# Language 42

for more information `L42.is`

## Core Language Syntax

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

## Auxiliary syntax

$\mathcal{E}^{\mathsf{c}} ::= \square \mid \mathcal{E}^{\mathsf{c}}\texttt{.}m\texttt{(}\overline{x\texttt{:}e}\texttt{)} \mid e_0{}^{\mathsf{c}}\texttt{.}m\texttt{(}\overline{x\texttt{:}e^{\mathsf{c}}}_1\,x\texttt{:}\mathcal{E}^{\mathsf{c}}\,\overline{x\texttt{:}e_2}\texttt{)} \mid \texttt{loop}\;\mathcal{E}^{\mathsf{c}} \mid \varrho\;\mathcal{E}^{\mathsf{c}}$
$\quad \mid \texttt{(}\overline{d^{\mathsf{c}}}_1\;T\;x\texttt{=}\mathcal{E}^{\mathsf{c}}\;\overline{d}_2\overline{\mathcal{K}}\;e\texttt{)} \mid \texttt{(}\overline{d^{\mathsf{c}}}\texttt{catch}\;\varrho\;x\;\texttt{on}\;T\;\mathcal{E}^{\mathsf{c}}\;e\texttt{)} \mid \texttt{(}\overline{d^{\mathsf{c}}}\,\overline{\mathcal{K}^{\mathsf{c}}}\,\mathcal{E}^{\mathsf{c}}\texttt{)}$
$\quad \texttt{using}\;\pi\;\texttt{check}\;\texttt{.}m\texttt{(}\overline{e^{\mathsf{c}}}_1\mathcal{E}^{\mathsf{c}}\,\overline{e}_2\texttt{)}\;e \mid \texttt{using}\;\pi\;\texttt{check}\;\texttt{.}m\texttt{(}\overline{e^{\mathsf{c}}}\texttt{)}\;\mathcal{E}^{\mathsf{c}}$

$\mathcal{E}^{\star} ::= \square \mid \mathcal{E}^{\star}\texttt{.}m\texttt{(}\overline{e}\texttt{)} \mid e_0\texttt{.}m\texttt{(}\overline{x\texttt{:}e}_1\,x\texttt{:}\mathcal{E}^{\star}\,\overline{x\texttt{:}e_2}\texttt{)} \mid \texttt{loop}\;\mathcal{E}^{\star} \mid \varrho\;\mathcal{E}^{\star}$
$\quad \mid \texttt{(}\overline{d}_1\;T\;x\texttt{=}\mathcal{E}^{\star}\;\overline{d}_2\overline{\mathcal{K}}\;e\texttt{)} \mid \texttt{(}\overline{d}\texttt{catch}\;\varrho\;x\;\texttt{on}\;T\;\mathcal{E}^{\star}\;e\texttt{)} \mid \texttt{(}\overline{d}\,\overline{\mathcal{K}}\,\mathcal{E}^{\star}\texttt{)}$
$\quad \texttt{using}\;\pi\;\texttt{check}\;\texttt{.}m\texttt{(}\overline{x\texttt{:}e}_0\,x\texttt{:}\mathcal{E}^{\star}\,\overline{x\texttt{:}e}_1\texttt{)}\;e \mid \texttt{using}\;\pi\;\texttt{check}\;\texttt{.}m\texttt{(}x\texttt{:}\overline{v}\texttt{)}\;\mathcal{E}^{\star}$

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

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

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

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

**Definition:** $\mathsf{compiled}(\_)$

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

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

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

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

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

**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 $\overline{\texttt{var}}$ 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 $\texttt{(}e\texttt{)}$ and $e$ to be equivalent, and from now on we omit documentations $doc$ when is not relevant. We consider terms of the form $\texttt{(}e\texttt{)}$ to be equivalent to the corresponding $e$ and terms of the form $\texttt{(}\overline{dv}\texttt{catch}\;\varrho\;x\;\texttt{()}\;e\texttt{)}$ to be equivalent to the corresponding $\texttt{(}\overline{dv}\,e\texttt{)}$. Also, values of form $\texttt{(}\;T\;x\texttt{=}\texttt{(}\;T\;y\texttt{=}ey\texttt{)}\;x\texttt{)}$ are equivalent to the corresponding $\texttt{(}\;T\;y\texttt{=}ey\texttt{)}$ 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 $\mathcal{E}^p = \mathcal{E}^{p\prime}[\texttt{using}\;\pi\;\texttt{check}\;\texttt{.}m\texttt{(}\overline{x\texttt{:}v}\texttt{)}\;\mathcal{E}^{p\prime\prime}]$ and $\mathsf{plugin}(\pi\;m\;\texttt{(}\overline{x\texttt{:}v}\texttt{)}\;\mathcal{E}^{p\prime\prime}[e])$ is well defined.

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

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

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

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

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

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

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

**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 $\mathsf{norm}_p(\pi') \in \mathsf{norm}_p(\overline{\pi}[\mathsf{from}\;\pi] \cup \pi) \cup \texttt{Any}$
$\quad$ with $p(\pi) = \texttt{\{}\_\texttt{\}}^{-;\overline{\pi}}$
$\texttt{capsule} \leq \texttt{mut} \leq \texttt{lent} \leq \texttt{read}$, $\texttt{capsule} \leq \texttt{immutable} \leq \texttt{read}$ and $\emptyset \leq \% \leq \texttt{\^{}}$

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

$h_1 \ldots h_n \cup \overline{\mathcal{M}} = h_1 \cup (\ldots \cup (h_n \cup \overline{\mathcal{M}}))$
$h \cup \overline{\mathcal{M}} = h\overline{\mathcal{M}}$ 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$(\Gamma)$, 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$(\,(\overline{dv}\ T\ y = \mathcal{E}^p\ \overline{d}\overline{K} e), \overline{x}) = \langle (\overline{dv}\ \overline{dv}'\ T\ y = \mathcal{E}^{p\prime}\ \overline{d}\overline{K} e), \emptyset\rangle$

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

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

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

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

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

$\overline{dv} = \overline{dv}_1\ \overline{dv}_2$ 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$(\mathcal{E}^{p\prime}[\,(\overline{dv}\mathcal{E}^p)], x)$ = dec$(\mathcal{E}^{p\prime}[\,(\overline{dv}\ T\ y = \mathcal{E}^p\ \overline{d}\overline{K} e)], x) = \overline{dv}(x)$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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 $\mathcal{L}_0 = \{\mathcal{H}{<:}\pi_1 \ldots \pi_n \overline{\mathcal{M}}\}$,

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

## Definitions2

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

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

**Definition:** norm$_p(\pi)$, norm$_p(T)$, norm$_p(\overline{he})$

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

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

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

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

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

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

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

$= \mu\ \mathtt{method}\ T'_0\ m(\ T'_1\ x_1 \ldots\ T'_n\ x_n)\ \mathtt{exception}$ 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$(p, \pi)$   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 $p(\pi) = \mathcal{L}^c \in \{\{\_\}^{\oplus}, \{\_\}^{\circledast}\}$

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

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

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

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 $\Gamma$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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 $e$   $p \vdash \mathcal{L} \rightarrow \{\_\}^{\overline{\pi}}$

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

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

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

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

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

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

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

superOf$_p(\mathcal{L}) = \overline{\pi}_1[$from $\pi_1] \cup \ldots \cup \overline{\pi}_n[$from $\pi_n]$, $\overline{h}_1[$from $\pi_1] \ldots \overline{h}_n[$from $\pi_n]$

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

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

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

are they used somewhere?

## Extraction of types

(ET-+)
$$\frac{\mathcal{L}_1 \xrightarrow{p} \ldots \xrightarrow{p} \mathcal{L}_2 \qquad \mathcal{L}_2 \not\xrightarrow{p} \_}{\mathcal{L}_1 \xrightarrow[\max]{p} \mathcal{L}_2}$$

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

(ET-LABEL)
$$\frac{}{\mathcal{L} \xrightarrow{p} \{\mathcal{H}\overline{\mathcal{M}}\}^{\mathsf{stageOf}_p(\mathcal{L},\overline{e})}}$$
with
$\mathcal{L} = \{\mathcal{H}\overline{\mathcal{M}}\}^{\circledast}$
$\overline{e} = \{(\mathcal{L}p)(\pi) | \pi \in \mathsf{used}^{\oplus}(\mathcal{L})\} \setminus \updownarrow$
$\cup\{\mathcal{L}(::C) | ::C \in \mathsf{dom}(\mathcal{L})\}$
$\forall \mathcal{L}^{\mathfrak{c}} \in e, \mathcal{L}^{\mathfrak{c}} = \{\}^{\circledcirc}$

need a rule for neverLess? read comment under

(P-OK)
$$\frac{\vdash p : \text{ok if } p \neq \emptyset \qquad p \vdash \ddot{\mathcal{L}}_0 : \text{ok}}{\vdash \ddot{\mathcal{L}}p : \text{ok}}$$
with
$\ddot{\mathcal{L}}_0 = \ddot{\mathcal{L}}[C{:}\texttt{error void}]$
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
$h = \mu\, \texttt{method}\ T\ m\texttt{(}T_1\ x_1 \ldots T_n x_n\texttt{)}\ \texttt{exception}\, \overline{\pi}$
$\Gamma = x_1 : T_1 \ldots x_n : T_n, \texttt{this} : \mu\ \texttt{Outer}_0$
$\Sigma = \emptyset; \emptyset; \texttt{this}, x_1, \ldots, x_n$
$\mathsf{exeOk}^{\oplus}(p, \Gamma)$

(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)
$$\frac{\widehat{\mathcal{L}^{\mathfrak{c}}}[C{:}\updownarrow], p \vdash \mathcal{L}' : \text{ok } \forall C{:}\mathcal{L}^{\mathfrak{c}\prime}, \mathcal{L}^{\mathfrak{c}}(C) = C{:}\mathcal{L}^{\mathfrak{c}} \qquad \mathcal{L}^{\mathfrak{c}}, p \vdash h\overline{e} : \text{ok} \forall h\overline{e}, \mathcal{L}^{\mathfrak{c}}(h) = h\overline{e}}{p \vdash \mathcal{L}^{\mathfrak{c}} : \text{ok}}$$
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)
$$\frac{}{p;\Gamma;\Sigma;\Phi \vdash \pi : \texttt{type}\ \pi}$$
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)
$$\frac{}{p;\Gamma;\Sigma;\Phi \vdash \pi : \texttt{type Any}}$$
with
either $p(\pi) = \widehat{\mathcal{L}^{\mathfrak{c}}}$
or $\pi \in \{\texttt{Any}, \texttt{Void}, \texttt{Library}\}$

(LIB-T)
$$\frac{}{p;\Gamma;\Sigma;\Phi \vdash \mathcal{L} : \texttt{immutable Library}}$$
with
$p \vdash \mathcal{L} \rightarrow \widehat{\mathcal{L}^{\mathfrak{c}}}$
if not $\mathsf{exe}(p)$ then $p \vdash \widehat{\mathcal{L}^{\mathfrak{c}}}$ : ok and
$\widehat{\mathcal{L}^{\mathfrak{c}}} = \{\_\_\}^{\oplus,-}$

(USING-T)
$$\frac{p;\Gamma;\Sigma;\Phi \vdash e_i : T_i' \leq T_i \text{ for all } i \in 0..n}{p;\Gamma;\Sigma;\Phi \vdash \texttt{using}\ \pi\ \texttt{check .}m\texttt{(}\overline{x{:}e}\texttt{)}\ e_0 : T_0}$$
with
$\overline{x{:}e} = x_1{:}e_1 \ldots x_n{:}e_n$
$\mathsf{plugin}(p, \pi, .m(x_1 \ldots x_n)) = plg; T_0 T_1 \ldots T_n$

(T-VAR)
$$\frac{}{p;\Gamma;\overline{x}_0;\overline{x}_1 \ldots \overline{x}_n;\_;\_ \vdash x : T}$$
with
$\mathsf{norm}_p(\Gamma(x)) = \mu\ \pi\alpha$
$x \notin \overline{x}_0$
$$T = \begin{cases} \texttt{lent}\ \pi\alpha & \text{if } x \in \overline{x}_1 \ldots \overline{x}_n \\ \mu\ \pi\alpha & \text{otherwise} \end{cases}$$

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

(T-VOID)
$$\frac{}{p;\Gamma;\Sigma;\Phi \vdash \texttt{void} : \texttt{capsule Void}}$$

(LOOP-T)
$$\frac{p;\Gamma;\Sigma;\Phi \vdash e : \texttt{immutable Void}}{p;\Gamma,\_;\Sigma;\Phi \vdash \texttt{loop}\ e : \texttt{immutable Void}}$$
with
$T$ not of form $\texttt{capsule}\ \_\_\ \forall x{:}T \in \Gamma$

(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 $\mu \in \{\texttt{read}, \texttt{lent}, \texttt{mut}\}$ then $\overline{x}' = \overline{x}$
otherwise $\overline{x}' = \emptyset$

(T-METHCALL)
$$\frac{p;\Gamma_0;\Sigma_0;\Phi \vdash e_0 : \_ \leq \mu_0\ \pi_0 \qquad p;\Gamma_i;\Sigma_i;\Phi \vdash e_i : T_i' \leq T_i \quad \forall i \in 1..n}{p;\Gamma;\Sigma;\Phi \vdash e_0.m\texttt{(}x_1{:}e_1 \ldots x_n{:}e_n\texttt{)} : T'}$$
with
$\mathsf{meth}_p(\pi_0.m, x_1 \ldots x_n) =$
$\mu_0\ \texttt{method}\ T\ m\texttt{(}T_1\ x_1 \ldots T_n\ x_n\texttt{)}\ \texttt{exception}\ \overline{\pi}\ \_$
$p \vdash \overline{\pi} \leq \Phi$
$\Gamma_i \subseteq \Gamma\ \forall i \in 0..n$
$\Gamma_i(x) = \Gamma_j(x) = \texttt{capsule}\ \_\_$ 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)
$$\frac{p;\Gamma;\Sigma';\Phi \vdash e : \texttt{mut}\ \pi\alpha}{p;\Gamma;\Sigma;\Phi \vdash e : \texttt{capsule}\ \pi\alpha}$$
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)
$$\frac{p;\Gamma;\Sigma';\Phi \vdash e : \mu\ \pi\alpha}{p;\Gamma;\Sigma;\Phi \vdash e : \texttt{immutable}\ \pi\alpha}$$
with
$\mu \in \{\texttt{read}, \texttt{lent}\}$
$\Sigma = \overline{x};\overline{x}_1 \ldots \overline{x}_n;\overline{\overline{x}}$
$\overline{x}_0 = \mathsf{dom}^{\mathsf{mut}}(\Gamma)\setminus\overline{x}_1 \ldots \overline{x}_n$
$\Sigma' = \overline{x}, \mathsf{dom}^{\mathsf{mut}\leq}(\Gamma);\overline{x}_0 \ldots \overline{x}_n;\overline{\overline{x}}$

(T-BLOCK-COMPLETE/PARTIAL)
$$\frac{p;\Gamma_i[\mathcal{K},\Sigma];\Sigma;\Phi[\mathcal{K}] \vdash e_i : T_i' \leq \mathsf{toPartial}(T_i) \quad \forall i \in 1..n \qquad p;\Gamma_0;\Sigma\,\mathsf{FV}(e_1 \ldots e_n) \cup x_1 \ldots x_n;\Phi \vdash e : T \qquad p;\Gamma_0 \setminus \mathsf{dom}(\Gamma');\Sigma;\Phi \vdash \texttt{catch}\ \varrho\ x\ \texttt{(}\mathcal{O}_i\texttt{)} : T \quad \forall i \in 1..k}{p;\Gamma;\Sigma;\Phi \vdash \texttt{(}T_1\ x_1\ \texttt{=}e_1 \ldots T_n\ x_n\ \texttt{=}e_n\mathcal{K}e\texttt{)} : T}$$
with
$\mathcal{K} = \texttt{catch}\ \varrho\ x\ \texttt{(}\mathcal{O}_1 \ldots \mathcal{O}_k\texttt{)}$
$\Gamma' = x_1 : T_1 \ldots x_n : T_n$
$\Gamma_i(x) = \Gamma_j(x) = \texttt{capsule}\ \_\_$ 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
$\Gamma_i \subseteq \Gamma, \mathsf{toPh}(\mathsf{complete}(\Gamma'))\ \forall i \in 1..n$
$\Gamma_0 \subseteq \Gamma, \mathsf{toPartial}(\Gamma')$

(K-T-ANY)
$$\frac{p;\Gamma;\Sigma;\Phi[\mathcal{K}_i] \vdash \mathcal{K}_i : T \quad \forall i \in 1..2}{p;\Gamma;\Sigma;\Phi \vdash \texttt{catch}\ \varrho\ x\ \texttt{(on}\ \mu\ \texttt{Any}\ e\texttt{)} : T}$$
with
either $\varrho = \texttt{exception}, \mu' = \mu = \texttt{immutable}$
or $\varrho = \texttt{return}, \texttt{type} \in \{\mu', \mu\}, \mu' \neq \mu$
$\mathcal{K}_1, \mathcal{K}_2 = \texttt{catch}\ \varrho\ x\ \texttt{(on}\ \mu\ \texttt{Library}\ e\texttt{)}, \texttt{catch}\ \varrho\ x\ \texttt{(on}\ \mu'\ \texttt{Void}\ e\texttt{)}$

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

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

(DEC-K-PROM-T)
$$\frac{p;\Gamma;\Sigma;\Phi \vdash \texttt{(}\overline{d}_0\ \overline{d}_1\texttt{catch return}\ x\ \texttt{(on}\ \mu'\ \pi\ x\texttt{)}\ e_0\texttt{)} : T}{p;\Gamma;\Sigma;\Phi \vdash \texttt{(}\overline{d}_0\ \overline{d}_1\texttt{catch return}\ x\ \texttt{(on}\ \mu\ \pi\ x\texttt{)}\ e_0\texttt{)} : T}$$
with
$\mu = \texttt{capsule}$ and $\mu' = \texttt{mut}$
or $\mu = \texttt{immutable}$ and $\mu' \in \{\texttt{mut}, \texttt{read}\}$

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

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

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

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

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

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

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

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

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

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

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

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

(R-USING-OUT)
$$\mathcal{E}^p[\mathtt{using}\ \pi\ \mathtt{check}\ m\,(\overline{x{:}v})\ e] \longrightarrow_p \mathcal{E}^p[e]$$
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)
$$\frac{e_1 \longrightarrow_{p'} e_2 \qquad \mathcal{L} \xrightarrow[\max]{p} \mathcal{L}' \qquad \vdash p' : \mathrm{ok} \qquad p'\circledast; \emptyset; \emptyset; \emptyset \vdash e_1 : \mathtt{Library}}{\mathcal{E}^p[\mathcal{L}] \longrightarrow_p \mathcal{E}^p[\mathcal{L}[C{:}e_2]]}$$
with
$\mathcal{L} = \{\_\texttt{<:}\_\overline{\mathcal{M}}^{\mathsf{c}}\ C{:}e^{\mathsf{c}}\_\}$
$p' = \mathcal{L}'^{\ominus}[C : \updownarrow]\ p$
$e_1 = \mathsf{norm}_{p'}(e^{\mathsf{c}})$

(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
$\mathcal{L} = \{\_\texttt{<:}\_\overline{\mathcal{M}}^{\mathsf{c}} mh\ \mathcal{E}^{\mathsf{c}}[\mathcal{L}_1]\_\}$
$p' = \mathcal{L}'\ p$

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

(R-ONMISS)
$$\mathcal{E}^p[e_1] \longrightarrow_p \mathcal{E}^p[\,(\overline{dv}\ T'\ x\mathtt{=}\varrho\ v\mathtt{catch}\ \varrho\ z\ \overline{\mathcal{O}}e_0)\,]$$
with
$e_1 = (\overline{dv}\ \overline{dv}'\ T'\ x\mathtt{=}e'\ \overline{d}\mathtt{catch}\ \varrho\ z\ \mathtt{on}\ T\ e\ \overline{\mathcal{O}}e_0)$
$\mathsf{throws}_p(e') = \varrho'\ v$
$\mu\ \pi = \mathsf{norm}_p(T)$
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 $\mathcal{L}_0 ... \mathcal{L}_n$ such that $\emptyset|\mathcal{L}_0 \rightarrow \emptyset|... \rightarrow \emptyset|\mathcal{L}_n$.
Normal forms are results: either library literals well-typed in $\circledast$ 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

$\texttt{immutable}$, $\texttt{trait}$, $\texttt{exception}\emptyset$ in $mh$ and $\texttt{<:}\emptyset$ in $\mathcal{H}$ are represented with the empty string.
$EOL$ can be omitted after the $\texttt{reuse}$ sequence of character if no members are present. White-space consists of <space>, $EOL$ and '$\texttt{,}$'.

## 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. $\texttt{with}\,\emptyset\,\emptyset\,\emptyset\,\_$ is not well formed; $\texttt{with}\,\overline{x}\,\overline{\mathcal{I}}\,\overline{d}\;\texttt{(}\overline{\mathcal{O}^w}\overline{\mathcal{B}}\texttt{)}$ is well formed if the number of types $T_1 ... T_n$ in each $\texttt{on}$ 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 $\texttt{with}$, variables introduces in the $\overline{\mathcal{I}}$ can be updated only if they are declared $\texttt{var}$.
There must not be any whitespace preceding the symbol '$\texttt{"}$' in string expressions or $\pi$ in number expression.
For all blocks of form $\texttt{\{}\overline{d}_1\mathcal{K}_1...\overline{d}_n\mathcal{K}_n\texttt{\}}$, terminating($\texttt{\{}\overline{d}_1\mathcal{K}_1...\overline{d}_n\mathcal{K}_n\texttt{\}}$) holds. Method names in method calls (using the dot) must be of form $x$ or $\texttt{\#}x$.
The $\texttt{return}$ keyword can not be used inside any $\texttt{if}$,$\texttt{case}$ or $\texttt{while}$ condition or inside the expression of a $x\,\texttt{in}\,e$.

## 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 $\texttt{'@private}$ are renamed to others that does not sintactically occurs into the importing program.

**Definition:** terminating(_)

terminating($\varrho\;e$) $=$ terminating($\texttt{loop}\;e$) $=$ true
terminating($\texttt{if}\;\_\;e\;\texttt{else}\;\mathcal{B}$) $=$
  terminating($e$) and terminating($\mathcal{B}$)
terminating($\texttt{(}\overline{d}_1\mathcal{K}_1...\overline{d}_n\mathcal{K}_n\,e\texttt{)}$) $=$ terminating($\mathcal{K}_1$)
  and ... and terminating($\mathcal{K}_n$) and terminating($e$)
terminating($\texttt{\{}\overline{d}_1\mathcal{K}_1...\overline{d}_n\mathcal{K}_n\texttt{\}}$) $=$ terminating($\mathcal{K}_1$)
  and ... and terminating($\mathcal{K}_n$) and terminating($\overline{d}_n$)
terminating($\texttt{catch}\;\varrho\;x\;\texttt{(}doc\overline{\mathcal{O}}\texttt{default}\;e\texttt{)}$) $=$
  terminating($\texttt{catch}\;\varrho\;x\;\texttt{(}doc\overline{\mathcal{O}}$) and terminating($e$)
terminating($\texttt{catch\_}\;\texttt{(on\_}e_1...\texttt{on\_}e_n\texttt{)}$) $=$
  terminating($e_1$) and ... and terminating($e_n$)
terminating($\texttt{if}\;\_\;e$) $=$ terminating($\mathcal{W}\overline{x}\,\overline{d}\,e$) $=$
  terminating($\overline{d}\,e$) $=$ terminating($e$)
terminating(_) $=$ false otherwise

## Atomic Language Terms

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

## Complete Language Syntax

| | | | |
|---|---|---|---|
| $e$ | ::= | $\mathcal{L}$ \| $x$ \| $\pi$ \| $\texttt{void}$ \| $num\;\overline{num\,e}$ \| $e\texttt{"}string\texttt{"}$ \| $\varrho\;e$ \| $x\;eqOp\;e$ \| $unOp\;e$ \| $\mathcal{B}$ \| $e_1\,binOp\,e_2$ \| $e\;\texttt{(}doc\;ps\texttt{)}$ \| $\texttt{if}\;e\;\mathcal{B}\;\overline{\texttt{else}\;e'}$ \| $\texttt{while}\;e\;\mathcal{B}$ \| $\mathcal{W}$ \| $e\texttt{.}m\texttt{(}doc\;ps\texttt{)}$ \| $e\,\texttt{[}doc\;ps_1\texttt{;}\;doc_1...\;ps_n\texttt{;}\;doc_n\texttt{]}$ \| $e\,\texttt{[}doc\;\mathcal{W}\texttt{]}$ \| $e\;doc$ \| $\texttt{loop}\;e$ \| $\texttt{...}$ \| $\texttt{using}\;\pi\;\texttt{check}\;m\texttt{(}doc\;ps\texttt{)}\;e$ | expression |
| $\mathcal{B}$ | ::= | $\texttt{(}doc\;\overline{d}_1\mathcal{K}_1...\overline{d}_n\mathcal{K}_n\;e\texttt{)}$ \| $\texttt{\{}doc\;\overline{d}_1\mathcal{K}_1...\overline{d}_n\mathcal{K}_n\texttt{\}}$ | expr-block |
| $\mathcal{K}$ | ::= | $\texttt{catch}\;\varrho\;x\;\texttt{(}doc\overline{\mathcal{O}}\overline{\texttt{default}\;e}\texttt{)}$ | catch-match |
| $d$ | ::= | $\overline{\texttt{var}}\;\overline{T}\;x\texttt{=}e$ \| $e$<$e \neq x$> \| $C\texttt{:}e$ | statement |
| $\mathcal{W}$ | ::= | $\texttt{with}\;\overline{x}\;\overline{\mathcal{I}}\;\overline{d}\;\texttt{(}\overline{\mathcal{O}^w}\overline{\texttt{default}\;e}\texttt{)}$ \| $\texttt{with}\;\overline{\mathcal{I}}\;\mathcal{B}$ | with |
| $\mathcal{O}^w$ | ::= | $\texttt{on}\;\overline{T}\;e$ \| $\texttt{on}\;\overline{T}\texttt{case}\;e\;\mathcal{B}$ \| $\texttt{case}\;e\;\mathcal{B}$ | type-case |
| $\mathcal{H}$ | ::= | $m\texttt{(}\overline{\mathcal{F}}\texttt{)}$ \| $\texttt{interface}$ \| $\texttt{trait}$ | lib. node h. |
| $\pi$ | ::= | $C\overline{::C}$\|$\texttt{Outer}^n\overline{::C}$\|$\texttt{Any}$\|$\texttt{Void}$\|$\texttt{Library}$ | node path |
| $\mathcal{L}$ | ::= | $\texttt{\{}doc\;\mathcal{H}\;\texttt{<:}\overline{\pi}\overline{\mathcal{M}}\texttt{\}}$ \| $\texttt{\{reuse}\;url\;EOL\;\overline{\mathcal{M}}\;\texttt{\}}$ | library literal |
| $ps$ | ::= | $\overline{e}\;\overline{x\texttt{:}e}$ | parameters |
| $\mathcal{I}$ | ::= | $\overline{\texttt{var}}\;x\;\texttt{in}\;e$ | iterator decl. |
| $\mathcal{O}$ | ::= | $\texttt{on}\;T\;e$ \| $\texttt{on}\;T\;\texttt{case}\;e\;\mathcal{B}$ | signal handler |

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

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

**Definition:** c-f-type$(\mu\;m\texttt{(}\mathcal{F}_1...\mathcal{F}_n\texttt{)}) = \overline{h}$

(1) $\texttt{type method}\;\mu\;\texttt{Outer}_0\;m\texttt{(}$toPh$(T_1\;x_1...\;T_n\;x_n)\texttt{)}\;\texttt{exception}\,\emptyset \in$ c-f-type$(\mu\;m\texttt{(}\overline{\texttt{var}}_1\;T_1\;x_1...\overline{\texttt{var}}_n\;T_n\;x_n\texttt{)})$
(2a) $\texttt{mut method immutable Void}\;x\texttt{(}\;\mu\,\pi\;\texttt{that)}\;\texttt{exception}\,\emptyset \in$ c-f-type$(_\;m\texttt{(}\overline{\mathcal{F}}_1\;\texttt{var}\;\mu\pi\;x\;\overline{\mathcal{F}}_2\texttt{)})$
(2b) $\texttt{mut method immutable Void}\;x\texttt{(}\;\pi\overline{mx}\;\texttt{that)}\;\texttt{exception}\,\emptyset \in$ c-f-type$(_\;m\texttt{(}\overline{\mathcal{F}}_1\;\texttt{var}\;\pi\overline{mx}\;x\;\overline{\mathcal{F}}_2\texttt{)})$
(3) $\texttt{mut method}\;T\;\texttt{\#}x\texttt{()}\;\texttt{exception}\,\emptyset \in$ c-f-type$(_\;m\texttt{(}\overline{\mathcal{F}}_1\;\overline{\texttt{var}}\;T\;x\;\overline{\mathcal{F}}_2\texttt{)})$
(4) $\texttt{read method}$ mut&LentToRead$(\mu\;\pi)\;x\texttt{()}\;\texttt{exception}\,\emptyset \in$ c-f-type$(_\;m\texttt{(}\overline{\mathcal{F}}_1\;\overline{\texttt{var}}\;\mu\;\pi\;x\;\overline{\mathcal{F}}_2\texttt{)})$

**Definition:** $\llbracket e \rrbracket_\Theta$ simple cases, where $\Theta ::= \Gamma;\ T;\ \overline{\mathcal{L}}$

(dw) $\llbracket$ `{reuse` $url\,\overline{\mathcal{M}}$ `}` $\rrbracket_{\_;\_;\overline{\mathcal{L}}} =$ `{`downloadFromWeb($url$) $\overline{\mathcal{M}}'$ `}` iff $\llbracket$ `{` $\overline{\mathcal{M}}$ `}` $\rrbracket_{\emptyset;\_;\{\overline{\mathcal{M}}\}} =$ `{` $\overline{\mathcal{M}}'$ `}`
otherwise $\llbracket$ `{` $\mathcal{H}$ `<:` $\overline{\pi}\,\overline{\mathcal{M}}$ `}` $\rrbracket_{\overline{\mathcal{L}}} =$ `{` $^{\mathcal{M}}\llbracket\mathcal{H}\rrbracket_\Theta$ `<:` $\llbracket\overline{\pi}\rrbracket_\Theta\, {}^{\mathcal{M}}\llbracket\overline{\mathcal{M}}\rrbracket_\Theta$ `}`
with $\Theta = \emptyset$; `immutable Void`; `{` $\mathcal{H}$ `<:` $\overline{\pi}\,\overline{\mathcal{M}}$ `}`, $\overline{\mathcal{L}}$

(cons) $\llbracket$ `{` $\mu\ m($ $\overline{\mathcal{F}}$ $)\ \overline{\mathcal{M}}$ `}` $\rrbracket_\Theta = \llbracket$ `{` c-f-type($\mu\ m($ $\overline{\mathcal{F}}$ $)$ )$\overline{\mathcal{M}}$ `}` $\rrbracket_\Theta$

(lt) $\llbracket nume \rrbracket_\Theta = \llbracket e$`.#numberParser({'@String`$EOL'$ $\llbracket$`'`$num$`'`$\rrbracket EOL$`})`$\rrbracket_\Theta$
$\llbracket e$`"`$char$`"`$\rrbracket_\Theta = \llbracket e$`.#stringParser({'@String`$EOL'$ $\llbracket$`'`$char$`'`$\rrbracket EOL$`})`$\rrbracket_\Theta$
$\llbracket e$`"`$EOLcharEOL$`"`$\rrbracket_\Theta = \llbracket e$`.#stringParser({'@String`$EOL'$ $\llbracket$`'`$charEOL$`'`$\rrbracket EOL$`})`$\rrbracket_\Theta$

(cf) $\llbracket$`while` $e\ \mathcal{B}\rrbracket_\Theta = \llbracket$ `(loop (` $e$`.#checkTrue()` $\mathcal{B}$`) catch exception (on Void void) void)` $\rrbracket_\Theta$
$\llbracket$`if` $a\ \mathcal{B}_1$ `else` $\mathcal{B}_2\rrbracket_\Theta = \llbracket$ `(`$a$`.#checkTrue() catch exception (onVoid` $\mathcal{B}_2$`)` $\mathcal{B}_1$ `void)` $\rrbracket_\Theta$
$\llbracket$`if` $e\ \mathcal{B}_1$ `else` $\mathcal{B}_2\rrbracket_\Theta = \llbracket$ `(` $y$`=`$e$ `if` $y\ \mathcal{B}_1$ `else` $\mathcal{B}_2$`)` $\rrbracket_\Theta$ with $y$ fresh and $e \neq a$
$\llbracket$ `(` $\overline{d}_1\mathcal{K}_1 \ldots \overline{d}_n\mathcal{K}_n\ e$ `)` $\rrbracket_\Theta = \llbracket$ `(` $\overline{d}_1\mathcal{K}_1$ `(`... `(` $\overline{d}_n\mathcal{K}_n\ e$ `)` ...`)` `)` $\rrbracket_\Theta$
$\llbracket$`void`$\rrbracket_\Theta =$ `void` $\quad \llbracket\pi\rrbracket_{\_,\overline{\mathcal{L}}} = {}^{\pi}\llbracket\pi\rrbracket_{\overline{\mathcal{L}}}$ $\quad \llbracket e\ doc\rrbracket_\Theta = \llbracket$ `(` $doc\ e$ `)` $\rrbracket_\Theta$
$\llbracket x\rrbracket_\_ = x$ $\quad \llbracket x$`:=`$e\rrbracket_\Theta = x$`.inner(`$\llbracket e\rrbracket_\Theta$`)` $\quad \llbracket \varrho\ e\rrbracket_\Theta = \varrho\ \llbracket e\rrbracket_\Theta$
$\llbracket$`loop` $e\rrbracket_{\Gamma,T\,\overline{\mathcal{L}}} =$ `loop` $\llbracket e\rrbracket_{\Gamma,\texttt{immutable Void}\,\overline{\mathcal{L}}}$
$\llbracket e$`.`$m$`(`$ps$`)`$\rrbracket_{\Gamma,T\,\overline{\mathcal{L}}} = \llbracket e\rrbracket_{\Gamma,T\,\overline{\mathcal{L}}}$`.`$m$`(`$\llbracket ps\rrbracket_{\Gamma,\mathsf{guessType}_\Gamma(e)::m,\overline{\mathcal{L}}}$`)`
$\llbracket$`using` $\pi$ `check .`$m$`(`$ps$`)` $e\rrbracket_{\Gamma,T,\overline{\mathcal{L}}} =$ `using` $\llbracket\pi\rrbracket_\Theta$ `check .`$m$`(`$\llbracket ps\rrbracket_{\Gamma,\texttt{immutable Void}\overline{\mathcal{L}}}$`)` $\llbracket e\rrbracket_{\Gamma,T,\overline{\mathcal{L}}}$

(rt) $\llbracket$ `{` $\overline{d}_1\mathcal{K}_1 \ldots \overline{d}_n\mathcal{K}_n$ `}` $\rrbracket_{\Gamma,T,\overline{\mathcal{L}}} = \llbracket$ `((` $\overline{d}_1\mathcal{K}_1\ldots\overline{d}_n\mathcal{K}_n$`void) catch return` $y$ `(on` $T\ y$`) error void)` $\rrbracket_{\Gamma,T,\overline{\mathcal{L}}}$

(vd1) $\llbracket e_0\rrbracket_\Theta = \llbracket$ `(` $C$`:{mut (var` $T$ `inner) }`
$\overline{d}_1\ T\ x$`=`$e\,\overline{d}_2\,d'(\overline{d}_3\ \overline{\mathcal{K}}\ e_1)[x\,eqOp := z\,eqOp][x := z$`.#inner()`$]$`)` $\rrbracket_\Theta$
with $e_0 =$ `(` $\overline{d}_1$`var` $T\ x$`=`$e\,\overline{d}_2\,\overline{d}_3\overline{\mathcal{K}}e_1$`)` $\quad d' =$ `mut Outer`$_0$`::`$C$ $z$`=Outer`$_0$`::`$C$ `(inner:`$x$`)`
not $x$`:=_` inside $\overline{d}_2$ and either $\overline{d}_3 = \emptyset$ or $\overline{d}_3 = \overline{d}\_$ and $x$`:=_` inside $\overline{d}$
$\llbracket$`(`$\overline{d}$`var` $x$`=`$e\,\overline{d}'\,\overline{\mathcal{K}}e_1$`)`$\rrbracket_{\Gamma;T';\overline{\mathcal{L}}} = \llbracket$`(`$\overline{d}$`var` $T\ x$`=`$e\,\overline{d}'\,\overline{\mathcal{K}}e_1$`)`$\rrbracket_{\Gamma;T';\overline{\mathcal{L}}}$ with $T = \mathsf{guessType}_{\Gamma,\Gamma\mathsf{of}\overline{d}}(e)$
$\llbracket$ `(` $\overline{d}_1\ e\ \overline{d}_2\overline{\mathcal{K}}e_1$ `)` $\rrbracket_\Theta = \llbracket$ `(` $\overline{d}_1$ `immutable void` $x$`=`$e\,\overline{d}_2\overline{\mathcal{K}}e_1$ `)` $\rrbracket_\Theta$
$\llbracket$ `(` $d_1\ldots d_n\overline{\mathcal{K}}e$ `)` $\rrbracket_{\Gamma;T;\overline{\mathcal{L}}} =$ `(` ${}^{d}\llbracket d_1\rrbracket_\Theta\ldots{}^{d}\llbracket d_n\rrbracket_\Theta\,{}^{\mathcal{K}}\llbracket\overline{\mathcal{K}}\rrbracket_{\Gamma;T;\overline{\mathcal{L}}}\llbracket e\rrbracket_\Theta$ `)`
with $\Theta = \Gamma, \Gamma\mathsf{of}(d_1\ldots d_n);\ T;\ \overline{\mathcal{L}}$ $\quad {}^{d}\llbracket C\!:\!e\rrbracket_\_ = C\!:\!e$ $\quad {}^{d}\llbracket\ T\ x$`=`$e\rrbracket_{\Gamma\_\overline{\mathcal{L}}} =\ T\ x$`=`$\llbracket e\rrbracket_{\Gamma T\overline{\mathcal{L}}}$
${}^{\mathcal{K}}\llbracket$`catch` $\varrho\ \overline{\mathcal{O}}$`default` $e\rrbracket_\Theta = {}^{\mathcal{K}}\llbracket$`catch` $\varrho\ x\ \overline{\mathcal{O}}$`default` $e\rrbracket_\Theta$
${}^{\mathcal{K}}\llbracket$`catch` $\varrho\ x\ \overline{\mathcal{O}}\rrbracket_\Theta = \llbracket$`catch` $\varrho\ x$ `on Any (with` $x$ `(`$\overline{\mathcal{O}}$`default` $\varrho\ x$`) )` $\rrbracket_\Theta$ iff `on` $T$ `if` $e\ \mathcal{B} \in \overline{\mathcal{O}}$
${}^{\mathcal{K}}\llbracket$`catch` $\varrho\ x\ \overline{\mathcal{O}}$ `default` $e\rrbracket_\Theta = \llbracket$`catch` $\varrho\ x$ `on Any (with` $x$ `(`$\overline{\mathcal{O}}$`default` $e$`) )` $\rrbracket_\Theta$ iff `on` $T$ `if` $e\ \mathcal{B} \in \overline{\mathcal{O}}$
otherwise ${}^{\mathcal{K}}\llbracket$`catch` $\varrho\ x\ \mathcal{O}_1\ldots\mathcal{O}_n\rrbracket_\Theta =$ `catch` $\varrho\ x\ {}^{\mathcal{K}}\llbracket\mathcal{O}_1\rrbracket_{x,\Theta}\ldots{}^{\mathcal{K}}\llbracket\mathcal{O}_n\rrbracket_{x,\Theta}$ and
${}^{\mathcal{K}}\llbracket$`catch` $\varrho\ x\ \mathcal{O}_1\ldots\mathcal{O}_n$`default` $e\rrbracket_\Theta =$ `catch` $\varrho\ x\ {}^{\mathcal{K}}\llbracket\mathcal{O}_1\rrbracket_{x,\Theta}\ldots{}^{\mathcal{K}}\llbracket\mathcal{O}_n\rrbracket_{x,\Theta}$`default` $\llbracket e\rrbracket_{x:\texttt{immutable Any},\Theta}$
${}^{\mathcal{K}}\llbracket$`on` $T\ \mathcal{B}\rrbracket_{x,\Theta} =$ `on` $\llbracket T\rrbracket_\Theta\ \llbracket\mathcal{B}\rrbracket_{x:T,\Theta}$(note:`on` $T$ `case` $e\ \mathcal{B}$ is managed in the catch)

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

(init) $\llbracket e$`[`$ps_1$`;` ...`;` $ps_n$`;`]$\rrbracket_\Theta = \llbracket e$`.#begin().#add(`$ps_1$`)`...`.#add(`$ps_n$`).#end()`$\rrbracket_\Theta$

(op) $\llbracket e_1\ binOp\ e_2\rrbracket_{\overline{\mathcal{L}}} = \llbracket e_1$`.`$\llbracket$`'`$binOp$`'`$\rrbracket$`(`$e_2$`)`$\rrbracket_{\overline{\mathcal{L}}}$
$\llbracket x\ eqOp\ e\rrbracket_\Theta = \llbracket x$`:=`$x$`.#inner().`$\llbracket$`'`$eqOp$`'`$\rrbracket$`(`$e$`)`$\rrbracket_\Theta$
$\llbracket unOp\ e\rrbracket_{\overline{\mathcal{L}}} = \llbracket e\rrbracket_{\overline{\mathcal{L}}}$`.`$\llbracket$`'`$unOp$`'`$\rrbracket$`()`
$\llbracket e$`(`$ps$`)`$\rrbracket_{\overline{\mathcal{L}}} = \llbracket e$`.#apply(`$ps$`)`$\rrbracket_{\overline{\mathcal{L}}}$

**Definition:** $\llbracket doc\rrbracket_p$

$\llbracket doc\rrbracket_p$ replaces all substrings of the form `@`$\pi$ and `@(`$e$`)` with `@`$\llbracket\pi\rrbracket_p$ and `@(`$\llbracket e\rrbracket_p$`)`.
This applies to all documentations excluding the one in multi-line string literals.

**Definition:** ${}^{\mathcal{W}}\llbracket e\rrbracket = e$

${}^{\mathcal{W}}\llbracket e\rrbracket$ propagate on the structure, and

(a) ${}^{\mathcal{W}}\llbracket$`with` $\overline{x}\ \overline{\mathcal{I}}\ \overline{d}$ `(`$\overline{\mathcal{O}^w}$`)` $\rrbracket = {}^{\mathcal{W}}\llbracket$`with` $\overline{x}\ \overline{\mathcal{I}}\ \overline{d}$ `(`$\overline{\mathcal{O}^w}$`default void)` $\rrbracket$

(b) ${}^{\mathcal{W}}\llbracket$`with` $\overline{x}\ \overline{d}$ `(`$\overline{\mathcal{O}^w}$ `default` $e_2$`)` $\rrbracket = {}^{\mathcal{W}}\llbracket$ `(` $\overline{d}$ `with`$x_1\ldots x_n$ `(`$\overline{\mathcal{O}^w}$ `default` $e_2$`) )` $\rrbracket$
with `with` $\overline{x}\ \overline{d} =$ `with`$x_1\_\ldots x_n\_$

(c) ${}^{\mathcal{W}}\llbracket$`with`$\overline{x}$ `(`$\overline{\mathcal{O}^w}$`default` $e$`)` $\rrbracket = {}^{\mathcal{W}}\llbracket{}^{\mathcal{W}}\llbracket\overline{\mathcal{O}^w}$`default` $e\rrbracket_{\overline{x}}\rrbracket$

(ca) ${}^{\mathcal{W}}\llbracket$`on` $T_1\ldots T_n$`case` $\overline{e_0}\ e_1\ \overline{\mathcal{O}^w}$ `default` $e\rrbracket_{x_1\ldots x_n} =$ `(` $\overline{e}$ cast$^{T_1}(y_1 \leftarrow x_1)\ldots$cast$^{T_n}(y_n \leftarrow x_n)$
`catch exception Void` ${}^{\mathcal{W}}\llbracket\overline{\mathcal{O}^w}$`default` $e\rrbracket_{x_1\ldots x_n}(e_1[x_1\,T_1 := y_1]\ldots[x_n\,T_n := y_n])$ `void)`
with $y_1\ldots y_n$ fresh and either $\overline{\texttt{case}\ e_0} = \overline{e} = \emptyset$
or $\overline{\texttt{case}\ e_0} =$ `case` $e_0$ and $\overline{e} =$ `with` $x_1\ldots x_n$ `(on` $T_1\ldots T_n$ `(if` $e_0$ `(void) else (exception void) )`

(cb) ${}^{\mathcal{W}}\llbracket$`case` $e_0\ e_1\ \overline{\mathcal{O}}$ `default` $e\rrbracket_{x_1\ldots x_n} =$ `if` $e_0\ e_1$ `else (`${}^{\mathcal{W}}\llbracket\overline{\mathcal{O}}$ `default` $e\rrbracket_{x_1\ldots x_n}$`)`

(cc) ${}^{\mathcal{W}}\llbracket$`default` $e\rrbracket_\_ = e$

(d) ${}^{\mathcal{W}}\llbracket$`with` $\overline{x}\ \overline{\mathcal{I}}\ \overline{d}$ `(`$\overline{\mathcal{O}^w}$ `default` $e_2$`)` $\rrbracket = {}^{\mathcal{W}}\llbracket$`with` $\overline{\mathcal{I}}$ `(` $\overline{d}$ `with`$x_1\ldots x_n$ `(`$\overline{\mathcal{O}^w}$ `default` $e_2$`) )` $\rrbracket$
with `with` $\overline{x}\ \overline{\mathcal{I}}\ \overline{d} =$ `with`$x_1\_\ldots x_n\_$

(e) ${}^{\mathcal{W}}\llbracket$`with` $\overline{\mathcal{I}}\ \mathcal{B}\rrbracket = {}^{\mathcal{W}}\llbracket$declareIts($\overline{\mathcal{I}}$,`(`
`loop(` $d_1\mathcal{K}_1\ldots d_n\mathcal{K}_n\mathcal{B}[x_1$`:=`$x_1$`.#inner()`$]\ldots[x_n$`:=`$x_n$`.#inner()`$]$`)`
`catch exception (on Void void) void)` $)\rrbracket$
with $\overline{\mathcal{I}} = \_\,x_1$ `in` $\_,\ldots\_\,x_n$ `in` $\_$, $\quad d_i\mathcal{K}_i =$ next$_i(x_1\ldots x_n)$

(initW) ${}^{\mathcal{W}}\llbracket e$ `[with` $\overline{x}\ \overline{\mathcal{I}}\ \overline{d}$ `(`$\mathcal{O}^w{}_1\ldots\mathcal{O}^w{}_n$`default` $e$`) ]` $\rrbracket =$
${}^{\mathcal{W}}\llbracket$ `(var` $x$`=`$e$`.#begin() with` $\overline{x}\ \overline{\mathcal{I}}\ \overline{d}$ `(`$\mathcal{O}^{w\prime}{}_1\ldots\mathcal{O}^{w\prime}{}_n$`default` $e'$`)` $x$`.#end() )` $\rrbracket$
with $\mathcal{O}^w{}_i =$ `on` $\overline{T}$ `if`$e\,e$, $\mathcal{O}^{w\prime}{}_i =$ `on` $\overline{T}$ `if`$e$ $x$`:=`$x$`.#add(`$e$`)` and either $\overline{\texttt{default}\ e} = \overline{\texttt{default}\ e'} = \emptyset$
or $\overline{\texttt{default}\ e} =$ `default` $e$ and $\overline{\texttt{default}\ e'} =$ `default` $x$`:=`$x$`.#add(`$e$`)`

**Definition:** $e[x := y]$

$e_0[x := e_1]$ propagate on the structure, and
$(x\,eqOp\,e_0)[x := e] = x\,eqOp(e_0[x := e])$
$x[x := e] = e$

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

$e[x\,\mu$ `Any` $:= y] = e$, otherwise $e[x\,T := y] = e[x := y]$

**Definition:** $\llbracket e\rrbracket_p$ auxiliary definitions

${}^{ps}\llbracket e_0\ \overline{x\!:\!e}\rrbracket_{\Gamma,T::m,\overline{\mathcal{L}}} = {}^{ps}\llbracket$`that:`$e_0\ \overline{x\!:\!e}\rrbracket_{\Gamma,T::m,\overline{\mathcal{L}}}$
${}^{ps}\llbracket x_1\!:\!e_1\ldots x_n\!:\!e_n\rrbracket_{\Gamma,T::m,\overline{\mathcal{L}}} =$
$x_1\!:\!\llbracket e_1\rrbracket_{\Gamma,T::m\,(x_1\ldots x_n)::x_1,\overline{\mathcal{L}}}\cdots$
$x_n\!:\!\llbracket e_n\rrbracket_{\Gamma,T::m,\,(x_1\ldots x_n)::x_n,\overline{\mathcal{L}}}$
${}^{ps}\llbracket x_1\!:\!e_1\ldots x_n\!:\!e_n\rrbracket_{\Gamma,T,\overline{\mathcal{L}}} =$
$x_1\!:\!\llbracket e_1\rrbracket_{\Gamma,T,\overline{\mathcal{L}}}\ldots x_n\!:\!\llbracket e_n\rrbracket_{\Gamma,T,\overline{\mathcal{L}}}$
${}^{\mathcal{M}}\llbracket\mathcal{M}_1\ldots\mathcal{M}_n\rrbracket_p = {}^{\mathcal{M}}\llbracket\mathcal{M}_1\rrbracket_p\ldots{}^{\mathcal{M}}\llbracket\mathcal{M}_n\rrbracket_p$
${}^{\mathcal{M}}\llbracket mh\,e\rrbracket_p = {}^{\mathcal{M}}\llbracket{}^{\mathcal{M}}\llbracket mh\rrbracket_p\ \llbracket e\rrbracket_{\emptyset;\mathsf{Tof}(mh);p}\rrbracket$
${}^{\mathcal{M}}\llbracket mh\mathcal{E}^\star[$ `(`$\overline{d}$`catch exception` $x$ `(`$\overline{\mathcal{O}}$`default` $e$`)` $e_0$`)` $]\rrbracket =$
${}^{\mathcal{M}}\llbracket mh\mathcal{E}^\star[$ `(`$\overline{d}$`catch exception` $x$ `(`$\overline{\mathcal{O}}$`on Any` $e$`)` $e_0$`)` $]\rrbracket$
${}^{\mathcal{M}}\llbracket mh\mathcal{E}^\star[$ `(`$\overline{d}$`catch return` $x$ `(`$\overline{\mathcal{O}}$`default` $e$`)` $e_0$`)` $]\rrbracket =$
${}^{\mathcal{M}}\llbracket mh\mathcal{E}^\star[$ `(`$\overline{d}$`catch return` $x$ `(`$\overline{\mathcal{O}}$`on` $T\ e$`)` $e_0$`)` $]\rrbracket$
where $T$ is obtained using $\mathcal{E}^\star$ as the innermost
`catch`$\_$`return (on` $T\_$`)` or `Any` if no such $\mathcal{E}^\star$ exists
${}^{\mathcal{M}}\llbracket mh\mathcal{E}^\star[$ `(`$\overline{d}_1\ C\!:\!e\,\overline{d}_2\overline{\mathcal{K}}e_0$`)` $]\rrbracket =$
${}^{\mathcal{M}}\llbracket C\!:\!e\rrbracket\,{}^{\mathcal{M}}\llbracket mh\mathcal{E}^\star[$ `(`$\overline{d}_1\,\overline{d}_2\overline{\mathcal{K}}e_0$`)` $]\rrbracket$
otherwise ${}^{\mathcal{M}}\llbracket mh\,e\rrbracket = mh\,e$
${}^{\mathcal{M}}\llbracket C\!:\!\mathcal{E}^\star[\ldots]\rrbracket_p = {}^{\mathcal{M}}\llbracket C\!:\!\mathcal{E}^\star[e]\rrbracket_p$Where $e$ is found
on the local system depending on the original
position of such ... symbol in the source and $C$
${}^{\mathcal{M}}\llbracket\mu$ `method` $T\ m$`(` $\overline{T\ x}$`)` `exception` $\overline{\pi}\rrbracket_p =$
$\mu$ `method` ${}^{\pi}\llbracket T\rrbracket_p\ \llbracket$`'`$m$`'`$\rrbracket$`(`${}^{\pi}\llbracket\ \overline{T\ x}\rrbracket_p$`)` `exception` ${}^{\pi}\llbracket\overline{\pi}\rrbracket_p$
${}^{\mathcal{M}}\llbracket$`method`$m$`(`$\overline{x}$`)`$\rrbracket_p =$ `method`$\llbracket$`'`$m$`'`$\rrbracket$`(`$\overline{x}$`)`
${}^{\mathcal{M}}\llbracket C\!:\rrbracket_p = C\!:$
${}^{\pi}\llbracket C\!::\!C\rrbracket_{\mathcal{L}_0\ldots\mathcal{L}_n} =$ `Outer`$_k$`::`$C\!::\!C$
where $\mathcal{L}_k(::C)$ well defined and
$\forall i < k : \mathcal{L}_i(::C)$ not well defined
${}^{\pi}\llbracket C\!::\!C\rrbracket_{\mathcal{L}_0\ldots\mathcal{L}_n} =$ `Outer`$_n$`::`$C\!::\!C$
where $\forall i \in 0..n : \mathcal{L}_i(::C)$ not well defined

**Definition:** Tof

Tof($C$:) = `immutable Library`
Tof($\mu$ `method` $T\ m$`(` $\overline{T\ x}$`)` `exception` $\overline{\pi}$) = $T$
Tof(`method`$m$`(`$\overline{x}$`)` ) = `Outer`$_0$`.`$m$`(`$\overline{x}$`)`

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

declareIts($\emptyset$, $e$) = $e$
declareIts(`var` $x_0$ `in` $e_0\overline{\mathcal{I}}$, $e$) = `(`
$x_0$`=`$e_0$ `((`declareIts($\overline{\mathcal{I}}$, $e$)
`catch exception` $y$ `(default (`$x_0$`.#close() exception` $y$`) )`
`void)`
`catch return` $y'$ `(default (`$x_0$`.#close() exception` $y'$`) )`
$x_0$`.#close() )`

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

next$_i(z_0\ldots z_n) = z_i$`.#next()`
`catch exception (on Void (`
`(`$z_{i+1}$`.#next() catch exception (on Void void) void)`
... `(`$z_n$`.#next() catch exception (on Void void) void)`
`(`$z_0$`.#checkEnd() catch exception (on Void void) void)`
... `(`$z_n$`.#checkEnd() catch exception (on Void void) void)`
`exception void) )`

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

cast$^{\mu\ \pi}(y \leftarrow x) = \mu\ \pi\ y$`= (return` $x$
`catch return` $z$ `(on` $\mu\ \pi\ z$ `on` $\mu$ `Any exception void)`
`error void)` $\quad$ with $z$ fresh

**Definition:** xsOf($ps$)

$e_0 x_1 : e_1 \ldots x_n : e_n =$ `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:** $\mathcal{L}_1 \oplus \mathcal{L}_2 = \mathcal{L}$

$\{\mathcal{H}_1 \texttt{<:} \overline{\pi}_1 \overline{\mathcal{M}}_1\} \oplus \{\mathcal{H}_2 \texttt{<:} \overline{\pi}_2\overline{\mathcal{M}}_2\} = \{\mathcal{H}_1 \oplus \mathcal{H}_2 \texttt{<:} \overline{\pi}_1\overline{\pi}_2 \overline{\mathcal{M}}_1 \oplus \overline{\mathcal{M}}_2\}$

$\texttt{interface} \oplus \texttt{interface} = \texttt{interface}$

$\emptyset \oplus \emptyset = \emptyset$

$\texttt{interface} \oplus \emptyset = \emptyset \oplus \texttt{interface} = \emptyset$

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

$C\texttt{:}doc_1 \mathcal{L}_1 \oplus C\texttt{:}doc_2\mathcal{L}_2\overline{\mathcal{M}} = C\texttt{:}(doc_1 \oplus doc_2 \mathcal{L}_1 \oplus \mathcal{L}_2)\overline{\mathcal{M}}$

$h_1\overline{e} \oplus h_2\overline{\mathcal{M}} = h_1 \oplus h_2\overline{e}\overline{\mathcal{M}} = (h_1 \oplus h_2)\overline{e}\overline{\mathcal{M}}$

where $h_1, h_2$ differs only for the documentation

$\texttt{method}\, doc\, mx\, e$ 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 $\mathcal{L}(\pi) =$

$\{\mathcal{H}\overline{\mathcal{M}}_1 \texttt{method Void}\ m\ (\texttt{Void}\ x_1 ... \texttt{Void}\ x_n)\ (\texttt{this}.m'(x'_1\texttt{:}x_1 ... x'_n\texttt{:}x_n))\ \overline{\mathcal{M}}_2\}$

**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

$(C\texttt{:}e)[\mathsf{renUsage}_{\ddot{\mathcal{L}}p}\overline{\pi.mx\mapsto mx'}] =$

$C\texttt{:}(e[\mathsf{renUsage}_{\ddot{\mathcal{L}}[C:\updownarrow]p}\overline{\pi.mx\mapsto mx'}])$,

$\{\mathcal{H}\overline{\mathcal{M}}\}[\mathsf{renUsage}_p \overline{\pi.mx\mapsto mx'}] = \{\mathcal{H}[\mathsf{renUsage}_{\widehat{\mathcal{L}}p}\overline{\texttt{Outer}_1\texttt{::}\pi.mx\mapsto mx'}]$

$\overline{\mathcal{M}}[\mathsf{renUsage}_{\widehat{\mathcal{L}}p}\overline{\texttt{Outer}_1\texttt{::}\pi.mx\mapsto mx'}]\}$(first time not add outer1)

with $p;\emptyset \vdash \{\mathcal{H}\overline{\mathcal{M}}\} \to \widehat{\mathcal{L}}$ and

$e.m(x_1\texttt{:}e_1...x_n\texttt{:}e_n)\ [\mathsf{renUsage}_{\Gamma,p}\overline{\pi.mx\mapsto mx'}] =$

$e.m'(x'_1\texttt{:}e_1...x'_n\texttt{:}e_n)\ [\mathsf{renUsage}_{\Gamma,p}\overline{\pi.mx\mapsto mx'}]$

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:** $\mathcal{L}[\mathsf{redirectDefinition}\,\pi\mapsto\pi'] = \mathcal{L}'$

$\mathcal{L}[\mathsf{redirectDefinition}\,\pi\mapsto\texttt{Library}] = \mathcal{L}[\mathsf{redirectDefinition}\,\pi\mapsto\texttt{Void}] =$

$\mathcal{L}[\mathsf{redirectDefinition}\,\pi\mapsto\texttt{Any}] = \mathcal{L}[\mathsf{redirectDefinition}\,\pi\mapsto\texttt{Outer}_{n+1}\texttt{::}\_] = \{\,\}$

$\mathcal{L}[\mathsf{redirectDefinition}\,\texttt{Outer}_0\texttt{::}\overline{C}_0\mapsto\texttt{Outer}_0\texttt{::}\overline{C}_1] =$

$\mathcal{L}(\overline{C}_0)[\mathsf{from}\ \overline{C}_0\pi'][\mathsf{encapsulateIn}\,\overline{C}_1]$ iff $[\mathsf{to}\,\overline{C}_1] = \texttt{Outer}_1\texttt{::}\pi'\texttt{::}C'$

otherwise $\mathcal{L}[\mathsf{redirectDefinition}\,\texttt{Outer}_0\texttt{::}\overline{C}_0\mapsto\texttt{Outer}_0\texttt{::}\overline{C}_1] =$

$\mathcal{L}(\overline{C}_0)[\mathsf{remove\ n\ outers}][\mathsf{encapsulateIn}\,\overline{C}_1]$ if $[\mathsf{to}\,\overline{C}_1] = \texttt{Outer}_0\texttt{::}C_1\texttt{::}...\texttt{::}C_n$

**Definition:** $\overline{C}_0[\mathsf{to}\,\overline{C}_1] = \pi$

$\overline{C}_0[\mathsf{to}\,\emptyset] = \texttt{Outer}_0\texttt{::}\overline{C}_0$

$C\texttt{::}\overline{C}_0[\mathsf{to}\,C\texttt{::}\overline{C}_0] = \overline{C}_0[\mathsf{to}\,\overline{C}_0]$

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

**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

$(C\texttt{:}e)[\mathsf{renUsage}_{\ddot{\mathcal{L}}p}\,\pi]\pi' =$

$C\texttt{:}(e[\mathsf{renUsage}_{\ddot{\mathcal{L}}[C:\updownarrow]p}\,\pi]\pi')$,


## 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:** $\mathcal{L}_1 \oplus \mathcal{L}_2 = \mathcal{L}$

$\{\mathcal{H}_1 \texttt{<:} \overline{\pi}_1 \overline{\mathcal{M}}_1\} \oplus \{\mathcal{H}_2 \texttt{<:} \overline{\pi}_2\overline{\mathcal{M}}_2\} = \{\mathcal{H}_1 \oplus \mathcal{H}_2 \texttt{<:} \overline{\pi}_1\overline{\pi}_2 \overline{\mathcal{M}}_1 \oplus \overline{\mathcal{M}}_2\}$

$\texttt{interface} \oplus \texttt{interface} = \texttt{interface}$

$\emptyset \oplus \emptyset = \emptyset$

$\texttt{interface} \oplus \emptyset = \emptyset \oplus \texttt{interface} = \emptyset$

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

$C\texttt{:}doc_1 \mathcal{L}_1 \oplus C\texttt{:}doc_2\mathcal{L}_2\overline{\mathcal{M}} = C\texttt{:}(doc_1 \oplus doc_2 \mathcal{L}_1 \oplus \mathcal{L}_2)\overline{\mathcal{M}}$

$h_1\overline{e} \oplus h_2\overline{\mathcal{M}} = h_1 \oplus h_2\overline{e}\overline{\mathcal{M}} = (h_1 \oplus h_2)\overline{e}\overline{\mathcal{M}}$

where $h_1, h_2$ differs only for the documentation

$\texttt{method}\, doc\, mx\, e$ 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 $\mathcal{L}(\pi) =$

$\{\mathcal{H}\overline{\mathcal{M}}_1 \texttt{method Void}\ m\ (\texttt{Void}\ x_1 ... \texttt{Void}\ x_n)\ (\texttt{this}.m'(x'_1\texttt{:}x_1 ... x'_n\texttt{:}x_n))\ \overline{\mathcal{M}}_2\}$

**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

$(C\texttt{:}e)[\mathsf{renUsage}_{\ddot{\mathcal{L}}p}\overline{\pi.mx\mapsto mx'}] =$

$C\texttt{:}(e[\mathsf{renUsage}_{\ddot{\mathcal{L}}[C:\updownarrow]p}\overline{\pi.mx\mapsto mx'}])$,

$\{\mathcal{H}\overline{\mathcal{M}}\}[\mathsf{renUsage}_p \overline{\pi.mx\mapsto mx'}] = \{\mathcal{H}[\mathsf{renUsage}_{\widehat{\mathcal{L}}p}\overline{\texttt{Outer}_1\texttt{::}\pi.mx\mapsto mx'}]$

$\overline{\mathcal{M}}[\mathsf{renUsage}_{\widehat{\mathcal{L}}p}\overline{\texttt{Outer}_1\texttt{::}\pi.mx\mapsto mx'}]\}$(first time not add outer1)

with $p;\emptyset \vdash \{\mathcal{H}\overline{\mathcal{M}}\} \to \widehat{\mathcal{L}}$ and

$e.m(x_1\texttt{:}e_1...x_n\texttt{:}e_n)\ [\mathsf{renUsage}_{\Gamma,p}\overline{\pi.mx\mapsto mx'}] =$

$e.m'(x'_1\texttt{:}e_1...x'_n\texttt{:}e_n)\ [\mathsf{renUsage}_{\Gamma,p}\overline{\pi.mx\mapsto mx'}]$

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:** $\mathcal{L}[\mathsf{redirectDefinition}\,\pi\mapsto\pi'] = \mathcal{L}'$

$\mathcal{L}[\mathsf{redirectDefinition}\,\pi\mapsto\texttt{Library}] = \mathcal{L}[\mathsf{redirectDefinition}\,\pi\mapsto\texttt{Void}] =$

$\mathcal{L}[\mathsf{redirectDefinition}\,\pi\mapsto\texttt{Any}] = \mathcal{L}[\mathsf{redirectDefinition}\,\pi\mapsto\texttt{Outer}_{n+1}\texttt{::}\_] = \{\,\}$

$\mathcal{L}[\mathsf{redirectDefinition}\,\texttt{Outer}_0\texttt{::}\overline{C}_0\mapsto\texttt{Outer}_0\texttt{::}\overline{C}_1] =$

$\mathcal{L}(\overline{C}_0)[\mathsf{from}\ \overline{C}_0\pi'][\mathsf{encapsulateIn}\,\overline{C}_1]$ iff $[\mathsf{to}\,\overline{C}_1] = \texttt{Outer}_1\texttt{::}\pi'\texttt{::}C'$

otherwise $\mathcal{L}[\mathsf{redirectDefinition}\,\texttt{Outer}_0\texttt{::}\overline{C}_0\mapsto\texttt{Outer}_0\texttt{::}\overline{C}_1] =$

$\mathcal{L}(\overline{C}_0)[\mathsf{remove\ n\ outers}][\mathsf{encapsulateIn}\,\overline{C}_1]$ if $[\mathsf{to}\,\overline{C}_1] = \texttt{Outer}_0\texttt{::}C_1\texttt{::}...\texttt{::}C_n$

**Definition:** $\overline{C}_0[\mathsf{to}\,\overline{C}_1] = \pi$

$\overline{C}_0[\mathsf{to}\,\emptyset] = \texttt{Outer}_0\texttt{::}\overline{C}_0$

$C\texttt{::}\overline{C}_0[\mathsf{to}\,C\texttt{::}\overline{C}_0] = \overline{C}_0[\mathsf{to}\,\overline{C}_0]$

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

**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

$(C\texttt{:}e)[\mathsf{renUsage}_{\ddot{\mathcal{L}}p}\,\pi]\pi' =$

$C\texttt{:}(e[\mathsf{renUsage}_{\ddot{\mathcal{L}}[C:\updownarrow]p}\,\pi]\pi')$,