Language 42

for more information `L42.is`

Core Language Syntax

$\mathcal{L} ::= \{\, doc\ \mathcal{H}\texttt{<:}\overline{\pi}\ \overline{doc'\,\mathcal{M}}\,\}^{\circledcirc} \mid \updownarrow$	library literal
$\mathcal{M} ::= h \mid mh\ e$	class member
$\mathcal{H} ::= \mathtt{interface} \mid \emptyset$	class header
$e ::= a \mid \mathtt{loop}\ e \mid e.m(doc\ \overline{x\texttt{:}e}) \mid (doc\ \overline{d}\,e)$	expression
$\mid (doc\ \overline{d}\overline{\mathcal{K}}e) \mid \varrho\ e \mid \mathtt{using}\ \pi\ \mathtt{check}\ m(doc\ \overline{x\texttt{:}e})\ e$	expression
$a ::= x \mid \pi \mid \mathtt{void} \mid \mathcal{L}$	atomic value
$\varrho ::= \mathtt{exception} \mid \mathtt{error} \mid \mathtt{return}$	signal
$d ::= T\ x\ doc\texttt{=}e$	binding def.
$\mathcal{K} ::= \mathtt{catch}\ \varrho\ x\ (doc\overline{\mathcal{O}})$	catch-match
$\mathcal{O} ::= \mathtt{on}\ T\ doc\ e$	on-case
$h ::= \mu\ \mathtt{method}\ doc\ T\ doc'\ m(\overline{T\ x\ doc})\ \mathtt{exception}\ \overline{\pi}\ doc'$	typed m. header
$mh ::= h \mid \mathtt{method}\ doc\ m(\overline{x}) \mid C\texttt{:}doc$	member header
$m ::= x \mid \texttt{\#}x$	method name
$\pi ::= \mathtt{Outer}^n\texttt{::}\overline{C} \mid \mathtt{Any} \mid \mathtt{Void} \mid \mathtt{Library}$	path
$\alpha ::= \emptyset \mid \texttt{\^{}} \mid \%$	ph annotation
$T ::= \mu\ \pi\ \alpha \mid \pi\,\overline{mx}\,\alpha$	type annotation
$mx ::= \texttt{::}m(\overline{x})\ \overline{\texttt{::}x}$	typeLink
$\mu ::= \mathtt{immutable} \mid \mathtt{mut} \mid \mathtt{read} \mid \mathtt{lent} \mid \mathtt{capsule} \mid \mathtt{type}$	modifiers
$\circledcirc ::= \ominus \mid \oplus \mid \circledast$	stage

Auxiliary syntax

$\mathcal{E}^c ::= \square \mid \mathcal{E}^c.m(\overline{x\texttt{:}e}) \mid e_0^c.m(\overline{x\texttt{:}e^c}_1\ x\texttt{:}\mathcal{E}^c\ \overline{x\texttt{:}e}_2) \mid \mathtt{loop}\ \mathcal{E}^c \mid \varrho\ \mathcal{E}^c$
$\mid (\overline{d^c}_1\ T\ x\texttt{=}\mathcal{E}^c\ \overline{d}_2\overline{\mathcal{K}}\ e) \mid (\overline{d^c}\,\mathtt{catch}\ \varrho\ x\ \mathtt{on}\ T\ \mathcal{E}^c\ e) \mid (\overline{d^c}\,\overline{\mathcal{K}^c}\,\mathcal{E}^c)$
$\mathtt{using}\ \pi\ \mathtt{check}\ .m(\overline{e^c}_1\mathcal{E}^c\overline{e}_2)\ e \mid \mathtt{using}\ \pi\ \mathtt{check}\ .m(\overline{e^c})\ \mathcal{E}^c$

$\mathcal{E}^\star ::= \square \mid \mathcal{E}^\star.m(\overline{e}) \mid e_0.m(\overline{x\texttt{:}e}_1\ x\texttt{:}\mathcal{E}^\star\ \overline{x\texttt{:}e}_2) \mid \mathtt{loop}\ \mathcal{E}^\star \mid \varrho\ \mathcal{E}^\star$
$\mid (\overline{d}_1\ T\ x\texttt{=}\mathcal{E}^\star\ \overline{d}_2\overline{\mathcal{K}}\ e) \mid (\overline{d}\,\mathtt{catch}\ \varrho\ x\ \mathtt{on}\ T\ \mathcal{E}^\star\ e) \mid (\overline{d}\,\overline{\mathcal{K}}\,\mathcal{E}^\star)$
$\mathtt{using}\ \pi\ \mathtt{check}\ .m(\overline{x\texttt{:}e}_0 x\texttt{:}\mathcal{E}^\star\ \overline{x\texttt{:}e}_1)\ e \mid \mathtt{using}\ \pi\ \mathtt{check}\ .m(\overline{x\texttt{:}v})\ \mathcal{E}^\star$

$v^p ::= a \mid (\overline{dv^p}\ v^p)$ value

$dv^p ::= \mu\ \pi'\ x \texttt{=} \pi.m(x_1\texttt{:}a_1 ... x_n\texttt{:}a_n) \mid \mathtt{immutable}\ \pi\ x \texttt{=} (\overline{dv^p}\ v^p)$
<if $\mathsf{meth}_p(\pi.m(x_1...x_n)) = h\ \mathtt{field}$,
$\mu \neq \mathtt{capsule}$ and $p(\pi')$ not interface>

$\mathcal{E}^p ::= \square \mid (\overline{dv}\ T\ x \texttt{=} \mathcal{E}^p\ \overline{d}\overline{\mathcal{K}}e) \mid (\overline{dv}\mathcal{E}^p) \mid \varrho\ \mathcal{E}^p \mid \mathcal{E}^p.m(\overline{x\texttt{:}e})$
$\mid v_0.m(\overline{x\texttt{:}v}\ x\texttt{:}\mathcal{E}^p\ \overline{x\texttt{:}e}) \mid \mathtt{using}\ \pi\ \mathtt{check}\ .m(\overline{x\texttt{:}v}\ x\texttt{:}\mathcal{E}^p\ \overline{x\texttt{:}e})\ e$
$\mid \mathtt{using}\ \pi\ \mathtt{check}\ .m(\overline{x\texttt{:}v})\ \mathcal{E}^p$

$p ::= \mathcal{L}_0, ..., \mathcal{L}_n{}^{\circledast}$	program type
$\Gamma ::= \overline{x : T}$	
$\Sigma ::= \overline{x}; \overline{x}_1...\overline{x}_n; \overline{x}'_1...\overline{x}'_k$	seal env
$\Phi ::= \overline{T}; \overline{\pi}$	throw env

Definition: compiled(_)

compiled(e) iff compiled($\mathcal{L}$) holds $\forall\mathcal{L}$ inside e
compiled($\{\mathcal{H}\texttt{<:}\overline{\pi}\overline{\mathcal{M}}\}^{\circledcirc}$) iff compiled($\mathcal{M}$)$\forall\mathcal{M} \in \overline{\mathcal{M}}$
compiled(h)
compiled($C\texttt{:}\mathcal{L}$) iff compiled($\mathcal{L}$)
compiled($mh\,e$) iff compiled(e)
We write e^c as a metavariable to represent an e where compiled(e) holds. Same notation is used for $\mathcal{L}^c$ and $\mathcal{M}^c$.

Definition: $\Gamma(x)$, $\overline{d}(x)$, $\mathcal{L}(C)$, $\mathcal{L}(mx)$, $p(\pi)$

$\Gamma(x)$: $(\_, x : T, \_)(x) = T$
$\overline{d}(x)$: $(\overline{d}_1\ T\ x\texttt{=}e\ \overline{d}_2)(x) = e$
$\mathcal{L}(\_)$: extract the corresponding element in $\mathcal{L}$
$p(\pi)$: $(\mathcal{L}_0...\mathcal{L}_n)(\pi) = \mathcal{L}_i(\overline{\texttt{::}C})$ if $\mathsf{norm}_p(\pi) = \mathtt{Outer}_i\texttt{::}\overline{C}$
$\mathcal{L}(\overline{\texttt{::}C})$: $\mathcal{L}(\texttt{::}C_1...\texttt{::}C_n) = \mathcal{L}(C_1)...(C_n)$ where $e(C) = e$ iff e not $\mathcal{L}$
$(\Gamma; \overline{x}; \overline{x}_1...\overline{x}_n)$, $\overline{dv}$ and $\overline{\mathcal{M}^c}$ are maps, thus order is irrelevant.
The above function notations _(_) each implicitly defines a domain dom(_) as the set of all inputs for which the function is defined

Definition: $\mathcal{L}[\mathcal{M}]$

$\{\mathcal{H}\,\overline{\pi}'\,\overline{\mathcal{M}}\,C\texttt{:}\_\}^{\circledcirc}[C\texttt{:}\mathcal{L}] = \{\mathcal{H}\,\overline{\pi}'\,\overline{\mathcal{M}}\,C\texttt{:}\mathcal{L}\}^{\circledcirc}$
$\{\mathcal{H}\,\overline{\pi}'\,\overline{\mathcal{M}}\,mh\ e_1\}^{\circledcirc}[mh\ e_2] = \{\mathcal{H}\,\overline{\pi}'\,\overline{\mathcal{M}}\,mh\ e_2\}^{\circledcirc}$

Definition: _ inside _

e_0 inside e_1 holds iff $e_1 = \mathcal{E}^\star[e_0]$

Notations

Symbols

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; for example $\overline{T}$ and $\overline{\mathtt{var}}$ 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. We consider terms (e) and e to be equivalent, and from now on we omit documentations doc when is not relevant. We consider terms of the form (e) to be equivalent to the corresponding e and terms of the form $(\overline{dv}\,\mathtt{catch}\ \varrho\ x\ ()\ e)$ to be equivalent to the corresponding $(\overline{dv}\,e)$. Also, values of form $(T\ x\texttt{=}(T\ y\texttt{=}ey)\ x)$ are equivalent to the corresponding $(T\ y\texttt{=}ey)$ if $x \notin \mathsf{FV}(e)$. In any moment a type of form $\pi\overline{mx}$ is considered in the context of a program p, we consider it equivalent to the corresponding resolved type $\mathsf{norm}_p(\pi\overline{mx})$.
The following symbols $\ominus\ \oplus\ \circledast\ \updownarrow\ \%$ are used only internally in the formalism, and are not present in the source code.

Syntax well formedness

All parameter names declared within a given method header must be unique. Local names do not hide each others: any method body/expression declaring a name already in scope is not well formed. All methods in a given class must be uniquely identified by their name m and the sequence of their parameter names $\overline{x}$. All nested class names C in a class must be unique. All fields names in a given header must be unique. `this` is not a valid field or parameter name.

$\mathcal{E}^p[e]$ is ill formed if $\mathcal{E}^p = \mathcal{E}^{p\prime}[\mathtt{using}\ \pi\ \mathtt{check}\ .m(\overline{x\texttt{:}v})\ \mathcal{E}^{p\prime\prime}]$ and $\mathsf{plugin}(\pi\ m\ (\overline{x\texttt{:}v})\ \mathcal{E}^{p\prime\prime}[e])$ is well defined.

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

$\mathtt{Outer}^n\texttt{::}\overline{C}[\mathsf{from}\ \mathtt{Outer}^m\texttt{::}C_1...\texttt{::}C_k] = \mathtt{Outer}^m\texttt{::}C_1...\texttt{::}C_{k-n}\texttt{::}\overline{C}$ if $n \leq k$
$\mathtt{Outer}^n\texttt{::}\overline{C}[\mathsf{from}\ \mathtt{Outer}^m\texttt{::}C_1...\texttt{::}C_k] = \mathtt{Outer}^{m+n-k}\texttt{::}\overline{C}$ if $n > k$
$\mathtt{Any}[\mathsf{from}\ \_] = \mathtt{Any}$ $\quad\mathtt{Library}[\mathsf{from}\ \_] = \mathtt{Library}$ $\quad\mathtt{Void}[\mathsf{from}\ \_] = \mathtt{Void}$

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

$e[\mathsf{from}\ \pi]$ propagate on the structure, and $\mathcal{L}[\mathsf{from}\ \pi] = \mathcal{L}[\mathsf{from}\ \pi]_0$
$\{\mathcal{H}\overline{\mathcal{M}}\}[\mathsf{from}\ \pi]_j = \{\mathcal{H}[\mathsf{from}\ \pi]_{j+1}\overline{\mathcal{M}}[\mathsf{from}\ \pi]_{j+1}\}$
$\mathtt{Outer}^{j+n}\texttt{::}\overline{C}_0[\mathsf{from}\ \pi]_j = \mathtt{Outer}^{j+k}\texttt{::}\overline{C}_1$ with $\mathtt{Outer}^n\texttt{::}\overline{C}_0[\mathsf{from}\ \pi] = \mathtt{Outer}^k\texttt{::}\overline{C}_1$
$\mathtt{Outer}^n\texttt{::}\overline{C}[\mathsf{from}\ \pi]_j = \mathtt{Outer}^n\texttt{::}\overline{C}$ with $n < j$
$doc[\mathsf{from}\ \pi]_j$ replaces all substrings of the form $\texttt{@}\pi_0$ and $\texttt{@}(e)$ with $\texttt{@}\pi_0[\mathsf{from}\ \pi]_j$ and $\texttt{@}(e_0[\mathsf{from}\ \pi]_j)$
All cases for other expressions/terms propagate to submembers

Definition: $\Gamma[\mathcal{K}, \Sigma] = \Gamma'$

with $\mathcal{K} = \mathtt{catch}\ \mathtt{error}\ x\ \overline{\mathcal{O}}$ and $\Sigma = \_; \_; \overline{x}_1...\overline{x}_n$
$\Gamma'(x) = \Gamma(x)$ iff $\forall\overline{x}_i$ such that $x \in \overline{x}_i$, $\overline{x} \cap \mathsf{FV}(\mathcal{K}) = \emptyset$
$\Gamma'(x) = \mathsf{mutableLentToReadable}(\Gamma(x))$ otherwise
otherwise $\Gamma' = \Gamma$

Definition: $\Phi[\mathcal{K}]$

$\overline{T}; \overline{\pi}[\mathtt{catch}\ \mathtt{return}\ \mathcal{O}_1...\mathcal{O}_n] = \overline{T}[\mathcal{O}_1]...[\mathcal{O}_n]; \overline{\pi}$
$\mu\pi_1...\mu\ \pi_n[\mathtt{on}\ \mu'\ \pi_0\_] = \mu'\ \pi_0...\mu'\ \pi_n$ if $\mu \leq \mu'$
otherwise $\mu\pi_1...\mu\ \pi_n[\mathtt{on}\ \mu'\ \pi_0\_] = \mu'\ \pi_0$
$\overline{T}; \overline{\pi}[\mathtt{catch}\ \mathtt{exception}\ x\,\mathtt{on}\ \mathtt{immutable}\ \pi_1\_...\mathtt{on}\ \mathtt{immutable}\ \pi_n\_] = \overline{T}; \overline{\pi}, \pi_1...\pi_n \setminus \mathtt{Any}$
otherwise $\Phi[\mathcal{K}] = \Phi$

Definition: $p \vdash \overline{\pi} \leq \Phi$

$p \vdash \overline{\pi}_1 \leq \overline{T}; \overline{\pi}_2$ iff $\forall\pi_1 \in \overline{\pi}_1.\exists\pi_2 \in \overline{\pi}_2$ such that $p \vdash \pi_1 \leq \pi_2$

Definition: $\Delta \vdash e : T \leq T'$, $p \vdash T \leq T'$, $p \vdash \pi \leq \pi'$

$p; \Gamma; \Sigma; \Phi \vdash e : T \leq T'$ iff $p; \Gamma; \Sigma; \Phi \vdash e : T$ and $p \vdash T \leq T'$
$p \vdash \mu\ \pi\alpha \leq \mu'\ \pi'\alpha'$ iff $\mu \leq \mu'$, $\alpha \leq \alpha'$ and $p \vdash \pi \leq \pi'$
$p \vdash \pi \leq \pi'$ iff $\mathsf{norm}_p(\pi') \in \mathsf{norm}_p(\overline{\pi}[\mathsf{from}\ \pi] \cup \pi) \cup \mathtt{Any}$
with $p(\pi) = \{\_\}^{\_;\overline{\pi}}$
$\mathtt{capsule} \leq \mathtt{mut} \leq \mathtt{lent} \leq \mathtt{read}$, $\mathtt{capsule} \leq \mathtt{immutable} \leq \mathtt{read}$ and $\emptyset \leq \% \leq \texttt{\^{}}$

Definition: $\overline{h} \cup \overline{\mathcal{M}}$

$h_1...h_n \cup \overline{\mathcal{M}} = h_1 \cup (... \cup (h_n \cup \overline{\mathcal{M}}))$
$h \cup \overline{\mathcal{M}} = h\overline{\mathcal{M}}$ iff $\mathsf{dom}(h)$ disjoint $\mathsf{dom}(\overline{\mathcal{M}})$
$h\pi \cup mh^s e\overline{\mathcal{M}} = h\pi e\overline{\mathcal{M}}$ iff $\mathsf{dom}(h) = \mathsf{dom}(mh^s e)$
$h\pi \cup h\overline{\mathcal{M}} = h\overline{\mathcal{M}}$

Definitions1

Definition: complete(Γ), dom$^{\mathsf{mut}}(\Gamma)$, dom$^{\mathsf{mut}\leq}(\Gamma)$, mutTolent$(T)$

complete$(\Gamma) = \{x : \mu\ \pi | \Gamma(x) = \mu\ \pi\}$

dom$^{\mathsf{mut}}(\Gamma) = \{x : \mathtt{mut}\ \pi\alpha | \Gamma(x) = \mathtt{mut}\ \pi\alpha\}$

dom$^{\mathsf{mut}\leq}(\Gamma) = \{x : \mu\ \pi\alpha | \Gamma(x) = \mu\ \pi\alpha, \mathtt{mut} \leq \mu\}$

mutTolent$(\mathtt{mut}\ \pi\alpha) = \mathtt{lent}\ \pi\alpha$

mutTolent$(\mu\ \pi\alpha) = \mu\ \pi\alpha$ otherwise

mut&LentToRead$(\mathtt{mut}\ \pi\alpha)$ = mut&LentToRead$(\mathtt{lent}\ \pi\alpha) = \mathtt{read}\ \pi\alpha$

mut&LentToRead$(\mu\ \pi\alpha) = \mu\ \pi\alpha$ otherwise

Definition: move$(\mathcal{E}^p, \overline{x}) = \langle \mathcal{E}^{p\prime}, \overline{dv}' \rangle$

move$(\square, \overline{x}) = \langle \square, \emptyset \rangle$

assuming move$(\mathcal{E}^p, \overline{x}) = \langle \mathcal{E}^{p\prime}, \overline{dv}' \rangle$, then

move$(\mathtt{(}\overline{dv}\ T\ y\ \mathtt{=}\mathcal{E}^p\ \overline{d\mathcal{K}e}\mathtt{)}, \overline{x}) = \langle \mathtt{(}\overline{dv}\ \overline{dv}'\ T\ y\ \mathtt{=}\mathcal{E}^{p\prime}\ \overline{d\mathcal{K}e}\mathtt{)}, \emptyset \rangle$

and move$(\mathtt{(}\overline{dv}\mathcal{E}^p\mathtt{)}, \overline{x}) = \langle \mathtt{(}\overline{dv}\ \overline{dv}'\mathcal{E}^{p\prime}\mathtt{)}, \emptyset \rangle$ with $\overline{x} \subseteq$ dom$(\overline{dv})$

move$(\mathcal{E}^p\mathtt{.}m\ \mathtt{(}\overline{x\mathtt{:}e}\mathtt{)}, \overline{x}) = \langle \mathcal{E}^{p\prime}\mathtt{.}m\ \mathtt{(}\overline{x\mathtt{:}e}\mathtt{)}, \overline{dv}' \rangle$

move$(v\mathtt{.}m\ \mathtt{(}\overline{x\mathtt{:}v}, y\mathtt{:}\mathcal{E}^p, \overline{x\mathtt{:}e}\mathtt{)}, \overline{x}) = \langle v\mathtt{.}m\ \mathtt{(}\overline{x\mathtt{:}v}, y\mathtt{:}\mathcal{E}^{p\prime}, \overline{x\mathtt{:}e}\mathtt{)}, \overline{dv}' \rangle$

move$(\mathtt{(}\overline{dv}\mathcal{E}^p\mathtt{)}, \overline{x}) = \langle \mathtt{(}\overline{dv}_2\mathcal{E}^{p\prime}\mathtt{)}, \overline{dv}'\ \overline{dv}_1 \rangle$

move$(\mathtt{(}\overline{dv}\ T\ y\ \mathtt{=}\mathcal{E}^p\ \overline{d\mathcal{K}e}\mathtt{)}, \overline{x}) = \langle \mathtt{(}\overline{dv}_2\ T\ y\ \mathtt{=}\mathcal{E}^{p\prime}\ \overline{d\mathcal{K}e}\mathtt{)}, \overline{dv}'\ \overline{dv}_1 \rangle$

$\overline{dv} = \overline{dv}_1\ \overline{dv}_2$ with $\overline{dv}_1$ inductively defined by

$x \in$ dom$(\overline{dv}_1)$ iff $x \in$ dom$(\overline{dv})$ and $x \in \overline{x} \cup$ FV$(\overline{dv}')$

$x \in$ dom$(\overline{dv}_1)$ iff $x \in$ dom$(\overline{dv})$, $\overline{dv}_1(\_) = v$ and $x \in$ FV(v)

Definition: dec$(\mathcal{E}^p, x)$

dec$(\mathcal{E}^{p\prime}[\mathtt{(}\overline{dv}\mathcal{E}^p\mathtt{)}], x)$ = dec$(\mathcal{E}^{p\prime}[\mathtt{(}\overline{dv}\ T\ y\ \mathtt{=}\mathcal{E}^p\ \overline{d\mathcal{K}e}\mathtt{)}], x) = \overline{dv}(x)$

if $x \in$ dom$(\overline{dv})$

Definition: class$(\mathcal{E}^p, v)$

class$(\mathcal{E}^p, x) = C$ if dec$(\mathcal{E}^p, x) = _\ x\ \mathtt{=}C\mathtt{.}m\ \mathtt{(_)}$

class$(\mathcal{E}^p, x)$ = class$(\mathcal{E}^p, \mathtt{(}\overline{dv}v\mathtt{)})$

if dec$(\mathcal{E}^p, x) = \mathtt{immutable}\ _\ x\ \mathtt{=}\ \mathtt{(}\overline{dv}v\mathtt{)}$

class$(\mathcal{E}^p, \pi) = \pi$

class$(\mathcal{E}^p, \mathtt{void}) = \mathtt{Void}$, class$(\mathcal{E}^p, \mathcal{L}) = \mathtt{Library}$

class$(\mathcal{E}^p, \mathtt{(}\overline{dv}v\mathtt{)})$ = class$(\mathcal{E}^p[\mathtt{(}\overline{dv}\square\mathtt{)}], v)$

Definition: $\overline{dv}[x.m = a] = \overline{dv}'$

$\overline{dv}\ T\ x\ \mathtt{=}\pi\mathtt{.}m\mathtt{(}\overline{x\mathtt{:}a}y\mathtt{:}\_\overline{x\mathtt{:}a}'\mathtt{)}\ [x.\#y = a] = \overline{dv}\ T\ x\ \mathtt{=}\pi\mathtt{.}m\mathtt{(}\overline{x\mathtt{:}a}y\mathtt{:}a\overline{x\mathtt{:}a}'\mathtt{)}$

Definition: abstract$_p(\mathcal{L})$, coherent$_p(\mathcal{L})$

abstract$_p(\mathcal{L})$ holds if not coherent$_p(\mathcal{L})$

or $\mathcal{L}$ has a nested class C:$\mathcal{L}'$ such that abstract$_p(\mathcal{L}')$

coherent$_p(\mathtt{\{interface_\}}^{\circledcirc})$ holds

coherent$_p(\mathtt{\{}\mathcal{H}\mathtt{<:}\overline{\pi}'\overline{h}\ \overline{\mathcal{M}}\mathtt{\}}^{\circledcirc})$ if no element of for $h \in \overline{\mathcal{M}}$ and either

$\overline{h} = \emptyset$ or $\mathtt{type\ method}\ \mu\ \mathtt{Outer}_0\ m\mathtt{(}\ \overline{T\ x}\mathtt{)}\ \mathtt{exception}\ \_ \in \overline{h}$

and for every other $h \in \overline{h}$, coherent$_p(\mu, \overline{T\ x}, h)$ holds

coherent$_p(\mu, \overline{T\ x}, h)$ = coherent$_p(\mu$, norm$_p(\overline{T\ x}), h)$

coherent$_p(\mu, \mu_1\ \pi_1\hat{}\ x_1 \ldots \mu_n\ \pi_n\hat{}\ x_1, h)$ iff

exists i such that coherent$_p(\mu_i\ \pi_i\ x_i, h)$ holds, and

$\mu \neq \mathtt{type}$, $\mu \in \{\mathtt{read}, \mathtt{lent}\}$ if $\mathtt{read}$ or $\mathtt{lent} \in \{\mu_1 \ldots \mu_n\}$,

$\mu \in \{\mathtt{capsule}, \mathtt{immutable}\}$ iff $\{\mu_1 \ldots \mu_n\} = \{\mathtt{immutable}, \mathtt{capsule}\}$

with $\mu \in \{\mathtt{type}, \mathtt{immutable}, \mathtt{read}\}$, $\mu' \neq \mathtt{type}$, $\mu'' \in \{\mathtt{mut}, \mathtt{lent}\}$, norm$_p(T') = \mathtt{V}$

(a) coherent$_p(\mu\ \pi\ x, \mu'\ \mathtt{method}\ T\ \#x\mathtt{()}\ \mathtt{exception}\ _)$ iff $p \vdash \mu\ \pi \leq$ norm$_p(T)$

(b) coherent$_p(\mu\ \pi\ x, \mu''\ \mathtt{method}\ T'\ \#x\mathtt{(}\ T\ \mathtt{that}\mathtt{)}\ \mathtt{exception}\ _)$ iff $p \vdash$ norm$_p(T)$

with $\mu \in \{\mathtt{mut}, \mathtt{lent}\}$, $\mu' \notin \{\mathtt{type}, \mathtt{mut}\}$, norm$_p(T') = \mathtt{Void}$

(c) coherent$_p(\mu\ \pi\ x, \mathtt{mut\ method}\ T\ \#x\mathtt{()}\ \mathtt{exception}\ _)$ iff $p \vdash \mu\ \pi \leq$ norm$_p(T)$

(d) coherent$_p(\mu\ \pi\ x, \mu'\ \mathtt{method}\ T\ \#x\mathtt{()}\ \mathtt{exception}\ _)$ iff $p \vdash \mu'\ \pi \leq$ norm$_p(T)$

(e) coherent$_p(\mu\ \pi\ x, \mathtt{mut\ method}\ T'\ \#x\mathtt{(}\ T\ \mathtt{that}\mathtt{)}\ \mathtt{exception}\ _)$

iff $p \vdash$ norm$_p(T) \leq \mu\ \pi$

(f) coherent$_p(\mu\ \pi\ x, \mathtt{lent\ method}\ T'\ \#x\mathtt{(}\ T\ \mathtt{that}\mathtt{)}\ \mathtt{exception}\ _)$

iff $p \vdash$ norm$_p(T) \leq \mathtt{capsule}\ \pi$

with $\mu' \neq \mathtt{type}$, norm$_p(T') = \mathtt{Void}$

(g) coherent$_p(\mathtt{capsule}\ \pi\ x, \mu'\ \mathtt{method}\ T\ \#x\mathtt{()}\ \mathtt{exception}\ _)$

iff $p \vdash \mu'\ \pi \leq$ norm$_p(T)$

(h) coherent$_p(\mathtt{capsule}\ \pi\ x, \mathtt{mut\ method}\ T'\ \#x\mathtt{(}\ T\ \mathtt{that}\mathtt{)}\ \mathtt{exception}\ _)$

iff $p \vdash$ norm$_p(T) \leq \mathtt{mut}\ \pi$

Definition: originalMeth$_p(\pi_1 \ldots \pi_n, \overline{mx}_0) = \overline{mx}$

originalMeth$_p(\mathcal{L}_0) = \overline{mx}_0 \setminus \ldots \setminus \overline{mx}_n$ with $\mathcal{L}_0 = \mathtt{\{}\mathcal{H}\mathtt{<:}\pi_1 \ldots \pi_n\overline{\mathcal{M}}\mathtt{\}}$,

$\mathcal{L}_1 = p(\pi_1) \ldots \mathcal{L}_n = p(\pi_n)$, dom$(\mathcal{L}_i) = \overline{mx}_i\overline{C}_i$

Definitions2

Definition: meth$_p(\pi\mathtt{.}m\ \mathtt{(}\overline{x}\mathtt{)})$

meth$_p(\pi\mathtt{.}m\ \mathtt{(}\overline{x}\mathtt{)})$ = norm$_p(p(\pi)(m\ \mathtt{(}\overline{x}\mathtt{)})$[from π])

Definition: norm$_p(\pi)$, norm$_p(T)$, norm$_p(h\overline{e})$

norm$_p(\mathtt{Outer}^{i+1}\mathtt{::}C\mathtt{::}\overline{C})$ = norm$_p(\mathtt{Outer}^{i}\mathtt{::}\overline{C})$

iff $p(\mathtt{Outer}^{i+1}) = \mathtt{\{}\mathcal{H}\ \overline{\mathcal{M}}\ C\mathtt{:}\updownarrow\mathtt{\}}^{\circledcirc}$ norm$_p(\pi) = \pi$ otherwise

norm$_p(\mu\ \pi\alpha) = \mu$ norm$_p(\pi)\alpha$

norm$_p(\pi' mx\hat{}) = \mu\ \pi\hat{}$ iff norm$_p(\pi' mx) = \mu\ \pi\alpha$

norm$_p(\pi_1 mx_1 mx_2\overline{mx})$ = norm$_p(\pi_2 mx_2\overline{mx})$

iff norm$_p(\pi_1 mx_1) = \mu\ \pi_2\alpha$

norm$_p(\pi\mathtt{::}m\ \mathtt{(}\overline{x}\mathtt{)})$ = norm$_p(T)$,

norm$_p(\pi\mathtt{::}m\ \mathtt{(}\overline{x}\mathtt{)}\mathtt{::}x_i)$ = norm$_p(T_i)$

and norm$_p(\pi\mathtt{::}m\ \mathtt{(}\overline{x}\mathtt{)}\mathtt{::this}) = \mu\ \mathtt{Outer}_0$

iff meth$_p(\pi\mathtt{.}m\ \mathtt{(}\overline{x}\mathtt{)}) = \mu\ \mathtt{method}\ T\ m\mathtt{(}T_1\ x_1 \ldots T_n\ x_n\mathtt{)}\ \mathtt{exception}\ _$

norm$_p(\pi\overline{mx})$ is undefined iff $p(\pi) = \emptyset$ or we run into a cycle

norm$_p(\mu\ \mathtt{method}\ T_0\ m\mathtt{(}\ T_1\ x_1 \ldots\ T_n\ x_n\mathtt{)}\ \mathtt{exception}\ \overline{\pi}\overline{e})$

$= \mu\ \mathtt{method}\ T_0'\ m\mathtt{(}\ T_1'\ x_1 \ldots\ T_n'\ x_n\mathtt{)}\ \mathtt{exception}$ norm$_p(\overline{\pi})$norm$_p(\overline{e})$

with T_i' = norm$_p(T_i)$

Definition: exe$^{\oplus}(p)$ exe$^{\oplus}(p, \pi)$ exe(p) exe(p, π) exeOk$^{\oplus}(p, \Gamma)$

exe$^{\oplus}(p)$ = exe$^{\oplus}(p, \mathtt{Outer}_0)$

exe$^{\oplus}(p, \mathtt{Any})$, exe$^{\oplus}(p, \mathtt{Void})$ and exe$^{\oplus}(p, \mathtt{Library})$ holds.

exe$^{\oplus}(p, \pi)$ iff $p(\pi) = \mathcal{L}^{\mathtt{c}} \in \{\mathtt{\{\_\}}^{\oplus}, \mathtt{\{\_\}}^{\circledast}\}$

exe$(p\overline{\circledast})$ iff $\overline{\circledast} = \circledast$

exe(p, π) holds iff $p(\pi) = \mathcal{L}^{\mathtt{c}} = \mathtt{\{\_\}}^{\circledast}$

exeOk$^{\oplus}(p, x_1\mathtt{:}_\ \pi_1_ \ldots x_n\mathtt{:}_\ \pi_n_)$

iff either not exe$^{\oplus}(p)$ or $\forall i \in 1..n$ exe$^{\oplus}(p, \pi_i)$

Definition: toPartial$(_)$, toPh$(_)$

toPartial$(\mu\ \pi) = \mu\ \pi\%$

toPartial$(\mu\ \pi\alpha) = \mu\ \pi\alpha$ otherwise

toPh$(\mu\ \pi_) = \mu\ \pi\hat{}$ and toPh$(\pi\overline{mx}_) = \pi\overline{mx}\hat{}$

those notions trivially extends to Γ

Definition: throws$_p(e) = \varrho\ v$

throws$_p(\varrho\ v) = \varrho\ v$

with e not a value, and throws$_p(e) = \varrho\ v$

throws$_p(e\mathtt{.}m\mathtt{(_)})$ = throws$_p(v\mathtt{.}m\mathtt{(}\overline{x\mathtt{:}v}x\mathtt{:}e_\mathtt{)})$ = throws$_p(\varrho\ e) = \varrho\ v$

throws$_p(\mathtt{(}\overline{dv}e\mathtt{)}) = \varrho\ \mathtt{(}\overline{dv}v\mathtt{)}$

throws$_p(\mathtt{(}\overline{dv}, \overline{dv}'\ T\ x\mathtt{=}e\ \overline{d}\ e_0\mathtt{)}) = \varrho\ \mathtt{(}\overline{dv}v\mathtt{)}$

Definition: used$(\mathcal{L}^{\mathtt{c}}) = \overline{\pi}$ used$^{\oplus}(\mathcal{L}^{\mathtt{c}}) = \overline{\pi}$

used$^{\oplus}(\mathtt{\{}\mathcal{H}\mathtt{<:}\overline{\pi}, \overline{\mathcal{M}}\mathtt{\}}\text{-}) = \overline{\pi} \cup$ used$^{\oplus}(\overline{\mathcal{M}})$

$\mathtt{Outer}_k\mathtt{::}\overline{C} \in$ used$^{\oplus}(C\mathtt{:}\mathcal{L}^{\mathtt{c}})$ iff $\mathtt{Outer}_{k+1}\mathtt{::}\overline{C} \in \mathcal{L}^{\mathtt{c}}$

$\pi \in$ used$^{\oplus}(h\overline{e})$ iff $\pi \in$ used$^{\oplus}(\overline{e})$ or π inside h

$\pi \in$ used$^{\oplus}(e)$ iff $\pi\mathtt{.}\_$ inside e

used$(\mathcal{L}^{\mathtt{c}})$ is defined as used$^{\oplus}(\widehat{\mathcal{L}^{\mathtt{c}}})$ but in addition

$\pi \in$ used$(mh\ e)$ iff $\pi \in$ used(e)

$\mathtt{Outer}_k\mathtt{::}\overline{C} \in$ used(e) iff $\mathtt{Outer}_{k+1}\mathtt{::}\overline{C} \in$ used$(\mathcal{L}^{\mathtt{c}})$ and $\mathcal{L}^{\mathtt{c}}$ inside e

Definition: plugin$(p, \pi\ m\ \mathtt{(}\overline{x\mathtt{:}v}\mathtt{)}\ e_0) = e$

if plg; $T\_$ = plugin$(p, \pi, m\ \mathtt{(}x_1 \ldots x_n\mathtt{)})$

with x as implicit reduction step identifier

execute$(x, plg, p, \mathcal{E}^p, v_1 \ldots v_n, e_0) = e$ and $e \in \{v, \mathtt{error}\ v\}$

$p; \emptyset; \emptyset \vdash e : T' \leq T$

either for all $\mathcal{L}$ inside e $\quad p \vdash \mathcal{L} \to \mathtt{\{\_\}}^{\overline{\pi}}$

or exists $\mathcal{L}$ inside v such that $p \vdash \mathcal{L} \to \mathtt{\{\}}^{\ominus}$

or exists π inside v such that $p(\pi) = \mathtt{\{\}}^{\ominus}$

if all of the former holds, then plugin$(\pi\ m\ \mathtt{(}x_1\mathtt{:}v_1 \ldots x_n\mathtt{:}v_n\mathtt{)}\ e_0) = e$

functions plugin$(_, _, _)$ and $_$execute$(_, _, _, _, _, _)$ are defined by the specific 42 implementation; the step identifier is a fresh variable that identify unequivocally the current reduction step.

Definition: stageOf$_p(\mathcal{L}^{\mathtt{c}}, \overline{e}) = \circledcirc$

stageOf$_p(\mathcal{L}) = \ominus$ iff $\overline{e}$not of form $\overline{\mathcal{L}^{\mathtt{c}}}$ or $\mathtt{\{\_\}}^{\ominus} \in \overline{e}$

otherwse stageOf$_p(\mathcal{L}) = \oplus$ iff abstract$_p(\mathcal{L})$ or $\mathtt{\{\_\}}^{\oplus} \in \overline{e}$

Definition: superOf$_p(\mathcal{L}^{\mathtt{c}}) = \overline{\pi}, \overline{\mathcal{M}}$

superOf$_p(\mathcal{L}) = \overline{\pi}_1$[from π_1]$\cup \ldots \cup \overline{\pi}_n$[from π_n], $\overline{h}_1$[from π_1]$\ldots\overline{h}_n$[from π_n]

norm$_{\mathcal{L}p}(\overline{\pi}) = \{\pi_1 \ldots \pi_n\}$, $(\mathcal{L}p)(\pi_i) = \mathtt{\{interface<:}\overline{\pi}_i\overline{h}_i\ \overline{C\mathtt{:}e}_i\mathtt{\}}^{\circledcirc}$

all originalMeth$_{\mathcal{L}p}(\overline{\pi}_i$[from π_i], dom$(\overline{h}_i))$ are disjoint, $\mathcal{L} = \mathtt{\{}\mathcal{H}\mathtt{<:}\overline{\pi}\overline{\mathcal{M}}\mathtt{\}}$

Definition: HB$(\mathcal{E})$, FV(e), $e[x = v]$

are they used somewhere?

Extraction of types

(ET-+)

$$\frac{\mathcal{L}_1 \xrightarrow{p} \ldots \xrightarrow{p} \mathcal{L}_2 \quad \mathcal{L}_2 \not\xrightarrow{p} \_}{\mathcal{L}_1 \xrightarrow[\max]{p} \mathcal{L}_2}$$

(ET-DEEP)

$$\frac{\mathcal{L}_1 \xrightarrow{\mathcal{L},p} \mathcal{L}_2}{\mathcal{L} \xrightarrow{p} \mathcal{L}[C{:}\mathcal{L}_2]}$$

with
$\mathcal{L} = \{\mathcal{H}\_\overline{\mathcal{M}}\,C{:}\mathcal{L}_1\}$

(ET-SUB)

$$\frac{}{\mathcal{L} \xrightarrow{p} \{\mathcal{H}{<:}\overline{\pi} \cup \overline{\pi}'\overline{h} \cup \overline{\mathcal{M}}\}^{\circledast}}$$

with
$\mathcal{L} = \{\mathcal{H}{<:}\overline{\pi}\,\overline{\mathcal{M}}\}$
$\mathsf{superOf}_p(\mathcal{L}) = \overline{\pi}',\ \overline{h}$
$\overline{h} \cup \overline{\mathcal{M}}$ not have untyped headers

(ET-LABEL)

$$\frac{}{\mathcal{L} \xrightarrow{p} \{\mathcal{H}\overline{\mathcal{M}}\}^{\mathsf{stageOf}_p(\mathcal{L},\overline{e})}}$$

with
$\mathcal{L} = \{\mathcal{H}\overline{\mathcal{M}}\}^{\circledast}$
$\overline{e} = \{(\mathcal{L}p)(\pi) \mid \pi \in \mathsf{used}^{\oplus}(\mathcal{L})\} \setminus \updownarrow$
$\quad \cup\{\mathcal{L}(::C) \mid ::C \in \mathsf{dom}(\mathcal{L})\}$
$\forall \mathcal{L}^{\mathfrak{c}} \in e, \mathcal{L}^{\mathfrak{c}} = \{\}^{\circledcirc}$

need a rule for neverLess? read comment under

(P-OK)

$$\frac{\vdash p : \text{ok if} p \neq \emptyset \quad p \vdash \ddot{\mathcal{L}}_0 : \text{ok}}{\vdash \ddot{\mathcal{L}}p : \text{ok}}$$

with
$\ddot{\mathcal{L}}_0 = \ddot{\mathcal{L}}[C{:}\mathtt{error\ void}]$
if exists C such that $\ddot{\mathcal{L}}(C) = \updownarrow$
otherwise $\ddot{\mathcal{L}}_0 = \ddot{\mathcal{L}}$

(METH-T-DEF)

$$\frac{p;\Gamma;\Sigma;\emptyset;\overline{\pi} \vdash e : T' \leq \mathsf{toPartial}(T) \text{ if } \overline{e} = e}{p \vdash \mathcal{M} : \text{ok}}$$

with
$\mathsf{norm}_p(\mathcal{M}) = h\overline{e}$ fully normalized
$h = \mu\ \mathtt{method}\ T\ m\mathtt{(}T_1\ x_1 \ldots T_n x_n\mathtt{)}\ \mathtt{exception}\ \overline{\pi}$
$\Gamma = x_1 : T_1 \ldots x_n : T_n,\ \mathtt{this} : \mu\ \mathtt{Outer}_0$
$\Sigma = \emptyset;\emptyset;\mathtt{this}, x_1, \ldots, x_n$
$\mathsf{exeOk}^{\oplus}(p,\Gamma)$

(CHECK-CT1)

$$\frac{\mathcal{L}[C{:}\updownarrow], p \vdash \mathcal{L}' : \text{ ok } \forall C{:}\mathcal{L}', \mathcal{L}(C) = C{:}\mathcal{L}'}{p \vdash \mathcal{L} : \text{ok}}$$

with
$\mathcal{L} \notin \{\{\_\}^{\oplus}, \{\_\}^{\circledast}\}$

(CHECK-CT2)

$$\frac{\widehat{\mathcal{L}^{\mathfrak{c}}}[C{:}\updownarrow], p \vdash \mathcal{L}' : \text{ ok } \forall C{:}\mathcal{L}^{\mathfrak{c}\prime}, \mathcal{L}^{\mathfrak{c}}(C) = C{:}\mathcal{L}^{\mathfrak{c}} \quad \mathcal{L}^{\mathfrak{c}}, p \vdash h\overline{e} : \text{ok}\forall h\overline{e}, \mathcal{L}^{\mathfrak{c}}(h) = h\overline{e}}{p \vdash \mathcal{L}^{\mathfrak{c}} : \text{ok}}$$

with
$\mathcal{L}^{\mathfrak{c}} \in \{\{\_\}^{\oplus}, \{\_\}^{\circledast}\}$

[Marco: by being at least plus, $\mathcal{L}^{\mathfrak{c}}$ is fully normalized/able]

Type for expressions

(PATH-PATH)

$$\frac{}{p;\Gamma;\Sigma;\Phi \vdash \pi : \mathtt{type}\ \pi}$$

with
$p(\pi) = \widehat{\mathcal{L}^{\mathfrak{c}}}$ not interface
if $\mathsf{exe}^{\oplus}(p)$ then $\mathsf{exe}^{\oplus}(p,\pi)$
if $\mathsf{exe}(p)$ then $\mathsf{exe}(p,\pi)$

(PATH-ANY)

$$\frac{}{p;\Gamma;\Sigma;\Phi \vdash \pi : \mathtt{type\ Any}}$$

with
either $p(\pi) = \widehat{\mathcal{L}^{\mathfrak{c}}}$
or $\pi \in \{\mathtt{Any}, \mathtt{Void}, \mathtt{Library}\}$

(LIB-T)

$$\frac{}{p\circledast;\Gamma;\Sigma;\Phi \vdash \mathcal{L} : \mathtt{immutable\ Library}}$$

with
$p \vdash \mathcal{L} \rightarrow \widehat{\mathcal{L}^{\mathfrak{c}}}$
$p \vdash \widehat{\mathcal{L}^{\mathfrak{c}}} : \text{ok}$
if $\mathsf{exe}^{\oplus}(p)$ then $\widehat{\mathcal{L}^{\mathfrak{c}}} = \{\_\_\}^{\overline{\oplus},\_}$

(METH-UNKNOWN-T)

$$\frac{p;\Gamma;\Sigma;\Phi \vdash e_i : T_i \text{ for all } i \in 0..n}{p;\Gamma;\Sigma;\Phi \vdash e_0.m\mathtt{(}x_1{:}e_1 \ldots x_n{:}e_n\mathtt{)} : T}$$

with
either $\forall T.p;\Gamma;\Sigma;\Phi \vdash e_0 : T$
or not $\mathsf{exe}(p)$, not $\mathsf{exe}^{\oplus}(p)$ and
$T_0 = \_\ \pi{::}C$ where $p(\pi) \in \{\emptyset, \updownarrow\}$

(USING-T)

$$\frac{p;\Gamma;\Sigma;\Phi \vdash e_i : T'_i \leq T_i \text{ for all } i \in 0..n}{p;\Gamma;\Sigma;\Phi \vdash \mathtt{using}\ \pi\ \mathtt{check}\ .m\mathtt{(}\overline{x{:}e}\mathtt{)}\ e_0 : T_0}$$

with
$\overline{x{:}e} = x_1{:}e_1 \ldots x_n{:}e_n$
$\mathsf{plugin}(p,\pi,.m\mathtt{(}x_1\ldots x_n\mathtt{)}) = plg;\ T_0 T_1 \ldots T_n$

(T-BLOCK-COMPLETE/PARTIAL)

$$\frac{p;\Gamma_i[\mathcal{K},\Sigma];\Sigma;\Phi[\mathcal{K}] \vdash e_i : T'_i \leq \mathsf{toPartial}(T_i) \quad \forall i \in 1..n \qquad p;\Gamma_0;\Sigma\,\mathsf{FV}(e_1\ldots e_n) \cup x_1\ldots x_n;\Phi \vdash e : T \qquad p;\Gamma_0 \setminus \mathsf{dom}(\Gamma');\Sigma;\Phi \vdash \mathtt{catch}\ \varrho\ x\ \mathtt{(}\mathcal{O}_i\mathtt{)} : T \quad \forall i \in 1..k}{p;\Gamma;\Sigma;\Phi \vdash \mathtt{(}T_1\ x_1\ \mathtt{=}e_1 \ldots T_n\ x_n\ \mathtt{=}e_n\mathcal{K}e\mathtt{)} : T}$$

with
$\mathcal{K} = \mathtt{catch}\ \varrho\ x\ \mathtt{(}\mathcal{O}_1 \ldots \mathcal{O}_k\mathtt{)}$
$\Gamma' = x_1 : T_1 \ldots x_n : T_n$
$\Gamma_i(x) = \Gamma_j(x) = \mathtt{capsule}\ \_\_$ implies $i = j$
$\mathsf{exeOk}^{\oplus}(p,\Gamma_0)$
either
$\Gamma_i \subseteq \mathsf{complete}(\Gamma), \mathsf{toPh}(\mathsf{complete}(\Gamma'))\ \forall i \in 1..n$
$\Gamma_0 \subseteq \Gamma, \Gamma'$
or
$\Gamma_i \subseteq \Gamma, \mathsf{toPh}(\mathsf{complete}(\Gamma'))\ \forall i \in 1..n$
$\Gamma_0 \subseteq \Gamma, \mathsf{toPartial}(\Gamma')$

(T-VAR)

$$\frac{}{p;\Gamma;\overline{x}_0;\overline{x}_1\ldots\overline{x}_n;\_;\_ \vdash x : T}$$

with
$\mathsf{norm}_p(\Gamma(x)) = \mu\ \pi\alpha$
$x \notin \overline{x}_0$
$T = \begin{cases} \mathtt{lent}\ \pi\alpha & \text{if } x \in \overline{x}_1 \ldots \overline{x}_n \\ \mu\ \pi\alpha & \text{otherwise} \end{cases}$

(THROW-T)

$$\frac{p;\Gamma;\Sigma;\overline{T}\overline{\pi} \vdash e : \_ \leq \mu\ \pi}{p;\Gamma;\Sigma;\overline{T}\overline{\pi} \vdash \varrho\ e : T}$$

with
if $\varrho = \mathtt{return}$ then $\mu\ \pi \in \overline{T}$,
otherwise $\mu = \mathtt{immutable}$
if $\varrho = \mathtt{exception}$ then $\pi \in \overline{\pi}$

(T-VOID)

$$\frac{}{p;\Gamma;\Sigma;\Phi \vdash \mathtt{void} : \mathtt{capsule\ Void}}$$

(LOOP-T)

$$\frac{p;\Gamma;\Sigma;\Phi \vdash e : \mathtt{immutable\ Void}}{p;\Gamma,\_;\Sigma;\Phi \vdash \mathtt{loop}\ e : \mathtt{immutable\ Void}}$$

with
T not of form $\mathtt{capsule}\ \_\_\ \forall x{:}T \in \Gamma$

(T-UNLOCK)

$$\frac{p;\Gamma;\Sigma';\Phi \vdash e : \_ \leq \mu\ \pi\alpha}{p;\Gamma;\Sigma;\Phi \vdash e : \mathsf{mutTolent}(\mu\ \pi\alpha)}$$

with
$\Sigma = \overline{x};\overline{x}_0,\overline{x}_1\ldots\overline{x}_n;\overline{\overline{x}}$
$\Sigma' = \overline{x}';\overline{x}_1\ldots\overline{x}_n, \mathsf{dom}^{\mathsf{mut}}(\Gamma)\setminus\overline{x}_0\ldots\overline{x}_n;\overline{\overline{x}}$
if $\mu \in \{\mathtt{read}, \mathtt{lent}, \mathtt{mut}\}$ then $\overline{x}' = \overline{x}$
otherwise $\overline{x}' = \emptyset$

(T-METHCALL)

$$\frac{p;\Gamma_0;\Sigma_0;\Phi \vdash e_0 : \_ \leq \mu_0\ \pi_0 \qquad p;\Gamma_i;\Sigma_i;\Phi \vdash e_i : T'_i \leq T_i \quad \forall i \in 1..n}{p;\Gamma;\Sigma;\Phi \vdash e_0.m\mathtt{(}x_1{:}e_1\ldots x_n{:}e_n\mathtt{)} : T'}$$

with
$\mathsf{meth}_p(\pi_0.m, x_1\ldots x_n) =$
$\quad \mu_0\ \mathtt{method}\ T\ m\mathtt{(}T_1\ x_1 \ldots T_n\ x_n\mathtt{)}\ \mathtt{exception}\ \overline{\pi}\ \_$
$p \vdash \overline{\pi} \leq \Phi$
$\Gamma_i \subseteq \Gamma\ \forall i \in 0..n$
$\Gamma_i(x) = \Gamma_j(x) = \mathtt{capsule}\ \_\_$ implies $i = j$
$T' = \begin{cases} T \text{ if } T'_i = \mu_i\ \pi_i \text{for all } i \in 1..n \\ \mathsf{toPartial}(T) \text{ otherwise} \end{cases}$
$\Sigma = \overline{x};\overline{\overline{x}};\overline{x}_1\ldots\overline{x}_k$
$\overline{x}'_i = \mathsf{FV}(e_0\ldots e_n \setminus e_i)$
$\Sigma_i = \overline{x};\overline{\overline{x}};\overline{x}_1\overline{x}'_i\ldots\overline{x}_k\overline{x}'_i$

(T-MUTABLE)

$$\frac{p;\Gamma;\Sigma';\Phi \vdash e : \mathtt{mut}\ \pi\alpha}{p;\Gamma;\Sigma;\Phi \vdash e : \mathtt{capsule}\ \pi\alpha}$$

with
$\Sigma = \overline{x};\overline{x}_1\ldots\overline{x}_n;\overline{\overline{x}}$
$\Sigma' = \overline{x};\overline{x}_1\ldots\overline{x}_n, \mathsf{dom}^{\mathsf{mut}}(\Gamma)\setminus\overline{x}_1\ldots\overline{x}_n;\overline{\overline{x}}$

(T-READABLE)

$$\frac{p;\Gamma;\Sigma';\Phi \vdash e : \mu\ \pi\alpha}{p;\Gamma;\Sigma;\Phi \vdash e : \mathtt{immutable}\ \pi\alpha}$$

with
$\mu \in \{\mathtt{read}, \mathtt{lent}\}$
$\Sigma = \overline{x};\overline{x}_1\ldots\overline{x}_n;\overline{\overline{x}}$
$\overline{x}_0 = \mathsf{dom}^{\mathsf{mut}}(\Gamma)\setminus\overline{x}_1\ldots\overline{x}_n$
$\Sigma' = \overline{x}, \mathsf{dom}^{\mathsf{mut}\leq}(\Gamma);\overline{x}_0\ldots\overline{x}_n;\overline{\overline{x}}$

(K-T-ANY)

$$\frac{p;\Gamma;\Sigma;\Phi[\mathcal{K}_i] \vdash \mathcal{K}_i : T \quad \forall i \in 1..2}{p;\Gamma;\Sigma;\Phi \vdash \mathtt{catch}\ \varrho\ x\ \mathtt{(on}\ \mu\ \mathtt{Any}\ e\mathtt{)} : T}$$

with
either $\varrho = \mathtt{exception}, \mu' = \mu = \mathtt{immutable}$
or $\varrho = \mathtt{return}, \mathtt{type} \in \{\mu', \mu\}, \mu' \neq \mu$
$\mathcal{K}_1, \mathcal{K}_2 = \mathtt{catch}\ \varrho\ x\ \mathtt{(on}\ \mu\ \mathtt{Library}\ e\mathtt{)}, \mathtt{catch}\ \varrho\ x\ \mathtt{(on}\ \mu'\ \mathtt{Void}\ e\mathtt{)}$

(K-T)

$$\frac{p;x{:}T',\Gamma;\Sigma;\Phi \vdash e : T}{p;\Gamma;\Sigma;\Phi \vdash \mathtt{catch}\ \varrho\ x\ \mathtt{(on}\ T'\ e\mathtt{)} : T}$$

(DEC-F-T)

$$\frac{p;\Gamma;\Sigma;\Phi \vdash \mathtt{(}\overline{d}_0\mathcal{K}\ \mathtt{(}\overline{d}_1\mathcal{K}e_0\mathtt{))} : T}{p;\Gamma;\Sigma;\Phi \vdash \mathtt{(}\overline{d}_0\ \overline{d}_1\mathcal{K}e_0\mathtt{)} : T}$$

(DEC-K-PROM-T)

$$\frac{p;\Gamma;\Sigma;\Phi \vdash \mathtt{(}\overline{d}_0\ \overline{d}_1\mathtt{catch\ return}\ x\ \mathtt{(on}\ \mu'\ \pi\ x\mathtt{)}\ e_0\mathtt{)} : T}{p;\Gamma;\Sigma;\Phi \vdash \mathtt{(}\overline{d}_0\ \overline{d}_1\mathtt{catch\ return}\ x\ \mathtt{(on}\ \mu\ \pi\ x\mathtt{)}\ e_0\mathtt{)} : T}$$

with
$\mu = \mathtt{capsule}$ and $\mu' = \mathtt{mut}$
or $\mu = \mathtt{immutable}$ and $\mu' \in \{\mathtt{mut}, \mathtt{read}\}$

Reduction rules

(GARBAGE)

$$\mathcal{E}^p[\,(\overline{dv}\ \overline{d}\mathcal{K}e)\,] \longrightarrow_p \mathcal{E}^p[\,(\overline{d}\mathcal{K}e)\,]$$
with
$\overline{dv} \neq \emptyset$
$\mathsf{FV}(\,(\overline{d}e)\,) \cap \mathsf{dom}(\overline{dv}) = \emptyset$

(METHCALL)

$$\mathcal{E}^p[v.m\,(x_1{:}v_1 \ldots x_n{:}v_n)\,] \longrightarrow_p \mathcal{E}^p[\,(\mu_0\ \pi\ \texttt{this} = v\ T_1\ x_1 = v_1 \ldots T_n\ x_n = v_n\ e)\,]$$
with
$\mathsf{class}(\mathcal{E}^p, v) = \pi$
$\mathsf{meth}_p(\pi.m\,(x_1 \ldots x_n)\,) = \mu_0\ \texttt{method}\ T\ m(T_1\ x_1 \ldots T_n\ x_n)\ \texttt{exception}\ \_\ e$

(PRIMCALLREC)

$$\mathcal{E}^p[v.m\,(x_1{:}v_1 \ldots x_n{:}v_n)\,] \longrightarrow_p \mathcal{E}^p[\,(\mu\ \pi\ z = v\,z.m\,(x_1{:}v_1 \ldots x_n{:}v_n)\,)\,]$$
with
$\mathsf{class}(\mathcal{E}^p, v) = \pi$
$\mathsf{meth}_p(\pi.m\,(x_1 \ldots x_n)\,) = \mu\ \texttt{method}\ T\ m(T_1\ x_1 \ldots T_n\ x_n)\ \texttt{exception}\ \_$
v is a block

(PRIMCALLARG)

$$\mathcal{E}^p[v_0.m\,(\overline{x{:}a}x_i{:}v\overline{x{:}v})\,] \longrightarrow_p \mathcal{E}^p[\,(T'_i\ z = v_i\,v_0.m\,(\overline{x{:}a}x_i{:}z\overline{x{:}v})\,)\,]$$
with
v is a block
$\mathsf{class}(\mathcal{E}^p, v_0) = \pi$
$x_1{:}\_ \ldots x_n{:}\_ = \overline{x{:}a}x_i{:}v\overline{x{:}v}$
$\mathsf{meth}_p(\pi.m\,(x_1 \ldots x_n)\,) = \mu\ \texttt{method}\ T\ m(T_1\ x_1 \ldots T_n\ x_n)\ \texttt{exception}\ \_$
$T'_i = \mu_i\ \pi_i$ and $T_i = \mu_i\ \pi_i\_$

(FIELDAOBJ)

$$\mathcal{E}^p[x.m\,()\,] \longrightarrow_p \mathcal{E}^p[a_i]$$
with
$\mathsf{dec}(\mathcal{E}^p, x) = \_\ x = \pi.m'\,(x_1{:}a_1 \ldots x_n{:}a_n)$
$m = x_i$ or $m = \#x_i$
$\mathsf{meth}_p(\pi.m()) = \mu\ \texttt{method}\ T\ m()\ \texttt{exception}\ \_$

(FIELDABLOCK)

$$\mathcal{E}^p[x.m\,()\,] \longrightarrow_p \mathcal{E}^p[\,(\texttt{immutable}\ \pi'\ z = (\overline{dv}\,v.m\,()\,)\,z)\,]$$
with
$\mathsf{dec}(\mathcal{E}^p, x) = \texttt{immutable}\ \pi\_\ x = (\overline{dv}\,v)$
$\mathsf{meth}_p(\pi.m()) = \mu\ \texttt{method}\ \_\ \pi'\ m()\ \texttt{exception}\ \_$

(LOOP)

$$\mathcal{E}^p[\texttt{loop}\ e] \longrightarrow_p \mathcal{E}^p[e']$$
with
$e' = (\texttt{immutable Void}\ x{=}e\ \ \texttt{loop}\ e)$

(BLOCKELIM)

$$\mathcal{E}^p[\,(\overline{dv}'\mu\ \pi\alpha\ x = (\overline{dv}\,v)\ \overline{d}\mathcal{K}e)\,] \longrightarrow_p \mathcal{E}^p[\,(\overline{dv}'\overline{dv}\ \mu\ \pi\alpha\ x = v\ \overline{d}\mathcal{K}e)\,]$$
with
$\mu \geq \texttt{mut}$

(SUBST)

$$\mathcal{E}^p[\,(\overline{dv}'\ \mu\ \pi\ x = v\ \overline{d}\mathcal{K}e)\,] \longrightarrow_p \mathcal{E}^p[\,(\overline{dv}'\ \overline{d}\mathcal{K}e)\,[x = v]]$$
with
either $v = a$ or $\mu = \texttt{capsule}$

(NORMALDEC)

$$\mathcal{E}^p[\,(\overline{dv}'\ \mu\ \pi\alpha\ x = e'\ \overline{d}e)\,] \longrightarrow_p \mathcal{E}^p[\,(\overline{dv}'\ dv\ \overline{d}e)\,]$$
with
$\mathsf{class}(\mathcal{E}^p, e') = \pi'$
$dv = \mu\ \pi'\ x = e'$ [Marco: dv is important]
either $\alpha \neq \emptyset$ or $\pi' \neq \pi$

(NEW)

$$\mathcal{E}^p[\pi.m\,(x_1{:}a_1 \ldots x_n{:}a_n)\,] \longrightarrow_p \mathcal{E}^p[\,(\mu\ \pi\ z = \pi.m\,(x_1{:}a_1 \ldots x_n{:}a_n)\,z)\,]$$
with
$\mathsf{meth}_p(\pi.m\,(x_1 \ldots x_n)\,) = \texttt{type method}\ \mu\ \pi\ m(\_)\ \texttt{exception}\ \_$

(FIELDU)

$$\mathcal{E}^p[\mathcal{E}^p{}_1[x.m\,(\texttt{that}{:}a)]] \longrightarrow_p \mathcal{E}^p[\,(\overline{d}[x.m = a]\overline{\mathcal{K}}e)\,]$$
with
$\mathsf{move}_p(\mathcal{E}^p{}_1, \mathsf{FV}(a)) = \langle \mathcal{E}^p{}_2, \emptyset \rangle$
$\mathcal{E}^p{}_2[\texttt{void}] = (\overline{d}\mathcal{K}e)$
$\mathcal{E}^p{}_1 = (\overline{dv}\_\_), \overline{dv}(x) = \mu\ \pi\ x = \_$ and $\mu \in \{\texttt{mut}, \texttt{lent}\}$
$\mathsf{class}(\mathcal{E}^p{}_1, x) = \pi$
$\mathsf{meth}_p(\pi.m\,(\texttt{that})\,) = \_\texttt{method}\ \_\ m()\ \texttt{exception}\ \_$

(R-USING)

$$\mathcal{E}^p[e_0] \longrightarrow_p \mathcal{E}^p[e_1]$$
with
$e_0 = \texttt{using}\ \pi\ \texttt{check}\ m\ (\overline{x{:}v})\ e$
$e_1 = \mathsf{plugin}(p, \pi\ m\ (\overline{x{:}v})\ e)$

(R-USING-OUT)

$$\mathcal{E}^p[\texttt{using}\ \pi\ \texttt{check}\ m\ (\overline{x{:}v})\ e] \longrightarrow_p \mathcal{E}^p[e]$$
with
$\mathsf{plugin}(p, \pi\ m\ (\overline{x{:}v})\ e)$ undefined
either e is a v or $\mathsf{throws}_p(e) = \varrho\ v$

(R-META)

$$\frac{e_1 \longrightarrow_{p'} e_2 \qquad \mathcal{L} \xrightarrow[\max]{p} \mathcal{L}' \qquad \vdash p' : \mathrm{ok} \qquad p'\circledast;\emptyset;\emptyset;\emptyset \vdash e_1 : \texttt{Library}}{\mathcal{E}^p[\mathcal{L}] \longrightarrow_p \mathcal{E}^p[\mathcal{L}[C{:}e_2]]}$$
with
$\mathcal{L} = \{\_ <:\_ \overline{\mathcal{M}^{\mathsf{c}}}\ C{:}e^{\mathsf{c}}\_\}$
$p' = \mathcal{L}'^{\ominus}[C : \updownarrow]\ p$
$e_1 = \mathsf{norm}_{p'}(e^{\mathsf{c}})$

(R-META-METHOD)

$$\frac{\mathcal{L}_1 \longrightarrow_{p'} \mathcal{L}_2 \qquad \mathcal{L} \xrightarrow[\max]{p} \mathcal{L}'}{\mathcal{E}^p[\mathcal{L}] \longrightarrow_p \mathcal{E}^p[\mathcal{L}[mh\ \mathcal{E}^{\mathsf{c}}[\mathcal{L}_2]]]}$$
with
$\mathcal{L} = \{\_ <:\_ \overline{\mathcal{M}^{\mathsf{c}}}\ mh\ \mathcal{E}^{\mathsf{c}}[\mathcal{L}_1]\_\}$
$p' = \mathcal{L}'\ p$

(R-CAPTURE)

$$\mathcal{E}^p[e_1] \longrightarrow_p \mathcal{E}^p[\,(\overline{dv}\ \mu\ \pi\ z{=}v\ \ e)\,]$$
with
$e_1 = (\overline{dv}\ \overline{dv}'\ T'\ x{=}e'\ \overline{d}\texttt{catch}\ \varrho\ z\ \texttt{on}\ T\ e\ \overline{\mathcal{O}}e_0)$
$\mathsf{throws}_p(e') = \varrho\ v$
$\mu\ \pi = \mathsf{norm}_p(T)$
$p \vdash \mathsf{class}(\mathcal{E}^p[\,(\overline{dv}\square)\,], v) \leq \pi$

(R-ONMISS)

$$\mathcal{E}^p[e_1] \longrightarrow_p \mathcal{E}^p[\,(\overline{dv}\ T'\ x{=}\varrho\ v\texttt{catch}\ \varrho\ z\ \overline{\mathcal{O}}e_0)\,]$$
with
$e_1 = (\overline{dv}\ \overline{dv}'\ T'\ x{=}e'\ \overline{d}\texttt{catch}\ \varrho\ z\ \texttt{on}\ T\ e\ \overline{\mathcal{O}}e_0)$
$\mathsf{throws}_p(e') = \varrho'\ v$
$\mu\ \pi = \mathsf{norm}_p(T)$
either $\varrho \neq \varrho'$ or not $p \vdash \mathsf{class}(\mathcal{E}^p[\,(\overline{dv}\square)\,], v) \leq \pi$

(R-KOUT)

$$\mathcal{E}^p[\,(\overline{dv}\mathcal{K}e)\,] \longrightarrow_p \mathcal{E}^p[\,(\overline{dv}e)\,]$$

Desugering and compilation process

With $\mathcal{L}$ as a source in the sugared language $^{\mathcal{W}}[\![\mathcal{L}]\!]_{\emptyset;\texttt{immutable Void};\emptyset}$ is the corresponding desugared term. An **execution process** is a sequence $\mathcal{L}_0 \ldots \mathcal{L}_n$ such that $\emptyset|\mathcal{L}_0 \rightarrow \emptyset|\ldots \rightarrow \emptyset|\mathcal{L}_n$.
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$(plg, p, \sigma, \overline{d}, \overline{v}, e)$). **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.
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 formedness restriction of the core syntax applies here. Moreover in a $\mathcal{B}$ all $\overline{d}_i$ 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` $\overline{x}\,\overline{\mathcal{I}}\,\overline{d}$ `(`$\overline{\mathcal{O}^w}\overline{\mathcal{B}}$`)` is well formed if the number of types $T_1 \ldots T_n$ in each `on` is the same of the sum of the cardinalities of $\overline{x}$, $\overline{\mathcal{I}}$ and $\overline{d}$. In a $\mathcal{O}^w$ body, variables whose type have been made more specific can still beeing updated using the more general type, thus a well formed $\mathcal{O}^w$ body can not read a variable after updating it. In a `with`, variables introduces in the $\overline{\mathcal{I}}$ can be updated only if they are declared **`var`**.
There must not be any whitespace preceding the symbol '`"`' in string expressions or π in number expression.
For all blocks of form `{`$\overline{d}_1\mathcal{K}_1 \ldots \overline{d}_n\mathcal{K}_n$`}`, terminating(`{`$\overline{d}_1\mathcal{K}_1 \ldots \overline{d}_n\mathcal{K}_n$`}`) holds. Method names in method calls (using the dot) must be of form x or `#`x.
The **`return`** keyword can not be used inside any **`if`**,**`case`** or **`while`** condition or inside the expression of a x**`in`**e.

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.

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$(\varrho\ e)$ = terminating(`loop` e) = true
- terminating(`if` _ e `else` $\mathcal{B}$) = terminating(e) and terminating$(\mathcal{B})$
- terminating(`(`$\overline{d}_1\mathcal{K}_1 \ldots \overline{d}_n\mathcal{K}_n\, e$`)`) = terminating$(\mathcal{K}_1)$ and … and terminating$(\mathcal{K}_n)$ and terminating(e)
- terminating(`{`$\overline{d}_1\mathcal{K}_1 \ldots \overline{d}_n\mathcal{K}_n$`}`) = terminating$(\mathcal{K}_1)$ and … and terminating$(\mathcal{K}_n)$ and terminating$(\overline{d}_n)$
- terminating(`catch` $\varrho\ x$ `(`$doc\overline{\mathcal{O}}$`default` e`)`) = terminating(`catch` $\varrho\ x$ `(`$doc\overline{\mathcal{O}}$) and terminating$(e)$
- terminating(`catch`_ `(on`_e_1…`on`_e_n`)`) = terminating(e_1) and … and terminating(e_n)
- terminating(`if` _ e) = terminating$(\mathcal{W}\overline{x}\,\overline{d}\,e)$ = terminating$(\overline{d}\,e)$ = terminating(e)
- terminating(_) = false otherwise

Atomic Language Terms

ϱ	::= `return` | `error` | `exception`	results
μ	::= `type` | `mut` | `read` | `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	::= x | `#`x | $\emptyset$ | $unOp$ | $eqOp$ | $binOp$	method names
doc	::= $strLine_1 \ldots strLine_n$<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
$digit$	::= `0` | `1` | `2` | `3` | `4` | `5` | `6` | `7` | `8` | `9`	
$digit+$	::= $digit$ | `.` $digit$ | `-`$digit$	
num	::= $\overline{dataOp}\,digit\,\overline{digit+}$	
$unOp$	::= `!` | `~`	
$eqOp$	::= `+=` | `-=` | `*=` | `/=` | `&=` | `|=` | `>>=` | `<<=` | `++=` | `**=` | `:=`	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` | $num\ \overline{num}\,e$ | e`"`$string$`"` | $\varrho\ e$ | $x\ eqOp\ e$ | $unOp\ e$	expression
	| $\mathcal{B}$ | $e_1\,binOp\,e_2$ | e`(`$doc\ ps$`)` | `if` $e\,\mathcal{B}\,\overline{\texttt{else}\ e'}$ | `while` $e\,\mathcal{B}$ | $\mathcal{W}$ | $e.m$`(`$doc\ ps$`)`	
	| e`[`$doc\ ps_1$`;`$doc_1 \ldots ps_n$`;`doc_n`]` | e`[`$doc\ \mathcal{W}$`]` | $e\ doc$ | `loop` e | `...`	
	| `using` π `check` m`(`$doc\ ps$`)` e	
$\mathcal{B}$	::= `(`$doc\ \overline{d}_1\mathcal{K}_1 \ldots \overline{d}_n\mathcal{K}_n\ e$`)` | `{`$doc\ \overline{d}_1\mathcal{K}_1 \ldots \overline{d}_n\mathcal{K}_n$`}`	expr-block
$\mathcal{K}$	::= `catch` $\varrho\ x$ `(`$doc\overline{\mathcal{O}}\,\overline{\texttt{default}\ e}$`)`	catch-match
d	::= $\overline{\texttt{var}}\ \overline{T}\ x$`=`$e$ | e<$e \neq x$> | $\overline{C}$`:`e	statement
$\mathcal{W}$	::= `with` $\overline{x}\,\overline{\mathcal{I}}\,\overline{d}$ `(`$\overline{\mathcal{O}^w}\,\overline{\texttt{default}\ e}$`)` | `with` $\overline{\mathcal{I}}\,\mathcal{B}$	with
$\mathcal{O}^w$	::= `on` $\overline{T}\,e$ | `on` $\overline{T}$ `case` $e\,\mathcal{B}$ | `case` $e\,\mathcal{B}$	type-case
$\mathcal{H}$	::= m`(`$\overline{\mathcal{F}}$`)` | `interface` | `trait`	lib. node h.
π	::= C`::`$\overline{C}$|`Outer`^{n}`::`$\overline{C}$|`Any`|`Void`|`Library`	node path
$\mathcal{L}$	::= `{` $doc\ \mathcal{H}$ `<:`$\overline{\pi}\,\overline{\mathcal{M}}$ `}` | `{` `reuse` $url\ EOL\ \overline{\mathcal{M}}$ `}`	library literal
ps	::= $\overline{e}\ \overline{x\texttt{:}e}$	parameters
$\mathcal{I}$	::= $\overline{\texttt{var}}\ x$ `in` e	iterator decl.
$\mathcal{O}$	::= `on` $T\ e$ | `on` T `case` $e\,\mathcal{B}$	signal handler

Definition: guessType$_\Gamma(e)$ note:guessType$_\Gamma$(`{`$\overline{d}_1\mathcal{K}_1 \ldots \overline{d}_n\mathcal{K}_n$`}`) is correctly undefined

- guessType$_\Gamma(\mathcal{L})$ = `immutable Library`, guessType$_\Gamma(x) = \Gamma(x)$, guessType$_\Gamma(\pi)$ = `type` π
- guessType$_\Gamma$(`void`) = guessType$_\Gamma$(`loop` e) = guessType$_\Gamma(x\ eqOp\ e)$ = guessType$_\Gamma(\varrho\ e)$ = guessType$_\Gamma$(`if` $e\,\mathcal{B}_1\,\overline{\texttt{else}\ \mathcal{B}_2}$) = guessType$_\Gamma$(`while` $e\,\mathcal{B}$) = guessType$_\Gamma(\mathcal{W}\,\overline{\mathcal{B}})$ = `immutable Void`
- guessType$_\Gamma(num\,e)$ = guessType$_\Gamma$(e`.#numberParser({trait})`)
- guessType$_\Gamma$(e`"_"`) = guessType$_\Gamma$(e`.#stringParser({trait})`)
- guessType$_\Gamma(unOp\ e)$ = guessType$_\Gamma$(e.[['$unOp$']]`()`)
- guessType$_\Gamma(e_1\,binOp\,e_2)$ = guessType$_\Gamma$(e_1.[['$binOp$']]`(`e_2`)`)
- guessType$_\Gamma$(e`(`ps`)`) = guessType$_\Gamma$(e`.#apply(`ps`)`)
- guessType$_\Gamma$($e.m$`(`ps`)`) = $T\,m$(xsOf(ps)) iff guessType$_\Gamma(e) = \pi\overline{mx} = T$
- guessType$_\Gamma$($e.m$`(`ps`)`) = $\pi.m$(xsOf(ps)) iff guessType$_\Gamma(e) = \mu\,\pi\,\hat{}$
- guessType$_\Gamma$(`(`$\overline{d}_1\mathcal{K}_1 \ldots \overline{d}_n\mathcal{K}_n\,e$`)`) = guessType$_{\Gamma'}(e)$ with Γ' = guessType$_\Gamma(\overline{d}_1 \ldots \overline{d}_n)$ and guessType$_\Gamma() = \Gamma$ guessType$_\Gamma(e\,\overline{d})$ = guessType$_\Gamma(C\texttt{:}e\,\overline{d})$ = guessType$_\Gamma(\overline{d})$
  - guessType$_\Gamma$($\overline{\texttt{var}}\ T\ x$`=`$e\,\overline{d}$) = guessType$_{\Gamma, x\mapsto T}(\overline{d})$
  - guessType$_\Gamma$($\overline{\texttt{var}}\ x$`=`$e\,\overline{d}$) = guessType$_{\Gamma, x\mapsto \text{guessType}_\Gamma(e)}(\overline{d})$
- guessType$_\Gamma$(e`[`ps_1`;`$\ldots ps_n$`;]`) = guessType$_\Gamma$(e`.#apply().#add(`ps_1`)`…`.#add(`ps_n`)`)
- guessType$_\Gamma$(e`[`$\mathcal{W}\overline{\mathcal{B}}$`]`) = guessType$_\Gamma$($e$`.#apply()`)
- guessType$_\Gamma(e\ doc)$ = guessType$_\Gamma$(`using` π `check` m`(`$doc\ ps$`)` e) = guessType$_\Gamma(e)$

Definition: c-f-type($\mu\ m$`(`$\mathcal{F}_1 \ldots \mathcal{F}_n$`)`) = $\overline{h}$

(1) `type method` μ `Outer`$_0$ m(toPh$(T_1\,x_1 \ldots T_n\,x_n)$`)` `exception` $\emptyset \in$ c-f-type($\mu\ m$`(`$\overline{\texttt{var}}_1\ T_1\,x_1 \ldots \overline{\texttt{var}}_n\ T_n\,x_n$`)`

(2a) `mut method immutable Void` x`(` $\mu\,\pi$ `that)` `exception` $\emptyset \in$ c-f-type(_ m`(`$\overline{\mathcal{F}}_1$ `var` $\mu\pi\ x\ \overline{\mathcal{F}}_2$`)`)

(2b) `mut method immutable Void` x`(` $\pi\overline{mx}$ `that)` `exception` $\emptyset \in$ c-f-type(_ m`(`$\overline{\mathcal{F}}_1$ `var` $\pi\overline{mx}\ x\ \overline{\mathcal{F}}_2$`)`)

(3) `mut method` T `#`x`()` `exception` $\emptyset \in$ c-f-type(_ m`(`$\overline{\mathcal{F}}_1\ \overline{\texttt{var}}\ T\ x\ \overline{\mathcal{F}}_2$`)`)

(4) `read method` mut&LentToRead$(\mu\ \pi)$ x`()` `exception` $\emptyset \in$ c-f-type(_ m`(`$\overline{\mathcal{F}}_1\ \overline{\texttt{var}}\ \mu\ \pi\ x\ \overline{\mathcal{F}}_2$`)`)

Definition: $[\![e]\!]_\Theta$ simple cases, where $\Theta ::= \Gamma;\ T;\ \overline{\mathcal{L}}$

(dw) $[\![\mathtt{\{reuse}\ url\ \overline{\mathcal{M}}\mathtt{\}}]\!]_{\_;\_;\overline{\mathcal{L}}} = \mathtt{\{}\text{downloadFromWeb}(url)\ \overline{\mathcal{M}}'\mathtt{\}}$ iff $[\![\mathtt{\{}\overline{\mathcal{M}}\mathtt{\}}]\!]_{\emptyset;\_;\mathtt{\{}\overline{\mathcal{M}}\mathtt{\}}} = \mathtt{\{}\overline{\mathcal{M}}'\mathtt{\}}$

otherwise $[\![\mathtt{\{}\mathcal{H}\ \mathtt{<:}\overline{\pi}\ \overline{\mathcal{M}}\mathtt{\}}]\!]_{\overline{\mathcal{L}}} = \mathtt{\{}{}^{\mathcal{M}}[\![\mathcal{H}]\!]_\Theta\mathtt{<:}[\![\overline{\pi}]\!]_\Theta{}^{\mathcal{M}}[\![\overline{\mathcal{M}}]\!]_\Theta\mathtt{\}}$

with $\Theta = \emptyset;\ \mathtt{immutable\ Void};\ \mathtt{\{}\mathcal{H}\ \mathtt{<:}\overline{\pi}\ \overline{\mathcal{M}}\mathtt{\}},\overline{\mathcal{L}}$

(cons) $[\![\mathtt{\{}\mu\ m(\overline{\mathcal{F}})\ \overline{\mathcal{M}}\mathtt{\}}]\!]_\Theta = [\![\mathtt{\{}\text{c-f-type}(\mu\ m(\overline{\mathcal{F}})\ )\overline{\mathcal{M}}\mathtt{\}}]\!]_\Theta$

(lt) $[\![nume]\!]_\Theta = [\![e\mathtt{.\#numberParser(\{'\,@String}EOL'\ [\![\text{'}num\text{'}]\!]EOL\mathtt{\})}]\!]_\Theta$

$[\![e\mathtt{"}char\mathtt{"}]\!]_\Theta = [\![e\mathtt{.\#stringParser(\{'\,@String}EOL'\ [\![\text{'}char\text{'}]\!]EOL\mathtt{\})}]\!]_\Theta$

$[\![e\mathtt{"}EOLcharEOL\mathtt{"}]\!]_\Theta = [\![e\mathtt{.\#stringParser(\{'\,@String}EOL'\ [\![\text{'}charEOL\text{'}]\!]EOL\mathtt{\})}]\!]_\Theta$

(cf) $[\![\mathtt{while}\ e\ \mathcal{B}]\!]_\Theta = [\![\ \mathtt{(loop\ (}e\mathtt{.\#checkTrue()}\ \mathcal{B}\mathtt{)\ catch\ exception\ (on\ Void\ void)\ void)}\ ]\!]_\Theta$

$[\![\mathtt{if}\ a\ \mathcal{B}_1\ \mathtt{else}\ \mathcal{B}_2]\!]_\Theta = [\![\ \mathtt{(}a\mathtt{.\#checkTrue()\ catch\ exception\ (onVoid}\ \mathcal{B}_2\mathtt{)}\ \mathcal{B}_1\ \mathtt{void)}\ ]\!]_\Theta$

$[\![\mathtt{if}\ e\ \mathcal{B}_1\ \mathtt{else}\ \mathcal{B}_2]\!]_\Theta = [\![\ \mathtt{(}\ y\mathtt{=}e\ \mathtt{if}\ y\ \mathcal{B}_1\ \mathtt{else}\ \mathcal{B}_2\mathtt{)}\ ]\!]_\Theta$ with y fresh and $e \neq a$

$[\![\ \mathtt{(}\overline{d}_1\mathcal{K}_1\ldots\overline{d}_n\mathcal{K}_n\ e\mathtt{)}\ ]\!]_\Theta = [\![\ \mathtt{(}\overline{d}_1\mathcal{K}_1\ \mathtt{(}\ldots\mathtt{(}\overline{d}_n\mathcal{K}_n\ e\mathtt{)}\ldots\mathtt{)\,)}\ ]\!]_\Theta$

$[\![\mathtt{void}]\!]_\Theta = \mathtt{void}$ $\quad [\![\pi]\!]_{\_,\overline{\mathcal{L}}} = {}^{\pi}[\![\pi]\!]_{\overline{\mathcal{L}}}$ $\quad [\![e\ doc]\!]_\Theta = [\![\ \mathtt{(}doc\ e\mathtt{)}\ ]\!]_\Theta$

$[\![x]\!]_\_ = x$ $\quad [\![x\mathtt{:=}e]\!]_\Theta = x\mathtt{.inner(}[\![e]\!]_\Theta\mathtt{)}$ $\quad [\![\varrho\ e]\!]_\Theta = \varrho\ [\![e]\!]_\Theta$

$[\![\mathtt{loop}\ e]\!]_{\Gamma,T\,\overline{\mathcal{L}}} = \mathtt{loop}\ [\![e]\!]_{\Gamma,\mathtt{immutable\ Void}\overline{\mathcal{L}}}$

$[\![e\mathtt{.}m\mathtt{(}ps\mathtt{)}\ ]\!]_{\Gamma,T\,\overline{\mathcal{L}}} = [\![e]\!]_{\Gamma,T\,\overline{\mathcal{L}}}\mathtt{.}m\mathtt{(}[\![ps]\!]_{\Gamma,\text{guessType}_\Gamma(e)::m,\overline{\mathcal{L}}}\mathtt{)}$

$[\![\mathtt{using}\ \pi\ \mathtt{check\ .}m\mathtt{(}ps\mathtt{)}\ \ e]\!]_{\Gamma,T,\overline{\mathcal{L}}} = \mathtt{using}\ [\![\pi]\!]_\Theta\ \mathtt{check\ .}m\mathtt{(}[\![ps]\!]_{\Gamma,\mathtt{immutable\ Void}\overline{\mathcal{L}}}\mathtt{)}\ \ [\![e]\!]_{\Gamma,T,\overline{\mathcal{L}}}$

(rt) $[\![\mathtt{\{}\overline{d}_1\mathcal{K}_1\ldots\overline{d}_n\mathcal{K}_n\mathtt{\}}]\!]_{\Gamma,T,\overline{\mathcal{L}}} = [\![\ \mathtt{(\,(}\overline{d}_1\mathcal{K}_1\ldots\overline{d}_n\mathcal{K}_n\mathtt{void)\ catch\ return}\ y\ \ \mathtt{(on}\ T\ y\mathtt{)\ error\ void)}\ ]\!]_{\Gamma,T,\overline{\mathcal{L}}}$

(vd1) $[\![e_0]\!]_\Theta = [\![\ \mathtt{(}C\mathtt{:\{mut\ (var}\ T\ \mathtt{inner)\ \}}$

$\overline{d}_1\ T\ x\mathtt{=}e\,\overline{d}_2\,d'(\overline{d}_3\ \overline{\mathcal{K}}\ e_1)[x\,eqOp \coloneqq z\,eqOp][x \coloneqq z\mathtt{.\#inner()}]\mathtt{)}\ ]\!]_\Theta$

with $e_0 = \mathtt{(}\overline{d}_1\mathtt{var}\ T\ x\mathtt{=}e\,\overline{d}_2\,\overline{d}_3\overline{\mathcal{K}}e_1\mathtt{)}$ $\qquad d' = \mathtt{mut\ Outer_0::}C\ z\mathtt{=Outer_0::}C\ \mathtt{(\ inner:}x\mathtt{)}$

not $x\mathtt{:=}\_$ inside $\overline{d}_2$ and either $\overline{d}_3 = \emptyset$ or $\overline{d}_3 = d\_$ and $x\mathtt{:=}\_$ inside d

$[\![\mathtt{(}\overline{d}\mathtt{var}\ x\mathtt{=}e\,\overline{d}'\overline{\mathcal{K}}e_1\mathtt{)}]\!]_{\Gamma;T';\overline{\mathcal{L}}} = [\![\mathtt{(}\overline{d}\mathtt{var}\ T\ x\mathtt{=}e\,\overline{d}'\overline{\mathcal{K}}e_1\mathtt{)}]\!]_{\Gamma;T';\overline{\mathcal{L}}}$ with $T = \text{guessType}_{\Gamma,\Gamma\text{of}\,\overline{d}}(e)$

$[\![\ \mathtt{(}\overline{d}_1\,e\ \overline{d}_2\overline{\mathcal{K}}e_1\mathtt{)}\ ]\!]_\Theta = [\![\ \mathtt{(}\overline{d}_1\ \mathtt{immutable\ void}\ x\mathtt{=}e\,\overline{d}_2\overline{\mathcal{K}}e_1\mathtt{)}\ ]\!]_\Theta$

$[\![\ \mathtt{(}d_1\ldots d_n\overline{\mathcal{K}}e\mathtt{)}\ ]\!]_{\Gamma;T;\overline{\mathcal{L}}} = \mathtt{(}{}^{d}[\![d_1]\!]_\Theta\ldots{}^{d}[\![d_n]\!]_\Theta{}^{\mathcal{K}}[\![\overline{\mathcal{K}}]\!]_{\Gamma;T;\overline{\mathcal{L}}}[\![e]\!]_\Theta\mathtt{)}$

with $\Theta = \Gamma, \Gamma\text{of}(d_1\ldots d_n);\ T;\ \overline{\mathcal{L}}$ $\qquad {}^{d}[\![C\mathtt{:}e]\!]_\_ = C\mathtt{:}e$ $\qquad {}^{d}[\![\ T\ x\mathtt{=}e]\!]_{\Gamma\,\_\,\overline{\mathcal{L}}} = \ T\ x\mathtt{=}[\![e]\!]_{\Gamma\,T\,\overline{\mathcal{L}}}$

${}^{\mathcal{K}}[\![\mathtt{catch}\ \varrho\ \overline{\mathcal{O}}\mathtt{default}\ e]\!]_\Theta = {}^{\mathcal{K}}[\![\mathtt{catch}\ \varrho\ x\ \overline{\mathcal{O}}\mathtt{default}\ e]\!]_\Theta$

${}^{\mathcal{K}}[\![\mathtt{catch}\ \varrho\ x\ \overline{\mathcal{O}}]\!]_\Theta = [\![\mathtt{catch}\ \varrho\ x\ \mathtt{on\ Any\ (with}\ x\ \ \mathtt{(}\overline{\mathcal{O}}\mathtt{default}\ \varrho\ x\mathtt{)\ )}\ ]\!]_\Theta$ iff $\mathtt{on}\ T\ \mathtt{if}\ e\ \mathcal{B} \in \overline{\mathcal{O}}$

${}^{\mathcal{K}}[\![\mathtt{catch}\ \varrho\ x\ \overline{\mathcal{O}}\ \mathtt{default}\ e]\!]_\Theta = [\![\mathtt{catch}\ \varrho\ x\ \mathtt{on\ Any\ (with}\ x\ \ \mathtt{(}\overline{\mathcal{O}}\mathtt{default}\ \ e\mathtt{)\ )}\ ]\!]_\Theta$ iff $\mathtt{on}\ T\ \mathtt{if}\ e\ \mathcal{B} \in \overline{\mathcal{O}}$

otherwise ${}^{\mathcal{K}}[\![\mathtt{catch}\ \varrho\ x\ \mathcal{O}_1\ldots\mathcal{O}_n]\!]_\Theta = \mathtt{catch}\ \varrho\ x\ {}^{\mathcal{K}}[\![\mathcal{O}_1]\!]_{x,\Theta}\ldots{}^{\mathcal{K}}[\![\mathcal{O}_n]\!]_{x,\Theta}$ and

${}^{\mathcal{K}}[\![\mathtt{catch}\ \varrho\ x\ \mathcal{O}_1\ldots\mathcal{O}_n\mathtt{default}\ e]\!]_\Theta = \mathtt{catch}\ \varrho\ x\ {}^{\mathcal{K}}[\![\mathcal{O}_1]\!]_{x,\Theta}\ldots{}^{\mathcal{K}}[\![\mathcal{O}_n]\!]_{x,\Theta}\mathtt{default}\ [\![e]\!]_{x:\mathtt{immutable\ Any},\Theta}$

${}^{\mathcal{K}}[\![\mathtt{on}\ T\ \mathcal{B}]\!]_{x,\Theta} = \mathtt{on}\ [\![T]\!]_\Theta\ [\![\mathcal{B}]\!]_{x:T,\Theta}$(note:$\mathtt{on}\ T\ \mathtt{case}\,e\ \mathcal{B}$ is managed in the catch)

Definition: $[\![e]\!]$ case collections initialization and operators

(init) $[\![e\ \mathtt{[}\ ps_1\mathtt{;}\ \ldots\mathtt{;}\ ps_n\mathtt{;}\ \mathtt{]}\ ]\!]_\Theta = [\![e\mathtt{.\#begin()\ .\#add(}ps_1\mathtt{)}\ \ldots\mathtt{.\#add(}ps_n\mathtt{)\ .\#end()}\ ]\!]_\Theta$

(op) $[\![e_1\ binOp\ e_2]\!]_{\overline{\mathcal{L}}} = [\![e_1\mathtt{.}[\![\text{'}binOp\text{'}]\!]\mathtt{(}e_2\mathtt{)}\ ]\!]_{\overline{\mathcal{L}}}$

$[\![x\ eqOp\ e]\!]_\Theta = [\![x\mathtt{:=}x\mathtt{.\#inner()\,.}[\![\text{'}eqOp\text{'}]\!]\mathtt{(}e\mathtt{)}\ ]\!]_\Theta$

$[\![unOp\ e]\!]_{\overline{\mathcal{L}}} = [\![e]\!]_{\overline{\mathcal{L}}}\mathtt{.}[\![\text{'}unOp\text{'}]\!]\mathtt{()}$

$[\![e\ \mathtt{(}ps\mathtt{)}\ ]\!]_{\overline{\mathcal{L}}} = [\![e\mathtt{.\#apply(}ps\mathtt{)}\ ]\!]_{\overline{\mathcal{L}}}$

Definition: $[\![doc]\!]_p$

$[\![doc]\!]_p$ replaces all substrings of the form $\mathtt{@}\pi$ and $\mathtt{@\ (}e\mathtt{)}$ with $\mathtt{@}[\![\pi]\!]_p$ and $\mathtt{@\ (}[\![e]\!]_p\mathtt{)}$.
This applies to all documentations excluding the one in multi-line string literals.

Definition: ${}^{\mathcal{W}}[\![e]\!] = e$

${}^{\mathcal{W}}[\![e]\!]$ propagate on the structure, and

(a) ${}^{\mathcal{W}}[\![\mathtt{with}\ \overline{x}\ \overline{\mathcal{I}}\ \overline{d}\ \ \mathtt{(}\overline{\mathcal{O}^w}\mathtt{)}\ ]\!] = {}^{\mathcal{W}}[\![\mathtt{with}\ \overline{x}\ \overline{\mathcal{I}}\ \overline{d}\ \ \mathtt{(}\overline{\mathcal{O}^w}\mathtt{default\ void)}\ ]\!]$

(b) ${}^{\mathcal{W}}[\![\mathtt{with}\ \overline{x}\ \overline{d}\ \ \mathtt{(}\overline{\mathcal{O}^w}\ \mathtt{default}\ e_2\mathtt{)}\ ]\!] = {}^{\mathcal{W}}[\![\ \mathtt{(}\overline{d}\ \mathtt{with}x_1\ldots x_n\ \mathtt{(}\overline{\mathcal{O}^w}\ \mathtt{default}\ e_2\mathtt{)\ )}\ ]\!]$

with $\mathtt{with}\ \overline{x}\ \overline{d} = \mathtt{with}x_1\_\ldots x_n\_$

(c) ${}^{\mathcal{W}}[\![\mathtt{with}\overline{x}\ \mathtt{(}\overline{\mathcal{O}^w}\mathtt{default}\ e\mathtt{)}\ ]\!] = {}^{\mathcal{W}}[\![{}^{\mathcal{W}}[\![\overline{\mathcal{O}^w}\mathtt{default}\ e]\!]_{\overline{x}}]\!]$

(ca) ${}^{\mathcal{W}}[\![\mathtt{on}\ T_1\ldots T_n\overline{\mathtt{case}\ e_0}\ e_1\ \overline{\mathcal{O}^w}\ \mathtt{default}\ e]\!]_{x_1\ldots x_n} = \mathtt{(}\overline{e}\ \text{cast}^{T_1}(y_1 \leftarrow x_1)\ldots\text{cast}^{T_n}(y_n \leftarrow x_n)$

$\mathtt{catch\ exception\ Void}\ {}^{\mathcal{W}}[\![\overline{\mathcal{O}^w}\mathtt{default}\ e]\!]_{x_1\ldots x_n}(e_1[x_1\,T_1 \coloneqq y_1]\ldots[x_n\,T_n \coloneqq y_n])\ \mathtt{void)}$

with $y_1\ldots y_n$ fresh and either $\overline{\mathtt{case}\ e_0} = \overline{e} = \emptyset$

or $\overline{\mathtt{case}\ e_0} = \mathtt{case}\ e_0$ and $\overline{e} = \mathtt{with}\ x_1\ldots x_n\ \mathtt{(on}\ T_1\ldots T_n\ \mathtt{(if}\ e_0\ \ \mathtt{(void)\ else\ \ (exception\ void)\ )}$

(cb) ${}^{\mathcal{W}}[\![\mathtt{case}\ e_0\ \ e_1\ \ \overline{\mathcal{O}}\ \mathtt{default}\ e]\!]_{x_1\ldots x_n} = \mathtt{if}\ e_0\ \ e_1\ \mathtt{else\ (}{}^{\mathcal{W}}[\![\overline{\mathcal{O}}\ \mathtt{default}\ e]\!]_{x_1\ldots x_n}\mathtt{)}$

(cc) ${}^{\mathcal{W}}[\![\mathtt{default}\ e]\!]_\_ = e$

(d) ${}^{\mathcal{W}}[\![\mathtt{with}\ \overline{x}\ \overline{\mathcal{I}}\ \overline{d}\ \ \mathtt{(}\overline{\mathcal{O}^w}\ \mathtt{default}\ e_2\mathtt{)}\ ]\!] = {}^{\mathcal{W}}[\![\mathtt{with}\ \overline{\mathcal{I}}\ \ \mathtt{(}\overline{d}\ \mathtt{with}x_1\ldots x_n\ \mathtt{(}\overline{\mathcal{O}^w}\ \mathtt{default}\ e_2\mathtt{)\ )}\ ]\!]$

with $\mathtt{with}\ \overline{x}\ \overline{\mathcal{I}}\ \overline{d} = \mathtt{with}x_1\_\ldots x_n\_$

(e) ${}^{\mathcal{W}}[\![\mathtt{with}\ \overline{\mathcal{I}}\ \mathcal{B}]\!] = {}^{\mathcal{W}}[\![\text{declareIts}(\overline{\mathcal{I}},\mathtt{(}$

$\mathtt{loop(}\ d_1\mathcal{K}_1\ldots d_n\mathcal{K}_n\mathcal{B}[x_1\mathtt{:=}x_1\mathtt{.\#inner()}]\ldots[x_n\mathtt{:=}x_n\mathtt{.\#inner()}]\mathtt{)}$

$\mathtt{catch\ exception\ (on\ Void\ void)\ void)}\ )]\!]$

with $\overline{\mathcal{I}} = \_\ x_1\ \mathtt{in}\ \_\ldots\_\ x_n\ \mathtt{in}\ \_,$ $\quad d_i\mathcal{K}_i = \text{next}_i(x_1\ldots x_n)$

(initW) ${}^{\mathcal{W}}[\![e\ \mathtt{[with}\ \overline{x}\ \overline{\mathcal{I}}\ \overline{d}\ \mathtt{(}\mathcal{O}^w{}_1\ldots\mathcal{O}^w{}_n\overline{\mathtt{default}\ e}\mathtt{)\ ]}\ ]\!] =$

${}^{\mathcal{W}}[\![\ \mathtt{(var}\ \ x\mathtt{=}e\mathtt{.\#begin()\ with}\ \overline{x}\ \overline{\mathcal{I}}\ \overline{d}\ \ \mathtt{(}\mathcal{O}^{w\prime}{}_1\ldots\mathcal{O}^{w\prime}{}_n\overline{\mathtt{default}\ e'}\mathtt{)}\ x\mathtt{.\#end()\ )}\ ]\!]$

with $\mathcal{O}^w{}_i = \mathtt{on}\ \overline{T}\ \mathtt{if}\,e\,e, \mathcal{O}^{w\prime}{}_i = \mathtt{on}\ \overline{T}\ \mathtt{if}\,e\ x\mathtt{:=}x\mathtt{.\#add(}e\mathtt{)}$ and either $\overline{\mathtt{default}\ e} = \overline{\mathtt{default}\ e'} = \emptyset$

or $\overline{\mathtt{default}\ e} = \mathtt{default}\ e$ and $\overline{\mathtt{default}\ e'} = \mathtt{default}\ x\mathtt{:=}x\mathtt{.\#add(}e\mathtt{)}$

Definition: $e[x \coloneqq y]$

$e_0[x \coloneqq e_1]$ propagate on the structure, and

$(x\,eqOp\,e_0)[x \coloneqq e] = x\,eqOp(e_0[x \coloneqq e])$

$x[x \coloneqq e] = e$

Definition: $e[x\,T \coloneqq y]$

$e[x\,\mu\ \mathtt{Any} \coloneqq y] = e$, otherwise $e[x\,T \coloneqq y] = e[x \coloneqq y]$

Definition: $[\![e]\!]_p$ auxiliary definitions

${}^{ps}[\![e_0\ \overline{x\mathtt{:}e}]\!]_{\Gamma,T::m,\overline{\mathcal{L}}} = {}^{ps}[\![\ \mathtt{that:}e_0\ \overline{x\mathtt{:}e}]\!]_{\Gamma,T::m,\overline{\mathcal{L}}}$

${}^{ps}[\![x_1\mathtt{:}e_1\ldots x_n\mathtt{:}e_n]\!]_{\Gamma,T::m,\overline{\mathcal{L}}} =$

$x_1\mathtt{:}[\![e_1]\!]_{\Gamma,T::m\ \mathtt{(}x_1\ldots x_n\mathtt{)}::x_1,\overline{\mathcal{L}}}\cdots$

$x_n\mathtt{:}[\![e_n]\!]_{\Gamma,T::m,\ \mathtt{(}x_1\ldots x_n\mathtt{)}::x_n,\overline{\mathcal{L}}}$

${}^{ps}[\![x_1\mathtt{:}e_1\ldots x_n\mathtt{:}e_n]\!]_{\Gamma,T,\overline{\mathcal{L}}} =$

$x_1\mathtt{:}[\![e_1]\!]_{\Gamma,T,\overline{\mathcal{L}}}\ldots x_n\mathtt{:}[\![e_n]\!]_{\Gamma,T,\overline{\mathcal{L}}}$

${}^{\mathcal{M}}[\![\mathcal{M}_1\ldots\mathcal{M}_n]\!]_p = {}^{\mathcal{M}}[\![\mathcal{M}_1]\!]_p\ldots{}^{\mathcal{M}}[\![\mathcal{M}_n]\!]_p$

${}^{\mathcal{M}}[\![mh\,e]\!]_p = {}^{\mathcal{M}}[\![{}^{\mathcal{M}}[\![mh]\!]_p\ [\![e]\!]_{\emptyset;\text{Tof}(mh);p}]\!]$

${}^{\mathcal{M}}[\![mh\mathcal{E}^\star[\ \mathtt{(}\overline{d}\mathtt{catch\ exception}\ x\ \mathtt{(}\overline{\mathcal{O}}\mathtt{default}\ e\mathtt{)}\ e_0\mathtt{)}\ ]]\!] =$

${}^{\mathcal{M}}[\![mh\mathcal{E}^\star[\ \mathtt{(}\overline{d}\mathtt{catch\ exception}\ x\ \mathtt{(}\overline{\mathcal{O}}\mathtt{on\ Any}\ e\mathtt{)}\ e_0\mathtt{)}\ ]]\!]$

${}^{\mathcal{M}}[\![mh\mathcal{E}^\star[\ \mathtt{(}\overline{d}\mathtt{catch\ return}\ x\ \mathtt{(}\overline{\mathcal{O}}\mathtt{default}\ e\mathtt{)}\ e_0\mathtt{)}\ ]]\!] =$

${}^{\mathcal{M}}[\![mh\mathcal{E}^\star[\ \mathtt{(}\overline{d}\mathtt{catch\ return}\ x\ \mathtt{(}\overline{\mathcal{O}}\mathtt{on}\ T\ e\mathtt{)}\ e_0\mathtt{)}\ ]]\!]$

where T is obtained using $\mathcal{E}^\star$ as the innermost $\mathtt{catch\_return\ (on}\ T\mathtt{\_)}$ or $\mathtt{Any}$ if no such $\mathcal{E}^\star$ exists

${}^{\mathcal{M}}[\![mh\mathcal{E}^\star[\ \mathtt{(}\overline{d}_1\ C\mathtt{:}e\,\overline{d}_2\overline{\mathcal{K}}e_0\mathtt{)}\ ]]\!] =$

${}^{\mathcal{M}}[\![C\mathtt{:}e]\!]{}^{\mathcal{M}}[\![mh\mathcal{E}^\star[\ \mathtt{(}\overline{d}_1\,\overline{d}_2\overline{\mathcal{K}}e_0\mathtt{)}\ ]]\!]$

otherwise ${}^{\mathcal{M}}[\![mh\,e]\!] = mh\,e$

${}^{\mathcal{M}}[\![C\mathtt{:}\mathcal{E}^\star[\mathtt{\ldots}]]\!]_p = {}^{\mathcal{M}}[\![C\mathtt{:}\mathcal{E}^\star[e]]\!]_p$Where e is found on the local system depending on the original position of such $\mathtt{\ldots}$ symbol in the source and C

${}^{\mathcal{M}}[\![\mu\ \mathtt{method}\ T\ m\mathtt{(}\ \overline{T\ x}\mathtt{)}\ \mathtt{exception}\ \overline{\pi}]\!]_p =$

$\mu\ \mathtt{method}\ {}^{\pi}[\![T]\!]_p\ [\![\text{'}m\text{'}]\!]\mathtt{(}{}^{\pi}[\![\ \overline{T\ x}]\!]_p\mathtt{)}\ \mathtt{exception}\ {}^{\pi}[\![\overline{\pi}]\!]_p$

${}^{\mathcal{M}}[\![\mathtt{method}\,m\mathtt{(}\overline{x}\mathtt{)}\ ]\!]_p = \mathtt{method}[\![\text{'}m\text{'}]\!]\mathtt{(}\overline{x}\mathtt{)}$

${}^{\mathcal{M}}[\![C\mathtt{:}]\!]_p = C\mathtt{:}$

${}^{\pi}[\![C\mathtt{::}\overline{C}]\!]_{\mathcal{L}_0\ldots\mathcal{L}_n} = \mathtt{Outer}_k\mathtt{::}C\mathtt{::}\overline{C}$

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

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

${}^{\pi}[\![C\mathtt{::}\overline{C}]\!]_{\mathcal{L}_0\ldots\mathcal{L}_n} = \mathtt{Outer}_n\mathtt{::}C\mathtt{::}\overline{C}$

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

Definition: Tof

$\text{Tof}(\ C\mathtt{:}) = \mathtt{immutable\ Library}$

$\text{Tof}(\mu\ \mathtt{method}\ T\ m\mathtt{(}\ \overline{T\ x}\mathtt{)}\ \mathtt{exception}\ \overline{\pi}) = T$

$\text{Tof}(\mathtt{method}\,m\mathtt{(}\overline{x}\mathtt{)}\ ) = \mathtt{Outer_0.}m\mathtt{(}\overline{x}\mathtt{)}$

Definition: $\text{declareIts}(\overline{\mathcal{I}}, e_0)$

$\text{declareIts}(\emptyset, e) = e$

$\text{declareIts}(\mathtt{var}\ x_0\ \mathtt{in}\ e_0\overline{\mathcal{I}}, e) = \mathtt{(}$

$x_0\mathtt{=}e_0$ $\quad \mathtt{(\,(}\text{declareIts}(\overline{\mathcal{I}}, e)$

$\mathtt{catch\ exception}\ y\ \ \mathtt{(default\ (}x_0\mathtt{.\#close()\ exception}\ y\mathtt{)\ )}$

$\mathtt{void)}$

$\mathtt{catch\ return}\ y'\ \ \mathtt{(default\ (}x_0\mathtt{.\#close()\ exception}\ y'\mathtt{)\ )}$

$x_0\mathtt{.\#close()\ )}$

Definition: $\text{next}_i(\overline{x})$

$\text{next}_i(z_0\ldots z_n) = z_i\mathtt{.\#next()}$

$\mathtt{catch\ exception\ \ (on\ Void\ (}$

$\mathtt{(}z_{i+1}\mathtt{.\#next()\ catch\ exception\ \ (on\ Void\ void)\ void)}$

$\ldots\ \mathtt{(}z_n\mathtt{.\#next()\ catch\ exception\ \ (on\ Void\ void)\ void)}$

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

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

$\mathtt{exception\ void)\ )}$

Definition: $\text{cast}^{\mu\ \pi}(y \leftarrow x)$

$\text{cast}^{\mu\ \pi}(y \leftarrow x) = \ \mu\ \pi\ y\mathtt{=\ (return}\ x$

$\mathtt{catch\ return}\ z\ \ \mathtt{(on}\ \mu\ \pi\,z\ \mathtt{on}\ \mu\ \mathtt{Any\ exception\ void)}$

$\mathtt{error\ void)}$ $\qquad$ with z fresh

Definition: $\text{xsOf}(ps)$

$e_0x_1 : e_1\ldots x_n : e_n = \mathtt{that}x_1\ldots x_n$

$x_1 : e_1\ldots x_n : e_n = x_1\ldots x_n$

Plugin auxiliary definitions

Definition: $\mathcal{L}_1 = \mathcal{L}_2$
$\mathcal{L}$ is equivalent to a version where all the `'@private`
members have been consistently renamed

Definition: $\mathcal{L}_1 \oplus \mathcal{L}_2 = \mathcal{L}$
$\{\mathcal{H}_1 <: \overline{\pi}_1 \overline{\mathcal{M}}_1\} \oplus \{\mathcal{H}_2 <: \overline{\pi}_2\overline{\mathcal{M}}_2\} = \{\mathcal{H}_1 \oplus \mathcal{H}_2 <: \overline{\pi}_1\overline{\pi}_2 \overline{\mathcal{M}}_1 \oplus \overline{\mathcal{M}}_2\}$
$\texttt{interface} \oplus \texttt{interface} = \texttt{interface}$
$\emptyset\oplus\emptyset=\emptyset$
$\texttt{interface}\oplus\emptyset = \emptyset\oplus\texttt{interface}=\emptyset$
only if `interface`is virgin
$\overline{\mathcal{M}}_1 \mathcal{M} \oplus \overline{\mathcal{M}}_2 = \overline{\mathcal{M}}_1 \oplus (\mathcal{M}\oplus\overline{\mathcal{M}}_2)$
$\emptyset \oplus \overline{\mathcal{M}} = \overline{\mathcal{M}}$
$\mathcal{M}\oplus\overline{\mathcal{M}} = \mathcal{M}\overline{\mathcal{M}}$ if $\{\mathcal{M}\overline{\mathcal{M}}\}$ is well formed, otherwise
$C{:}doc_1\mathcal{L}_1 \oplus C{:}doc_2\mathcal{L}_2\overline{\mathcal{M}} = C{:}(doc_1\oplus doc_2 \mathcal{L}_1\oplus\mathcal{L}_2)\overline{\mathcal{M}}$
$h_1\overline{e}\oplus h_2\overline{\mathcal{M}} = h_1 \oplus h_2\overline{e}\overline{\mathcal{M}} = (h_1\oplus h_2)\overline{e}\overline{\mathcal{M}}$
where h_1, h_2 differs only for the documentation
$\texttt{method}\, doc\, m x\, e$ can not be sum with h

Definition: $p \vdash \mathcal{L}_1 \rtimes \mathcal{L}_2 = \mathcal{L}$
$\mathcal{L}_1\rtimes\mathcal{L}_2 = \mathcal{L}_1[_p \mathsf{mapMx}(\mathcal{L}_2)][_p\mathsf{mapC}(\mathcal{L}_2)]$
$\pi_1 \mapsto \pi_2[\mathsf{from}\ \pi_1] \in \mathsf{mapC}(\mathcal{L})$ iff $\mathcal{L}(\pi_1) = \{\,(\pi_2\ \texttt{that})\ \_\}$
$\pi\, m\,(x_1...x_n)\mapsto m'\,(x'_1...x'_n) \in \mathsf{mapMx}(\mathcal{L})$ iff $\mathcal{L}(\pi) =$
$\{\mathcal{H}\overline{\mathcal{M}}_1 \texttt{method Void}\ m\ (\texttt{Void}\ x_1...\texttt{Void}\ x_n)\ (\texttt{this}.m'(x'_1{:}x_1...x'_n{:}x_n))\ \overline{\mathcal{M}}_2\}$

Definition: $\mathcal{L}[_p\overline{\pi.mx\mapsto mx'}] = \mathcal{L}'$
$\mathcal{L}[_p \pi_1.mx_1\mapsto mx'_1 ... \pi_n.mx_n \mapsto mx'_n] = \mathcal{L}_0\oplus...\oplus\mathcal{L}_n$
where:
$p' = \mathsf{removeTopLevel}{\updownarrow}(p)$
$\mathcal{L}' = \mathcal{L}[\mathsf{renUsage}_{p'}\overline{\pi.mx\mapsto mx'}]$
$\mathcal{L}_0 = \mathcal{L}'[\mathsf{remove}\pi_1.mx_1]...[\mathsf{remove}\pi_n.mx_n]$
$\mathcal{L}_i = \mathcal{L}'[\mathsf{retainOnly}\pi_i.mx_i\mapsto mx'_i]$
on purpose not put $\updownarrow$ when composing $\mathcal{L}p$
to stop normalization scope

Definition: $\mathcal{L}[\mathsf{renUsage}_p\overline{\pi.mx\mapsto mx'}] = \mathcal{L}'$
$\mathcal{L}[\mathsf{renUsage}_p\overline{\pi.mx\mapsto mx'}]$ and $e[\mathsf{renUsage}_{\Gamma,p}\overline{\pi.mx\mapsto mx'}]$
propagate on the subterms, but
$(C{:}e)[\mathsf{renUsage}_{\ddot{\mathcal{L}}p}\overline{\pi.mx\mapsto mx'}] =$
$C{:}(e[\mathsf{renUsage}_{\ddot{\mathcal{L}}[C:\updownarrow]p}\overline{\pi.mx\mapsto mx'}])$,
$\{\mathcal{H}\overline{\mathcal{M}}\}[\mathsf{renUsage}_p\overline{\pi.mx\mapsto mx'}] = \{\mathcal{H}[\mathsf{renUsage}_{\widehat{\mathcal{L}}p}\overline{\texttt{Outer}_1::\pi.mx\mapsto mx'}]$
$\overline{\mathcal{M}}[\mathsf{renUsage}_{\widehat{\mathcal{L}}p}\overline{\texttt{Outer}_1::\pi.mx\mapsto mx'}]\}$(first time not add outer1)
with $p;\emptyset\vdash\{\mathcal{H}\overline{\mathcal{M}}\}\to\widehat{\mathcal{L}}$ and
$e.m(x_1{:}e_1...x_n{:}e_n)\ [\mathsf{renUsage}_{\Gamma,p}\overline{\pi.mx\mapsto mx'}] =$
$e.m'(x'_1{:}e_1...x'_n{:}e_n)\ [\mathsf{renUsage}_{\Gamma,p}\overline{\pi.mx\mapsto mx'}]$
iff $\pi.m\ (x_1...x_n)\mapsto m'\ (x'_1...x'_n) \in \overline{\pi.mx\mapsto mx'}$
$\mathsf{norm}_p(\mathsf{guessType}_\Gamma(e)) = \_\ \pi'\_$ and $\mathsf{norm}_p(\pi) = \pi'$
Actually, smarter way for block is used, looking in catches

Definition: $\mathcal{L}[_p\overline{\pi\mapsto\pi'}] = \mathcal{L}'$
$\mathcal{L}[_p\overline{\pi\mapsto\pi'}] = \mathcal{L}_0\oplus...\oplus\mathcal{L}_n$
where:
$p' = \mathsf{removeTopLevel}{\updownarrow}(p)$
$\mathcal{L}' = \mathcal{L}[\mathsf{renUsage}_{p'}\overline{\pi\mapsto\pi'}]$
$\mathcal{L}_0 = \mathcal{L}'[\mathsf{remove}\pi_1]...[\mathsf{remove}\pi_n]$
$\mathcal{L}_i = \mathcal{L}'[\mathsf{redirectDefinition}\,\pi_i\mapsto\pi'_i]$

Definition: $\mathcal{L}[\mathsf{redirectDefinition}\,\pi\mapsto\pi'] = \mathcal{L}'$
$\mathcal{L}[\mathsf{redirectDefinition}\,\pi\mapsto\texttt{Library}] = \mathcal{L}[\mathsf{redirectDefinition}\,\pi\mapsto\texttt{Void}] =$
$\mathcal{L}[\mathsf{redirectDefinition}\,\pi\mapsto\texttt{Any}] = \mathcal{L}[\mathsf{redirectDefinition}\,\pi\mapsto\texttt{Outer}_{n+1}::\_] = \{\}$
$\mathcal{L}[\mathsf{redirectDefinition}\,\texttt{Outer}_0::\overline{C}_0\mapsto\texttt{Outer}_0::\overline{C}_1] =$
$\mathcal{L}(\overline{C}_0)[\mathsf{from}\ \overline{C}_0\pi'][\mathsf{encapsulateIn}\overline{C}_1]$ iff $[\mathsf{to}\overline{C}_1] = \texttt{Outer}_1::\pi'::C'$
otherwise $\mathcal{L}[\mathsf{redirectDefinition}\,\texttt{Outer}_0::\overline{C}_0\mapsto\texttt{Outer}_0::\overline{C}_1] =$
$\mathcal{L}(\overline{C}_0)[\text{remove n outers}][\mathsf{encapsulateIn}\overline{C}_1]$ if $[\mathsf{to}\overline{C}_1] = \texttt{Outer}_0::C_1::..::C_n$

Definition: $\overline{C}_0[\mathsf{to}\overline{C}_1] = \pi$
$\overline{C}_0[\mathsf{to}\emptyset] = \texttt{Outer}_0::\overline{C}_0$
$C::\overline{C}_0[\mathsf{to}\,C::\overline{C}_0] = \overline{C}_0[\mathsf{to}\overline{C}_0]$
otherwise $\overline{C}_0[\mathsf{to}\,C_1::...::C_n] = \texttt{Outer}_n::\overline{C}_0$

Definition: $\mathcal{L}[\mathsf{renUsage}_p\pi_1]\pi'_1 = \mathcal{L}'$
$\mathcal{L}[\mathsf{renUsage}_p\pi]\pi'$ and $e[\mathsf{renUsage}_p\pi]\pi'$
propagate on the subterms, but
$(C{:}e)[\mathsf{renUsage}_{\ddot{\mathcal{L}}p}\pi]\pi' =$
$C{:}(e[\mathsf{renUsage}_{\ddot{\mathcal{L}}[C:\updownarrow]p}\pi]\pi')$,

Plugin auxiliary definitions

Definition: $\mathcal{L}_1 = \mathcal{L}_2$
$\mathcal{L}$ is equivalent to a version where all the `'@private`
members have been consistently renamed

Definition: $\mathcal{L}_1 \oplus \mathcal{L}_2 = \mathcal{L}$
$\{\mathcal{H}_1 <: \overline{\pi}_1 \overline{\mathcal{M}}_1\} \oplus \{\mathcal{H}_2 <: \overline{\pi}_2\overline{\mathcal{M}}_2\} = \{\mathcal{H}_1 \oplus \mathcal{H}_2 <: \overline{\pi}_1\overline{\pi}_2 \overline{\mathcal{M}}_1 \oplus \overline{\mathcal{M}}_2\}$
$\texttt{interface} \oplus \texttt{interface} = \texttt{interface}$
$\emptyset\oplus\emptyset=\emptyset$
$\texttt{interface}\oplus\emptyset = \emptyset\oplus\texttt{interface}=\emptyset$
only if `interface`is virgin
$\overline{\mathcal{M}}_1 \mathcal{M} \oplus \overline{\mathcal{M}}_2 = \overline{\mathcal{M}}_1 \oplus (\mathcal{M}\oplus\overline{\mathcal{M}}_2)$
$\emptyset \oplus \overline{\mathcal{M}} = \overline{\mathcal{M}}$
$\mathcal{M}\oplus\overline{\mathcal{M}} = \mathcal{M}\overline{\mathcal{M}}$ if $\{\mathcal{M}\overline{\mathcal{M}}\}$ is well formed, otherwise
$C{:}doc_1\mathcal{L}_1 \oplus C{:}doc_2\mathcal{L}_2\overline{\mathcal{M}} = C{:}(doc_1\oplus doc_2 \mathcal{L}_1\oplus\mathcal{L}_2)\overline{\mathcal{M}}$
$h_1\overline{e}\oplus h_2\overline{\mathcal{M}} = h_1 \oplus h_2\overline{e}\overline{\mathcal{M}} = (h_1\oplus h_2)\overline{e}\overline{\mathcal{M}}$
where h_1, h_2 differs only for the documentation
$\texttt{method}\, doc\, m x\, e$ can not be sum with h

Definition: $p \vdash \mathcal{L}_1 \rtimes \mathcal{L}_2 = \mathcal{L}$
$\mathcal{L}_1\rtimes\mathcal{L}_2 = \mathcal{L}_1[_p \mathsf{mapMx}(\mathcal{L}_2)][_p\mathsf{mapC}(\mathcal{L}_2)]$
$\pi_1 \mapsto \pi_2[\mathsf{from}\ \pi_1] \in \mathsf{mapC}(\mathcal{L})$ iff $\mathcal{L}(\pi_1) = \{\,(\pi_2\ \texttt{that})\ \_\}$
$\pi\, m\,(x_1...x_n)\mapsto m'\,(x'_1...x'_n) \in \mathsf{mapMx}(\mathcal{L})$ iff $\mathcal{L}(\pi) =$
$\{\mathcal{H}\overline{\mathcal{M}}_1 \texttt{method Void}\ m\ (\texttt{Void}\ x_1...\texttt{Void}\ x_n)\ (\texttt{this}.m'(x'_1{:}x_1...x'_n{:}x_n))\ \overline{\mathcal{M}}_2\}$

Definition: $\mathcal{L}[_p\overline{\pi.mx\mapsto mx'}] = \mathcal{L}'$
$\mathcal{L}[_p \pi_1.mx_1\mapsto mx'_1 ... \pi_n.mx_n \mapsto mx'_n] = \mathcal{L}_0\oplus...\oplus\mathcal{L}_n$
where:
$p' = \mathsf{removeTopLevel}{\updownarrow}(p)$
$\mathcal{L}' = \mathcal{L}[\mathsf{renUsage}_{p'}\overline{\pi.mx\mapsto mx'}]$
$\mathcal{L}_0 = \mathcal{L}'[\mathsf{remove}\pi_1.mx_1]...[\mathsf{remove}\pi_n.mx_n]$
$\mathcal{L}_i = \mathcal{L}'[\mathsf{retainOnly}\pi_i.mx_i\mapsto mx'_i]$
on purpose not put $\updownarrow$ when composing $\mathcal{L}p$
to stop normalization scope

Definition: $\mathcal{L}[\mathsf{renUsage}_p\overline{\pi.mx\mapsto mx'}] = \mathcal{L}'$
$\mathcal{L}[\mathsf{renUsage}_p\overline{\pi.mx\mapsto mx'}]$ and $e[\mathsf{renUsage}_{\Gamma,p}\overline{\pi.mx\mapsto mx'}]$
propagate on the subterms, but
$(C{:}e)[\mathsf{renUsage}_{\ddot{\mathcal{L}}p}\overline{\pi.mx\mapsto mx'}] =$
$C{:}(e[\mathsf{renUsage}_{\ddot{\mathcal{L}}[C:\updownarrow]p}\overline{\pi.mx\mapsto mx'}])$,
$\{\mathcal{H}\overline{\mathcal{M}}\}[\mathsf{renUsage}_p\overline{\pi.mx\mapsto mx'}] = \{\mathcal{H}[\mathsf{renUsage}_{\widehat{\mathcal{L}}p}\overline{\texttt{Outer}_1::\pi.mx\mapsto mx'}]$
$\overline{\mathcal{M}}[\mathsf{renUsage}_{\widehat{\mathcal{L}}p}\overline{\texttt{Outer}_1::\pi.mx\mapsto mx'}]\}$(first time not add outer1)
with $p;\emptyset\vdash\{\mathcal{H}\overline{\mathcal{M}}\}\to\widehat{\mathcal{L}}$ and
$e.m(x_1{:}e_1...x_n{:}e_n)\ [\mathsf{renUsage}_{\Gamma,p}\overline{\pi.mx\mapsto mx'}] =$
$e.m'(x'_1{:}e_1...x'_n{:}e_n)\ [\mathsf{renUsage}_{\Gamma,p}\overline{\pi.mx\mapsto mx'}]$
iff $\pi.m\ (x_1...x_n)\mapsto m'\ (x'_1...x'_n) \in \overline{\pi.mx\mapsto mx'}$
$\mathsf{norm}_p(\mathsf{guessType}_\Gamma(e)) = \_\ \pi'\_$ and $\mathsf{norm}_p(\pi) = \pi'$
Actually, smarter way for block is used, looking in catches

Definition: $\mathcal{L}[_p\overline{\pi\mapsto\pi'}] = \mathcal{L}'$
$\mathcal{L}[_p\overline{\pi\mapsto\pi'}] = \mathcal{L}_0\oplus...\oplus\mathcal{L}_n$
where:
$p' = \mathsf{removeTopLevel}{\updownarrow}(p)$
$\mathcal{L}' = \mathcal{L}[\mathsf{renUsage}_{p'}\overline{\pi\mapsto\pi'}]$
$\mathcal{L}_0 = \mathcal{L}'[\mathsf{remove}\pi_1]...[\mathsf{remove}\pi_n]$
$\mathcal{L}_i = \mathcal{L}'[\mathsf{redirectDefinition}\,\pi_i\mapsto\pi'_i]$

Definition: $\mathcal{L}[\mathsf{redirectDefinition}\,\pi\mapsto\pi'] = \mathcal{L}'$
$\mathcal{L}[\mathsf{redirectDefinition}\,\pi\mapsto\texttt{Library}] = \mathcal{L}[\mathsf{redirectDefinition}\,\pi\mapsto\texttt{Void}] =$
$\mathcal{L}[\mathsf{redirectDefinition}\,\pi\mapsto\texttt{Any}] = \mathcal{L}[\mathsf{redirectDefinition}\,\pi\mapsto\texttt{Outer}_{n+1}::\_] = \{\}$
$\mathcal{L}[\mathsf{redirectDefinition}\,\texttt{Outer}_0::\overline{C}_0\mapsto\texttt{Outer}_0::\overline{C}_1] =$
$\mathcal{L}(\overline{C}_0)[\mathsf{from}\ \overline{C}_0\pi'][\mathsf{encapsulateIn}\overline{C}_1]$ iff $[\mathsf{to}\overline{C}_1] = \texttt{Outer}_1::\pi'::C'$
otherwise $\mathcal{L}[\mathsf{redirectDefinition}\,\texttt{Outer}_0::\overline{C}_0\mapsto\texttt{Outer}_0::\overline{C}_1] =$
$\mathcal{L}(\overline{C}_0)[\text{remove n outers}][\mathsf{encapsulateIn}\overline{C}_1]$ if $[\mathsf{to}\overline{C}_1] = \texttt{Outer}_0::C_1::..::C_n$

Definition: $\overline{C}_0[\mathsf{to}\overline{C}_1] = \pi$
$\overline{C}_0[\mathsf{to}\emptyset] = \texttt{Outer}_0::\overline{C}_0$
$C::\overline{C}_0[\mathsf{to}\,C::\overline{C}_0] = \overline{C}_0[\mathsf{to}\overline{C}_0]$
otherwise $\overline{C}_0[\mathsf{to}\,C_1::...::C_n] = \texttt{Outer}_n::\overline{C}_0$

Definition: $\mathcal{L}[\mathsf{renUsage}_p\pi_1]\pi'_1 = \mathcal{L}'$
$\mathcal{L}[\mathsf{renUsage}_p\pi]\pi'$ and $e[\mathsf{renUsage}_p\pi]\pi'$
propagate on the subterms, but
$(C{:}e)[\mathsf{renUsage}_{\ddot{\mathcal{L}}p}\pi]\pi' =$
$C{:}(e[\mathsf{renUsage}_{\ddot{\mathcal{L}}[C:\updownarrow]p}\pi]\pi')$,