Language 42

for more information `L42.is`

Core Language Syntax

$\mathcal{L}$	$::= \texttt{\{}\, doc\ \mathcal{H}\texttt{<:}\,\overline{\pi}\ doc'\,\overline{\mathcal{M}}\,\texttt{\}}^{\overbracket{\circledcirc}} \mid \updownarrow$	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
ϱ	$::= \texttt{exception} \mid \texttt{error} \mid \texttt{return}$	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
π	$::= \texttt{Outer}^{n}\overline{\texttt{::}C} \mid \texttt{Any} \mid \texttt{Void} \mid \texttt{Library}$	path
α	$::= \emptyset \mid \texttt{\^{}} \mid \%$	ph annotation
T	$::= \mu\ \pi\ \alpha \mid \pi\overline{mx}\alpha$	type annotation
mx	$::= \texttt{::}m\texttt{(}\overline{x}\texttt{)}\overbracket{\texttt{::}x}$	typeLink
μ	$::= \texttt{immutable} \mid \texttt{mut} \mid \texttt{read} \mid \texttt{lent} \mid \texttt{capsule} \mid \texttt{type}$	modifiers
$\circledcirc$	$::= \ominus \mid \oplus \mid \circledast$	stage

Auxiliary syntax

$\mathcal{E}^{\mathsf{c}}\ ::= \square \mid \mathcal{E}^{\mathsf{c}}\texttt{.}m\texttt{(}\overline{x\texttt{:}e}\texttt{)} \mid e_0{}^{\mathsf{c}}\texttt{.}m\texttt{(}\overline{x\texttt{:}e^{\mathsf{c}}}_1 x\texttt{:}\mathcal{E}^{\mathsf{c}}\overline{x\texttt{:}e}_2\texttt{)} \mid \texttt{loop}\ \mathcal{E}^{\mathsf{c}} \mid \varrho\ \mathcal{E}^{\mathsf{c}}$

$\mid \texttt{(}\overline{d^{\mathsf{c}}}_1\ T\ x\texttt{=}\mathcal{E}^{\mathsf{c}}\ \overline{d}_2\overline{\mathcal{K}}\ e\texttt{)} \mid \texttt{(}\overline{d^{\mathsf{c}}}\texttt{catch}\ \varrho\ x\ \texttt{on}\ T\ \mathcal{E}^{\mathsf{c}}\ e\texttt{)} \mid \texttt{(}\overline{d^{\mathsf{c}}}\overline{\mathcal{K}^{\mathsf{c}}}\mathcal{E}^{\mathsf{c}}\texttt{)}$

$\texttt{using}\ \pi\ \texttt{check}\ \texttt{.}m\texttt{(}\overline{e^{\mathsf{c}}}_1\mathcal{E}^{\mathsf{c}}\overline{e}_2\texttt{)}\ e \mid \texttt{using}\ \pi\ \texttt{check}\ \texttt{.}m\texttt{(}\overline{e^{\mathsf{c}}}\texttt{)}\ \mathcal{E}^{\mathsf{c}}$

$\mathcal{E}^{\star}\ ::= \square \mid \mathcal{E}^{\star}\texttt{.}m\texttt{(}\overline{e}\texttt{)} \mid e_0\texttt{.}m\texttt{(}\overline{x\texttt{:}e}_1 x\texttt{:}\mathcal{E}^{\star}\overline{x\texttt{:}e}_2\texttt{)} \mid \texttt{loop}\ \mathcal{E}^{\star} \mid \varrho\ \mathcal{E}^{\star}$

$\mid \texttt{(}\overline{d}_1\ T\ x\texttt{=}\mathcal{E}^{\star}\ \overline{d}_2\overline{\mathcal{K}}\ e\texttt{)} \mid \texttt{(}\overline{d}\texttt{catch}\ \varrho\ x\ \texttt{on}\ T\ \mathcal{E}^{\star}\ e\texttt{)} \mid \texttt{(}\overline{d}\overline{\mathcal{K}}\mathcal{E}^{\star}\texttt{)}$

$\texttt{using}\ \pi\ \texttt{check}\ \texttt{.}m\texttt{(}\overline{x\texttt{:}e}_0 x\texttt{:}\mathcal{E}^{\star}\overline{x\texttt{:}e}_1\texttt{)}\ e \mid \texttt{using}\ \pi\ \texttt{check}\ \texttt{.}m\texttt{(}x\texttt{:}\overline{v}\texttt{)}\ \mathcal{E}^{\star}$

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

$dv^p ::= \mu\ \pi'\ x\ \texttt{=}\pi\texttt{.}m\ \texttt{(}x_1\texttt{:}a_1 ... x_n\texttt{:}a_n\texttt{)} \mid \texttt{immutable}\ \pi\ x\ \texttt{=}\texttt{(}\overline{dv^p}\, v^p\texttt{)}$

<if $\mathsf{meth}_p(\pi\texttt{.}m\ \texttt{(}x_1 ... x_n\texttt{)}) = h\ \texttt{field}$,

$\mu \neq \texttt{capsule}$ and $p(\pi')$ not interface>

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

$\mid v_0\texttt{.}m\texttt{(}\overline{x\texttt{:}v}\ x\texttt{:}\mathcal{E}^p\ \overline{x\texttt{:}e}\texttt{)} \mid \texttt{using}\ \pi\ \texttt{check}\ \texttt{.}m\texttt{(}\overline{x\texttt{:}v}x\texttt{:}\mathcal{E}^p\overline{x\texttt{:}e}\texttt{)}\ e$

$\mid \texttt{using}\ \pi\ \texttt{check}\ \texttt{.}m\texttt{(}\overline{x\texttt{:}v}\texttt{)}\ \mathcal{E}^p$

$p ::= \mathcal{L}_0, ..., \mathcal{L}_n\overbracket{\circledast}$ 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: $\mathsf{compiled}(\_)$

$\mathsf{compiled}(e)$ iff $\mathsf{compiled}(\mathcal{L})$ holds $\forall\mathcal{L}$ inside e

$\mathsf{compiled}(\texttt{\{}\mathcal{H}\texttt{<:}\,\overline{\pi}\overline{\mathcal{M}}\texttt{\}}^{\overbracket{\circledcirc}})$ 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^{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: $\Gamma(x),\quad \overline{d}(x),\quad \mathcal{L}(C),\quad \mathcal{L}(mx),\quad p(\pi)$

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

$\overline{d}(x):\ (\overline{d}_1\ \overbracket{T}\ x\texttt{=}e\ \overline{d}_2)(x) = e$

$\mathcal{L}(\_):$ extract the corresponding element in $\mathcal{L}$

$p(\pi):\ (\mathcal{L}_0 ... \mathcal{L}_n)(\pi) = \mathcal{L}_i(\overline{\texttt{::}C})$ if $\mathsf{norm}_p(\pi) = \texttt{Outer}_i\overline{\texttt{::}C}$

$\mathcal{L}(\overline{\texttt{::}C}):\ \mathcal{L}(\texttt{::}C_1 ... \texttt{::}C_n) = \mathcal{L}(C_1)...(C_n)$ where $e(C) = e$ iff e not $\mathcal{L}$

$(\Gamma; \overline{x}; \overline{x}_1 ... \overline{x}_n)$, $\overline{dv}$ and $\overline{\mathcal{M}}^{\mathsf{c}}$ 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: $\mathcal{L}[\mathcal{M}]$

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

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

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 $\overbracket{\_}$ to denote optionality; for example $\overbracket{T}$ and $\overbracket{\texttt{var}}$ 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 $\texttt{(}\ T\ x\texttt{=(}\ T\ y\texttt{=}e\,y\texttt{)}\ x\texttt{)}$ are equivalent to the corresponding $\texttt{(}\ T\ y\texttt{=}e\,y\texttt{)}$ 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: $\pi_0[\mathsf{from}\ \pi_1] = \pi_2$

$\texttt{Outer}^{n}\overline{\texttt{::}C}[\mathsf{from}\ \texttt{Outer}^{m}\texttt{::}C_1 ... \texttt{::}C_k] = \texttt{Outer}^{m}\texttt{::}C_1 ... \texttt{::}C_{k-n}\overline{\texttt{::}C}$ if $n \leq k$

$\texttt{Outer}^{n}\overline{\texttt{::}C}[\mathsf{from}\ \texttt{Outer}^{m}\texttt{::}C_1 ... \texttt{::}C_k] = \texttt{Outer}^{m+n-k}\overline{\texttt{::}C}$ 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 $\mathcal{L}[\mathsf{from}\ \pi] = \mathcal{L}[\mathsf{from}\ \pi]_0$

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

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

$\texttt{Outer}^{n}\overline{\texttt{::}C}[\mathsf{from}\ \pi]_j = \texttt{Outer}^{n}\overline{\texttt{::}C}$ with $n < j$

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

All cases for other expressions/terms propagate to submembers

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

with $\overline{\mathcal{K}} = \texttt{catch error}x\ \overline{\mathcal{O}}$ and $\Sigma = \_; \_; \overline{x}_1 ... \overline{x}_n$

$\Gamma'(x) = \Gamma(x)$ iff $\forall\overline{x}_i$ such that $x \in \overline{x}_i, \overline{x} \cap \mathsf{FV}(\overline{\mathcal{K}}) = \emptyset$

$\Gamma'(x) = \mathsf{mutableLentToReadable}(\Gamma(x))$ otherwise

otherwise $\Gamma' = \Gamma$

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

$\overline{T}; \overline{\pi}[\texttt{catch return}\ \mathcal{O}_1 ... \mathcal{O}_n] = \overline{T}[\mathcal{O}_1]...[\mathcal{O}_n]; \overline{\pi}$

$\mu\pi_1 ... \mu\ \pi_n[\texttt{on}\ \mu'\ \pi_0\_] = \mu'\ \pi_0 ... \mu'\ \pi_n$ if $\mu \leq \mu'$

otherwise $\mu\pi_1 ... \mu\ \pi_n[\texttt{on}\ \mu'\ \pi_0\_] = \mu'\ \pi_0$

$\overline{T}; \overline{\pi}[\texttt{catch exception}\ x\texttt{on immutable}\ \pi_1\_ ... \texttt{on immutable}\ \pi_n\_] = \overline{T}; \overline{\pi}, \pi_1 ... \pi_n \setminus \texttt{Any}$

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 $\mathsf{norm}_p(\pi') \in \mathsf{norm}_p(\overline{\pi}[\mathsf{from}\ \pi] \cup \pi) \cup \texttt{Any}$

with $p(\pi) = \texttt{\{\_\}}^{-;\overline{\pi}}$

$\texttt{capsule} \leq \texttt{mut} \leq \texttt{lent} \leq \texttt{read},\quad \texttt{capsule} \leq \texttt{immutable} \leq \texttt{read}$ 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: $\mathsf{complete}(\Gamma)$, $\mathsf{dom}^{\mathsf{mut}}(\Gamma)$, $\mathsf{dom}^{\mathsf{mut}\leq}(\Gamma)$, $\mathsf{mutTolent}(T)$

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

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

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

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

$\mathsf{mutTolent}(\mu\ \pi\alpha) = \mu\ \pi\alpha$ otherwise

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$\mathsf{coherent}_p(\texttt{\{interface\_\}}^{\overline{\circledcirc}})$ holds

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Definitions2

Definition: $\mathsf{meth}_p(\pi\texttt{.}m\,\texttt{(}\overline{x}\texttt{)}\,)$

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

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

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

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

$\mathsf{norm}_p(\mu\ \pi\alpha) = \mu\ \mathsf{norm}_p(\pi)\alpha$

$\mathsf{norm}_p(\pi' mx\texttt{\^{}}) = \mu\ \pi\texttt{\^{}}$ iff $\mathsf{norm}_p(\pi' mx) = \mu\ \pi\alpha$

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

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

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

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

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

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

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

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

$= \mu\ \texttt{method}\ T'_0\ m\texttt{(}\ T'_1\ x_1 \ldots\ T'_n\ x_n\texttt{)}\ \texttt{exception}\ \mathsf{norm}_p(\overline{\pi})\mathsf{norm}_p(\overline{e})$

with $T'_i = \mathsf{norm}_p(T_i)$

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

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

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

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

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

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

$\mathsf{exeOk}^{\oplus}(p, x_1\texttt{:}\_\ \pi_1\_ \ldots x_n\texttt{:}\_\ \pi_n\_)$

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

Definition: $\mathsf{toPartial}(\_)$, $\mathsf{toPh}(\_)$

$\mathsf{toPartial}(\mu\ \pi) = \mu\ \pi\%$

$\mathsf{toPartial}(\mu\ \pi\alpha) = \mu\ \pi\alpha$ otherwise

$\mathsf{toPh}(\mu\ \pi\_) = \mu\ \pi\texttt{\^{}}$ and $\mathsf{toPh}(\pi\overline{mx}\_) = \pi\overline{mx}\texttt{\^{}}$

those notions trivially extends to Γ

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

$\mathsf{throws}_p(\varrho\ v) = \varrho\ v$

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

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

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

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

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

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

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

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

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

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

$\pi \in \mathsf{used}(mh\ e)$ iff $\pi \in \mathsf{used}(e)$

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

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

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

with x as implicit reduction step identifier

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

are they used somewhere?

Extraction of types

(ET-+)

$$\frac{\mathcal{L}_1 \xrightarrow{p} \ldots \xrightarrow{p} \mathcal{L}_2 \qquad \mathcal{L}_2 \not\xrightarrow{p} \_}{\mathcal{L}_1 \underset{\max}{\xrightarrow{p}} \mathcal{L}_2}$$

(ET-DEEP)

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

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

(ET-SUB)

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

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

(ET-LABEL)

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

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

(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)

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

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

(CHECK-CT2)

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

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

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

Type for expressions

(PATH-PATH)

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

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

(LIB-T)

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

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

(METH-UNKNOWN-T)

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

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

(USING-T)

$$\frac{p; \Gamma; \Sigma; \Phi \vdash e_i : T_i' \leq T_i \text{ for all } i \in 0..n}{p; \Gamma; \Sigma; \Phi \vdash \texttt{using}\ \pi\ \texttt{check .}m\texttt{(}\overline{x{:}e}\texttt{)}\ e_0 : T_0}$$

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

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

(T-METHCALL)

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

with
$\mathsf{meth}_p(\pi_0\texttt{.}m, x_1 \ldots x_n) =$
$\mu_0\ \texttt{method}\ T\ m\texttt{(}T_1\ x_1 \ldots T_n\ x_n\texttt{)}\ \texttt{exception}\ \overline{\pi}\ \_$
$p \vdash \overline{\pi} \leq \Phi$
$\Gamma_i \subseteq \Gamma\ \forall i \in 0..n$
$\Gamma_i(x) = \Gamma_j(x) = \texttt{capsule}$ __ 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
$\Sigma = \overline{x}; \overline{x}_1 \ldots \overline{x}_n; \overline{\overline{x}}$
$\Sigma' = \overline{x}; \overline{x}_1 \ldots \overline{x}_n, \mathsf{dom}^{\mathsf{mut}}(\Gamma) \backslash \overline{x}_1 \ldots \overline{x}_n; \overline{\overline{x}}$

(T-READABLE)

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

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

(T-BLOCK-COMPLETE/PARTIAL)

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

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)

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

(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)

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

with

$\overline{dv} \neq \emptyset$

$\mathsf{FV}(\,\mathtt{(}\overline{d}e\mathtt{)}) \cap \mathsf{dom}(\overline{dv}) = \emptyset$

(METHCALL)

$$\frac{}{\mathcal{E}^p[v\mathtt{.}m\ \mathtt{(}x_1\mathtt{:}v_1 ... x_n\mathtt{:}v_n\mathtt{)}\,] \longrightarrow_p \mathcal{E}^p[\,\mathtt{(}\mu_0\ \pi\ \mathtt{this}\ \mathtt{=}v\ T_1\ x_1\ \mathtt{=}v_1 ... T_n\ x_n\ \mathtt{=}v_n e\mathtt{)}\,]}$$

with

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

$\mathsf{meth}_p(\pi\mathtt{.}m\ \mathtt{(}x_1 ... x_n\mathtt{)}\,) = \mu_0\ \mathtt{method}\ T\ m\mathtt{(}T_1\ x_1 ... T_n\ x_n\mathtt{)}\ \mathtt{exception}\,\_\ e$

(PRIMCALLREC)

$$\frac{}{\mathcal{E}^p[v\mathtt{.}m\ \mathtt{(}x_1\mathtt{:}v_1 ... x_n\mathtt{:}v_n\mathtt{)}\,] \longrightarrow_p \mathcal{E}^p[\,\mathtt{(}\mu\ \pi\ z\ \mathtt{=}v z\mathtt{.}m\ \mathtt{(}x_1\mathtt{:}v_1 ... x_n\mathtt{:}v_n\mathtt{)}\,\mathtt{)}\,]}$$

with

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

$\mathsf{meth}_p(\pi\mathtt{.}m\ \mathtt{(}x_1 ... x_n\mathtt{)}\,) = \mu\ \mathtt{method}\ T\ m\mathtt{(}T_1\ x_1 ... T_n\ x_n\mathtt{)}\ \mathtt{exception}\,\_$

v is a block

(PRIMCALLARG)

$$\frac{}{\mathcal{E}^p[v_0\mathtt{.}m\ \mathtt{(}\overline{x\mathtt{:}a} x_i\mathtt{:}v\overline{x\mathtt{:}v}\mathtt{)}\,] \longrightarrow_p \mathcal{E}^p[\,\mathtt{(}T'_i\ z\ \mathtt{=}v_i v_0\mathtt{.}m\ \mathtt{(}\overline{x\mathtt{:}a} x_i\mathtt{:}z\overline{x\mathtt{:}v}\mathtt{)}\,\mathtt{)}\,]}$$

with

v is a block

$\mathsf{class}(\mathcal{E}^p, v_0) = \pi$

$x_1\mathtt{:}\_ ... x_n\mathtt{:}\_ = \overline{x\mathtt{:}a} x_i\mathtt{:}v\overline{x\mathtt{:}v}$

$\mathsf{meth}_p(\pi\mathtt{.}m\ \mathtt{(}x_1 ... x_n\mathtt{)}\,) = \mu\ \mathtt{method}\ T\ m\mathtt{(}T_1\ x_1 ... T_n\ x_n\mathtt{)}\ \mathtt{exception}\,\_$

$T'_i = \mu_i\ \pi_i$ and $T_i = \mu_i\ \pi_i\_$

(FIELDAOBJ)

$$\frac{}{\mathcal{E}^p[x\mathtt{.}m\ \mathtt{()}\,] \longrightarrow_p \mathcal{E}^p[a_i]}$$

with

$\mathsf{dec}(\mathcal{E}^p, x) = \_\ x\ \mathtt{=}\pi\mathtt{.}m'\ \mathtt{(}x_1\mathtt{:}a_1 ... x_n\mathtt{:}a_n\mathtt{)}$

$m = x_i$ or $m = \mathtt{\#}x_i$

$\mathsf{meth}_p(\pi\mathtt{.}m\mathtt{()}) = \mu\ \mathtt{method}\ T\ m\mathtt{()}\ \mathtt{exception}\,\_$

(FIELDABLOCK)

$$\frac{}{\mathcal{E}^p[x\mathtt{.}m\ \mathtt{()}\,] \longrightarrow_p \mathcal{E}^p[\,\mathtt{(immutable}\ \pi'\ z\ \mathtt{=}\ \mathtt{(}\overline{dv} v\mathtt{.}m\ \mathtt{()}\,\mathtt{)}\ z\mathtt{)}\,]}$$

with

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

$\mathsf{meth}_p(\pi\mathtt{.}m\mathtt{()}) = \mu\ \mathtt{method}\,\_\ \pi'\ m\mathtt{()}\ \mathtt{exception}\,\_$

(LOOP)

$$\frac{}{\mathcal{E}^p[\mathtt{loop}\ e] \longrightarrow_p \mathcal{E}^p[e']}$$

with

$e' = \mathtt{(}\ \mathtt{immutable}\ \mathtt{Void}\ x\mathtt{=}e$

$\mathtt{loop}\ e\mathtt{)}$

(BLOCKELIM)

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

with

$\mu \geq \mathtt{mut}$

(SUBST)

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

with

either $v = a$ or $\mu = \mathtt{capsule}$

(NORMALDEC)

$$\frac{}{\mathcal{E}^p[\,\mathtt{(}\overline{dv}'\ \mu\ \pi\alpha\ x\ \mathtt{=}e'\ \overline{d}e\mathtt{)}\,] \longrightarrow_p \mathcal{E}^p[\,\mathtt{(}\overline{dv}'\ dv\ \overline{d}e\mathtt{)}\,]}$$

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)

$$\frac{}{\mathcal{E}^p[\pi\mathtt{.}m\ \mathtt{(}x_1\mathtt{:}a_1 ... x_n\mathtt{:}a_n\mathtt{)}\,] \longrightarrow_p \mathcal{E}^p[\,\mathtt{(}\mu\ \pi\ z\ \mathtt{=}\pi\mathtt{.}m\ \mathtt{(}x_1\mathtt{:}a_1 ... x_n\mathtt{:}a_n\mathtt{)}\ z\mathtt{)}\,]}$$

with

$\mathsf{meth}_p(\pi\mathtt{.}m\ \mathtt{(}x_1 ... x_n\mathtt{)}\,) = \mathtt{type}\ \mathtt{method}\ \mu\ \pi\ m\mathtt{(\_)}\ \mathtt{exception}\,\_$

(FIELDU)

$$\frac{}{\mathcal{E}^p[\mathcal{E}^p{}_1[x\mathtt{.}m\ \mathtt{(that\!:\!}a\mathtt{)}\,]] \longrightarrow_p \mathcal{E}^p[\,\mathtt{(}\overline{d}[x.m = a]\overline{\mathcal{K}}e\mathtt{)}\,]}$$

with

$\mathsf{move}_p(\mathcal{E}^p{}_1, \mathsf{FV}(a)) = \langle \mathcal{E}^p{}_2, \emptyset \rangle$

$\mathcal{E}^p{}_2[\mathtt{void}] = \mathtt{(}\overline{d}\overline{\mathcal{K}}e\mathtt{)}$

$\mathcal{E}^p{}_1 = \mathtt{(}\overline{dv}\_\_\mathtt{)}\,, \overline{dv}(x) = \mu\ \pi\ x\ \mathtt{=}\_$ and $\mu \in \{\mathtt{mut}, \mathtt{lent}\}$

$\mathsf{class}(\mathcal{E}^p{}_1, x) = \pi$

$\mathsf{meth}_p(\pi\mathtt{.}m\ \mathtt{(that)}\,) = \_\mathtt{method}\,\_\ m\mathtt{()}\ \mathtt{exception}\,\_$

(R-USING)

$$\frac{}{\mathcal{E}^p[e_0] \longrightarrow_p \mathcal{E}^p[e_1]}$$

with

$e_0 = \mathtt{using}\ \pi\ \mathtt{check}\ m\ \mathtt{(}\overline{x\mathtt{:}v}\mathtt{)}\ e$

$e_1 = \mathsf{plugin}(p, \pi\ m\ \mathtt{(}\overline{x\mathtt{:}v}\mathtt{)}\ e)$

(R-USING-OUT)

$$\frac{}{\mathcal{E}^p[\mathtt{using}\ \pi\ \mathtt{check}\ m\ \mathtt{(}\overline{x\mathtt{:}v}\mathtt{)}\ e] \longrightarrow_p \mathcal{E}^p[e]}$$

with

$\mathsf{plugin}(p, \pi\ m\ \mathtt{(}\overline{x\mathtt{:}v}\mathtt{)}\ e)$ undefined

either e is a v or $\mathsf{throws}_p(e) = \varrho\ v$

(R-META)

$$\frac{\begin{array}{c} e_1 \longrightarrow_{p'} e_2 \\ \mathcal{L} \xrightarrow[\max]{p} \mathcal{L}' \\ \vdash p' : \mathrm{ok} \\ p'\circledast;\emptyset;\emptyset;\emptyset \vdash e_1 : \mathtt{Library} \end{array}}{\mathcal{E}^p[\mathcal{L}] \longrightarrow_p \mathcal{E}^p[\mathcal{L}[C\mathtt{:}e_2]]}$$

with

$\mathcal{L} = \mathtt{\{}\_\mathtt{<:}\_\overline{\mathcal{M}^{\mathsf{c}}}C\mathtt{:}e^{\mathsf{c}}\_\mathtt{\}}$

$p' = \mathcal{L}'^{\ominus}[C : \updownarrow]\ p$

$e_1 = \mathsf{norm}_{p'}(e^{\mathsf{c}})$

(R-META-METHOD)

$$\frac{\begin{array}{c} \mathcal{L}_1 \longrightarrow_{p'} \mathcal{L}_2 \\ \mathcal{L} \xrightarrow[\max]{p} \mathcal{L}' \end{array}}{\mathcal{E}^p[\mathcal{L}] \longrightarrow_p \mathcal{E}^p[\mathcal{L}[mh\,\mathcal{E}^{\mathsf{c}}[\mathcal{L}_2]]]}$$

with

$\mathcal{L} = \mathtt{\{}\_\mathtt{<:}\_\overline{\mathcal{M}^{\mathsf{c}}}mh\,\mathcal{E}^{\mathsf{c}}[\mathcal{L}_1]\_\mathtt{\}}$

$p' = \mathcal{L}'\ p$

(R-CAPTURE)

$$\frac{}{\mathcal{E}^p[e_1] \longrightarrow_p \mathcal{E}^p[\,\mathtt{(}\overline{dv}\ \mu\ \pi\ z\mathtt{=}v\ e\mathtt{)}\,]}$$

with

$e_1 = \mathtt{(}\overline{dv}\ \overline{dv}'\ T'\ x\mathtt{=}e'\ \overline{d}\mathtt{catch}\ \varrho\ z\ \mathtt{on}\ T\ e\,\overline{\mathcal{O}}e_0\mathtt{)}$

$\mathsf{throws}_p(e') = \varrho\ v$

$\mu\ \pi = \mathsf{norm}_p(T)$

$p \vdash \mathsf{class}(\mathcal{E}^p[\,\mathtt{(}\overline{dv}\square\mathtt{)}\,], v) \leq \pi$

(R-ONMISS)

$$\frac{}{\mathcal{E}^p[e_1] \longrightarrow_p \mathcal{E}^p[\,\mathtt{(}\overline{dv}\ T'\ x\mathtt{=}\varrho\ v\mathtt{catch}\ \varrho\ z\ \overline{\mathcal{O}}e_0\mathtt{)}\,]}$$

with

$e_1 = \mathtt{(}\overline{dv}\ \overline{dv}'\ T'\ x\mathtt{=}e'\ \overline{d}\mathtt{catch}\ \varrho\ z\ \mathtt{on}\ T\ e\,\overline{\mathcal{O}}e_0\mathtt{)}$

$\mathsf{throws}_p(e') = \varrho'\ v$

$\mu\ \pi = \mathsf{norm}_p(T)$

either $\varrho \neq \varrho'$ or not $p \vdash \mathsf{class}(\mathcal{E}^p[\,\mathtt{(}\overline{dv}\square\mathtt{)}\,], v) \leq \pi$

(R-KOUT)

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

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 ($\mathsf{plugin}(p^t, \pi, \texttt{.}m\texttt{(}\overline{x}\texttt{)})$) from library types (often containing an url doc) extracted from a program type p^t. Plugins monitor execution of code e ($\mathsf{execute}(plg, p, \sigma, \overline{d}, \overline{v}, e)$). **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 **var**.

There must not be any whitespace preceding the symbol '$\texttt{"}$' in string expressions or π in number expression.

For all blocks of form $\texttt{\{}\overline{d}_1\mathcal{K}_1 ... \overline{d}_n\mathcal{K}_n\texttt{\}}$, $\mathsf{terminating}(\texttt{\{}\overline{d}_1\mathcal{K}_1 ... \overline{d}_n\mathcal{K}_n\texttt{\}})$ holds. Method names in method calls (using the dot) must be of form x or $\texttt{\#}x$.

The **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: $\mathsf{downloadFromWeb}(\_)$

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: $\mathsf{terminating}(\_)$

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

Atomic Language Terms

$$\begin{array}{lll}
\varrho & ::= \texttt{return} \mid \texttt{error} \mid \texttt{exception} & \text{results}\\
\mu & ::= \texttt{type} \mid \texttt{mut} \mid \texttt{read} \mid \texttt{lent} \mid \texttt{capsule} \mid \texttt{immutable} & \text{type modifiers}\\
\mathit{ident} & ::= \text{< [\_,a..z,A..Z,,\$,\%] [\_,a..z,A..Z,\$,\%,0..9]*>} & \text{identifiers}\\
C & ::= \text{<}\mathit{ident}\text{ starting upper-case except } \texttt{Any}, \texttt{Void}, \texttt{Library}\text{>} & \text{Class names}\\
x, y, z & ::= \text{<}\mathit{ident}\text{ starting lower-case (or \_) except keywords>} & \text{variable names}\\
m & ::= x \mid \texttt{\#}x \mid \emptyset \mid \mathit{unOp} \mid \mathit{eqOp} \mid \mathit{binOp} & \text{method names}\\
\mathit{doc} & ::= \mathit{strLine}_1 ... \mathit{strLine}_n\text{<often omitted for brevity>} & \text{documentation}\\
\mathit{strLine} & ::= \text{<spaces> } \texttt{'}\text{<sequence of } \mathit{char} \text{ excluding } \mathit{EOL}\text{>}\mathit{EOL} & \text{line of documentation}\\
\mathit{string} & ::= \text{<any sequence of } \mathit{char} \text{ excluding (}\texttt{"}\text{) and } \mathit{EOL}\text{>} & \text{simple string}\\
 & \quad \mid \mathit{EOL}\ \mathit{doc}\text{<spaces><where } \mathit{doc} \text{ is not empty>} & \text{multi line string}\\
\mathit{char} & ::= \text{<a subset of all character; around} \sim \text{100 symbols>} & \text{source chars}\\
\mathit{digit} & ::= \texttt{0} \mid \texttt{1} \mid \texttt{2} \mid \texttt{3} \mid \texttt{4} \mid \texttt{5} \mid \texttt{6} \mid \texttt{7} \mid \texttt{8} \mid \texttt{9} & \\
\mathit{digit+} & ::= \mathit{digit} \mid \texttt{.}\,\mathit{digit} \mid \texttt{-}\mathit{digit} & \\
\mathit{num} & ::= \overline{\mathit{dataOp}}\,\mathit{digit}\,\overline{\mathit{digit+}} & \\
\mathit{unOp} & ::= \texttt{!} \mid \sim & \\
\mathit{eqOp} & ::= \texttt{+=} \mid \texttt{-=} \mid \texttt{*=} \mid \texttt{/=} \mid \texttt{\&=} \mid \texttt{\vert=} \mid \texttt{»=} \mid \texttt{«=} \mid \texttt{++=} \mid \texttt{**=} \mid \texttt{:=} & \text{requires } x \text{ as left value}\\
\mathit{boolOp} & ::= \texttt{\&} \mid \texttt{\vert} & \text{weak precedence}\\
\mathit{relOp} & ::= \texttt{<} \mid \texttt{>} \mid \texttt{==} \mid \texttt{!=} \mid \texttt{<=} \mid \texttt{>=} & \text{medium precedence}\\
\mathit{dataOp} & ::= \texttt{+} \mid \texttt{-} \mid \texttt{*} \mid \texttt{/} \mid \texttt{«} \mid \texttt{»} \mid \texttt{++} \mid \texttt{**} & \text{strong precedence}\\
\mathit{binOp} & ::= \mathit{boolOp} \mid \mathit{relOp} \mid \mathit{dataOp} & \text{binary operators}\\
\mathit{url} & ::= \text{<sequence of } \mathit{char} \text{ excluding <space>,}\mathit{EOL}\text{, } \texttt{\{} \text{ and } \texttt{\}}\text{>} &
\end{array}$$

Complete Language Syntax

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

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

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

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

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

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

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

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

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

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

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

otherwise $\llbracket\mathtt{\{}\mathcal{H}\ \mathtt{<:}\overline{\pi}\overline{\mathcal{M}}\mathtt{\}}\rrbracket_{\overline{\mathcal{L}}} = \mathtt{\{}{}^{\mathcal{M}}\llbracket\mathcal{H}\rrbracket_\Theta\mathtt{<:}\llbracket\overline{\pi}\rrbracket_\Theta{}^{\mathcal{M}}\llbracket\overline{\mathcal{M}}\rrbracket_\Theta\mathtt{\}}$

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

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

(lt) $\llbracket num\overline{e}\rrbracket_\Theta = \llbracket e\mathtt{.\#numberParser(\{'@String}EOL'\ \llbracket\overline{'num'}\rrbracket EOL\mathtt{\})}\rrbracket_\Theta$

$\llbracket e\mathtt{"}\overline{char}\mathtt{"}\rrbracket_\Theta = \llbracket e\mathtt{.\#stringParser(\{'@String}EOL'\ \llbracket\overline{'char'}\rrbracket EOL\mathtt{\})}\rrbracket_\Theta$

$\llbracket e\mathtt{"}EOL\overline{char}EOL\mathtt{"}\rrbracket_\Theta = \llbracket e\mathtt{.\#stringParser(\{'@String}EOL'\ \llbracket\overline{'charEOL'}\rrbracket EOL\mathtt{\})}\rrbracket_\Theta$

(cf) $\llbracket\mathtt{while}\ e\ \mathcal{B}\rrbracket_\Theta = \llbracket\mathtt{(loop\ (}e\mathtt{.\#checkTrue()}\ \mathcal{B}\mathtt{)\ catch\ exception\ (on\ Void\ void)\ void)}\rrbracket_\Theta$

$\llbracket\mathtt{if}\ a\ \mathcal{B}_1\ \mathtt{else}\ \mathcal{B}_2\rrbracket_\Theta = \llbracket\mathtt{(}a\mathtt{.\#checkTrue()\ catch\ exception\ (onVoid}\ \mathcal{B}_2\mathtt{)}\ \mathcal{B}_1\ \mathtt{void)}\rrbracket_\Theta$

$\llbracket\mathtt{if}\ e\ \mathcal{B}_1\ \mathtt{else}\ \mathcal{B}_2\rrbracket_\Theta = \llbracket\mathtt{(}\ y\mathtt{=}e\ \mathtt{if}\ y\ \mathcal{B}_1\ \mathtt{else}\ \mathcal{B}_2\mathtt{)}\rrbracket_\Theta$ with y fresh and $e \neq a$

$\llbracket\mathtt{(}\overline{d}_1\mathcal{K}_1...\overline{d}_n\mathcal{K}_n e\mathtt{)}\rrbracket_\Theta = \llbracket\mathtt{(}\overline{d}_1\mathcal{K}_1\ \mathtt{(}...\ \mathtt{(}\overline{d}_n\mathcal{K}_n e\mathtt{)}\ ...\mathtt{))}\rrbracket_\Theta$

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

$\llbracket x\rrbracket_\_ = x$ $\quad \llbracket x\mathtt{:=}e\rrbracket_\Theta = x\mathtt{.inner(}\llbracket e\rrbracket_\Theta\mathtt{)}$ $\quad \llbracket\varrho\ e\rrbracket_\Theta = \varrho\llbracket e\rrbracket_\Theta$

$\llbracket\mathtt{loop}\ e\rrbracket_{\Gamma,T\,\overline{\mathcal{L}}} = \mathtt{loop}\ \llbracket e\rrbracket_{\Gamma,\mathtt{immutable\ Void}\overline{\mathcal{L}}}$

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

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

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

(vd1) $\llbracket e_0\rrbracket_\Theta = \llbracket\mathtt{(}C\mathtt{:\{mut\ (var}\ T\ \mathtt{inner)\}}$

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

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 $\overline{d}_2$ $\quad$ and either $\overline{d}_3 = \emptyset$ or $\overline{d}_3 = d\_$ and $x\mathtt{:=}\_$ inside d

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

$\llbracket\mathtt{(}\overline{d}_1 e\ \overline{d}_2\overline{\mathcal{K}}e_1\mathtt{)}\rrbracket_\Theta = \llbracket\mathtt{(}\overline{d}_1\ \mathtt{immutable\ void}\ x\mathtt{=}e\overline{d}_2\overline{\mathcal{K}}e_1\mathtt{)}\rrbracket_\Theta$

$\llbracket\mathtt{(}d_1...d_n\overline{\mathcal{K}}e\mathtt{)}\rrbracket_{\Gamma;T;\overline{\mathcal{L}}} = \mathtt{(}{}^d\llbracket d_1\rrbracket_\Theta...{}^d\llbracket d_n\rrbracket_\Theta{}^{\mathcal{K}}\llbracket\overline{\mathcal{K}}\rrbracket_{\Gamma;T;\overline{\mathcal{L}}}\llbracket e\rrbracket_\Theta\mathtt{)}$

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

${}^{\mathcal{K}}\llbracket\mathtt{catch}\ \varrho\ \overline{\mathcal{O}\mathtt{default}\ e}\rrbracket_\Theta = {}^{\mathcal{K}}\llbracket\mathtt{catch}\ \varrho\ x\ \overline{\mathcal{O}\mathtt{default}\ e}\rrbracket_\Theta$

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

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

otherwise ${}^{\mathcal{K}}\llbracket\mathtt{catch}\ \varrho\ x\ \mathcal{O}_1...\mathcal{O}_n\rrbracket_\Theta = \mathtt{catch}\ \varrho\ x\ {}^{\mathcal{K}}\llbracket\mathcal{O}_1\rrbracket_{x,\Theta}...{}^{\mathcal{K}}\llbracket\mathcal{O}_n\rrbracket_{x,\Theta}$ and

${}^{\mathcal{K}}\llbracket\mathtt{catch}\ \varrho\ x\ \mathcal{O}_1...\mathcal{O}_n\mathtt{default}\ e\rrbracket_\Theta = \mathtt{catch}\ \varrho\ x\ {}^{\mathcal{K}}\llbracket\mathcal{O}_1\rrbracket_{x,\Theta}...{}^{\mathcal{K}}\llbracket\mathcal{O}_n\rrbracket_{x,\Theta}\mathtt{default}\ \llbracket e\rrbracket_{x:\mathtt{immutable\ Any},\Theta}$

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

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

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

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

$\llbracket x\ eqOp\ e\rrbracket_\Theta = \llbracket x\mathtt{:=}x\mathtt{.\#inner().}\llbracket 'eqOp'\rrbracket\mathtt{(}e\mathtt{)}\rrbracket_\Theta$

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

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

Definition: $\llbracket doc\rrbracket_p$

$\llbracket doc\rrbracket_p$ replaces all substrings of the form $\mathtt{@}\pi$ and $\mathtt{@(}e\mathtt{)}$ with $\mathtt{@}\llbracket\pi\rrbracket_p$ and $\mathtt{@(}\llbracket e\rrbracket_p\mathtt{)}$. 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) ${}^{\mathcal{W}}\llbracket\mathtt{with}\ \overline{x}\ \overline{\mathcal{I}}\ \overline{d}\ \mathtt{(}\overline{\mathcal{O}^w}\mathtt{)}\rrbracket = {}^{\mathcal{W}}\llbracket\mathtt{with}\ \overline{x}\ \overline{\mathcal{I}}\ \overline{d}\ \mathtt{(}\overline{\mathcal{O}^w}\mathtt{default\ void)}\rrbracket$

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

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

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

(ca) ${}^{\mathcal{W}}\llbracket\mathtt{on}\ T_1...T_n\overline{\mathtt{case}\ e_0}\ e_1\ \overline{\mathcal{O}^w}\ \mathtt{default}\ e\rrbracket_{x_1...x_n} = \mathtt{(}\overline{e}\ \text{cast}^{T_1}(y_1 \leftarrow x_1)...\text{cast}^{T_n}(y_n \leftarrow x_n)$

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

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

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

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

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

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

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

(e) ${}^{\mathcal{W}}\llbracket\mathtt{with}\ \overline{\mathcal{I}}\ \mathcal{B}\rrbracket = {}^{\mathcal{W}}\llbracket\text{declareIts}(\overline{\mathcal{I}},\mathtt{(}$

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

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

with $\overline{\mathcal{I}} = \_x_1\ \mathtt{in}\ \_...\_x_n\ \mathtt{in}\ \_,$ $\quad d_i\mathcal{K}_i = \text{next}_i(x_1...x_n)$

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

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

with $\mathcal{O}^w{}_i = \overline{\mathtt{on}\ \overline{T}\ \mathtt{if}\,e}\,e,\ \mathcal{O}^{w'}{}_i = \overline{\mathtt{on}\ \overline{T}\ \mathtt{if}\,e}\ x\mathtt{:=}x\mathtt{.\#add(}e\mathtt{)}$ 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: $\llbracket e\rrbracket_p$ auxiliary definitions

${}^{ps}\llbracket e_0\ \overline{x\mathtt{:}e}\rrbracket_{\Gamma,T::m,\overline{\mathcal{L}}} = {}^{ps}\llbracket\mathtt{that:}e_0\ \overline{x\mathtt{:}e}\rrbracket_{\Gamma,T::m,\overline{\mathcal{L}}}$

${}^{ps}\llbracket x_1\mathtt{:}e_1...x_n\mathtt{:}e_n\rrbracket_{\Gamma,T::m,\overline{\mathcal{L}}} =$

$x_1\mathtt{:}\llbracket e_1\rrbracket_{\Gamma,T::m\ \mathtt{(}x_1...x_n\mathtt{)}::x_1,,\overline{\mathcal{L}}}\cdots$

$x_n\mathtt{:}\llbracket e_n\rrbracket_{\Gamma,T::m,\ \mathtt{(}x_1...x_n\mathtt{)}::x_n,\overline{\mathcal{L}}}$

${}^{ps}\llbracket x_1\mathtt{:}e_1...x_n\mathtt{:}e_n\rrbracket_{\Gamma,T,\overline{\mathcal{L}}} =$

$x_1\mathtt{:}\llbracket e_1\rrbracket_{\Gamma,T,\overline{\mathcal{L}}}...x_n\mathtt{:}\llbracket e_n\rrbracket_{\Gamma,T,\overline{\mathcal{L}}}$

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

${}^{\mathcal{M}}\llbracket mh\,e\rrbracket_p = {}^{\mathcal{M}}\llbracket{}^{\mathcal{M}}\llbracket mh\rrbracket_p\ \llbracket e\rrbracket_{\emptyset;\text{Tof}(mh);p}\rrbracket$

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

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

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

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

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

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

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

otherwise ${}^{\mathcal{M}}\llbracket mh\,e\rrbracket = mh\,e$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$\mathtt{void)}$

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

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

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

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

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

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

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

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

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

$\mathtt{exception\ void))}$

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

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

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

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

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

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

$x_1 : e_1...x_n : e_n = x_1...x_n$

Plugin auxiliary definitions

Definition: $\mathcal{L}_1 \oplus \mathcal{L}_2 = \mathcal{L}$

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

$\texttt{trait} \oplus \mathcal{H} = \mathcal{H} \oplus \texttt{trait} = \mathcal{H}$

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

$\overline{\mathcal{M}}_1\mathcal{M} \oplus \overline{\mathcal{M}}_2 = \overline{\mathcal{M}}_1 \oplus (\mathcal{M} \oplus \overline{\mathcal{M}}_2)$

$\emptyset \oplus \overline{\mathcal{M}} = \overline{\mathcal{M}}$

$\mathcal{M} \oplus \overline{\mathcal{M}} = \mathcal{M}\overline{\mathcal{M}}$ 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

$\texttt{method}\,doc\,mx\,e \oplus h\overline{\mathcal{M}} = h \oplus \texttt{method}\,doc\,mx\,e\overline{\mathcal{M}} =$

$\texttt{method}(\mathsf{docsOf}(h) \oplus doc)\,mx\,e\overline{\mathcal{M}}$ fix in the code

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 $\mathcal{L}(\pi) =$

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

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 $p; \emptyset \vdash \{\mathcal{H}\overline{\mathcal{M}}\} \rightarrow \widehat{\mathcal{L}}$ 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 $\mathcal{L}[\mathsf{redirectDefinition}\,\texttt{Outer}_0{::}\overline{C}_0 \mapsto \texttt{Outer}_0{::}\overline{C}_1] =$

$\mathcal{L}(\overline{C}_0)[\mathsf{remove\ n\ outers}][\mathsf{encapsulateIn}\,\overline{C}_1]$ 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

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

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

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

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

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

and $\pi_0[\mathsf{renUsage}_p\pi]\pi' = \mathsf{norm}_p(\pi')$ iff $\mathsf{norm}_p(\pi_0) = \mathsf{norm}_p(\pi)$

Plugin auxiliary definitions

Definition: $\mathcal{L}_1 \oplus \mathcal{L}_2 = \mathcal{L}$

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

$\texttt{trait} \oplus \mathcal{H} = \mathcal{H} \oplus \texttt{trait} = \mathcal{H}$

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

$\overline{\mathcal{M}}_1\mathcal{M} \oplus \overline{\mathcal{M}}_2 = \overline{\mathcal{M}}_1 \oplus (\mathcal{M} \oplus \overline{\mathcal{M}}_2)$

$\emptyset \oplus \overline{\mathcal{M}} = \overline{\mathcal{M}}$

$\mathcal{M} \oplus \overline{\mathcal{M}} = \mathcal{M}\overline{\mathcal{M}}$ 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

$\texttt{method}\,doc\,mx\,e \oplus h\overline{\mathcal{M}} = h \oplus \texttt{method}\,doc\,mx\,e\overline{\mathcal{M}} =$

$\texttt{method}(\mathsf{docsOf}(h) \oplus doc)\,mx\,e\overline{\mathcal{M}}$ fix in the code

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 $\mathcal{L}(\pi) =$

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

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 $p; \emptyset \vdash \{\mathcal{H}\overline{\mathcal{M}}\} \rightarrow \widehat{\mathcal{L}}$ 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 $\mathcal{L}[\mathsf{redirectDefinition}\,\texttt{Outer}_0{::}\overline{C}_0 \mapsto \texttt{Outer}_0{::}\overline{C}_1] =$

$\mathcal{L}(\overline{C}_0)[\mathsf{remove\ n\ outers}][\mathsf{encapsulateIn}\,\overline{C}_1]$ 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

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

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

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

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

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

and $\pi_0[\mathsf{renUsage}_p\pi]\pi' = \mathsf{norm}_p(\pi')$ iff $\mathsf{norm}_p(\pi_0) = \mathsf{norm}_p(\pi)$