member
$\mathcal{H}$	$::= m\mathtt{(}\overline{\mathcal{F}}\mathtt{)}\ \mathtt{<:}\ \overline{\pi}\ doc \mid \mathtt{interface<:}\ \overline{\pi}\ doc \mid \mathtt{trait<:}\ \overline{\pi}\ doc$	class header
$\mathcal{F}$	$::= T\ x\ doc \mid \mathtt{var}\ T\ x\ doc$	field
e	$::= \mathcal{L} \mid x \mid \pi \mid \mathtt{void} \mid e.m\mathtt{(}doc\ \overline{e}\mathtt{)} \mid \mathtt{(}doc\ \overline{\mathcal{X}}\mathcal{K}e\mathtt{)} \mid s\ e$ $\mid \mathtt{using}\ \pi\ \mathtt{check}\ m\mathtt{(}doc\ \overline{e}\mathtt{)}\ e$	expression
s	$::= \mathtt{exception} \mid \mathtt{error} \mid \mathtt{return}$	signal
$\mathcal{X}$	$::= T\ x\ doc\mathtt{=}e$	binding def.
$\mathcal{K}$	$::= \mathtt{catch}\ x\ doc\ \overline{\mathcal{O}}$	
$\mathcal{O}$	$::= \mathtt{on}\ s\ \overline{\pi}\ doc\ e$	catch-match
mh^t	$::= \mu\ \mathtt{method}\ doc\ T\ doc'\ m\mathtt{(}\overline{T\ x\ doc}\mathtt{)}\ \mathtt{exception}\ \overline{\pi}\ doc$	typed m. header
mh^s	$::= \mathtt{method}\ doc\ m\mathtt{(}\overline{x}\mathtt{)}$	method selector
mh	$::= mh^t \mid mh^s$	method header
m	$::= x\overline{x} \mid \mathtt{\#}x\overline{x}$	method name
π	$::= \mathtt{Outer}^n\overline{::C} \mid \mathtt{Any} \mid \mathtt{Void} \mid \mathtt{Library}$	path
T	$::= \mu\ \pi \mid \mu\ \pi\mathtt{\hat{}} \mid \pi.m$	type annotation
μ	$::= \mathtt{immutable} \mid \mathtt{shared} \mid \mathtt{readable} \mid \mathtt{lent} \mid \mathtt{capsule} \mid \mathtt{type}$	modifiers

Notations

We represent with $\emptyset$ both the set of empty characters and empty lists and maps. x, y and z metavariables denote lower case identifiers, while C denotes upper case ones. We use $\overline{\ }$ to denote optionality; in particular, $\overline{T}, \overline{\mathtt{var}}, \overline{x}$ denote metavariables that can be either the empty string $\emptyset$ or in the form of the corresponding terms. In the same way, we use $\overline{\ }$ to denote multiplicity; in particular, $\overline{\mathcal{O}}$, $\overline{T\ x}$, $\overline{x\!:\!e}$ denote metavariables that can be a sequence of any number of the corresponding terms. Method names are wrote without $\overline{x}$ if followed by syntactic terms containing such parameter names. For example $e.x\mathtt{(}x_1\!:\!e_1...x_n\!:\!e_n\mathtt{)}$ instead of $e.xx_1...x_n\ \mathtt{(}x_1\!:\!e_1...x_n\!:\!e_n\mathtt{)}$. A $\mathcal{K}$ with empty $\overline{\mathcal{O}}$ is represented as $\emptyset$, and this will happens also in the rules.

Syntax well formedness

All parameter names declared within a given method header must be unique. All binding names declared within a given method body (forms $\mathcal{X}$ and $\mathtt{catch}\ x$) and its header must be unique. All methods in a given class must be uniquely identified by their name m and the sequence of their parameter names $\overline{x}$. All binding names declared within class initialization expressions of a given class must be unique. All fields names in a given header must be unique. To avoid syntactic ambiguities, expressions in the body of an $\mathtt{on}$, $\mathtt{using}$ are of form $\mathtt{(\_)}$

Compiled Language Syntax, compilation context and $\mathcal{E}^\star$

$\mathcal{L}^c ::= \mathtt{\{}\mathcal{H}\overline{\mathcal{M}^c}\mathtt{\}}$

$\mathcal{M}^c ::= C\!:\!\mathcal{L}^c \mid mh\ e^c \mid mh^t$

$e^c ::= \mathcal{L}^c \mid x \mid \pi \mid \mathtt{void} \mid e^c.m\mathtt{(}\overline{e^c}\mathtt{)}$
$\mid \mathtt{(}\overline{\mathcal{X}^c}\mathcal{K}^c e^c\mathtt{)} \mid s\ e^c \mid \mathtt{using}\ \pi\ \mathtt{check}\ .m\mathtt{(}\overline{e^c}\mathtt{)}\ e^c$

$\mathcal{X}^c ::= T\ x\mathtt{=}e^c$

$\mathcal{K}^c ::= \mathtt{catch}\ x\ \overline{\mathcal{O}^c}$

$\mathcal{O}^c ::= \mathtt{on}\ s\ \pi\ e^c$

$\mathcal{E}^c ::= \square \mid \mathcal{E}^c.m\mathtt{(}\overline{e}\mathtt{)} \mid e_0{}^c.m\mathtt{(}\overline{e^c}_1\mathcal{E}^c\overline{e}_2\mathtt{)} \mid s\ \mathcal{E}^c$
$\mid \mathtt{(}\overline{\mathcal{X}^c}_1\ T\ x\mathtt{=}\mathcal{E}^c\ \overline{\mathcal{X}}_2\mathcal{K}\ e\mathtt{)} \mid \mathtt{(}\overline{\mathcal{X}^c}\mathtt{catch}\ x\ \overline{\mathcal{O}^c}_1\mathtt{on}\ s\ \pi\ \mathcal{E}^c\ \overline{\mathcal{O}}_2 e\mathtt{)} \mid \mathtt{(}\overline{\mathcal{X}^c}\mathcal{K}^c\mathcal{E}^c\mathtt{)}$
$\mathtt{using}\ \pi\ \mathtt{check}\ .m\mathtt{(}\overline{e^c}_1\mathcal{E}^c\overline{e}_2\mathtt{)}\ e \mid \mathtt{using}\ \pi\ \mathtt{check}\ .m\mathtt{(}\overline{e^c}\mathtt{)}\ \mathcal{E}^c$

$\mathcal{E}^\star ::= \square \mid \mathcal{E}^\star.m\mathtt{(}\overline{e}\mathtt{)} \mid e_0.m\mathtt{(}\overline{e}_1\mathcal{E}^\star\overline{e}_2\mathtt{)} \mid s\ \mathcal{E}^\star$
$\mid \mathtt{(}\overline{\mathcal{X}}_1\ T\ x\mathtt{=}\mathcal{E}^\star\ \overline{\mathcal{X}}_2\mathcal{K}\ e\mathtt{)} \mid \mathtt{(}\overline{\mathcal{X}}\mathtt{catch}\ x\ \overline{\mathcal{O}}_1\mathtt{on}\ \_\ \mathcal{E}^\star\ \overline{\mathcal{O}}_2 e\mathtt{)} \mid \mathtt{(}\overline{\mathcal{X}}\mathcal{K}\mathcal{E}^\star\mathtt{)}$
$\mathtt{using}\ \pi\ \mathtt{check}\ .m\mathtt{(}\overline{e}_0\mathcal{E}^\star\overline{e}_1\mathtt{)}\ e \mid \mathtt{using}\ \pi\ \mathtt{check}\ .m\mathtt{(}\overline{v}\mathtt{)}\ \mathcal{E}^\star$

Values Syntax and evaluation Ctx

$v^p ::= \pi.m\mathtt{(}\overline{v^p}\mathtt{)}$ <with is-constr$(p(\pi), m)$> $\mid x \mid \mathtt{(}\overline{\mathcal{X}v^p}\ v^p\mathtt{)} \mid \mathtt{void} \mid \pi \mid \mathcal{L}^c$

$\mathcal{X}v^p ::= T\ x\mathtt{=}v^p$

<we write just v and $\mathcal{X}v$ when p is clear from the context>

$\mathcal{E}^p ::= \square.m\mathtt{(}\overline{e}\mathtt{)} \mid v_0.m\mathtt{(}\overline{v}\ \square\ \overline{e}\mathtt{)} \mid s\ \square \mid \mathtt{using}\ \pi\ \mathtt{check}\ .m\mathtt{(}\overline{v}\square\overline{e}\mathtt{)}\ e$

Auxiliary Syntax

$\sigma ::= \mathcal{X}v_0, ..., \mathcal{X}v_n$	run time env	$\Gamma ::= x_1 \mapsto T_1, ..., x_n \mapsto T_n$	
$p ::= \mathcal{L}^t{}_0, ..., \mathcal{L}^t{}_n$	program type	$\lambda ::= \ominus \mid \circledcirc \mid \oplus \mid \circledast$	stage
$\mathcal{L}^t ::= \mathtt{\{}\mathcal{H}\ \overline{\mathcal{M}^t}\mathtt{\}}^\lambda \mid \emptyset \mid \updownarrow$	class type	$\Delta ::= \Gamma; p^t; \overline{\mu}; \overline{T}; \overline{\pi}$	typing env
$\mathcal{M}^t ::= mh^t e \mid mh^t\mathtt{abstract} \mid mh^t\mathtt{constr} \mid mh^t\mathtt{field} \mid C\!:\!\mathcal{L}^t$	member type		

Definition: $\pi_0[\text{from } \pi_1] = \pi_2$

$\mathtt{Outer}^n\overline{::C}[\text{from } \mathtt{Outer}^m::C_1...::C_k] = \mathtt{Outer}^m::C_1...::C_{k-n}\overline{::C}$ if $n \le k$

$\mathtt{Outer}^n\overline{::C}[\text{from } \mathtt{Outer}^m::C_1...::C_k] = \mathtt{Outer}^{m+n-k}\overline{::C}$ if $n > k$

$\mathtt{Any}[\text{from }\_] = \mathtt{Any}$ $\quad \mathtt{Library}[\text{from }\_] = \mathtt{Library}$ $\quad \mathtt{Void}[\text{from }\_] = \mathtt{Void}$

Definition: $e_0[\text{from } \pi] = e_1$, $e_0[\text{from } \pi]_n = e_1$

$e[\text{from } \pi] = e[\text{from } \pi]_0$

$\mathtt{\{}\mathcal{H}\overline{\mathcal{M}}\mathtt{\}}[\text{from } \pi]_j = \mathtt{\{}\mathcal{H}[\text{from } \pi]_{j+1}\overline{\mathcal{M}}[\text{from } \pi]_{j+1}\mathtt{\}}$

$\mathtt{Outer}^{j+n}\overline{::C}_0[\text{from } \pi]_j = \mathtt{Outer}^{j+k}\overline{::C}_1$ with $\mathtt{Outer}^n\overline{::C}_0[\text{from } \pi] = \mathtt{Outer}^k\overline{::C}_1$

$\mathtt{Outer}^n\overline{::C}[\text{from } \pi]_j = \mathtt{Outer}^n\overline{::C}$ $\quad$ with $n < j$

$doc[\text{from } \pi]_j$ replaces all substrings of the form $\mathtt{@}\pi_0$ and $\mathtt{@\,(}e\mathtt{)}$ with $\mathtt{@}\pi_0[\text{from } \pi]_j$ and $\mathtt{@\,(}e_0[\text{from } \pi]_j\mathtt{)}$

All cases for other expressions/terms propagate to submembers

Definition: $\Gamma(x)$, $\quad \overline{\mathcal{X}}(x)$, $\quad p(\pi)$, $\quad \mathcal{L}(\pi)$

$\Gamma(x): \ (\_, x \mapsto T, \_)(x) = T$

$\overline{\mathcal{X}}(x): \ (\overline{\mathcal{X}}_1\ \overline{T}\ x\mathtt{=}e\ \overline{\mathcal{X}}_2)(x) = e$

$p(\pi): \ (\mathcal{L}_0...\mathcal{L}_n)(\mathtt{Outer}^i\overline{::C}) = \mathcal{L}_i(\overline{::C})$

$\mathcal{L}(\overline{::C}): \ \mathtt{\{}\mathcal{H}\ \overline{\mathcal{M}}_1\ C\!:\!\mathcal{L}\ \overline{\mathcal{M}}_2\mathtt{\}}(::C\overline{::C}) = \mathcal{L}(\overline{::C})$ $\quad$ and $\mathcal{L}() = \mathcal{L}$

Definition: $p^t(\pi)$

$(\mathcal{L}^t{}_0...\mathcal{L}^t{}_n)(\mathtt{Outer}^i\overline{::C}) = \mathcal{L}^t{}_i(\overline{::C})$ where $\mathcal{L}^t() = \mathcal{L}^t, \emptyset(\_) = \emptyset$ and

$\mathtt{\{}\mathcal{H}\ \overline{\mathcal{M}^t}_1 \updownarrow C_0 \mapsto \mathcal{L}^t\ \overline{\mathcal{M}^t}_2\mathtt{\}}^\lambda(::C_0\overline{::C}) = \mathcal{L}^t(\overline{::C})$

Definition: Type well formedness

$\overline{\mathcal{M}^t}$ and Γ are maps, thus order is irrelevant and

$\overline{\mathcal{M}^t}(m) = \mathcal{M}^t = \mu\ \mathtt{method}\ T\ m\mathtt{(}\ T_1\ x_1...\ T_n\ x_n\mathtt{)}\ \mathtt{exception}\ \overline{\pi}\_$ iff $\mathcal{M}^t \in \overline{\mathcal{M}^t}$

$\overline{\mathcal{M}^t}(C) = C \mapsto \mathcal{L}^t$ iff $C \mapsto \mathcal{L}^t \in \overline{\mathcal{M}^t}$

Definition: dom(_)

the above function notations _(_) each implicitly defines a domain dom(_) as the set of all inputs for which the function is defined

Definition: is-constr$(\mathcal{L}^t, m)$, is-set$(\mathcal{L}^t, m)$, is-get$(\mathcal{L}^t, m)$

is-constr$(\mathtt{\{}_\ \mu\ \mathtt{method}\ _\ m\mathtt{(_)}\ \mathtt{exception}\ \emptyset\mathtt{constr}\ _\mathtt{\}}^\lambda, m)$

is-get$(\mathtt{\{}_\ \mu\ \mathtt{method}\ _\ m\mathtt{()}\ \mathtt{exception}\ \emptyset\mathtt{field}\ _\mathtt{\}}^\lambda, m)$

is-set$(\mathtt{\{}_\ \mu\ \mathtt{method}\ _\ m\mathtt{(}T\ x\mathtt{)}\ \mathtt{exception}\ \emptyset\mathtt{field}\ _\mathtt{\}}^\lambda, m)$

Definition: c-f-type$(m(\mathcal{F}_1...\mathcal{F}_n)) = \overline{mh^t}$

(1) $\mathtt{type\ method\ lent\ Outer}_0\ m\mathtt{(}\ \mu_1\ \pi_1\mathtt{\hat{}}\ x_1...\ \mu_n\ \pi_n\mathtt{\hat{}}\ x_n\mathtt{)}\ \mathtt{exception}\ \emptyset\mathtt{constr}$
$\in$ c-f-type$(m\mathtt{(}\overline{\mathtt{var}}_1\ \mu_1\ \pi_1\ x_1...\overline{\mathtt{var}}_n\ \mu_n\ \pi_n\ x_n\mathtt{)})$
iff $\{\mathtt{lent}, \mathtt{readable}\} \cap \{\mu_1, ..., \mu_n\} \ne \emptyset$

(2) $\mathtt{type\ method\ shared\ Outer}_0\ m\mathtt{(}\ \mu_1\ \pi_1\mathtt{\hat{}}\ x_1...\ \mu_n\ \pi_n\mathtt{\hat{}}\ x_n\mathtt{)}\ \mathtt{exception}\ \emptyset\mathtt{constr}$
$\in$ c-f-type$(m\mathtt{(}\overline{\mathtt{var}}_1\ \mu_1\ \pi_1\ x_1...\overline{\mathtt{var}}_n\ \mu_n\ \pi_n\ x_n\mathtt{)})$
iff $\{\mathtt{lent}, \mathtt{readable}\} \cap \{\mu_1, ..., \mu_n\} = \emptyset$

(4) $\mathtt{shared\ method\ immutable\ Void}\ x\ \mathtt{that(}\ \mu\ \pi\ \mathtt{that)}\ \mathtt{exception}\ \emptyset\mathtt{field}$
$\in$ c-f-type$(m\mathtt{(}\overline{\mathcal{F}}_1\ \mathtt{var}\ \mu\pi\ x\ \overline{\mathcal{F}}_2\mathtt{)})$

(5) $\mathtt{shared\ method}\ \mu'\ \pi\ \mathtt{\#}x\mathtt{()}\ \mathtt{exception}\ \emptyset\mathtt{field} \in$ c-f-type$(m\mathtt{(}\overline{\mathcal{F}}_1\ \mu\pi\ x\ \overline{\mathcal{F}}_2\mathtt{)})$
with $\mu' = \mathtt{lent}$ iff $\mu = \mathtt{capsule}$, $\mu' = \mu$ otherwise

(6) $\mathtt{readable\ method\ readable}\ \pi\ x\mathtt{()}\ \mathtt{exception}\ \emptyset\mathtt{field} \in$ c-f-type$(m\mathtt{(}\overline{\mathcal{F}}_1\ \mu\pi\ x\ \overline{\mathcal{F}}_2\mathtt{)})$
iff $\mu \in \{\mathtt{capsule}, \mathtt{shared}, \mathtt{lent}\}$, otherwise
$\mathtt{readable\ method}\ \mu\ \pi\ x\mathtt{()}\ \mathtt{exception}\ \emptyset\mathtt{field} \in$ c-f-type$(m\mathtt{(}\overline{\mathcal{F}}_1\ \mu\pi\ x\ \overline{\mathcal{F}}_2\mathtt{)})$

Definition: _ inside _

$\mathtt{\hat{}}$ inside Γ holds iff $\Gamma(\_) = \_\mathtt{\hat{}}$

μ inside $T_1...T_n$ holds iff $\mu\_ \in T_1...T_n$ or $\mu\_\mathtt{\hat{}} \in T_1...T_n$

μ inside Γ holds iff $\Gamma(\_) = \mu\_$ or $\Gamma(\_) = \mu\_\mathtt{\hat{}}$

μ inside $(\Gamma; \_; \_; \_)$ holds iff μ inside Γ

x inside $\mathtt{\{}\mathcal{H}\_C\!:, e\_\mathtt{\}}, p$ iff $\overline{T}\ x\mathtt{=}\_$ inside e

$\mathcal{X}$ inside $(\mathcal{X}v_1, ..., \mathcal{X}v_n)$ iff $\mathcal{X} \in \mathcal{X}v_i$ with $i \in 1..n$

$\mathcal{X}$ inside e iff $e = \mathcal{E}^\star[\mathtt{(}\overline{\mathcal{X}}_0\mathcal{X}\overline{\mathcal{X}}_1\mathcal{K}e_0\mathtt{)}]$

e_0 inside e_1 holds iff $e_1 = \mathcal{E}^\star[e_0]$
or $e_1 = \mathcal{E}^\star[\mathtt{(}\overline{\mathcal{X}}\mathtt{catch}\ y\overline{\mathcal{O}}_1\mathtt{on}\ \pi_1\ \pi_2\ x\ e_2\overline{\mathcal{O}}_2 e_3\mathtt{)}]$ and $e_0 = x$

$\Delta \vdash e : T$

(LIB-T)
$$\frac{}{\Delta \vdash \mathcal{L} : \texttt{immutable Library}}$$
with
$p^{t\Delta} \vdash \mathcal{L} : \{\_\}^{\lambda_1}$
$\lambda_1 = \ominus$ iff $\lambda^\Delta \in \{\ominus, \circledcirc\}$
$\lambda_1 = \oplus$ iff $\lambda^\Delta \in \{\oplus, \circledast\}$

(PATH-STRICT)
$$\frac{}{\Delta \vdash \pi : \texttt{type}\ \pi}$$
with
$p^{t\Delta}(\pi) = \{\texttt{.}m(\_)\ \texttt{<:}\overline{\pi}\ \_\}^{\circledast}$
$\lambda^\Delta \neq \ominus$

(PATH-RELAX)
$$\frac{}{\Delta \vdash \pi : \texttt{type}\ \pi}$$
with
$p^{t\Delta}(\pi) = \{\mathcal{H}\ \_\}^{\lambda}$
$\lambda^\Delta \in \{\oplus, \ominus\}$
if $\lambda = \ominus$ then $\lambda^\Delta = \ominus$
$\mathcal{H}$ not of form $\texttt{interface<:}\_$

(PATH-ANY)
$$\frac{}{\Delta \vdash \pi : \texttt{type Any}}$$
with
$p^{t\Delta}(\pi) = \{\_\}^{\lambda}$
if $\lambda = \ominus$ then $\lambda^\Delta = \ominus$

(VOID-T)
$$\frac{}{\Delta \vdash \texttt{void} : \texttt{capsule Void}}$$

(BINDING-T)
$$\frac{}{\Delta \vdash x : T^{\setminus\overline{\mu}^\Delta\setminus}}$$
with
either $\Gamma^\Delta(x) = \overline{\texttt{var}}\ T$
or $x \notin \text{dom}(\Gamma^\Delta)$ and $\lambda^\Delta = \ominus$

(SATISFY-T)
$$\frac{\Delta[\overline{\mu} := \emptyset] \vdash e : T}{\Delta \vdash e : T}$$
with
$\forall \mu \in \overline{\mu}^\Delta$: not μ inside $T, \overline{T}^\Delta$

(LOOP-T)
$$\frac{\Delta \vdash e : \texttt{immutable Void}}{\Gamma \circ \Delta \vdash \texttt{loop}\ e : \texttt{immutable Void}}$$
with
not $\texttt{capsule}$ inside Δ

(DEC-FIRST-T)
$$\frac{\Delta \vdash (\overline{\mathcal{X}}_0 \mathcal{K} (\overline{\mathcal{X}}_1 \mathcal{K} e_0)) : T}{\Delta \vdash (\overline{\mathcal{X}}_0\ \overline{\mathcal{X}}_1 \mathcal{K} e_0) : T}$$

(DEC-LENT-T)
$$\frac{\Gamma_1 \circ \Delta \vdash e_1 : \texttt{lent}\pi \qquad (\Gamma_0 \circ \Delta)^{\setminus \text{shared}\setminus} \vdash (\texttt{shared}\pi\ x\texttt{=error void}\ \mathcal{K} e_0) : T}{\Gamma_0 \circ \Gamma_1 \circ \Delta \vdash (T_1\ x\texttt{=}e_1\ \mathcal{K}\ e_0) : T^{\setminus \text{shared}\setminus}}$$
with
not $\texttt{capsule}$ inside Δ

(DEC-T)
$$\frac{\begin{array}{c}\Gamma_0 \circ \Delta \vdash \mathcal{K} : T \mid \overline{T};\overline{\pi} \\ (\Gamma_1 \circ \Delta \circ \overline{T};\overline{\pi})^{\setminus\mathcal{K}\setminus} \vdash \overline{\mathcal{X}} : \Gamma'_0 \mid \Gamma' \\ \Gamma'_0 \circ \Gamma_0 \circ \Delta \vdash e_0 : T \\ \Gamma' \circ \Gamma_0 \circ \Delta \vdash e_0 : \_\end{array}}{\Gamma \circ \Gamma_0 \circ \Gamma_1 \circ \Delta \vdash (\overline{\mathcal{X}}\ \mathcal{K}\ e_0) : T}$$
with
not $\texttt{capsule}$ inside Δ
if $\hat{}$ inside $\Gamma_0, \Gamma_1, \Gamma^\Delta$ then $\hat{}$ inside T

(V-ASS-T)
$$\frac{\Delta \vdash (T\ x_0\texttt{=}e\ x_0) : T}{\Delta \vdash x\texttt{:=}e : \texttt{immutable Void}}$$
with
$\Gamma^\Delta(x) = \texttt{var}\ T$

(METH-INVK-T)
$$\frac{\begin{array}{c}\Delta[\overline{\mu} := \emptyset] \vdash e_0 : \_\pi \\ \Delta \vdash \text{invoke}_\pi(mh^t, e_0, ..., e_n, \texttt{error void}) : T\end{array}}{\Delta \vdash e_0.m(x_1\texttt{:}e_1 ... x_n\texttt{:}e_n) : T}$$
with
$p^{t\Delta}(\pi) = \{\mathcal{H}\overline{\mathcal{M}^t}\}$
$mh^t = \overline{\mathcal{M}^t}(\texttt{method .}m(x_1 ... x_n))$[from π]

(METH-UNKNOWN-T)
$$\frac{\begin{array}{c}\Delta[\overline{\mu} := \emptyset] \vdash e_0 : \_\pi \\ \Delta \vdash e : T\end{array}}{\Delta \vdash e_0.m(x_1\texttt{:}e_1 ... x_n\texttt{:}e_n) : T}$$
with
$p^{t\Delta}(\pi) = \emptyset$ and $\lambda^\Delta = \ominus$
$e = (y_0\texttt{=}e_0\ y_1\texttt{=}e_1 ... y_n\texttt{=}e_n\ \texttt{error void})$

(SIGNAL-T)
$$\frac{\Delta \vdash (\mu\ \pi\ x\texttt{=}e\ x) : \mu\ \pi}{\Delta \vdash s\ e : T}$$
with
if $s = \texttt{return}$ then $\mu\ \pi = \overline{T}^\Delta$
if $s \neq \texttt{return}$ then $\mu = \texttt{immutable}$
if $s = \texttt{exception}$ then $\pi \in \overline{\pi}^\Delta$
if $s = \texttt{error}$ or $\mu = \texttt{type}$ then $\pi = \texttt{Any}$

(USING-T)
$$\frac{\Delta \vdash (T_1\ y_1\texttt{=}e_1 ... T_n\ y_n\texttt{=}e_n\ (T_0\ y_0\texttt{=}e_0\ \texttt{catch}\ y\ \texttt{on error Any}\ (\texttt{error}\ y)\ y_0)) : T_0}{\Delta \vdash x\ \texttt{using}\ \pi\ \texttt{check .}m(x_1\texttt{:}e_1 ... x_n\texttt{:}e_n)\ e_0 : T_0}$$
with
$plg; T_0 ... T_n = \text{plugin}(p^{t\Delta}, \pi, \texttt{.}m(x_1 ... x_n))$
$y, y_0 ... y_n$ fresh

$\Delta \vdash \overline{\mathcal{X}} : \Gamma_0 \mid \Gamma, \quad \Delta \vdash \mathcal{K} : T \mid \overline{T};\overline{\pi}, \quad \Delta \vdash x\,\mathcal{O} : T \mid \overline{T};\overline{\pi}$

(CATCH-DISPATCH)
$$\frac{\forall i \in 1..n : \quad \Delta \vdash x\mathcal{O}_i : T \mid \overline{T}_i;\overline{\pi}_i}{\Delta \vdash \texttt{catch}\ x\ \mathcal{O}_1 ... \mathcal{O}_n : T \mid \overline{T};\overline{\pi}_1 \cup ... \cup \overline{\pi}_n}$$
with
$\forall i \in 1..n : \overline{T}_i \in \{\emptyset, \mu\ \_\}$
$\overline{T} = T$ iff $\overline{T}_1, ..., \overline{T}_n = \_, T, \emptyset, ..., \emptyset$
$\overline{T} = \emptyset$ otherwise

(ON)
$$\frac{x \mapsto \mu\ \pi \circ \Delta \vdash e : T}{\Delta \vdash x\ \texttt{on}\ s\ \pi\ e : T \mid \overline{T};\overline{\pi}}$$
with
either $s = \texttt{return}$ or $\mu = \texttt{immutable}$
either $s = \texttt{return}$ and $\overline{T};\overline{\pi} = \mu\ \pi;\emptyset$
or $s = \texttt{error}$ and $\overline{T};\overline{\pi} = \emptyset;\emptyset$
or $s = \texttt{exception}$ and $\overline{T};\overline{\pi} = \emptyset;\pi$

(BINDING-DEC)
$$\frac{\begin{array}{c}\forall i \in 1..n : \Gamma \circ \Gamma_i \circ \Delta \vdash e_i : T_i \\ \forall i \in 1..n : \text{toPh}(\Gamma) \circ \Gamma_i \circ \Delta_i \vdash e_i : \_\end{array}}{\Gamma_1 \circ ... \circ \Gamma_n \circ \Delta \vdash \overline{\mathcal{X}} : \Gamma \mid \Gamma'}$$
with
$\overline{\mathcal{X}} = \overline{\texttt{var}}_1\ T_1\ x_1\texttt{=}e_1 ... \overline{\texttt{var}}_n\ T_n\ x_n\texttt{=}e_n$
not $\texttt{capsule}$ inside Δ
$\Gamma = \{x_i \mapsto \overline{\texttt{var}}_i\ T_i \mid i \in 1..n, T_i = T'_i\}$
$\forall i \in 1..n : \Delta_i = \Delta[\overline{\mu} := \overline{\mu}^\Delta \cup \text{cost-of}_\Delta(T_i \le T'_i)]$
$\Gamma' = \Gamma$ iff $\forall i \in 1..n : \text{toPh}(\Gamma) \circ \text{obj}(\Gamma_i \circ \Delta_i) \vdash e_i : \_$
and not $\hat{}$ inside Γ otherwise $\Gamma' = \text{toPh}(\Gamma)$

(ON-EXC-ANY)
$$\frac{\begin{array}{c}x \mapsto \texttt{immutable Any} \circ \Delta \circ \emptyset; \texttt{Any} \vdash e : T \\ \forall \pi \in \overline{\pi} : \quad x \mapsto \texttt{immutable}\ \pi \circ \Delta \vdash e : T\end{array}}{\Delta \vdash x\ \texttt{on exception Any}\ e : T \mid \emptyset;\emptyset}$$

(ON-RET-ANY-1)
$$\frac{\begin{array}{c}x \mapsto \texttt{readable Any} \circ \Delta \circ \texttt{readable Any}; \emptyset \vdash e : T \\ x \mapsto \texttt{type Any} \circ \Delta \circ \texttt{type Any}; \emptyset \vdash e : T\end{array}}{\Delta \vdash x\ \texttt{on return Any}\ e : T \mid \emptyset;\emptyset}$$

(ON-RET-ANY-2)
$$\frac{\begin{array}{c}x \mapsto \texttt{readable Any} \circ \Delta \vdash e : \texttt{readable Any} \\ x \mapsto \texttt{type Any} \circ \Delta \vdash e : \texttt{type Any}\end{array}}{\Delta \vdash x\ \texttt{on return Any}\ e : T \mid T;\emptyset}$$

$p^t \vdash \mathcal{L} : \mathcal{L}^t, \quad p^t \vdash \mathcal{H}_0\overline{\mathcal{M}} : \mathcal{H}_1\overline{mh^t}, \quad p^t \vdash \mathcal{M} : \mathcal{M}^t$

(LIBRARY-T)
$$\frac{\begin{array}{c}\mathcal{L}^t, p^t \vdash \mathcal{H}_0\overline{\mathcal{M}}_0 : \mathcal{H}_1\overline{\mathcal{M}^t} \\ \forall i \in 1..n : \mathcal{L}^t, p^t \vdash \mathcal{M}_i : \mathcal{M}^t_i\end{array}}{p^t \vdash \{doc\ \mathcal{H}_0\overline{\mathcal{M}}\} : \mathcal{L}^t}$$
with
$\overline{\mathcal{M}}_0; \mathcal{M}_1 ... \mathcal{M}_n$ are impl;decl of $\overline{\mathcal{M}}$
$\mathcal{L}^t = \{doc\ \mathcal{H}_1\overline{\mathcal{M}^t}\ \mathcal{M}^t_1 ... \mathcal{M}^t_n\}^{\lambda_0}$

(NESTED-EMPTY-T)
$$\frac{\emptyset; \mathcal{L}^t_1, p^t; \ominus; \emptyset; \emptyset; \emptyset \vdash e : \texttt{immutable Library}}{\mathcal{L}^t_0, p^t \vdash C\texttt{:}e : C\texttt{:}\emptyset}$$
with
$\mathcal{L}^t_0 = \{\mathcal{H}\ \overline{\mathcal{M}^t}\ C\texttt{:}\emptyset\}^\lambda$
$\mathcal{L}^t_1 = \{\mathcal{H}\ \overline{\mathcal{M}^t}\ C\texttt{:}\updownarrow\}^\lambda$
$\lambda \in \{\ominus, \circledcirc\}$

(NESTED-T)
$$\frac{\{\mathcal{H}\overline{\mathcal{M}^t}\}^\lambda, \mathcal{L}^t_1, p^t \vdash \mathcal{L} : \{\mathcal{H}\overline{\mathcal{M}^t}\}^\lambda}{\mathcal{L}^t_0, p^t \vdash C\texttt{:}\mathcal{L} : C\texttt{:}\{\mathcal{H}\overline{\mathcal{M}^t}\}^\lambda}$$
with
$\mathcal{L}^t_0 = \{\mathcal{H}_1\ \overline{\mathcal{M}^t}\ C\texttt{:}\{\mathcal{H}\overline{\mathcal{M}^t}\}^\lambda\}^{\lambda_0}$
$\mathcal{L}^t_1 = \{\mathcal{H}_1\ \overline{\mathcal{M}^t}\ C\texttt{:}\updownarrow\}^{\lambda_0}$
$\lambda_0 \neq \ominus$
if $\lambda_0 \neq \circledcirc$ then $\lambda_0 = \lambda$

(METH-T)
$$\frac{}{p^t \vdash mh^t : mh^t\texttt{abstract}}$$

(METH-T-DEF)
$$\frac{\emptyset; \{\mathcal{H}\overline{\mathcal{M}^t}\}^\lambda, p^t; \emptyset; \emptyset; \overline{\pi} \vdash e_0 : T}{\{\mathcal{H}\overline{\mathcal{M}^t}\}^\lambda, p^t \vdash mh\texttt{:}e : \overline{\mathcal{M}^t}(mh)\ e}$$
with
$e_0 = \text{invoke}_{\texttt{Outer}^0}(\overline{\mathcal{M}^t}(mh), e)$
$\overline{\mathcal{M}^t}(mh) = \_\texttt{method}\ T\ m(\_)\ \texttt{exception}\ \overline{\pi}$

(INTERFACE-T)
$$\frac{}{p^t \vdash \texttt{interface<:}\overline{\pi} : \texttt{interface<:}\text{methodTypes}_{p^t}(\overline{\pi})}$$
with
$\forall \pi \in \overline{\pi} : p^t(\pi)$ is of form $\{\texttt{interface<:}\_\ \_\}$
$\text{singledecl}_{p^t}(\overline{\pi})$

(TRAIT-T)
$$\frac{\begin{array}{c}p^t \vdash \texttt{interface<:}\overline{\pi} : \texttt{interface<:}\overline{\pi}\ \overline{mh^t} \\ \forall i \in 1..n : p^t; \lambda \vdash \mathcal{M}_i : mh^t_i\end{array}}{p^t \vdash \texttt{trait<:}\overline{\pi}\ \mathcal{M}_1 ... \mathcal{M}_n : \texttt{trait<:}\overline{\pi}\ \overline{mh^t}}$$
with
$mh^t_1 ... mh^t_n \subseteq \overline{mh^t}$
$\lambda^{p^t} \neq \circledast$

(CONCRETE-T)
$$\frac{\begin{array}{c}p^t \vdash \texttt{interface<:}\overline{\pi} : \texttt{interface<:}\overline{\pi}\ \overline{mh^t}_0 \\ \forall i \in 1..n : p^t \vdash \mathcal{M}_i : mh^t_i\end{array}}{p^t \vdash m(\mathcal{F})\ \texttt{<:}\overline{\pi}\ \mathcal{M}_1 ... \mathcal{M}_n : \texttt{<:}\overline{\pi}\ \overline{mh^t}_0 \cup \overline{mh^t}_1}$$
with
$\overline{mh^t}_1 = \text{c-f-type}(m(\mathcal{F}))$
$mh^t_1 ... mh^t_n \subseteq \overline{mh^t}_0$
if $\lambda^{p^t} = \circledast$ then $mh^t_1 ... mh^t_n = \overline{mh^t}_0$

Definition: Δ notations

$\Gamma^{\Delta}: \Gamma^{(\Gamma;p^t;\overline{\mu};\overline{T};\overline{\pi})} = \Gamma$ $\qquad \overline{\mu}^{\Delta}: \overline{\mu}^{(\Gamma;p^t;\overline{\mu};\overline{T};\overline{\pi})} = \overline{\mu}$

$\overline{T}^{\Delta}: \overline{T}^{(\Gamma;p^t;\overline{\mu};\overline{T};\overline{\pi})} = \overline{T}$ $\qquad \overline{\pi}^{\Delta}: \overline{\pi}^{(\Gamma;p^t;\overline{\mu};\overline{T};\overline{\pi})} = \overline{\pi}$

$\lambda^{\Delta}: \lambda^{\Delta} = \lambda^{p^{t\Delta}}$ $\qquad p^{t\Delta}: p^{t(\Gamma;p^t;\overline{\mu};\overline{T};\overline{\pi})} = p^t$

$\lambda^{p^t}: \lambda^{\{\mathcal{H}\overline{\mathcal{M}^t}\}^{\lambda}, p^t} = \lambda$

$\Delta[\overline{\mu} := \overline{\mu}_1]: (\Gamma; p^t; \overline{\mu}_0; \overline{T}; \overline{\pi})[\overline{\mu} := \overline{\mu}_1] = \Gamma; p^t; \overline{\mu}_1; \overline{T}; \overline{\pi}$

$\Gamma \circ \Delta: \Gamma_0 \circ (\Gamma_1; p^t; \overline{\mu}; \overline{T}; \overline{\pi}) = (\Gamma_0, \Gamma_1; p^t; \overline{\mu}; \overline{T}; \overline{\pi})$

$\Delta \circ \overline{T}; \overline{\pi}: (\Gamma; p^t; \overline{\mu}; \overline{T}; \overline{\pi}_1) \circ T; \overline{\pi}_2 = \Gamma; p^t; \overline{\mu}; \overline{T}; \overline{\pi}_1, \overline{\pi}_2$

$(\Gamma; p^t; \overline{\mu}; \overline{T}; \overline{\pi}_1) \circ \emptyset; \overline{\pi}_2 = \Gamma; p^t; \overline{\mu}; \overline{T}; \overline{\pi}_1, \overline{\pi}_2$

Definition: extract variable x in binding z: $\sigma[\texttt{+=} z := x]$

(a) $(\mathcal{X}v_1, ..., \mathcal{X}v_n)[\texttt{+=} z := x] = \mathcal{X}v_1[\texttt{+=} z := x], ..., \mathcal{X}v_n[\texttt{+=} z := x]$

(b) $(\mathcal{X}v_1\ \texttt{var}\ \overline{T}\ x\texttt{=}v\ \mathcal{X}v_2)[\texttt{+=} z := x] = \mathcal{X}v_1\ (\texttt{var}\ \overline{T}\ x\texttt{=}v)[\texttt{+=} z := x]\ \mathcal{X}v_2$

(c) $\mathcal{X}v[\texttt{+=} z := x] = \mathcal{X}v$ with $x \notin \text{dom}(\mathcal{X}v)$

(d) $\texttt{var}\ \overline{T}\ x\texttt{=}v[\texttt{+=} z := x] = \overline{T}\ z\texttt{=}v\ \texttt{var}\ \overline{T}\ x\texttt{=}z$

Definition: make v visible in scope of x: $\sigma[x \nwarrow v]$

(a) $(\mathcal{X}v, \sigma)[x \nwarrow v] = \mathcal{X}v, \sigma$ if $x \in \text{dom}(\mathcal{X}v)$

(b) $(\mathcal{X}v, \mathcal{X}v_0, \sigma)[x \nwarrow v] = \mathcal{X}v_1, ((\mathcal{X}v \setminus \mathcal{X}v_1)\mathcal{X}v_0, \sigma)[x \nwarrow v]$

if $x \notin \text{dom}(\mathcal{X}v)$ and garbage-of$(\ \texttt{(}\mathcal{X}v\ v\texttt{)}\ ) = \mathcal{X}v_1$

Definition: update variable x with v: $\sigma[x := v]$

(a) $(\mathcal{X}v_1, ..., \mathcal{X}v_n)[x := v] = \mathcal{X}v_1[x := v], ..., \mathcal{X}v_n[x := v]$

(b) $(\mathcal{X}v_1\ \overline{\texttt{var}}\ \overline{T}\ x\texttt{=}v\ \mathcal{X}v_2)[x := v] = \mathcal{X}v_1\ (\overline{\texttt{var}}\ \overline{T}\ x\texttt{=}v)[x := v]\ \mathcal{X}v_2$

(c) $\mathcal{X}v[x := v] = \mathcal{X}v$ with $x \notin \text{dom}(\mathcal{X}v)$

(d) $\texttt{var}\ \overline{T}\ x\texttt{=}v_0[x := v] = \texttt{var}\ \overline{T}\ x\texttt{=}v$

Definition: extract field $x.y$ in binding z: $\sigma[\texttt{+=} z := x.y]$

(a) $(\mathcal{X}v_1, ..., \mathcal{X}v_n)[\texttt{+=} z := x.y] = \mathcal{X}v_1[\texttt{+=} z := x.y], ..., \mathcal{X}v_n[\texttt{+=} z := x.y]$

(b) $(\mathcal{X}v_1 \overline{\texttt{var}}\ \overline{T} x\texttt{=}v\ \mathcal{X}v_2)[\texttt{+=} z := x.y] = \mathcal{X}v_1 (\overline{\texttt{var}}\ \overline{T} x\texttt{=}v)[\texttt{+=} z := x.y]\ \mathcal{X}v_2$

(c) $\mathcal{X}v[\texttt{+=} z := x.y] = \mathcal{X}v$ with $x \notin \text{dom}(\mathcal{X}v)$

(d) $\overline{\texttt{var}}\ \overline{T}\ x\texttt{=}\pi.m\texttt{(}\overline{x{:}v_1}\ y{:}v\ \overline{x{:}v_2}\texttt{)}\ [\texttt{+=} z := x.y]$

$= \overline{\texttt{var}}\ \overline{T}\ x\texttt{=}\pi.m\texttt{(}\overline{x{:}v_1}\ y{:}z\ \overline{x{:}v_2}\texttt{)}\ \ \overline{T'}[\text{from } \pi]\ z\texttt{=}v$

if $p(\pi) = \texttt{\{.}m\texttt{(}\overline{\mathcal{F}_1}\ \_\ T'y\ \overline{\mathcal{F}_2}\texttt{)}\_\texttt{\}}$

(e) $\overline{\texttt{var}}\ \overline{T}\ x\texttt{= (}\mathcal{X}v_0\,v_0\texttt{)}\ [\texttt{+=} z := x.y] = \mathcal{X}v_2\ \ \overline{T}\ z\texttt{=}v_2\ \overline{\texttt{var}}\ \overline{T}\ x\texttt{= (}\mathcal{X}v_3\,v_1\texttt{)}$

(e1) iff $\overline{T}\ x\texttt{=}v_0[\texttt{+=} z := x.y] = \mathcal{X}v_1\ \ \overline{T}\ z\texttt{=}v_2\ \ \overline{T}\ x\texttt{=}v_1$

(e2) and ($x\texttt{=void}, \mathcal{X}v_0\ \mathcal{X}v_1)[x \nwarrow v_2] = \mathcal{X}v_2\ \ x\texttt{=void}, \mathcal{X}v_3$

Definition: update field $x.y$ with v: $\sigma[x.y := v]$

(a) $(\mathcal{X}v_1, ..., \mathcal{X}v_n)[x.y := v] = \mathcal{X}v_1[x.y := v], ..., \mathcal{X}v_n[x.y := v]$

(b) $(\mathcal{X}v_1\ \overline{\texttt{var}}\ \overline{T}\ x\texttt{=}v_0\ \mathcal{X}v_2)[x.y := v] = \mathcal{X}v_1\ (\overline{\texttt{var}}\ \overline{T}\ x\texttt{=}v_0)[x.y := v]\ \mathcal{X}v_2$

(c) $\mathcal{X}v[x.y := v] = \mathcal{X}v$ with $x \notin \text{dom}(\mathcal{X}v)$

(d) $\overline{\texttt{var}}\ \overline{T} x\texttt{=}\pi.m\texttt{(}\overline{x{:}v_1}\ y{:}\_\ \overline{x{:}v_2}\texttt{)}[x.y := v] = \overline{\texttt{var}}\ \overline{T} x\texttt{=}\pi.m\texttt{(}\overline{x{:}v_1}\ y{:}v\ \overline{x{:}v_2}\texttt{)}$

(e) $\overline{\texttt{var}}\ \overline{T}\ x\texttt{= (}\mathcal{X}v\,v_0\texttt{)}\ [x.y := v] = \overline{\texttt{var}}\ \overline{T}\ x\texttt{= (}\mathcal{X}v\,v_1\texttt{)}$

with $\overline{\texttt{var}}\ \overline{T}\ x\texttt{=}v_0[x.y := v] = \overline{\texttt{var}}\ \overline{T}\ x\texttt{=}v_1$

Definition: $\overline{\mathcal{M}}_0; \overline{\mathcal{M}}_1$ are impl;decl of $\overline{\mathcal{M}}$

$\emptyset; \emptyset$ are impl;decl of $\emptyset$ and assuming $\overline{\mathcal{M}}_0; \overline{\mathcal{M}}_1$ are impl;decl of $\overline{\mathcal{M}}$,

$mh^s{:}e\ \overline{\mathcal{M}}_0; \overline{\mathcal{M}}_1$ are impl;decl of $mh^s{:}e\ \overline{\mathcal{M}}$, otherwise

$\overline{\mathcal{M}}_0; \mathcal{M}, \overline{\mathcal{M}}_1$ are impl;decl of $\mathcal{M}, \overline{\mathcal{M}}$.

Definition: modifiers restriction $\mu_0{}^{\backslash\overline{\mu}\backslash} = \mu_1$

(a) $\mu^{\backslash\overline{\mu}\backslash} = \mu$ if $\mu \notin \overline{\mu}$

otherwise $\texttt{shared}^{\backslash\overline{\mu}\backslash} = \texttt{lent}$ if $\texttt{lent} \notin \overline{\mu}$

otherwise $\texttt{shared}^{\backslash\overline{\mu}\backslash} = \texttt{lent}^{\backslash\overline{\mu}\backslash} = \texttt{immutable}^{\backslash\overline{\mu}\backslash} = \texttt{readable}^{\backslash\overline{\mu}\backslash} = \texttt{readable}$

(b) $\mu\,\pi^{\backslash\overline{\mu}\backslash} = \mu^{\backslash\overline{\mu}\backslash}\ \pi$ and $\mu\,\pi\,\hat{}^{\backslash\overline{\mu}\backslash} = \mu^{\backslash\overline{\mu}\backslash}\ \pi\,\hat{}$

(c) $(T_1 ... T_n)^{\backslash\overline{\mu}\backslash} = T_1{}^{\backslash\overline{\mu}\backslash} ... T_n{}^{\backslash\overline{\mu}\backslash}$

(d) $(x_1 \mapsto T_1, ..., x_n \mapsto T_n)^{\backslash\overline{\mu}\backslash} = x_1 \mapsto T_1{}^{\backslash\overline{\mu}\backslash}, ..., x_n \mapsto T_n{}^{\backslash\overline{\mu}\backslash}$

(e) $(\Gamma; p^t; \lambda; \overline{\mu}_0; \overline{T}; \overline{\pi})^{\backslash\overline{\mu}\backslash} = \Gamma^{\backslash\overline{\mu}\backslash}; p^t; \lambda; \overline{\mu}_0; \overline{T}; \overline{\pi}$

(f) $\Delta^{\backslash\texttt{catch}\ x\ \texttt{on}\ s_1\ \pi_1\ e_1 ... \texttt{on}\ s_n\ \pi_n\ e_n\backslash} = \Delta$ iff $s_1 ... s_n \subseteq \{\texttt{return}, \texttt{exception}\}$

$\Delta^{\backslash\mathcal{K}\backslash} = \Delta^{\backslash\texttt{shared, lent}\backslash}$ otherwise

Definition: $\mu_1 \leq \mu_2$, cost-of$_{\Delta}(\overline{T} \leq T)$, cost-of$(\mu_1 \leq \mu_2)$

(a) $\mu \leq \mu$, $\texttt{shared} \leq \texttt{lent}$, and with $\mu \neq \texttt{type}$ $\texttt{capsule} \leq \mu$ and $\mu \leq \texttt{readable}$

(b) cost-of$(\mu_1 \leq \mu_2) = \emptyset$ if $\mu_1 \leq \mu_2$

(c) cost-of$(\texttt{shared} \leq \mu) = \texttt{shared}$ if $\mu \neq \texttt{type}$

(d) cost-of$(\texttt{lent} \leq \texttt{immutable}) = \texttt{shared}, \texttt{lent}$

(e) cost-of$(\texttt{readable} \leq \texttt{immutable}) = \texttt{shared}, \texttt{lent}, \texttt{readable}$

(f) cost-of$_{\Delta}(\mu_1\ \pi_1 \leq \mu_2\ \pi_2) =$ cost-of$(\mu_1 \leq \mu_2)$ with $p^{t\Delta} \vdash \pi_1 \leq \pi_0$

Definition: toPh(Γ)

toPh$(\emptyset) = \emptyset$

toPh$(x \mapsto \overline{\texttt{var}}\ \mu\ \pi\ \Gamma) =$ toPh$(x \mapsto \overline{\texttt{var}}\ \mu\ \pi\,\hat{}\ \Gamma) = x \mapsto \mu\ \pi\,\hat{}$ toPh(Γ)

Definition: obj(Δ)

obj$(\Gamma; p^t; \overline{\mu}) =$ obj$(\Gamma \setminus x \mapsto \overline{\texttt{var}}\ \mu\ \pi\,\hat{}; p^t; \overline{\mu})$ if $x \mapsto \overline{\texttt{var}}\ \mu\ \pi\,\hat{} \in \Gamma$

otherwise obj$(\Delta) = \Delta$

Definition: $\mathcal{X}_0 \setminus \mathcal{X}_1 = \mathcal{X}$

$\mathcal{X}$ is the biggest (possibly non-contiguous) subsequence of $\mathcal{X}_0$

such that $\forall x \in \text{dom}(\mathcal{X}_1)$: not $x \in \text{dom}(\mathcal{X}_0)$

Definition: garbage-of$(\ \texttt{(}\mathcal{X}v\ e\texttt{)}\ ) = \mathcal{X}v_1$

$\mathcal{X}v_1$ is the biggest (possibly non-contiguous) subsequence of $\mathcal{X}v$

such that $\forall x \in \text{dom}(\mathcal{X}v_1)$: not x inside $\texttt{(}\mathcal{X}v \setminus \mathcal{X}v_1\ e\texttt{)}$

Definition: invoke$_{\pi}(mh^t, e)$, invoke$_{\pi}(mh^t, e_0, ..., e_n, e)$

invoke$_{\pi}(mh^t, e) =$ invoke$_{\pi}(mh^t, \texttt{error void}, ..., \texttt{error void}, e)$

invoke$_{\pi}(mh^t, e_0, ..., e_n, e) = \texttt{(}\mu\ \pi\ \texttt{this=}e_0\ \ T_1 x_1\texttt{=}e_1 ...\ T_n x_n\texttt{=}e_n\ \ T\ x\texttt{=}e\ x\texttt{)}$

with $mh^t = \mu\ \texttt{method}\ T\ m\texttt{(}\ T_1\ x_1 ...\ T_n\ x_n\texttt{)}\ \texttt{exception}\ \overline{\pi}$

[$Marco$: alpha renaming for parameter names?]

Definition: inferType$_{p^t}(e)$=e_0, inferType$_{p^t}(\Gamma, e)$=T, classOf$_{p^t,\sigma}(v)$=π

inferType$_{p^t}(e) = \mathcal{E}^{\star}[\texttt{(}\mathcal{X}_0\overline{\texttt{var}}$ inferType$_{p^t}(\Gamma, e)\ x\texttt{=}e\mathcal{X}_1\mathcal{K}e_2\texttt{)}]$

with $e = \mathcal{E}^{\star}[\texttt{(}\mathcal{X}_0\overline{\texttt{var}}\ x\texttt{=}e\mathcal{X}_1\mathcal{K}e_2\texttt{)}]$ and $\Gamma = \{x \mapsto T \mid \overline{\texttt{var}}\ T\ x\texttt{=}\_$ inside $e\}$

$\cup\{x \mapsto \texttt{immutable}\ \pi \mid \mathcal{E}^{\star} = \mathcal{E}^{\star}_0[\ \texttt{(}\mathcal{X}\texttt{catch}\ x\ \overline{\mathcal{O}}_0\texttt{on}\ \pi\ \mathcal{E}^{\star}_1\overline{\mathcal{O}}_1\_\texttt{)}\ ]\}$

inferType$_{p^t}(\Gamma, x) = T$ iff $T = \Gamma(x)$

inferType$_{p^t}(\Gamma, e.m\texttt{(}x_1{:}e_1 ... x_n{:}e_n\texttt{)}) = T$ iff inferType$_{p^t}(\Gamma, e) = _\,\pi$,

$p^t(\pi)$=$\texttt{\{}\mathcal{H}\overline{\mathcal{M}^t}\texttt{\}}^{\lambda}and\overline{\mathcal{M}^t}(\texttt{method}\,m(x_1,...,x_n))$=$_\texttt{method}\ T m\texttt{(_)}\ \texttt{exception}\ _$

inferType$_{p^t}(\Gamma, \texttt{(}\mathcal{X}\mathcal{K}e\texttt{)}) =$ inferType$_{p^t}(\Gamma, e)$

inferType$_{p^t}(\Gamma, x\texttt{:=}\ e) =$ inferType$_{p^t}(\Gamma, \texttt{void}) = \texttt{immutable Void}$

inferType$_{p^t}(\Gamma, \mathcal{L}) = \texttt{immutable Library}$

inferType$_{p^t}(\Gamma, \pi) = \texttt{type Any}$ iff $p^t(\pi) = \texttt{\{interface<:\_\_\}}$

otherwise inferType$_{p^t}(\Gamma, \pi) = \texttt{type}\ \pi$

classOf$_{p^t,\sigma}(v) = \pi$ iff inferType$_{p^t}(\Gamma)e = \mu\,\pi$

where $\Gamma(x) = T$ iff $\_\ T\ x\texttt{=}v$ inside σ

Definition: resolve$_{p^t}(T_0) = T_1$

resolve$_{p^t}(\mu\,\pi_1\hat{}) = \mu\,\pi_1\hat{}$

resolve$_{p^t}(\pi\,m_0\,m_1\,\overline{m}) =$ resolve$_{p^t}(\pi_1\,m_1\,\overline{m})$ iff resolve$_{p^t}(\pi_0\,m) = \mu\,\pi_1\hat{}$

resolve$_{p^t}(\pi_0\,m) =$ resolve$_{p^t}(T)$

iff $p^t(\pi_0) = \texttt{\{}\mathcal{H}\overline{\mathcal{M}^t}\texttt{\}}^{\lambda}$ and $\overline{\mathcal{M}^t}(m) = \_\texttt{method}\ T\ m\texttt{(\_)}\ \texttt{exception}\ \_$

Definition: norm$_{p^t}(\pi)$

norm$_{p^t}(\texttt{Outer}^{i+1}\texttt{::}C\texttt{::}\overline{C}) =$ norm$_{p^t}(\texttt{Outer}^{i}\texttt{::}\overline{C})$

iff $p^t(\texttt{Outer}^{i+1}) = \texttt{\{}\mathcal{H}\overline{\mathcal{M}^t}\updownarrow C \mapsto \_\texttt{\}}^{\lambda}$ $\quad$ norm$_{p^t}(\pi) = \pi$ otherwise

Definition: super$_{p^t}(\overline{\pi})$, directS$_{p^t}(\pi_0)$

norm$_{p^t}(\pi_1) \in$ super$_{p^t}(\overline{\pi}_0)$

iff $\pi_0 \in \overline{\pi}_0$, $\pi_1 \in$ directS$_{p^t}(\pi_0) \cup$ super$_{p^t}$(directS$_{p^t}(\pi_0)$)

directS$_{p^t}(\pi_0) = \overline{\pi}_1$ iff p^t(norm$_{p^t}(\pi_0)$)[from norm$_{p^t}(\pi_0)$] $= \texttt{\{\_<:}\overline{\pi}_1\ \_\texttt{\}}$

Definition: $p^t \vdash \pi_0 \leq \pi_1$

$p^t \vdash \pi_0 \leq \pi_1$ iff $\pi_1 \in$ super$_{p^t}(\pi_0) \cup \pi_0 \cup \texttt{Any}$

Definition: $p^t \vdash T_0 \leq T_1$

$p^t \vdash T_0 \leq T_1$ iff $p^t \vdash$ resolve$_{p^t}(T_0) \leq$ resolve$_{p^t}(T_1)$

$p^t \vdash \mu_0\,\pi_0\hat{}_0 \leq \mu_1\,\pi_1\hat{}_1$ iff $p^t \vdash \pi_0 \leq \pi_1$and $\hat{}_0 \in \{\hat{}_1, \hat{}\}$ and

$\vdash \mu_0 \leq \mu_1$ with $\texttt{capsule} < \texttt{shared} < \texttt{lent} < \texttt{readable}$, $\texttt{capsule} < \texttt{immutable} < \texttt{readable}$

Definition: src$_{p^t}(mh^s, \pi)$

src$_{p^t}(mh^s, \pi)$ iff $p^t(\pi) = \texttt{\{\_<:}\overline{\pi}\,\overline{\mathcal{M}^t}\texttt{\}}^{\lambda}$ and

$\overline{\mathcal{M}^t}(mh^s)$ is defined and $\forall \pi_i \in$ super$_{p^t}(\overline{\pi})$:

$p^t(\pi_i) = \texttt{\{\_}\overline{\mathcal{M}^t}_i\texttt{\}}$ and $\overline{\mathcal{M}^t}_i(mh^s)$ is undefined

Definition: singledecl$_{p^t}(\overline{\pi})$

singledecl$_{p^t}(\overline{\pi})$ iff $\forall mh^s : \forall \pi_1, \pi_2 \in$ super$_{p^t}(\overline{\pi})$:

src$_{p^t}(mh^s, \pi_1)$and src$_{p^t}(mh^s, \pi_2)$ implies $\pi_1 = \pi_2$

Definition: methodTypes$_{p^t}(\overline{\pi})$

methodTypes$_{p^t}(\overline{\pi}) = \{\overline{\mathcal{M}^t}(mh^s)[\text{from } \pi] \mid$

$p^t(\pi) = \texttt{\{}\mathcal{H}\overline{\mathcal{M}^t}\texttt{\}}^{\lambda}, \pi \in$ super$_{p^t}(\overline{\pi})$, src$_{p^t}(mh^s, \pi)\}$

$\sigma_1|e_1 \xrightarrow[p;p^t]{} \sigma_2|e_2$

(R-METHOD)

$$\dfrac{}{\sigma|v_0.m(x_1{:}v_1 \ldots x_n{:}v_n) \xrightarrow[p;p^t]{} \sigma|\mathsf{invoke}_\pi(mh^t[\mathsf{from}\ \pi], v_0, \ldots, v_n, e[\mathsf{from}\ \pi])}$$

with
classOf$_\sigma(v_0) = \pi$
$p(\pi) = \{\mathcal{H}\ \overline{\mathcal{M}}_1\ mh{:}e\ \overline{\mathcal{M}}_2\}$
$p^t(\pi) = \{\mathcal{H}'\overline{\mathcal{M}^t}\}^{\circledast}$
$mh^t = \overline{\mathcal{M}^t}(mh) = \_\ \mathtt{method}\ \_\ m(x_1\_ \ldots x_n\_)\ \mathtt{exception}\ \_$

(R-EXPAND)

$$\dfrac{}{\sigma|(\overline{\mathcal{X}v}\ v_0).m(x_1{:}v_1 \ldots x_n{:}v_n) \xrightarrow[p;p^t]{} \sigma|(\overline{\mathcal{X}v}\ v_0.m(x_1{:}v_1 \ldots x_n{:}v_n))}$$

with
classOf$_\sigma((\overline{\mathcal{X}v}\ v_0)) = \pi$
either is-set$(p(\pi), .m(x_1 \ldots x_n))$ or is-get$(p(\pi), .m(x_1 \ldots x_n))$

(R-CTX)

$$\dfrac{\sigma_0|e_0 \xrightarrow[p;p^t]{} \sigma_1|e_1}{\sigma_0|\mathcal{E}^p[e_0] \xrightarrow[p;p^t]{} \sigma_1|\mathcal{E}^p[e_1]}$$

(R-LOOP)

$$\dfrac{}{\sigma|\mathtt{loop}\ e_0 \xrightarrow[p;p^t]{} \sigma|e_1}$$

with
$e_1 = (\mathtt{immutable\ Void}\ z\mathtt{=}e_0\ \mathtt{loop}\ e_0)$

(R-VAR)

$$\dfrac{}{\sigma|x \xrightarrow[p;p^t]{} \sigma[\mathrel{+}= z \mathrel{:}= x]|z}$$

with
$\mathtt{var}\ \_\ x\mathtt{=}\_$ inside σ

(R-VAR-ASS)

$$\dfrac{}{\sigma|x\mathtt{:=}v \xrightarrow[p;p^t]{} \sigma[x \nwarrow v][x \mathrel{:}= v]|\mathtt{void}}$$

with
$\mathtt{var}\ \_\ x\mathtt{=}\_$ inside σ

(R-VAL-CONSTR)

$$\dfrac{}{\sigma|v.m(\overline{x{:}v}) \xrightarrow[p;p^t]{} \sigma|\pi.m(\overline{x{:}v})}$$

with
v not of form π
classOf$_\sigma(v) = \pi$
$\overline{x{:}v} = x_1{:}\_ \ldots x_n{:}\_$
is-constr$(p(\pi), .m(x_1 \ldots x_n))$

(R-CONSTR-GET)

$$\dfrac{}{\sigma|\pi.m_0(x_1{:}v_1 \ldots x_n{:}v_n).m_1() \xrightarrow[p;p^t]{} \sigma|v_i}$$

with
is-constr$(p(\pi), .m_0(x_1 \ldots x_n))$
is-get$(p(\pi), .m_1())$
$\_x_i = m_1$

(R-CONSTR-SET)

$$\dfrac{}{\sigma|\pi.m_0(x_1{:}v_1 \ldots x_n{:}v_n).m_1(y{:}v) \xrightarrow[p;p^t]{} \sigma|\mathtt{void}}$$

with
is-constr$(p(\pi), .m_0(x_1 \ldots x_n))$
is-set$(p(\pi), .m_1(y))$

(R-ALIAS-GET)

$$\dfrac{}{\sigma|x.m() \xrightarrow[p;p^t]{} \sigma[\mathrel{+}= z \mathrel{:}= x.x_i]|z}$$

with
classOf$_\sigma(x) = \pi$
is-get$(p(\pi), .m())$
$\_x_i = m$

(R-ALIAS-SET)

$$\dfrac{}{\sigma|z.m(y{:}v) \xrightarrow[p;p^t]{} \sigma[z \nwarrow v][z.x_i \mathrel{:}= v]|\mathtt{void}}$$

with
classOf$_\sigma(z) = \pi$
is-set$(p(\pi), .m(y))$
$\_x_i = m$

(R-USING-PROP)

$$\dfrac{\sigma|e_0 \xrightarrow[p;p^t]{} \sigma|e_1}{\sigma|\mathtt{using}\ \pi\ \mathtt{check}\ .m(\overline{y{:}v})\ e_0 \xrightarrow[p;p^t]{} \sigma|\mathtt{using}\ \pi\ \mathtt{check}\ .m(\overline{y{:}v})\ e_1}$$

with
$plg; \_ =$ plugin$(p^t, \pi_1, \pi_2, .m(x_1 \ldots x_n))$
execute$(plg, p, \overline{r}, \sigma, \overline{\mathcal{X}}, y_1 \ldots y_n)$ is undefined

(R-USING)

$$\dfrac{}{\sigma|\mathtt{using}\ \pi\ \mathtt{check}\ .m(x_1{:}v_1 \ldots x_n{:}v_n)\ e_0 \xrightarrow[p;p^t]{} \sigma|e}$$

with
$plg;\ T\_ =$ plugin$(p^t, \pi, .m(x_1 \ldots x_n))$
$e =$ execute$(plg, p, \sigma, \overline{\mathcal{X}}, v_1 \ldots v_n, e_0)$ and $e \in \{v, \mathtt{error}\ v\}$
$\emptyset; p^t; \circledcirc; \emptyset; \emptyset; \emptyset \vdash e : T$
if $\exists i \in 1..n$ such that $\emptyset;p^t;\circledast;\emptyset;\emptyset;\emptyset \vdash (\sigma v_i) : \_$ then $\emptyset;p^t;\circledast;\emptyset;\emptyset;\emptyset \vdash v : \_$

(R-CAPTURE)

$$\dfrac{}{\sigma|(\overline{\mathcal{X}v}\ \mathtt{var}\ \overline{T}\ x\mathtt{=}\mathcal{E}^p[s\ v]\ \overline{\mathcal{X}}\ \mathcal{K}\ e) \xrightarrow[p;p^t]{} \sigma|(\overline{\mathcal{X}v}'\ z\mathtt{=}v\ e_0)}$$

with
$\mathcal{K} = \mathtt{catch}\ z\ \mathtt{on}\ s\ \pi_0\ e_0\ \overline{\mathcal{O}}$
$p^t \vdash$ classOf$_\sigma(v) \leq \pi_0$
$\overline{\mathcal{X}v}' = \overline{\mathcal{X}v} \setminus$ garbage-of$((\overline{\mathcal{X}v}\ v))$

(R-ON-OUT)

$$\dfrac{}{\sigma|(\overline{\mathcal{X}}_0\mathtt{catch}\ z\ \mathcal{O}\ \overline{\mathcal{O}}e_0) \xrightarrow[p;p^t]{} \sigma|(\overline{\mathcal{X}}_0\mathtt{catch}\ z\ \overline{\mathcal{O}}e_0)}$$

with
(R-CAPTURE) not applicable
either $\overline{\mathcal{X}}_0 = \overline{\mathcal{X}v}$
or $\overline{\mathcal{X}}_0 = \overline{\mathcal{X}v}\ \mathtt{var}\ \overline{T}\ x\mathtt{=}\mathcal{E}^p[s_0\ v]\ \overline{\mathcal{X}}$

(R-REFERENCE)

$$\dfrac{}{\sigma|\mathcal{E}^\star[x] \xrightarrow[p;p^t]{} \sigma|\mathcal{E}^\star[v]}$$

with
$(\overline{\mathcal{X}v}\ \overline{T}\ x\mathtt{=}v\ \overline{\mathcal{X}}\mathcal{K}e) = \mathcal{E}^\star[x]$
v of form z

(R-R)

$$\dfrac{\sigma_0|e_0 \xrightarrow[p;p^t]{} \sigma_1|e_1}{\sigma_0|(\overline{\mathcal{X}v}\ \mathtt{var}\ \overline{T}\ x\mathtt{=}e_0\overline{\mathcal{X}}\mathcal{K}e) \xrightarrow[p;p^t]{} \sigma_1|(\overline{\mathcal{X}v}\ \mathtt{var}\ \overline{T}\ x\mathtt{=}e_1\overline{\mathcal{X}}\mathcal{K}e)}$$

(R-PROP)

$$\dfrac{}{\sigma|(\overline{\mathcal{X}v}\ \mathtt{var}\ \overline{T}\ x\mathtt{=}\mathcal{E}^p[s\ v]\ \overline{\mathcal{X}}\ e) \xrightarrow[p;p^t]{} \sigma|s\ (\overline{\mathcal{X}v}'v)}$$

with
$\overline{\mathcal{X}v}' = \overline{\mathcal{X}v} \setminus$ garbage-of$((\overline{\mathcal{X}v}\ v))$

(R-EMBED)

$$\dfrac{}{\sigma|(\overline{\mathcal{X}v}\ \_\ \overline{T}\ x\mathtt{=}v\ \overline{\mathcal{X}}\mathcal{K}e_0) \xrightarrow[p;p^t]{} \sigma|\mathcal{E}^\star[v]}$$

with
$(\overline{\mathcal{X}v}\ \overline{\mathcal{X}}\mathcal{K}e_0) = \mathcal{E}^\star[x]$
not x inside $\mathcal{E}^\star[v]$

(R-IN)

$$\dfrac{\overline{\mathcal{X}v}_0, \sigma_0|e_0 \xrightarrow[p;p^t]{} \overline{\mathcal{X}v}_1, \sigma_1|e_1}{\sigma_0|(\overline{\mathcal{X}v}_0\ e_0) \xrightarrow[p;p^t]{} \sigma_1|(\overline{\mathcal{X}v}_1\ e_1)}$$

(R-GARBAGE)

$$\dfrac{}{\sigma|(\overline{\mathcal{X}v}\ e) \xrightarrow[p;p^t]{} \sigma|(\overline{\mathcal{X}v}'\ e)}$$

with
$\overline{\mathcal{X}v}' = \overline{\mathcal{X}v} \setminus$ garbage-of$((\overline{\mathcal{X}v}\ e))$

(R-OUT)

$$\dfrac{}{\sigma|(e) \xrightarrow[p;p^t]{} \sigma|e}$$

(R-META)

$$\dfrac{p^t{}_1 \vdash \mathcal{L}_0 : \{\mathcal{H}\ \overline{\mathcal{M}^t}\ C{\mapsto}\emptyset\}^{\circledcirc} \quad \emptyset; p^t{}_2; \emptyset; \emptyset; \emptyset \vdash e_1 : \mathtt{Library} \quad \emptyset|e_1 \xrightarrow[p_2;p^t{}_2]{} \emptyset|e_2}{\sigma|\mathcal{L}_0 \xrightarrow[p_1;p^t{}_1]{} \sigma|\{\mathcal{H}\overline{\mathcal{M}^c}_1\ C{:}e_2\ \overline{\mathcal{M}}_2\}}$$

with
$\mathcal{L}_0 = \{\mathcal{H}\overline{\mathcal{M}^c}_1\ C{:}e_1\ \overline{\mathcal{M}}_2\}$
$p_2; p^t{}_2 = \mathcal{L}_0, p_1; \{\mathcal{H}\ \overline{\mathcal{M}^t}{\updownarrow}C{\mapsto}\emptyset\}^{\circledcirc}, p^t{}_1$

(R-META-METHOD)

$$\dfrac{p^t{}_1 \vdash \mathcal{L}_0 : \{\mathcal{H}_0\overline{\mathcal{M}^t}\}^{\circledcirc} \quad \emptyset|\mathcal{L}_1 \xrightarrow[p_2;p^t{}_2]{} \emptyset|\mathcal{L}_2}{\sigma|\mathcal{L}_0 \xrightarrow[p_1;p^t{}_1]{} \sigma|\mathcal{L}_3}$$

with
$\mathcal{L}_0 = \{\mathcal{H}\overline{\mathcal{M}^c}_1\ mh\ \mathcal{E}^c[\mathcal{L}_1]\ \overline{\mathcal{M}}_2\}$
$\mathcal{L}_3 = \{\mathcal{H}\overline{\mathcal{M}^c}_1\ mh\ \mathcal{E}^c[\mathcal{L}_2]\ \overline{\mathcal{M}}_2\}$
$p_2; p^t{}_2 = \mathcal{L}_0, p_1; \{\mathcal{H}_0\overline{\mathcal{M}^t}\}^{\circledcirc}, p^t{}_1$

(R-INFER-1)

$$\dfrac{p^t \vdash \{\mathcal{H}_0\overline{\mathcal{M}}_0\ C{:}e_0\ \overline{\mathcal{M}}_1\} : \{\mathcal{H}_1\ \overline{\mathcal{M}^t}\}^{\ominus}}{\sigma|\{\mathcal{H}_0\overline{\mathcal{M}}_0\ C{:}e_0\ \overline{\mathcal{M}}_1\} \xrightarrow[p;p^t]{} \sigma|\{\mathcal{H}_0\overline{\mathcal{M}}_0\ C{:}e_1\ \overline{\mathcal{M}}_1\}}$$

with
$e_1 =$ inferType$_{p^t,\{\mathcal{H}_1\ \overline{\mathcal{M}^t}\}^{\ominus}}(e_0)$

(R-LOAD)

$$\dfrac{p^t \vdash \mathcal{L}^c : \{\_\}^{\circledast}}{\sigma|\{\mathcal{H}^c\ \overline{\mathcal{M}^c}\} \xrightarrow[p;p^t]{} \sigma|\{\mathcal{H}^c\ \overline{\mathcal{M}^c}\ C{:}\text{'}\mathtt{@private}\ \mathcal{L}^c\}}$$

(R-INFER-2)

$$\dfrac{p^t \vdash \{\mathcal{H}_0\overline{\mathcal{M}}_0\ mh{:}e_0\ \overline{\mathcal{M}}_1\} : \{\mathcal{H}_1\ \overline{\mathcal{M}^t}\}^{\ominus}}{\sigma|\{\mathcal{H}_0\overline{\mathcal{M}}_0\ mh{:}e_0\ \overline{\mathcal{M}}_1\} \xrightarrow[p;p^t]{} \sigma|\{\mathcal{H}_0\overline{\mathcal{M}}_0\ mh{:}e_1\ \overline{\mathcal{M}}_1\}}$$

with
$e =$ invoke$_{\mathtt{Outer}_0}(\overline{\mathcal{M}^t}(mh), e_0)$
$(\overline{\mathcal{X}}\ T\ x\mathtt{=}e_1 x) =$ inferType$_{p^t,\{\mathcal{H}_1\overline{\mathcal{M}^t}\}^{\ominus}}(e)$

Desugering and compilation process

With $\mathcal{L}$ as a source in the sugared language $[\![\mathcal{L}]\!]_\emptyset$ is the corresponding desugared term. An **execution process** is a sequence $\mathcal{L}_0 ... \mathcal{L}_n$ such that $\emptyset|\mathcal{L}_0 \underset{\emptyset}{\rightarrow} \emptyset|... \underset{\emptyset}{\rightarrow} \emptyset|\mathcal{L}_n$.

Normal forms are results: either library literals well-typed in ⊛ or representations of an error. **Plugins** are obtained (plugin$(p^t, \pi, .m(\overline{x}))$) from library types (often containing an url doc) extracted from a program type p^t. Plugins monitor execution of code e (execute$(plg, p, \sigma, \overline{\mathcal{X}}, \overline{v}, e)$). **Semantic extensions** are defined by providing different plug-ins implementations through some urls.

Concrete syntax

`immutable`, `trait`, `exception`$\emptyset$ in mh and `<:`$\emptyset$ in $\mathcal{H}$ are represented with the empty string. `catch` $x\ \emptyset$ is represented with the empty string, and is omitted also in the formalism.

EOL can be omitted after the `reuse` sequence of character if no members are present. White-space consists of <space>, EOL and '`,`'.

Well formedness

All the well formednes restriction of the core syntax applies here. Moreover in a $\mathcal{B}$ all $\overline{\mathcal{X}}_i$ except the first are not empty and only the last $\mathcal{K}_i$ can be omitted (having an empty $\overline{\mathcal{O}}$). A $\mathcal{B}$ can not be empty. `with` $\emptyset\ \emptyset\ \emptyset$ _ is not well formed; `with` $\overline{x}\ \overline{\mathcal{X}}\ \overline{\mathcal{I}}\ \overline{\mathcal{O}^s}\ \overline{\mathcal{O}^w}\overline{\mathcal{B}}$ is well formed if the number of types $T_1 ... T_n$ in each `on` is the same of the sum of the cardinalities of $\overline{x}$, $\overline{\mathcal{X}}$ and $\overline{\mathcal{I}}$. Moreover, $\overline{x}$ must be empty if $\overline{\mathcal{O}^w}$ is empty. $\overline{\mathcal{B}}$ is not empty if $\overline{\mathcal{O}^w}$ is empty.

There must not be any whitespace preceding the symbol '`"`' in string expressions or π in number expression.

For all blocks of form `{`$\overline{\mathcal{X}}_1\mathcal{K}_1 ... \overline{\mathcal{X}}_n\mathcal{K}_n$`}`, terminating$([\![$`(`$\overline{\mathcal{X}}_1\mathcal{K}_1 ... \overline{\mathcal{X}}_n\mathcal{K}_n$`void)`$]\!]_p)$ holds. Method names in method calls (using the dot) must be of form x or `#`x.

Operator precedence

Postfix unary operators (as method calls) have the strongest precedence of all, then prefix unary operators and finally binary operators. A sequence of identical binary operators associate from left to right, so that `a+b+c` is equivalent to `(a+b)+c`, but sequences of different operators with the same precedences, like `a+b*c`, are not well formed.

Atomic Language Terms

s	::= `return` | `error` | `exception`	results
μ	::= `type` | `shared` | `readable` | `lent` | `capsule` | `immutable`	type modifiers
$ident$	::= < [_,a..z,A..Z,$,%] [_,a..z,A..Z,$,%,0..9]*>	identifiers
C	::= <$ident$ starting upper-case except `Any`, `Void`, `Library`>	Class names
x, y, z	::= <$ident$ starting lower-case (or _) except keywords>	variable names
m	::= $\overline{x}$ | `#`$\overline{x}$ | $\emptyset\overline{x}$ | $unOp$ | $eqOp$ `that` | $binOp$ `that`	method names
doc	::= $strLine_1 ... strLine_n$<often omitted for brevity>	documentation
$strLine$	::= <spaces> `'`<sequence of $char$ excluding EOL>EOL	line of documentation
$string$	::= <any sequence of $char$ excluding (`"`) and EOL>	simple string
	| EOL doc<spaces><where doc is not empty>	multi line string
$char$	::= <a subset of all character; around ∼ 100 symbols>	source chars
num	::= `0` | `1` | `2` | `3` | `4` | `5` | `6` | `7` | `8` | `9` | `.`	
$unOp$	::= `!` | `~`	
$eqOp$	::= `+=` | `-=` | `*=` | `/=` | `&=` | `|=` | `>>=` | `<<=` | `++=` | `**=` | `:=`	requires x as left value
$boolOp$	::= `&` | `|`	weak precedence
$relOp$	::= `<` | `>` | `==` | `<=` | `>=`	medium precedence
$dataOp$	::= `+` | `-` | `*` | `/` | `<<` | `>>` | `++` | `**`	strong precedence
$binOp$	::= $boolOp$ | $relOp$ | $dataOp$	binary operators
url	::= <sequence of $char$ excluding <space>,EOL, `{` and `}`>	

Complete Language Syntax

e	::= $\mathcal{L}$ | x | π | `void` | $num\ \overline{num}\pi$ | π`"`$string$`"`	expression
	| $s\ e$ | $x\ eqOp\ e$ | $unOp\ e$ | $e_1\ binOp\ e_2$ | e`(`$doc\ ps$`)`	
	| `if` $e\ \mathcal{B}_1$ `else` $\mathcal{B}_2$ | `while` $e\ \mathcal{B}$ | $\mathcal{W}\overline{\mathcal{B}}$ | $e\ doc$ | e`.`m`(`$doc\ ps$`)` | $\mathcal{B}$	
	| e`[`$doc\ ps_1$`;` $doc_1 ... ps_n$`;` doc_n`]` | e`[`$doc\ \mathcal{W}\overline{\mathcal{B}}$`]`	
	| `using` π `check` m`(`$doc\ ps$`)` e	
$\mathcal{B}$	::= `(`$doc\ \overline{\mathcal{X}}_1\mathcal{K}_1 ... \overline{\mathcal{X}}_n\mathcal{K}_n\ e$`)` | `{`$doc\ \overline{\mathcal{X}}_1\mathcal{K}_1 ... \overline{\mathcal{X}}_n\mathcal{K}_n$`}`	expression-block
$\mathcal{K}$	::= `catch` $\overline{x}\ \overline{\mathcal{O}}$	result handler
ps	::= $\overline{e}\ \overline{x\text{:}e}$	parameters
$\mathcal{X}$	::= `var` $T\ x$`=`e | e<$e \neq x$> | C`:`e	statement
$\mathcal{I}$	::= `var` $T\ x$ `in` e | `var` $\overline{T_1\ x_1}$ `in` $\overline{T_2\ x_2\text{=}e}$	iterator decl.
$\mathcal{O}$	::= `on` $s\ \pi$`if` $e\ \mathcal{B}$	signal handler
$\mathcal{W}$	::= `with` $\overline{x}\ \overline{\mathcal{I}}\ \overline{\mathcal{X}}\ \overline{\mathcal{O}^s}\ \overline{\mathcal{O}^w}$	with
$\mathcal{O}^s$	::= `on start` $\mathcal{B}$	
$\mathcal{O}^w$	::= `on` $T_1 ... T_n$`if` $e\ \mathcal{B}$ | `if` $e\ \mathcal{B}$	type-case
$\mathcal{H}$	::= m`(`$\overline{\mathcal{F}}$`)` `<:`$\overline{\pi}$ | `interface` `<:`$\overline{\pi}$ | `trait` `<:`$\overline{\pi}$	lib. node header
$\mathcal{F}$	::= `var` $T\ doc\ x$	field
π	::= $C\overline{::C}$ | `Outer`$^n\overline{::C}$ | `Any` | `Void` | `Library`	node path
$\mathcal{L}$	::= `{` $doc\ \mathcal{H}\ \overline{\mathcal{M}}$ `}` | `{reuse` $url\ EOL\ \overline{\mathcal{M}}$ `}`	library literal
$\mathcal{M}$	::= mh^t | $mh\ e$ | C`:`$doc\ e$ | C`:...`doc	node members
mh^t	::= μ `method` $doc\ T\ m$`(`$\overline{T\ x}$`)` `exception` $\overline{\pi}$	typed meth. header
mh^s	::= `method` $doc\ m$`(`$\overline{x}$`)`	meth. selector
mh	::= mh^t | mh^s	meth. header
T	::= $\mu\ \pi$ | $\mu\ \pi$`^` | $\pi\overline{m}$	obj. and ph. type

FOO

Definition: downloadFromWeb(_)

If the url is a library address, the result is the corresponding library, where members annotated as `'@private` are renamed to others that does not sintactically occurs into the importing program.

Definition: terminating(_)

terminating($s\ e$) holds

terminating(e_0`.`m`(`x_1`:`$e_1 ... x_n$`:`e_n`)`) holds
iff $\exists i \in 1..n$ such that terminating(e_i) holds.

terminating(`(var` $T\ x$`=`e_1...`var` $T\ x$`=`$e_n\mathcal{K}e_0$`)`)
holds iff terminating($\mathcal{K}$) holds and
$\exists i \in 0..n$ such that terminating(e_i) holds.

terminating(`catch` $x\ \mathcal{O}_1 ... \mathcal{O}_n$)
holds iff $\forall i \in 1..n$: terminating($\mathcal{O}_i$) holds.

terminating(`on` $s\ \pi\ e$)
holds iff terminating(e) holds

Definition: $\llbracket e\rrbracket_{\Gamma,\overline{\mathcal{L}}}$ simple cases

(dw) $\llbracket \texttt{\{reuse}\ url\overline{\mathcal{M}}\texttt{\}}\rrbracket_{\Gamma,\overline{\mathcal{L}}} = \texttt{\{}\mathsf{downloadFromWeb}(url)\ \llbracket\overline{\mathcal{M}}\rrbracket_{\Gamma,\emptyset}\texttt{\}}$

otherwise $\llbracket \texttt{\{}\mathcal{H}\ \overline{\mathcal{M}}\texttt{\}}\rrbracket_{\Gamma,\overline{\mathcal{L}}} = \texttt{\{}{}^{\mathcal{M}}\llbracket\mathcal{H}\rrbracket_{\overline{\mathcal{L}},\texttt{\{}\mathcal{H}\overline{\mathcal{M}}\texttt{\}}}\ {}^{\mathcal{M}}\llbracket\overline{\mathcal{M}}\rrbracket_{\overline{\mathcal{L}},\texttt{\{}\mathcal{H}\overline{\mathcal{M}}\texttt{\}}}\texttt{\}}$

(lt) $\llbracket num_1 ... num_n \pi\rrbracket_{\Gamma,\overline{\mathcal{L}}} = \llbracket\pi\rrbracket_{\Gamma,\overline{\mathcal{L}}}\texttt{.\#numberParser().}\llbracket\texttt{'}num_1\texttt{'}\rrbracket\texttt{()}....\llbracket\texttt{'}num_n\texttt{'}\rrbracket\texttt{().\#endParse()}$

$\llbracket\pi\texttt{"}char_1...char_n\texttt{"}\rrbracket_{\Gamma,\overline{\mathcal{L}}} = \llbracket\pi\rrbracket_{\Gamma,\overline{\mathcal{L}}}\texttt{.\#stringParser().}\llbracket\texttt{'}char_1\texttt{'}\rrbracket\texttt{()}....\llbracket\texttt{'}char_n\texttt{'}\rrbracket\texttt{().\#endParse()}$

$\llbracket\pi\texttt{"}EOLchar_1...char_nEOL\texttt{"}\rrbracket_{\Gamma,\overline{\mathcal{L}}} =$

$\llbracket\pi\rrbracket_{\Gamma,\overline{\mathcal{L}}}\texttt{.\#stringParser()}\llbracket\texttt{'}char_1\texttt{'}\rrbracket\texttt{()}...\llbracket\texttt{'}char_n\texttt{'}\rrbracket\texttt{()}\llbracket\texttt{'}EOL\texttt{'}\rrbracket\texttt{().\#endParse()}$

(cf) $\llbracket\texttt{while}\ e\ \mathcal{B}\rrbracket_{\Gamma,\overline{\mathcal{L}}} = \texttt{(}C\texttt{:\{\#apply\})}$

$\llbracket\texttt{(loop (}e\texttt{.\#checkTrue()}\ \mathcal{B}\texttt{) catch on exception Void (void) void)}\rrbracket_{\Gamma,\overline{\mathcal{L}}}$

$\llbracket\texttt{if}\ x\ \mathcal{B}_1\ \texttt{else}\ \mathcal{B}_2\rrbracket_{\Gamma,\overline{\mathcal{L}}} = \llbracket\texttt{(}x\texttt{.\#checkTrue() catch on exception Void}\ \mathcal{B}_2\ \mathcal{B}_1\ \texttt{void)}\rrbracket_{\Gamma,\overline{\mathcal{L}}}$

$\llbracket\texttt{if}\ e\ \mathcal{B}_1\ \texttt{else}\ \mathcal{B}_2\rrbracket_{\Gamma,\overline{\mathcal{L}}} = \llbracket\texttt{(}\ y\texttt{=}e\ \texttt{if}\ y\ \mathcal{B}_1\ \texttt{else}\ \mathcal{B}_2\texttt{)}\rrbracket_{\Gamma,\overline{\mathcal{L}}}$ with y fresh and $e \neq x$

$\llbracket\texttt{(}\overline{\mathcal{X}}_1\mathcal{K}_1...\overline{\mathcal{X}}_n\mathcal{K}_n e\texttt{)}\rrbracket_{\Gamma,\overline{\mathcal{L}}} = \llbracket\texttt{(}\overline{\mathcal{X}}_1\mathcal{K}_1\ \texttt{(}...\ \texttt{(}\overline{\mathcal{X}}_n\mathcal{K}_n e\texttt{)}...\texttt{))}\rrbracket_{\Gamma,\overline{\mathcal{L}}}$

$\llbracket\texttt{(}\mathcal{X}_1...\mathcal{X}_n\mathcal{K}e\texttt{)}\rrbracket_{\Gamma,\overline{\mathcal{L}}} = \texttt{(}{}^{\mathcal{X}}\llbracket\mathcal{X}_1...\mathcal{X}_n\rrbracket_{\Gamma',\overline{\mathcal{L}}}{}^{\mathcal{K}}\llbracket\mathcal{K}\rrbracket_{\Gamma',\overline{\mathcal{L}}}\ \llbracket e\rrbracket_{\Gamma',\overline{\mathcal{L}}}\texttt{)}$ with $\Gamma' = \mathsf{domVar}(\mathcal{X}_1...\mathcal{X}_n)$

$\llbracket\texttt{void}\rrbracket_{\Gamma,\overline{\mathcal{L}}} = \texttt{void}$ $\quad\llbracket\pi\rrbracket_{\Gamma,\overline{\mathcal{L}}} = {}^{\pi}\llbracket\pi\rrbracket_{\overline{\mathcal{L}}}$ $\quad\llbracket e\ doc\rrbracket_{\Gamma,\overline{\mathcal{L}}} = \llbracket\texttt{(}doc\ e\texttt{)}\rrbracket_{\Gamma,\overline{\mathcal{L}}}$

$\llbracket\texttt{loop}\ e\rrbracket_{\Gamma,\overline{\mathcal{L}}} = \texttt{loop}\ \llbracket e\rrbracket_{\Gamma,\overline{\mathcal{L}}}$ $\quad\llbracket s\ e\rrbracket_{\Gamma,\overline{\mathcal{L}}} = s\ \llbracket e\rrbracket_{\Gamma,\overline{\mathcal{L}}}$ $\quad\llbracket e\texttt{.}m\texttt{(}ps\texttt{)}\rrbracket_{\Gamma,\overline{\mathcal{L}}} = \llbracket e\rrbracket_{\overline{\mathcal{L}}}\texttt{.}m\texttt{(}\llbracket ps\rrbracket_{\Gamma,\overline{\mathcal{L}}}\texttt{)}$

$\llbracket\texttt{using}\ \pi\ \texttt{check .}m\texttt{(}ps\texttt{)}\ e\rrbracket_{\Gamma,\overline{\mathcal{L}}} = \texttt{using}\ \llbracket\pi\rrbracket_{\Gamma,\overline{\mathcal{L}}}\ \texttt{check .}m\texttt{(}\llbracket ps\rrbracket_{\Gamma,\overline{\mathcal{L}}}\texttt{)}\ \llbracket e\rrbracket_{\Gamma,\overline{\mathcal{L}}}$

(rt) $\llbracket\texttt{\{}\overline{\mathcal{X}}_1\mathcal{K}_1...\overline{\mathcal{X}}_n\mathcal{K}_n\texttt{\}}\rrbracket_{\Gamma,\overline{\mathcal{L}}} = \texttt{(Void}\ x\texttt{=}\llbracket\texttt{(}\overline{\mathcal{X}}_1\mathcal{K}_1...\overline{\mathcal{X}}_n\mathcal{K}_n\texttt{void)}\rrbracket_{\Gamma,\overline{\mathcal{L}}}$

$\texttt{catch}\ y\ \texttt{on return Any (}y\texttt{) error void)}$

Definition: $\llbracket e\rrbracket_{\Gamma,\overline{\mathcal{L}}}$ case variable decl $\texttt{on} - \texttt{catch}$

${}^{\mathcal{X}}\llbracket C\texttt{:}e\rrbracket_{\Gamma,\overline{\mathcal{L}}} = C\texttt{:}\llbracket e\rrbracket_{\emptyset,\overline{\mathcal{L}}}$

${}^{\mathcal{X}}\llbracket\texttt{var}\ T\ x\texttt{=}e\rrbracket_{\Gamma,\overline{\mathcal{L}}} = \Gamma(x)\texttt{:\{ (var}\ T\ \texttt{inner) \}}$

$\texttt{shared Outer}_0\texttt{::}C\ x\texttt{=Outer}_0\texttt{::}\Gamma(x)\texttt{.\#apply(inner:}\llbracket e\rrbracket_{\Gamma,\overline{\mathcal{L}}}\texttt{)}$

${}^{\mathcal{X}}\llbracket\ \overline{T}\ x\texttt{=}e\rrbracket_{\Gamma,\overline{\mathcal{L}}} = {}^{\pi}\llbracket\overline{T}\rrbracket_p\ x\texttt{=}\llbracket e\rrbracket_{\Gamma,\overline{\mathcal{L}}}$

${}^{\mathcal{X}}\llbracket e\rrbracket_{\Gamma,\overline{\mathcal{L}}} = \texttt{immutable Void}\ x\texttt{=}\llbracket e\rrbracket_{\Gamma,\overline{\mathcal{L}}}$ with x fresh

$\llbracket x\rrbracket_{\Gamma,\overline{\mathcal{L}}} = x$ with $x \notin \mathsf{dom}(\Gamma)$

$\llbracket x\rrbracket_{\Gamma,\overline{\mathcal{L}}} = x\texttt{.\#inner()}$ with $x \in \mathsf{dom}(\Gamma)$

$\llbracket x\texttt{:=}e\rrbracket_{\Gamma,\overline{\mathcal{L}}} = x\texttt{.inner(}\llbracket e\rrbracket_{\Gamma,\overline{\mathcal{L}}}\texttt{)}$

Definition: $\mathsf{cast}^{\Gamma,T}(x_0 \leftarrow x_1)$

$\mathsf{cast}^{\Gamma\ T}(x_0 \leftarrow x_1) = \mu\ \pi\ x_0\texttt{= (return}\ x_1$

$\texttt{catch}\ z\ \texttt{on return}\ \pi\ \texttt{(}z\texttt{) on return Any (exception void) error void)}$ with z fresh and $x_1 \notin \mathsf{dom}(\Gamma)$

$\mathsf{cast}^{\Gamma\ T}(x_0 \leftarrow x_1) = \Gamma(x_1)\texttt{:\{ (shared}\ \Gamma(x_1)\ \texttt{delegate, var}\ T\ \texttt{\_inner)}$

$\texttt{shared method}\ T\ \texttt{\#inner() exception}\ \emptyset\ \texttt{(this.\#\_inner())}$

$\texttt{shared method Void inner(}\ T\ \texttt{that) exception}\ \emptyset\ \texttt{(this.\_inner(that), this.delegate(that)) \}}$

$\texttt{shared Outer}_0\texttt{::}C\ x_0\texttt{=Outer}_0\texttt{::}\Gamma(x_1)\texttt{.\#apply(delegate:}x_1\texttt{, inner: (return}\ x_1$

$\texttt{catch}\ z\ \texttt{on return}\ \pi\ \texttt{(}z\texttt{) on return Any (exception void) error void))}$ with z fresh and $x_1 \in \mathsf{dom}(\Gamma)$

Definition: $\llbracket e\rrbracket_p$ case $\texttt{on} - \texttt{catch}$

${}^{\mathcal{K}}\llbracket\texttt{catch}\ \overline{\mathcal{O}}\rrbracket_p = {}^{\mathcal{K}}\llbracket\texttt{catch}\ x\ \overline{\mathcal{O}}\rrbracket_p$

${}^{\mathcal{K}}\llbracket\texttt{catch}\ x\ \mathcal{O}_1...\mathcal{O}_n\rrbracket_p = \texttt{catch}\ x\ {}^{\mathcal{K}}\llbracket\mathcal{O}_1\rrbracket_p...{}^{\mathcal{K}}\llbracket\mathcal{O}_n\rrbracket_p$ iff none of $\mathcal{O}_1...\mathcal{O}_n$ of form $\texttt{on}\ s\ \pi\texttt{if}\ e\ \mathcal{B}$

otherwise ${}^{\mathcal{K}}\llbracket\texttt{catch}\ x\ \mathcal{O}_1...\mathcal{O}_n\rrbracket_p = \llbracket\texttt{catch}\ x\ \texttt{on}\ s\ \texttt{Any (with}\ x\ \mathcal{O}_1...\mathcal{O}_n\ \texttt{(}s\ x\texttt{))}\rrbracket_p$

iff all of $\mathcal{O}_1...\mathcal{O}_n$ of form $\texttt{on}\ s\ \pi_i\overline{\texttt{if}\ e_i}\ \mathcal{B}_i$ that is, all captures the same s

otherwise ${}^{\mathcal{K}}\llbracket\texttt{catch}\ x\ \mathcal{O}_1...\mathcal{O}_n\rrbracket_p = \texttt{catch}\ x\ \overline{\mathcal{O}}'_1...\overline{\mathcal{O}}'_k\overline{\mathcal{O}}'_0$ that is, $\texttt{on}$s are grouped w.r.t. s_i

with $\{\overline{\mathcal{O}}_1...\overline{\mathcal{O}}_k\overline{\mathcal{O}}_0\} = \{\mathcal{O}_1...\mathcal{O}_n\}$, $\quad\overline{\mathcal{O}}_0 = \overline{\texttt{on}\pi_1\ \pi_2\ x\mathcal{B}}$,

$\forall i \in 1..k: \overline{\mathcal{O}}_i = \texttt{on}s_i\texttt{\_}...\texttt{on}s_i\texttt{\_}$ and $\forall\mathcal{O},\mathcal{O}'$ such that $\overline{\mathcal{O}}_i = \_\mathcal{O}\mathcal{O}'\_$: $\mathcal{O}_1...\mathcal{O}_n = \_\mathcal{O}\overline{\mathcal{O}}\mathcal{O}'\_$,

$\forall i,j \in 1..k$: either $i = j$ or $s_i \neq s_j$, $\quad\forall i \in 0..k$: $\texttt{catch}\ x\overline{\mathcal{O}}'_i = {}^{\mathcal{K}}\llbracket\texttt{catch}\ x\overline{\mathcal{O}}_i\rrbracket_p$

${}^{\mathcal{K}}\llbracket\texttt{on}\ s\ \pi\ \mathcal{B}\rrbracket_p = \texttt{on}\ s\ \llbracket\pi\rrbracket_p\ \llbracket\mathcal{B}\rrbracket_p$ (note: $\texttt{on}\ s\ \pi\texttt{if}\ e\ \mathcal{B}$ is managed in the catch)

Definition: $\llbracket e\rrbracket$ case $\texttt{with}$

(a) $\llbracket\texttt{with}\ \overline{\mathcal{X}}\ \overline{\mathcal{O}^s}\ \mathcal{B}\rrbracket_p = \llbracket\texttt{(}\overline{\mathcal{X}}\overline{\mathcal{B}}\ \mathcal{B}\ \texttt{void)}\rrbracket_p$ with either $\overline{\mathcal{O}^s} = \emptyset = \overline{\mathcal{B}}$ or $\overline{\mathcal{O}^s} = \texttt{on start}\ \overline{\mathcal{B}}$

(b) $\llbracket\mathcal{W}\rrbracket_p = \llbracket\mathcal{W}\ \texttt{(void)}\rrbracket_p$

(c) $\llbracket\texttt{with}\ x_1...x_n\texttt{on}\ T_1...T_n\mathcal{B}_1\mathcal{B}_2\rrbracket_{\Gamma,\overline{\mathcal{L}}} = \llbracket\texttt{(}\mathsf{cast}^{\Gamma,\Gamma',T_1}(y_1 \leftarrow x_1)...\mathsf{cast}^{\Gamma,\Gamma',T_n}(y_n \leftarrow x_n)$

$\texttt{catch exception Void}\ \mathcal{B}_2\ (\mathcal{B}_1[x_1\ T_1 := y_1]...[x_n\ T_n := y_n])\ \texttt{void)}\rrbracket_p$

with $y_1...y_n$ fresh and $y_i\texttt{:}C_i \in \Gamma'$ iff $x_i \in \mathsf{dom}(\Gamma)$ and C_i fresh

(d) $\llbracket\texttt{with}\ x_1...x_n\texttt{on}\ T_1...T_n\texttt{if}\ e_0\ \mathcal{B}_1\ \mathcal{B}_2\rrbracket_p = \llbracket\texttt{(}$

$\texttt{type Any}\ y\texttt{=\{with}\ x_1...x_n\texttt{on}\ T_1...T_n\ \texttt{(if}\ e_0\ \texttt{(return Void)) return Any\}}$

$\texttt{with}\ y\ x_1...x_n\texttt{on Void}\ T_1...T_n\mathcal{B}_1\ \mathcal{B}_2\texttt{)}\rrbracket_p$

(e) $\llbracket\mathcal{W}\texttt{if}\ e\mathcal{B}_1\ \mathcal{B}_2\rrbracket_p = \llbracket\mathcal{W}\ \texttt{(if}\ e\mathcal{B}_1\texttt{else}\mathcal{B}_2\texttt{)}\rrbracket_p$

(f) $\llbracket\mathcal{W}\ \texttt{on}\ T_1...T_n\overline{\texttt{if}\ e}\mathcal{B}_1\ \mathcal{B}_2\rrbracket_p = \llbracket\mathcal{W}\ \texttt{(with}x_1...x_n\texttt{on}\ T_1...T_n\overline{\texttt{if}\ e}\mathcal{B}_1\texttt{else}\mathcal{B}_2\texttt{)}\rrbracket_p$

with $\mathcal{W} = \texttt{with}x_1\_...x_n\_\overline{\mathcal{O}}$

(g) $\llbracket\texttt{with}\ \overline{x}\ \overline{\mathcal{I}}\ \overline{\mathcal{X}}\ \overline{\mathcal{O}^s}\mathcal{B}\rrbracket_p = \llbracket\texttt{with}\ \overline{\mathcal{I}}\ \overline{\mathcal{X}}\ \overline{\mathcal{O}^s}\mathcal{B}\rrbracket_p$

(h) $\llbracket\texttt{with}\ \overline{\mathcal{I}}\ \overline{\mathcal{X}}\ \overline{\mathcal{O}^s}\mathcal{B}\rrbracket_p = \llbracket\mathsf{declareIts}_{\overline{T}'_0z_0...\overline{T}'_nz_n,\llbracket\overline{\mathcal{I}}\rrbracket_p}(\texttt{(}\overline{\mathcal{O}^s}\texttt{loop(}\overline{\mathcal{X}}\texttt{(}\mathcal{X}_0\mathcal{K}_0...\mathcal{X}_n\mathcal{K}_n\mathcal{B}'\texttt{)))},\rrbracket)_p$

with $\llbracket\overline{\mathcal{I}}\rrbracket_p = \overline{\texttt{var}}_0\ x_0\ \overline{T}_0\ \texttt{in}\ \overline{T}'_0\ z_0\texttt{=}e_0...\overline{\texttt{var}}_n\ x_n\ \overline{T}_n\ \texttt{in}\ \overline{T}'_n\ z_n\texttt{=}e_n$,

$\mathcal{X}_i\mathcal{K}_i = \mathsf{next}_i(\overline{\texttt{var}}_i\ x_i, \overline{T}_0z_0...\overline{T}_nz_n)$ and $\mathcal{B}' = \mathsf{replaceAssign}_{x_0,z_0...x_n,z_n}(\mathcal{B})$

Definition: $\llbracket e\rrbracket$ case collections initialization and operators

(initW) $\llbracket e\ \texttt{[}\ \overline{T}\texttt{with}\ \overline{x}\ \overline{\mathcal{I}}\ \overline{\mathcal{X}}\overline{\mathcal{O}^s}\overline{\texttt{on}\ \overline{T}_1\texttt{if}e_1}\mathcal{B}_1\ ...\ \overline{\texttt{on}\ \overline{T}_n\texttt{if}e_n}\mathcal{B}_n\overline{\mathcal{B}}\texttt{]}\rrbracket_p = \llbracket\texttt{(}\ \overline{T}\ x\texttt{=}e\texttt{.\#apply()}$

$\texttt{with}\ \overline{x}\ \overline{\mathcal{I}}\ \overline{\mathcal{X}}\ \overline{\mathcal{O}^s}\overline{\mathcal{O}}\overline{\texttt{on}\ \overline{T}_1\ \texttt{if}e_1}\ \texttt{(}x\texttt{.\#add(}\mathcal{B}_1\texttt{))}\ ...\ \overline{\texttt{on}\ \overline{T}_n\ \texttt{if}e_n}\ \texttt{(}x\texttt{.\#add(}\mathcal{B}_n\texttt{))}\ \overline{\texttt{(}x\texttt{.\#add(}\mathcal{B}\texttt{))}\ x}\texttt{)}\rrbracket_p$

with either $\overline{\texttt{(}x\texttt{.\#add(}\mathcal{B}\texttt{))}}$ and $\overline{\mathcal{B}}$ empty or $\overline{\texttt{(}x\texttt{.\#add(}\mathcal{B}\texttt{))}} = \texttt{(}x\texttt{.\#add(}\mathcal{B}\texttt{))}$ and $\overline{\mathcal{B}} = \mathcal{B}$

(init) $\llbracket e\ \texttt{[}ps_1\texttt{;}...\texttt{;}\ ps_n\texttt{;]}\rrbracket_p = \llbracket e\texttt{.\#apply().\#add(}ps_1\texttt{)}....\texttt{\#add(}ps_n\texttt{)}\rrbracket_p$

(op) $\llbracket e_1\ binOp\ e_2\rrbracket_{\overline{\mathcal{L}}} = \llbracket e_1\texttt{.}\llbracket\texttt{'}binOp\texttt{'}\rrbracket\texttt{(}e_2\texttt{)}\rrbracket_{\overline{\mathcal{L}}}$

$\llbracket unOp\ e\rrbracket_{\overline{\mathcal{L}}} = \llbracket e\rrbracket_{\overline{\mathcal{L}}}\texttt{.}\llbracket\texttt{'}unOp\texttt{'}\rrbracket\texttt{()}$

$\llbracket e\ \texttt{(}ps\texttt{)}\rrbracket_{\overline{\mathcal{L}}} = \llbracket e\texttt{.\#apply(}ps\texttt{)}\rrbracket_{\overline{\mathcal{L}}}$

Definition: $\llbracket\texttt{'\_'}\rrbracket$ char, number and operators

$\llbracket\texttt{'0'}\rrbracket = \texttt{\#\_0}...\llbracket\texttt{'9'}\rrbracket = \texttt{\#\_9}$ $\quad\llbracket\texttt{'a'}\rrbracket = \texttt{\#\_a}...\llbracket\texttt{'z'}\rrbracket = \texttt{\#\_z}$ $\quad\llbracket\texttt{'A'}\rrbracket = \texttt{\#\_A}...\llbracket\texttt{'Z'}\rrbracket = \texttt{\#\_Z}$

$\llbracket\texttt{'.'}\rrbracket = \texttt{\#\_dot}...\llbracket\texttt{'+'}\rrbracket = \texttt{\#\_plus}...\llbracket\texttt{'+='}\rrbracket = \texttt{\#\_plus}...\llbracket\texttt{'}\emptyset\texttt{'}\rrbracket = \texttt{\#apply}$

$\llbracket x\ eqOp\ e\rrbracket_{\overline{\mathcal{L}}} = \llbracket x\texttt{:=}x\texttt{.}\llbracket\texttt{'}eqOp\texttt{'}\rrbracket\texttt{(}e\texttt{)}\rrbracket_{\overline{\mathcal{L}}}$ with $eqOp \neq (\texttt{:=})$

Definition: $\mathsf{guessType}_\Gamma(e)$

$\mathsf{guessType}_\Gamma(\mathcal{L}) = \texttt{immutable Library}$

$\mathsf{guessType}_\Gamma(x) = \Gamma(x)$

$\mathsf{guessType}_\Gamma(\pi) = \texttt{type}\ \pi$

$\mathsf{guessType}_\Gamma(\texttt{void}) = \mathsf{guessType}_\Gamma(s\ e) = \mathsf{guessType}_\Gamma(x\ eqOp\ e) = \mathsf{guessType}_\Gamma(s\ e) =$

$\mathsf{guessType}_\Gamma(\texttt{if}\ e\ \mathcal{B}_1\ \overline{\texttt{else}\ \mathcal{B}_2}) = \mathsf{guessType}_\Gamma(\texttt{while}\ e\ \mathcal{B}) = \mathsf{guessType}_\Gamma(\mathcal{W}\overline{\mathcal{B}}) = \texttt{immutableVoid}$

$\mathsf{guessType}_\Gamma(num_1...num_n\pi) = \mathsf{guessType}_\Gamma(\pi\texttt{.\#numberParser().}\llbracket\texttt{'}num_1\texttt{'}\rrbracket\texttt{()}...$

$\texttt{.}\llbracket\texttt{'}num_n\texttt{'}\rrbracket\texttt{().\#endParse()})$

$\mathsf{guessType}_\Gamma(\pi\texttt{"}char_1...char_n\texttt{"}) = \pi\texttt{.\#stringParser().}\llbracket\texttt{'}char_1\texttt{'}\rrbracket\texttt{()}....\llbracket\texttt{'}char_n\texttt{'}\rrbracket\texttt{().\#endParse()}$

$\mathsf{guessType}_\Gamma(\pi\texttt{"}EOLchar_1...char_nEOL\texttt{"}) = \pi\texttt{.\#stringParser()}\llbracket\texttt{'}char_1\texttt{'}\rrbracket\texttt{()}...$

$\texttt{.}\llbracket\texttt{'}char_n\texttt{'}\rrbracket\texttt{()}\llbracket\texttt{'}EOL\texttt{'}\rrbracket\texttt{().\#endParse()}$

$\mathsf{guessType}_\Gamma(unOp\ e) = \mathsf{guessType}_\Gamma(e\texttt{.}\llbracket\texttt{'}unOp\texttt{'}\rrbracket\texttt{()})$

$\mathsf{guessType}_\Gamma(e_1\ binOp\ e_2) = \mathsf{guessType}_\Gamma(e_1\texttt{.}\llbracket\texttt{'}binOp\texttt{'}\rrbracket\texttt{(}e_2\texttt{)})$

$\mathsf{guessType}_\Gamma(e\ \texttt{(}ps\texttt{)}) = \mathsf{guessType}_\Gamma(e\texttt{.\#apply(}ps\texttt{)})$

$\mathsf{guessType}_\Gamma(e\texttt{.}m\texttt{(}ps\texttt{)}) = T\overline{m}$ iff $\mathsf{guessType}_\Gamma(e) = \pi\overline{m} = T$

$\mathsf{guessType}_\Gamma(e\texttt{.}m\texttt{(}ps\texttt{)}) = \pi m$ iff $\mathsf{guessType}_\Gamma(e) = \mu\ \pi\hat{}$

$\mathsf{guessType}_\Gamma(\texttt{(}\overline{\mathcal{X}}_1\mathcal{K}_1...\overline{\mathcal{X}}_n\mathcal{K}_n e\texttt{)}) = \mathsf{guessType}_{\Gamma'}(e)$ with $\Gamma' = \mathsf{guessType}_\Gamma(\overline{\mathcal{X}}_1...\overline{\mathcal{X}}_n)$

and $\mathsf{guessType}_\Gamma() = \Gamma$ $\quad\mathsf{guessType}_\Gamma(e\ \overline{\mathcal{X}}) = \mathsf{guessType}_\Gamma(C\texttt{:}e\ \overline{\mathcal{X}}) = \mathsf{guessType}_\Gamma(\overline{\mathcal{X}})$

$\mathsf{guessType}_\Gamma(\overline{\texttt{var}}\ T\ x\texttt{=}e\overline{\mathcal{X}}) = \mathsf{guessType}_{\Gamma,x\mapsto T}(\overline{\mathcal{X}})$

$\mathsf{guessType}_\Gamma(\overline{\texttt{var}}\ x\texttt{=}e\ \overline{\mathcal{X}}) = \mathsf{guessType}_{\Gamma,x\mapsto\mathsf{guessType}_\Gamma(e)}(\overline{\mathcal{X}})$

$\mathsf{guessType}_\Gamma(\texttt{\{}\overline{\mathcal{X}}_1\mathcal{K}_1...\overline{\mathcal{X}}_n\mathcal{K}_n\texttt{\}})$ is correctly undefined

$\mathsf{guessType}_\Gamma(e\ \texttt{[}ps_1\texttt{;}...\ ps_n\texttt{;]}) = \mathsf{guessType}_\Gamma(e\texttt{.\#apply().\#add(}ps_1\texttt{)}...\texttt{.\#add(}ps_n\texttt{)})$

$\mathsf{guessType}_\Gamma(e\ \texttt{[}\mathcal{W}\overline{\mathcal{B}}\texttt{]}) = \mathsf{guessType}_\Gamma(e\texttt{.\#apply()})$

$\mathsf{guessType}_\Gamma(e\ doc) = \mathsf{guessType}_\Gamma(\texttt{using}\ \pi\ \texttt{check}\ m(doc\ ps)\ e) = \mathsf{guessType}_\Gamma(e)$

Definition: $e[x := y]$

$e_0[x := e_1] = e_0$ iff $\forall\mathcal{E}^\star : e_0 \neq \mathcal{E}^\star[x]$

otherwise $\mathcal{E}^\star[x][x := e_1] = \mathcal{E}^\star[e_1][x := e_1]$

Definition: $e[x\ T := y]$

$e[x\mu\ \texttt{Any} := y] = e$, otherwise

$e[x\ T := y] = e[x := y]$

Definition: $\mathsf{replaceAssign}_{\overline{x,z}}(e)$

$\mathsf{replaceAssign}_{x,z\ \overline{x,z}}(\mathcal{E}^\star[x\ eqOp\ e]) =$

$\mathsf{replaceAssign}_{x,z\ \overline{x,z}}(\mathcal{E}^\star[x\ eqOp\ z\texttt{.\#update(}e\texttt{)}])$

iff $e \neq z\texttt{.\#update(\_)}$

$\mathsf{replaceAssign}_{x,z\ \overline{x,z}}(e) = \mathsf{replaceAssign}_{\overline{x,z}}(e)$

iff e_0 inside e implies $e_0 = z\texttt{.\#update(\_)}$

$\mathsf{replaceAssign}_{\emptyset,\emptyset}(e) = e$

Definition: $\llbracket e\rrbracket_p$ auxiliary definitions

${}^{ps}\llbracket e_0\ \overline{x\texttt{:}e}\rrbracket_p = {}^{ps}\llbracket\texttt{that:}e_0\ \overline{x\texttt{:}e}\rrbracket_p$

${}^{ps}\llbracket x_1\texttt{:}e_1...x_n\texttt{:}e_n\rrbracket_p = x_1\texttt{:}\llbracket e_1\rrbracket_p...x_n\texttt{:}\llbracket e_n\rrbracket_p$

${}^{\mathcal{M}}\llbracket\mathcal{M}_1...\mathcal{M}_n\rrbracket_p = {}^{\mathcal{M}}\llbracket\mathcal{M}_1\rrbracket_p...{}^{\mathcal{M}}\llbracket\mathcal{M}_n\rrbracket_p$

${}^{\mathcal{M}}\llbracket C\texttt{:}e\rrbracket_p = C\texttt{:}\llbracket e\rrbracket_p$

${}^{\mathcal{M}}\llbracket C\texttt{:...}\rrbracket_p = C\texttt{:}\llbracket e\rrbracket_p$ Where e is found on the local system depending on the original position of such **...** symbol in the source and C

${}^{\mathcal{M}}\llbracket m\texttt{(}\overline{\mathcal{F}}\texttt{) <:}\overline{\pi}\rrbracket_p = \llbracket\texttt{'}m\texttt{'}\rrbracket\texttt{(}\overline{\mathcal{F}}\texttt{) <:}{}^{\pi}\llbracket\overline{\pi}\rrbracket_p$

${}^{\mathcal{M}}\llbracket\texttt{interface <:}\overline{\pi}\rrbracket_p = \texttt{interface <:}{}^{\pi}\llbracket\overline{\pi}\rrbracket_p$

${}^{\mathcal{M}}\llbracket\texttt{trait <:}\overline{\pi}\rrbracket_p = \texttt{trait <:}{}^{\pi}\llbracket\overline{\pi}\rrbracket_p$

${}^{\mathcal{M}}\llbracket mh\,e\rrbracket_p = \overline{C\texttt{:}e}\ {}^{\mathcal{M}}\llbracket mh\rrbracket_p e'$

with $e' = \llbracket e\rrbracket_p$ without all the $\mathcal{X}$ of form $\overline{C\texttt{:}e}$

${}^{\mathcal{M}}\llbracket\mu\ \texttt{method}\ T\ m\texttt{(}\ \overline{T\ x}\texttt{) exception}\ \overline{\pi}\rrbracket_p =$

$\mu\ \texttt{method}\ {}^{\pi}\llbracket T\rrbracket_p\ \llbracket\texttt{'}m\texttt{'}\rrbracket\texttt{(}{}^{\pi}\llbracket\ \overline{T\ x_p}\rrbracket\texttt{) exception}\ {}^{\pi}\llbracket\overline{\pi}\rrbracket_p$

${}^{\mathcal{M}}\llbracket\texttt{method}m\texttt{(}\overline{x}\texttt{)}\rrbracket_p = \texttt{method}\llbracket\texttt{'}m\texttt{'}\rrbracket\texttt{(}\overline{x}\texttt{)}$

${}^{\mathcal{I}}\llbracket\overline{\texttt{var}}\ \overline{T}_1\ x_1\ \texttt{in}\ \overline{T}_2\ x_2\texttt{=}e\rrbracket = \overline{\texttt{var}}\ \overline{T}_1\ x_1\ \texttt{in}\ \overline{T}_2\ x_2\texttt{=}e$

${}^{\mathcal{I}}\llbracket\overline{\texttt{var}}\ \overline{T}_1\ x_1\ \texttt{in}\ e\rrbracket = \overline{\texttt{var}}\ \overline{T}_1\ x_1\ \texttt{in}\ \ x_2\texttt{=}e$ with x_2 fresh

${}^{\pi}\llbracket\overline{C\texttt{::}C}\rrbracket_{\mathcal{L}_0...\mathcal{L}_n} = \texttt{Outer}_k\texttt{::}\overline{C\texttt{::}C}$

where $\mathcal{L}_k(\texttt{::}C)$ well defined and

$\forall i < k : \mathcal{L}_i(\texttt{::}C)$ not well defined

${}^{\pi}\llbracket\overline{C\texttt{::}C}\rrbracket_{\mathcal{L}_0...\mathcal{L}_n} = \texttt{Outer}_n\texttt{::}\overline{C\texttt{::}C}$

where $\forall i \in 0..n : \mathcal{L}_i(\texttt{::}C)$ not well defined

Definition: $\mathsf{declareIts}_\Gamma(\overline{T}_0z_0...\overline{T}_nz_n, \overline{\mathcal{I}}, e)$

$\mathsf{declareIts}_\Gamma(\emptyset,\emptyset,e) = \texttt{(}e\texttt{catch on exception Void (void) void)}$

$\mathsf{declareIts}_\Gamma(\overline{T}_0z_0...\overline{T}_nz_n\ \overline{T}z, \overline{\mathcal{I}}\ \overline{\texttt{var}}\ \overline{T}\ x\ \texttt{in}\ e, e_0) =$

$\mathsf{declareIts}_\Gamma(\overline{T}z, \overline{T}_0z_0...\overline{T}_nz_n, \overline{\mathcal{I}}, \texttt{(}$

$\mathsf{declareIt}_\Gamma(\overline{T}z, \overline{\texttt{var}}\ \overline{T}\ x\ \texttt{in}\ e)$

$\texttt{(}e_0\ \texttt{catch}\ y\ \texttt{on return Any (}x\texttt{.\#close() return}\ y\texttt{)}$

$\texttt{on exception Any (}x\texttt{.\#close() exception}\ y\texttt{)}$

$x\texttt{.\#close()))})$

Definition: $\mathsf{declareIt}_\Gamma(\overline{T}z, \overline{\texttt{var}}\ \overline{T}\ x)\ \texttt{in}\ e$

$\mathsf{declareIt}_\Gamma(\overline{T}z, \texttt{var}\ T\ x\ \texttt{in}\ e) = \overline{\texttt{var}}\ \overline{T}\ x$

Definition: $\mathsf{next}_i(\overline{\texttt{var}}\ x, \overline{T}_0z_0...\overline{T}_nz_n)$

$\mathsf{next}_i(x, \overline{T}_0z_0...\overline{T}_nz_n) = \overline{\texttt{var}}\ \overline{T}_i\ x\texttt{=}z_i\texttt{.\#next()}$

$\texttt{catch on exception Void (}$

$\texttt{(}\ \overline{T}_{i+1}\ y\texttt{=}z_{i+1}\texttt{.\#next() catch on exception Void (void))}$

$...\texttt{(}\ \overline{T}_n\ y\texttt{=}z_n\texttt{.\#next() catch on exception Void (void))}$

$\texttt{(}z_0\texttt{.\#checkEnd() catch on exception Void (void))}$

$...\texttt{(}z_n\texttt{.\#checkEnd() catch on exception Void (void))}$

$\texttt{exception void)}$

Definition: $\llbracket doc\rrbracket_p$

$\llbracket doc\rrbracket_p$ replaces all substrings of the form $\texttt{@}\pi$ and $\texttt{@(}e\texttt{)}$ with $\texttt{@}\llbracket\pi\rrbracket_p$ and $\texttt{@(}\llbracket e\rrbracket_p\texttt{)}$

This applies to all documentations excluding the one in multi-line string literals.

Definition: $\mathsf{domVar}(\mathcal{X}_1...\mathcal{X}_n)$

$x\texttt{:}C \in \mathsf{domVar}(\mathcal{X}_1...\mathcal{X}_n)$ iff

$\mathcal{X}_i = \texttt{\_var}\ \overline{T}\ x\texttt{=}e\texttt{\_}$, C is fresh and $i \in 1..n$