

Language 42

for more information `L42.is`

Core Language Syntax

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

Auxiliary syntax

$\mathcal{E} ::= \square \mid \mathcal{E}.m(\bar{x}:e) \mid e_0^c.m(\bar{x}:\bar{e}_1x:\mathcal{E}\bar{x}:\bar{e}_2) \mid \text{loop } \mathcal{E} \mid \varrho \mathcal{E}$ $\mid (\bar{d}\bar{e}_1 \ T \ x = \mathcal{E} \ \bar{d}_2 \bar{\mathcal{K}} \ e) \mid (\bar{d}\text{catch } \varrho \ x \text{ on } T \ \mathcal{E}^* e) \mid (\bar{d}\bar{\mathcal{K}}\mathcal{E}^*)$ $\text{using } \pi \text{ check } .m(\bar{e}_1\mathcal{E}\bar{e}_2) \ e \mid \text{using } \pi \text{ check } .m(\bar{e}) \ \mathcal{E}^*$	
$\mathcal{E}^* ::= \square \mid \mathcal{E}^*.m(\bar{e}) \mid e_0.m(\bar{x}:\bar{e}_1x:\mathcal{E}^*\bar{x}:\bar{e}_2) \mid \text{loop } \mathcal{E}^* \mid \varrho \mathcal{E}^*$ $\mid (\bar{d}_1 \ T \ x = \mathcal{E}^* \ \bar{d}_2 \bar{\mathcal{K}} \ e) \mid (\bar{d}\text{catch } \varrho \ x \text{ on } T \ \mathcal{E}^* e) \mid (\bar{d}\bar{\mathcal{K}}\mathcal{E}^*)$ $\text{using } \pi \text{ check } .m(\bar{x}:\bar{e}_0x:\mathcal{E}^*\bar{x}:\bar{e}_1) \ e \mid \text{using } \pi \text{ check } .m(x:\bar{v}) \ \mathcal{E}^*$	
$v^p ::= a \mid (\bar{d}v^p \ v^p)$	value
$\bar{d}v^p ::= \mu \pi' \ x = \pi.m(x_1:a_1 \dots x_n:a_n) \mid \text{immutable } \pi \ x = (\bar{d}v^p \ v^p)$ $<\text{if meth}_p(\pi.m(x_1 \dots x_n)) = h \text{ field,}$ $\mu \neq \text{capsule and } p(\pi') \text{ not interface}>$	
$\mathcal{E}^p ::= \square \mid (\bar{d}v \ T \ x = \mathcal{E}^p \ \bar{d}\bar{\mathcal{K}} \ e) \mid (\bar{d}v \mathcal{E}^p) \mid \varrho \mathcal{E}^p \mid \mathcal{E}^p.m(\bar{x}:e)$ $\mid v_0.m(\bar{x}:\bar{v} \ x:\mathcal{E}^p \bar{x}:e) \mid \text{using } \pi \text{ check } .m(\bar{x}:\bar{v} \ x:\mathcal{E}^p \bar{x}:e) \ e$ $\mid \text{using } \pi \text{ check } .m(\bar{x}:\bar{v}) \ \mathcal{E}^p$	
$p ::= \mathcal{L}_0, \dots, \mathcal{L}_n^{\otimes}$	program type
$\Gamma ::= x : T$	
$\Sigma ::= \bar{x}; \bar{x}_1 \dots \bar{x}_n; \bar{x}'_1 \dots \bar{x}'_k$	seal env
$\Phi ::= \bar{T}; \bar{\pi}$	throw env

Definition: $\text{compiled}(_)$

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

$\text{compiled}(\{ \mathcal{H} <: \bar{\pi} \mathcal{M} \}^{\odot})$ iff $\text{compiled}(\mathcal{M}) \forall \mathcal{M} \in \bar{\mathcal{M}}$

$\text{compiled}(h)$

$\text{compiled}(C:\mathcal{L})$ iff $\text{compiled}(\mathcal{L})$

$\text{compiled}(mh \ e)$ iff $\text{compiled}(e)$

We write \mathcal{E} as a metavariable to represent an e where $\text{compiled}(e)$ holds. Same notation is used for \mathcal{L} and \mathcal{M} .

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

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

$\bar{d}(x) : (\bar{d}_1 \ \bar{T} \ x = e \ \bar{d}_2)(x) = e$

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

$p(\pi) : (\mathcal{L}_0 \dots \mathcal{L}_n)(\pi) = \mathcal{L}_i::\bar{C}$ if $\text{norm}_p(\pi) = \text{Outer}_i::\bar{C}$

$\mathcal{L}::\bar{C} : \mathcal{L}::C_1 \dots C_n = \mathcal{L}(C_1) \dots (C_n)$ where $e(C) = e$ iff e not \mathcal{L}

$(\Gamma; \bar{x}; \bar{x}_1 \dots \bar{x}_n), \bar{d}v$ and $\bar{\mathcal{M}}$ are maps, thus order is irrelevant.

The above function notations $_(_)$ each implicitly defines a domain $\text{dom}(_)$ as the set of all inputs for which the function is defined

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

$\{ \mathcal{H} \bar{\pi}' \bar{\mathcal{M}} C : _ \}^{\odot} [C:\mathcal{L}] = \{ \mathcal{H} \bar{\pi}' \bar{\mathcal{M}} C : \mathcal{L} \}^{\odot}$

$\{ \mathcal{H} \bar{\pi}' \bar{\mathcal{M}} mh \ e_1 \}^{\odot} [mh \ e_2] = \{ \mathcal{H} \bar{\pi}' \bar{\mathcal{M}} mh \ e_2 \}^{\odot}$

Definition: $_ \text{ inside } _$

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

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 $_$ to denote optionality; for example \bar{T} and $\bar{\text{var}}$ denote metavariables that can be either the empty string \emptyset or in the form of the corresponding terms. In the same way, we use $_$ to denote multiplicity. We consider terms (e) and e to be equivalent, and from now on we omit documentations doc when is not relevant. We consider terms of the form (e) to be equivalent to the corresponding e and terms of the form $(\bar{d}v \text{catch } \varrho \ x \ () \ e)$ to be equivalent to the corresponding $(\bar{d}ve)$. Also, values of form $(T \ x = (T \ y = ey) \ x)$ are equivalent to the corresponding $(T \ y = ey)$ if $x \notin \text{FV}(e)$. In any moment a type of form $\pi \bar{m} \alpha$ is considered in the context of a program p , we consider it equivalent to the corresponding resolved type $\text{norm}_p(\pi \bar{m} \alpha)$.

The following symbols $\ominus \oplus \otimes \dagger \%$ 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/exception 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 \bar{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 $\mathcal{E}^p = \mathcal{E}^{p'}[\text{using } \pi \text{ check } .m(\bar{x}:\bar{v}) \ \mathcal{E}^{p''}]$ and $\text{plugin}(\pi \ m(\bar{x}:\bar{v}) \ \mathcal{E}^{p''}[e])$ is well defined.

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

$\text{Outer}^n::\bar{C}[\text{from } \text{Outer}^m::C_1 \dots C_k] = \text{Outer}^m::C_1 \dots C_{k-n}::\bar{C}$ if $n \leq k$

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

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

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

$e[\text{from } \pi]$ propagate on the structure, and $\mathcal{L}[\text{from } \pi] = \mathcal{L}[\text{from } \pi]_0$

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

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

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

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

with $@ \pi_0[\text{from } \pi]_j$ and $@(e_0[\text{from } \pi]_j)$

All cases for other expressions/terms propagate to submembers

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

with $\bar{\mathcal{K}} = \text{catch error } x \ \bar{O}$ and $\Sigma = _; _; \bar{x}_1 \dots \bar{x}_n$

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

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

otherwise $\Gamma' = \Gamma$

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

$\bar{T}; \bar{\pi}[\text{catch return } \mathcal{O}_1 \dots \mathcal{O}_n] = \bar{T}[\mathcal{O}_1] \dots [\mathcal{O}_n]; \bar{\pi}$

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

otherwise $\mu \pi_1 \dots \mu \pi_n[\text{on } \mu' \pi_0 _] = \mu' \pi_0$

$\bar{T}; \bar{\pi}[\text{catch exception } x \text{ on } \text{immutable } \pi_1 \dots \text{on } \text{immutable } \pi_n _] = \bar{T}; \bar{\pi}, \pi_1 \dots \pi_n \setminus \text{Any}$

otherwise $\Phi[\bar{\mathcal{K}}] = \Phi$

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

$p \vdash \bar{\pi}_1 \leq \bar{T}; \bar{\pi}_2$ iff $\forall \pi_1 \in \bar{\pi}_1, \exists \pi_2 \in \bar{\pi}_2$ 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 $\text{norm}_p(\pi') \in \text{norm}_p(\bar{\pi}[\text{from } \pi] \cup \pi) \cup \text{Any}$

with $p(\pi) = \{ _ \}^{\vdash \bar{\pi}}$

$\text{capsule} \leq \text{mut} \leq \text{lent} \leq \text{read}, \text{capsule} \leq \text{immutable} \leq \text{read}$ and $\emptyset \leq \% \leq \wedge$

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

$h_1 \dots h_n \cup \bar{\mathcal{M}} = h_1 \cup (\dots \cup (h_n \cup \bar{\mathcal{M}}))$

$h \cup \bar{\mathcal{M}} = \bar{h} \bar{\mathcal{M}}$ iff $\text{dom}(h)$ disjoint $\text{dom}(\bar{\mathcal{M}})$

$h \pi \cup mh^s e \bar{\mathcal{M}} = h \pi e \bar{\mathcal{M}}$ iff $\text{dom}(h) = \text{dom}(mh^s e)$

$h \pi \cup h \bar{\mathcal{M}} = h \bar{\mathcal{M}}$

Definitions1

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

complete(Γ) = $\{x : \mu \pi \mid \Gamma(x) = \mu \pi\}$
 $\text{dom}^{\text{mut}}(\Gamma) = \{x : \text{mut } \pi \alpha \mid \Gamma(x) = \text{mut } \pi \alpha\}$
 $\text{dom}^{\text{mut}\leq}(\Gamma) = \{x : \mu \pi \alpha \mid \Gamma(x) = \mu \pi \alpha, \text{mut } \leq \mu\}$
 $\text{mutTolent}(\text{mut } \pi \alpha) = \text{lent } \pi \alpha$
 $\text{mutTolent}(\mu \pi \alpha) = \mu \pi \alpha$ otherwise
 $\text{mut\&LentToRead}(\text{mut } \pi \alpha) = \text{mut\&LentToRead}(\text{lent } \pi \alpha) = \text{read } \pi \alpha$
 $\text{mut\&LentToRead}(\mu \pi \alpha) = \mu \pi \alpha$ otherwise

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

move(\square, \bar{x}) = $\langle \square, \emptyset \rangle$
 assuming move(\mathcal{E}^p, \bar{x}) = $\langle \mathcal{E}^{p'}, \bar{dv}' \rangle$, then
 move($\langle \bar{dv} T y = \mathcal{E}^p \bar{dK}e \rangle, \bar{x}$) = $\langle \langle \bar{dv} \bar{dv}' T y = \mathcal{E}^{p'} \bar{dK}e \rangle, \emptyset \rangle$
 and move($\langle \bar{dv} \mathcal{E}^p \rangle, \bar{x}$) = $\langle \langle \bar{dv} \bar{dv}' \mathcal{E}^{p'} \rangle, \emptyset \rangle$ with $\bar{x} \subseteq \text{dom}(\bar{dv})$
 move($\mathcal{E}^p.m(\bar{x}:\bar{e}), \bar{x}$) = $\langle \mathcal{E}^{p'}.m(\bar{x}:\bar{e}), \bar{dv}' \rangle$
 move($v.m(\bar{x}:\bar{v}, y:\mathcal{E}^p, \bar{x}:\bar{e}), \bar{x}$) = $\langle v.m(\bar{x}:\bar{v}, y:\mathcal{E}^{p'}, \bar{x}:\bar{e}), \bar{dv}' \rangle$
 move($\langle \bar{dv} \mathcal{E}^p \rangle, \bar{x}$) = $\langle \langle \bar{dv} \bar{dv}' \mathcal{E}^{p'} \rangle, \bar{dv}' \bar{dv}_1 \rangle$
 move($\langle \bar{dv} T y = \mathcal{E}^p \bar{dK}e \rangle, \bar{x}$) = $\langle \langle \bar{dv} \bar{dv}' T y = \mathcal{E}^{p'} \bar{dK}e \rangle, \bar{dv}' \bar{dv}_1 \rangle$
 $\bar{dv} = \bar{dv}_1 \bar{dv}_2$ with \bar{dv}_1 inductively defined by
 $x \in \text{dom}(\bar{dv}_1)$ iff $x \in \text{dom}(\bar{dv})$ and $x \in \bar{x} \cup \text{FV}(\bar{dv}')$
 $x \in \text{dom}(\bar{dv}_1)$ iff $x \in \text{dom}(\bar{dv})$, $\bar{dv}_1(_) = v$ and $x \in \text{FV}(v)$

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

dec($\mathcal{E}^{p'}[\langle \bar{dv} \mathcal{E}^p \rangle], x$) = dec($\mathcal{E}^{p'}[\langle \bar{dv} T y = \mathcal{E}^p \bar{dK}e \rangle], x$) = $\bar{dv}(x)$
 if $x \in \text{dom}(\bar{dv})$

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

class(\mathcal{E}^p, x) = C if dec(\mathcal{E}^p, x) = $_ x = C.m(_)$
 class(\mathcal{E}^p, x) = class($\mathcal{E}^p, \langle \bar{dv} v \rangle$)
 if dec(\mathcal{E}^p, x) = $\text{immutable } _ x = \langle \bar{dv} v \rangle$
 class(\mathcal{E}^p, π) = π
 class($\mathcal{E}^p, \text{void}$) = Void , class($\mathcal{E}^p, \mathcal{L}$) = Library
 class($\mathcal{E}^p, \langle \bar{dv} v \rangle$) = class($\mathcal{E}^{p'}[\langle \bar{dv} \square \rangle], v$)

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

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

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

abstract $_p(\mathcal{L})$ holds if not coherent $_p(\mathcal{L})$
 or \mathcal{L} has a nested class $C:\mathcal{L}'$ such that abstract $_p(\mathcal{L}')$

coherent $_p(\{\text{interface } _ \}^{\odot})$ holds
 coherent $_p(\{\mathcal{H} <: \bar{\pi} \bar{h} \bar{\mathcal{M}}\}^{\odot})$ if no element of for $h \in \bar{\mathcal{M}}$ and either
 $\bar{h} = \emptyset$ or $\text{type method } \mu \text{Outer}_0 m(\bar{T} x) \text{exception } _ \in \bar{h}$
 and for every other $h \in \bar{h}$, coherent $_p(\mu, \bar{T} x, h)$ holds
 coherent $_p(\mu, \bar{T} x, h) = \text{coherent}_p(\mu, \text{norm}_p(\bar{T} x), h)$
 coherent $_p(\mu, \mu_1 \pi_1 \hat{x}_1 \dots \mu_n \pi_n \hat{x}_n, h)$ iff
 exists i such that coherent $_p(\mu_i \pi_i x_i, h)$ holds, and
 $\mu \neq \text{type}$, $\mu \in \{\text{read}, \text{lent}\}$ if read or $\text{lent} \in \{\mu_1 \dots \mu_n\}$,
 $\mu \in \{\text{capsule}, \text{immutable}\}$ iff $\{\mu_1 \dots \mu_n\} = \{\text{immutable}, \text{capsule}\}$

with $\mu \in \{\text{type}, \text{immutable}, \text{read}\}$, $\mu' \neq \text{type}$, $\mu'' \in \{\text{mut}, \text{lent}\}$, $\text{norm}_p(T') = v$
 (a) coherent $_p(\mu \pi x, \mu' \text{method } T \#x() \text{exception } _)$ iff $p \vdash \mu \pi \leq \text{norm}_p(T)$
 (b) coherent $_p(\mu \pi x, \mu'' \text{method } T' \#x(T \text{ that } _) \text{exception } _)$ iff $p \vdash \text{norm}_p(T)$

with $\mu \in \{\text{mut}, \text{lent}\}$, $\mu' \notin \{\text{type}, \text{mut}\}$, $\text{norm}_p(T') = \text{Void}$
 (c) coherent $_p(\mu \pi x, \text{mut method } T \#x() \text{exception } _)$ iff $p \vdash \mu \pi \leq \text{norm}_p(T)$
 (d) coherent $_p(\mu \pi x, \mu' \text{method } T \#x() \text{exception } _)$ iff $p \vdash \mu' \pi \leq \text{norm}_p(T)$
 (e) coherent $_p(\mu \pi x, \text{mut method } T' \#x(T \text{ that } _) \text{exception } _)$
 iff $p \vdash \text{norm}_p(T) \leq \mu \pi$
 (f) coherent $_p(\mu \pi x, \text{lent method } T' \#x(T \text{ that } _) \text{exception } _)$
 iff $p \vdash \text{norm}_p(T) \leq \text{capsule } \pi$

with $\mu' \neq \text{type}$, $\text{norm}_p(T') = \text{Void}$
 (g) coherent $_p(\text{capsule } \pi x, \mu' \text{method } T \#x() \text{exception } _)$
 iff $p \vdash \mu' \pi \leq \text{norm}_p(T)$
 (h) coherent $_p(\text{capsule } \pi x, \text{mut method } T' \#x(T \text{ that } _) \text{exception } _)$
 iff $p \vdash \text{norm}_p(T) \leq \text{mut } \pi$

Definition: originalMeth $_p(\pi_1 \dots \pi_n, \bar{m}x_0) = \bar{m}x$

originalMeth $_p(\mathcal{L}_0) = \bar{m}x_0 \setminus \dots \setminus \bar{m}x_n$ with $\mathcal{L}_0 = \{\mathcal{H} <: \pi_1 \dots \pi_n \bar{\mathcal{M}}\}$,
 $\mathcal{L}_1 = p(\pi_1) \dots \mathcal{L}_n = p(\pi_n)$, $\text{dom}(\mathcal{L}_i) = \bar{m}x_i \bar{C}_i$

Definitions2

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

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

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

norm $_p(\text{Outer}^{i+1}::\bar{C}::\bar{C}) = \text{norm}_p(\text{Outer}^i::\bar{C})$
 iff $p(\text{Outer}^{i+1}) = \{\mathcal{H} \bar{\mathcal{M}} \bar{C}::\bar{C}\}^{\odot}$ norm $_p(\pi) = \pi$ otherwise

norm $_p(\mu \pi \alpha) = \mu \text{norm}_p(\pi) \alpha$
 norm $_p(\pi' m x \hat{e}) = \mu \pi \hat{e}$ iff norm $_p(\pi' m x) = \mu \pi \alpha$
 norm $_p(\pi_1 m x_1 m x_2 \bar{m}x) = \text{norm}_p(\pi_2 m x_2 \bar{m}x)$

iff norm $_p(\pi_1 m x_1) = \mu \pi_2 \alpha$
 norm $_p(\pi::m(\bar{x})) = \text{norm}_p(T)$,
 norm $_p(\pi::m(\bar{x})::x_i) = \text{norm}_p(T_i)$
 and norm $_p(\pi::m(\bar{x})::\text{this}) = \mu \text{Outer}_0$
 iff meth $_p(\pi.m(\bar{x})) = \mu \text{method } T m(T_1 x_1 \dots T_n x_n) \text{exception } _$
 norm $_p(\pi \bar{m}x)$ is undefined iff $p(\pi) = \emptyset$ or we run into a cycle
 norm $_p(\mu \text{method } T_0 m(T_1 x_1 \dots T_n x_n) \text{exception } \bar{\pi} \bar{e})$
 = $\mu \text{method } T'_0 m(T'_1 x_1 \dots T'_n x_n) \text{exception } \text{norm}_p(\bar{\pi}) \text{norm}_p(\bar{e})$
 with $T'_i = \text{norm}_p(T_i)$

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

exe $^{\oplus}(p) = \text{exe}^{\oplus}(p, \text{Outer}_0)$
 exe $^{\oplus}(p, \text{Any})$, exe $^{\oplus}(p, \text{Void})$ and exe $^{\oplus}(p, \text{Library})$ holds.
 exe $^{\oplus}(p, \pi)$ iff $p(\pi) = \mathcal{L}^{\circ} \in \{\{_ \}^{\oplus}, \{_ \}^{\otimes}\}$
 exe (p^{\otimes}) iff $\otimes = \otimes$
 exe (p, π) holds iff $p(\pi) = \mathcal{L}^{\circ} = \{_ \}^{\otimes}$
 exeOk $^{\oplus}(p, x_1::_ \pi_1 \dots x_n::_ \pi_n)$
 iff either not exe $^{\oplus}(p)$ or $\forall i \in 1..n$ exe $^{\oplus}(p, \pi_i)$

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

toPartial($\mu \pi$) = $\mu \pi^{\circ}$
 toPartial($\mu \pi \alpha$) = $\mu \pi \alpha$ otherwise
 toPh($\mu \pi _$) = $\mu \pi^{\circ}$ and toPh($\pi \bar{m}x$) = $\pi \bar{m}x^{\circ}$
 those notions trivially extends to Γ

Definition: throws $_p(e) = q v$

throws $_p(q v) = q v$
 with e not a value, and throws $_p(e) = q v$
 throws $_p(e.m(_)) = \text{throws}_p(v.m(\bar{x}:\bar{v}x:e_)) = \text{throws}_p(q e) = q v$
 throws $_p(\langle \bar{dv} e \rangle) = q \langle \bar{dv} v \rangle$
 throws $_p(\langle \bar{dv}, \bar{dv}' T x = e \bar{d}e_0 \rangle) = q \langle \bar{dv} v \rangle$

Definition: used(\mathcal{L}) = $\bar{\pi}$ used $^{\oplus}(\mathcal{L}) = \bar{\pi}$

used $^{\oplus}(\{\mathcal{H} <: \bar{\pi}, \bar{\mathcal{M}}\}^-) = \bar{\pi} \cup \text{used}^{\oplus}(\bar{\mathcal{M}})$
 Outer $_k::\bar{C} \in \text{used}^{\oplus}(C:\mathcal{L}^{\circ})$ iff Outer $_{k+1}::\bar{C} \in \mathcal{L}^{\circ}$
 $\pi \in \text{used}^{\oplus}(h\bar{e})$ iff $\pi \in \text{used}^{\oplus}(\bar{e})$ or π inside h
 $\pi \in \text{used}^{\oplus}(e)$ iff $\pi.$ inside e
 used(\mathcal{L}°) is defined as used $^{\oplus}(\hat{\mathcal{L}}^{\circ})$ but in addition
 $\pi \in \text{used}(mh e)$ iff $\pi \in \text{used}(e)$
 Outer $_k::\bar{C} \in \text{used}(e)$ iff Outer $_{k+1}::\bar{C} \in \text{used}(\mathcal{L}^{\circ})$ and \mathcal{L}° inside e

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

if $p \text{lg}; T_ = \text{plugin}(p, \pi, m(x_1 \dots x_n))$
 with x as implicit reduction step identifier
 execute($x, p \text{lg}, p, \mathcal{E}^p, v_1 \dots v_n, e_0$) = e and $e \in \{v, \text{error } v\}$
 $p; \emptyset; \emptyset; \emptyset \vdash e : T' \leq T$
 either for all \mathcal{L} inside e $p \vdash \mathcal{L} \rightarrow \{_ \}^{\bar{\pi}}$
 or exists \mathcal{L} inside v such that $p \vdash \mathcal{L} \rightarrow \{_ \}^{\odot}$
 or exists π inside v such that $p(\pi) = \{_ \}^{\odot}$
 if all of the former holds, then plugin($\pi m(x_1:v_1 \dots x_n:v_n) e_0$) = e
 functions plugin($_, _, _$) and $_ \text{execute}(_, _, _, _, _, _)$ are defined by
 the specific 42 implementation; the step identifier is a fresh variable
 that identify unequivocally the current reduction step.

Definition: stageOf $_p(\mathcal{L}^{\circ}, \bar{e}) = \odot$

stageOf $_p(\mathcal{L}) = \odot$ iff \bar{e} not of form $\bar{\mathcal{L}}^{\circ}$ or $\{_ \}^{\odot} \in \bar{e}$
 otherwise stageOf $_p(\mathcal{L}) = \oplus$ iff abstract $_p(\mathcal{L})$ or $\{_ \}^{\oplus} \in \bar{e}$

Definition: superOf $_p(\mathcal{L}^{\circ}) = \bar{\pi}, \bar{\mathcal{M}}$

superOf $_p(\mathcal{L}) = \bar{\pi}_1[\text{from } \pi_1] \cup \dots \cup \bar{\pi}_n[\text{from } \pi_n]$, $\bar{h}_1[\text{from } \pi_1] \dots \bar{h}_n[\text{from } \pi_n]$
 norm $_{\mathcal{L}_p}(\bar{\pi}) = \{\pi_1 \dots \pi_n\}$, $(\mathcal{L}_p)(\pi_i) = \{\text{interface } <: \bar{\pi}_i \bar{h}_i \bar{C}::e_i\}^{\odot}$
 all originalMeth $_{\mathcal{L}_p}(\bar{\pi}_i[\text{from } \pi_i], \text{dom}(\bar{h}_i))$ are disjoint, $\mathcal{L} = \{\mathcal{H} <: \bar{\pi} \bar{\mathcal{M}}\}$

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

are they used somewhere?

Extraction of types

$$\begin{array}{c} \text{(ET-+)} \\ \frac{\mathcal{L}_1 \xrightarrow{p} \dots \xrightarrow{p} \mathcal{L}_2}{\mathcal{L}_2 \xrightarrow{p} _} \\ \frac{\mathcal{L}_1 \xrightarrow{p} \mathcal{L}_2}{\mathcal{L}_1 \xrightarrow{\text{max}} \mathcal{L}_2} \end{array} \quad \text{(ET-DEEP)} \quad \frac{\mathcal{L}_1 \xrightarrow{\mathcal{L}, p} \mathcal{L}_2}{\mathcal{L} \xrightarrow{p} \mathcal{L}[C:\mathcal{L}_2]} \quad \text{with } \mathcal{L} = \{\mathcal{H}_- \overline{\mathcal{M}} C:\mathcal{L}_1\}$$

$$\text{(ET-SUB)} \quad \frac{\mathcal{L} \xrightarrow{p} \{\mathcal{H} <: \overline{\pi} \cup \overline{\pi'} \overline{h} \cup \overline{\mathcal{M}}\}^{\oplus}}{\text{with } \mathcal{L} = \{\mathcal{H} <: \overline{\pi} \overline{\mathcal{M}}\} \text{ superOf}_p(\mathcal{L}) = \overline{\pi'}, \overline{h} \text{ } \overline{h} \cup \overline{\mathcal{M}} \text{ not have untyped headers}}$$

$$\text{(ET-LABEL)} \quad \frac{\mathcal{L} \xrightarrow{p} \{\mathcal{H} \overline{\mathcal{M}}\}^{\text{stageOf}_p(\mathcal{L}, \overline{e})}}{\text{with } \mathcal{L} = \{\mathcal{H} \overline{\mathcal{M}}\}^{\oplus} \quad \overline{e} = \{(\mathcal{L}p)(\pi) \mid \pi \in \text{used}^{\oplus}(\mathcal{L})\} \setminus \dagger \cup \{\mathcal{L}(\overline{C}) \mid \overline{C} \in \text{dom}(\mathcal{L})\} \quad \forall \mathcal{L}^e \in e, \mathcal{L}^e = \{\}^{\oplus}}$$

need a rule for neverLess? read comment under

$$\text{(P-OK)} \quad \frac{\vdash p : \text{ok if } p \neq \emptyset}{p \vdash \hat{\mathcal{L}}_0 : \text{ok}} \quad \text{with } \hat{\mathcal{L}}_0 = \hat{\mathcal{L}}[C:\text{error void}] \text{ if exists } C \text{ such that } \hat{\mathcal{L}}(C) = \dagger \text{ otherwise } \hat{\mathcal{L}}_0 = \hat{\mathcal{L}}$$

$$\text{(METH-T-DEF)} \quad \frac{p; \Gamma; \Sigma; \emptyset; \overline{\pi} \vdash e : T' \leq T \text{ if } \overline{e} = e}{p \vdash \mathcal{M} : \text{ok}} \quad \text{with } \text{norm}_p(\mathcal{M}) = h\overline{e} \text{ fully normalized } h = \mu \text{ method } T m(T_1 x_1 \dots T_n x_n) \text{ exception } \overline{\pi} \quad \Gamma = x_1 : T_1 \dots x_n : T_n, \text{ this} : \mu \text{ Outer}_0 \quad \Sigma = \emptyset; \emptyset; \text{ this}, x_1, \dots, x_n \quad \text{exeOk}^{\oplus}(p, \Gamma)$$

$$\text{(CHECK-CT1)} \quad \frac{\mathcal{L}[C:\dagger], p \vdash \mathcal{L}' : \text{ok } \forall C:\mathcal{L}', \mathcal{L}(C) = C:\mathcal{L}'}{p \vdash \mathcal{L} : \text{ok}} \quad \text{with } \mathcal{L} \notin \{\{_ \}^{\oplus}, \{_ \}^{\otimes}\} \quad \hat{\mathcal{L}}[C:\dagger], p \vdash \mathcal{L}' : \text{ok } \forall C:\mathcal{L}', \mathcal{L}^e(C) = C:\mathcal{L}^e \quad \mathcal{L}^e, p \vdash h\overline{e} : \text{ok } \forall h\overline{e}, \mathcal{L}^e(h) = h\overline{e}$$

$$\text{(CHECK-CT2)} \quad \frac{p \vdash \mathcal{L}^e : \text{ok}}{\text{with } \mathcal{L}^e \in \{\{_ \}^{\oplus}, \{_ \}^{\otimes}\}}$$

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

Type for expressions

$$\text{(PATH-PATH)} \quad \frac{p; \Gamma; \Sigma; \Phi \vdash \pi : \text{type } \pi}{\text{with } p(\pi) = \hat{\mathcal{L}} \text{ not interface } \text{if } \text{exe}^{\oplus}(p) \text{ then } \text{exe}^{\oplus}(p, \pi) \text{ if } \text{exe}(p) \text{ then } \text{exe}(p, \pi)}$$

$$\text{(PATH-ANY)} \quad \frac{p; \Gamma; \Sigma; \Phi \vdash \pi : \text{type Any}}{\text{with } \text{either } p(\pi) = \hat{\mathcal{L}} \text{ or } \pi \in \{\text{Any}, \text{Void}, \text{Library}\}}$$

$$\text{(LIB-T)} \quad \frac{p \oplus; \Gamma; \Sigma; \Phi \vdash \mathcal{L} : \text{immutable Library}}{\text{with } p \vdash \mathcal{L} \rightarrow \hat{\mathcal{L}} \quad p \vdash \hat{\mathcal{L}} : \text{ok} \quad \text{if } \text{exe}^{\oplus}(p) \text{ then } \hat{\mathcal{L}} = \{_ _ \}^{\oplus} _}$$

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

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

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

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

$$\text{(T-BLOCK-COMPLETE/PARTIAL)} \quad \frac{p; \Gamma_i[\mathcal{K}, \Sigma]; \Sigma; \Phi[\mathcal{K}] \vdash e_i : T'_i \leq \text{toPartial}(T_i) \quad \forall i \in 1..n \quad p; \Gamma_0; \Sigma \text{FV}(e_1 \dots e_n) \cup x_1 \dots x_n; \Phi \vdash e : T \quad p; \Gamma_0 \setminus \text{dom}(\Gamma'); \Sigma; \Phi \vdash \text{catch } q \ x (\mathcal{O}_i) : T \quad \forall i \in 1..k}{p; \Gamma; \Sigma; \Phi \vdash (T_1 x_1 = e_1 \dots T_n x_n = e_n \mathcal{K} e) : T}$$

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

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

$$\text{(THROW-T)} \quad \frac{p; \Gamma; \Sigma; \overline{T} \overline{\pi} \vdash e : _ \leq \mu \pi}{p; \Gamma; \Sigma; \overline{T} \overline{\pi} \vdash q e : T} \quad \text{with } \text{if } q = \text{return then } \mu \pi \in \overline{T}, \text{ otherwise } \mu = \text{immutable} \quad \text{if } q = \text{exception then } \pi \in \overline{\pi}$$

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

$$\text{(LOOP-T)} \quad \frac{p; \Gamma; \Sigma; \Phi \vdash e : \text{immutable Void}}{p; \Gamma, _ ; \Sigma; \Phi \vdash \text{loop } e : \text{immutable Void}}$$

$$\text{with } T \text{ not of form } \text{capsule } _ \quad \forall x:T \in \Gamma \quad p; \Gamma; \Sigma'; \Phi \vdash e : _ \leq \mu \pi \alpha$$

$$\text{(T-UNLOCK)} \quad \frac{p; \Gamma; \Sigma; \Phi \vdash e : \text{mutTolent}(\mu \pi \alpha)}{\text{with } \Sigma = \overline{x}; \overline{x}_0, \overline{x}_1 \dots \overline{x}_n; \overline{x} \quad \Sigma' = \overline{x}'; \overline{x}_1 \dots \overline{x}_n, \text{dom}^{\text{mut}}(\Gamma) \setminus \overline{x}_0 \dots \overline{x}_n; \overline{x} \quad \text{if } \mu \in \{\text{read}, \text{lent}, \text{mut}\} \text{ then } \overline{x}' = \overline{x} \text{ otherwise } \overline{x}' = \emptyset}$$

$$\text{(T-METHCALL)} \quad \frac{p; \Gamma_0; \Sigma_0; \Phi \vdash e_0 : _ \leq \mu_0 \pi_0 \quad p; \Gamma_i; \Sigma_i; \Phi \vdash e_i : T'_i \leq T_i \quad \forall i \in 1..n}{p; \Gamma; \Sigma; \Phi \vdash e_0.m(x_1:e_1 \dots x_n:e_n) : T'} \quad \text{with } \text{meth}_p(\pi_0.m, x_1 \dots x_n) = \mu_0 \text{ method } T m(T_1 x_1 \dots T_n x_n) \text{ exception } \overline{\pi} \quad p \vdash \overline{\pi} \leq \Phi \quad \Gamma_i \subseteq \Gamma \quad \forall i \in 0..n \quad \Gamma_i(x) = \Gamma_j(x) = \text{capsule } _ \text{ implies } i = j \quad T' = \begin{cases} T & \text{if } T'_i = \mu_i \pi_i \text{ for all } i \in 1..n \\ \text{toPartial}(T) & \text{otherwise} \end{cases} \quad \Sigma = \overline{x}; \overline{x}; \overline{x}_1 \dots \overline{x}_k \quad \overline{x}'_i = \text{FV}(e_0 \dots e_n \setminus e_i) \quad \Sigma_i = \overline{x}; \overline{x}; \overline{x}_1 \overline{x}'_i \dots \overline{x}_k \overline{x}'_i$$

$$\text{(T-MUTABLE)} \quad \frac{p; \Gamma; \Sigma'; \Phi \vdash e : \text{mut } \pi \alpha}{p; \Gamma; \Sigma; \Phi \vdash e : \text{capsule } \pi \alpha} \quad \text{with } \Sigma = \overline{x}; \overline{x}_1 \dots \overline{x}_n; \overline{x} \quad \Sigma' = \overline{x}; \overline{x}_1 \dots \overline{x}_n, \text{dom}^{\text{mut}}(\Gamma) \setminus \overline{x}_1 \dots \overline{x}_n; \overline{x} \quad p; \Gamma; \Sigma'; \Phi \vdash e : \mu \pi \alpha$$

$$\text{(T-READABLE)} \quad \frac{p; \Gamma; \Sigma; \Phi \vdash e : \text{immutable } \pi \alpha}{\text{with } \mu \in \{\text{read}, \text{lent}\} \quad \Sigma = \overline{x}; \overline{x}_1 \dots \overline{x}_n; \overline{x} \quad \overline{x}_0 = \text{dom}^{\text{mut}}(\Gamma) \setminus \overline{x}_1 \dots \overline{x}_n \quad \Sigma' = \overline{x}, \text{dom}^{\text{mut} \leq}(\Gamma); \overline{x}_0 \dots \overline{x}_n; \overline{x}}$$

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

$$\text{with } \text{either } q = \text{exception}, \mu' = \mu = \text{immutable} \text{ or } q = \text{return}, \text{type} \in \{\mu', \mu\}, \mu' \neq \mu \quad \mathcal{K}_1, \mathcal{K}_2 = \text{catch } q \ x (\text{on } \mu \text{ Library } e), \text{ catch } q \ x (\text{on } \mu' \text{ Void } e) \quad p; x:T', \Gamma; \Sigma; \Phi \vdash e : T$$

$$\text{(K-T)} \quad \frac{p; \Gamma; \Sigma; \Phi \vdash \text{catch } q \ x (\text{on } T' e) : T}{p; \Gamma; \Sigma; \Phi \vdash (\overline{d}_0 \overline{\mathcal{K}} (\overline{d}_1 \overline{\mathcal{K}} e_0)) : T} \quad \text{(DEC-F-T)} \quad \frac{}{p; \Gamma; \Sigma; \Phi \vdash (\overline{d}_0 \overline{d}_1 \overline{\mathcal{K}} e_0) : T}$$

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

Reduction rules

(GARBAGE)	$\frac{\mathcal{E}^p[(\overline{dv} \ \overline{d\mathcal{K}}e)] \longrightarrow_p \mathcal{E}^p[(\overline{d\mathcal{K}}e)]}{\text{with } \overline{dv} \neq \emptyset \\ \text{FV}((\overline{de})) \cap \text{dom}(\overline{dv}) = \emptyset}$	(METHCALL)	$\frac{\mathcal{E}^p[v.m(x_1:v_1 \dots x_n:v_n)] \longrightarrow_p \mathcal{E}^p[(\mu_0 \ \pi \ \text{this} = v \ T_1 \ x_1 = v_1 \dots T_n \ x_n = v_n e)]}{\text{with } \text{class}(\mathcal{E}^p, v) = \pi \\ \text{meth}_p(\pi.m(x_1 \dots x_n)) = \mu_0 \ \text{method } T \ m(T_1 \ x_1 \dots T_n \ x_n) \ \text{exception_} \ e}$		
(PRIMCALLREC)	$\frac{\mathcal{E}^p[v.m(x_1:v_1 \dots x_n:v_n)] \longrightarrow_p \mathcal{E}^p[(\mu \ \pi \ z = v \ z.m(x_1:v_1 \dots x_n:v_n))]}{\text{with } \text{class}(\mathcal{E}^p, v) = \pi \\ \text{meth}_p(\pi.m(x_1 \dots x_n)) = \mu \ \text{method } T \ m(T_1 \ x_1 \dots T_n \ x_n) \ \text{exception_} _ \\ v \text{ is a block}}$	(PRIMCALLARG)	$\frac{\mathcal{E}^p[v_0.m(\overline{x}:\overline{ax}_i:v\overline{x}:\overline{v})] \longrightarrow_p \mathcal{E}^p[(T'_i \ z = v_i v_0.m(\overline{x}:\overline{ax}_i:z\overline{x}:\overline{v}))]}{\text{with } v \text{ is a block} \\ \text{class}(\mathcal{E}^p, v_0) = \pi \\ x_1: \dots x_n: _ = \overline{x}:\overline{ax}_i:v\overline{x}:\overline{v} \\ \text{meth}_p(\pi.m(x_1 \dots x_n)) = \mu \ \text{method } T \ m(T_1 \ x_1 \dots T_n \ x_n) \ \text{exception_} _ \\ T'_i = \mu_i \ \pi_i \text{ and } T_i = \mu_i \ \pi_i _}$		
(FIELDABOJ)	$\frac{\mathcal{E}^p[x.m()] \longrightarrow_p \mathcal{E}^p[a_i]}{\text{with } \text{dec}(\mathcal{E}^p, x) = _ \ x = \pi.m'(x_1:a_1 \dots x_n:a_n) \\ m = x_i \text{ or } m = \#x_i \\ \text{meth}_p(\pi.m()) = \mu \ \text{method } T \ m() \ \text{exception_} _}$	(FIELDABLOCK)	$\frac{\mathcal{E}^p[x.m()] \longrightarrow_p \mathcal{E}^p[(\text{immutable } \pi' \ z = (\overline{dv} v.m()) \ z)]}{\text{with } \text{dec}(\mathcal{E}^p, x) = \text{immutable } \pi _ \ x = (\overline{dv} v) \\ \text{meth}_p(\pi.m()) = \mu \ \text{method_} \ \pi' \ m() \ \text{exception_} _}$	(LOOP)	$\frac{\mathcal{E}^p[\text{loop } e] \longrightarrow_p \mathcal{E}^p[e']}{\text{with } e' = (\text{immutable Void } x = e \ \text{loop } e)}$
(BLOCKELIM)	$\frac{\mathcal{E}^p[(\overline{dv}' \mu \ \pi \ \alpha \ x = (\overline{dv} v) \ \overline{d\mathcal{K}}e)] \longrightarrow_p \mathcal{E}^p[(\overline{dv}' \overline{dv} \mu \ \pi \ \alpha \ x = v \ \overline{d\mathcal{K}}e)]}{\text{with } \mu \geq \text{mut}}$	(SUBST)	$\frac{\mathcal{E}^p[(\overline{dv}' \mu \ \pi \ x = v \ \overline{d\mathcal{K}}e)] \longrightarrow_p \mathcal{E}^p[(\overline{dv}' \overline{d\mathcal{K}}e)[x = v]]}{\text{with } \text{either } v = a \text{ or } \mu = \text{capsule}}$		
(NORMALDEC)	$\frac{\mathcal{E}^p[(\overline{dv}' \mu \ \pi \ \alpha \ x = e' \ \overline{de})] \longrightarrow_p \mathcal{E}^p[(\overline{dv}' \overline{dv} \ \overline{de})]}{\text{with } \text{class}(\mathcal{E}^p, e') = \pi' \\ \overline{dv} = \mu \ \pi' \ x = e' [\text{Marco: } \overline{dv} \text{ is important}] \\ \text{either } \alpha \neq \emptyset \text{ or } \pi' \neq \pi}$	(NEW)	$\frac{\mathcal{E}^p[\pi.m(x_1:a_1 \dots x_n:a_n)] \longrightarrow_p \mathcal{E}^p[(\mu \ \pi \ z = \pi.m(x_1:a_1 \dots x_n:a_n) \ z)]}{\text{with } \text{meth}_p(\pi.m(x_1 \dots x_n)) = \text{type method } \mu \ \pi \ m(_) \ \text{exception_} _}$		
(FIELDU)	$\frac{\mathcal{E}^p[\mathcal{E}^p_1[x.m(\text{that}:a)]] \longrightarrow_p \mathcal{E}^p[(\overline{d}[x.m = a] \ \overline{\mathcal{K}}e)]}{\text{with } \text{move}_p(\mathcal{E}^p_1, \text{FV}(a)) = \langle \mathcal{E}^p_2, \emptyset \rangle \\ \mathcal{E}^p_2[\text{void}] = (\overline{d\mathcal{K}}e) \\ \mathcal{E}^p_1 = (\overline{dv} _, \overline{dv}(x) = \mu \ \pi \ x = _ \text{ and } \mu \in \{\text{mut}, \text{lent}\}) \\ \text{class}(\mathcal{E}^p_1, x) = \pi \\ \text{meth}_p(\pi.m(\text{that})) = _ \ \text{method_} \ m() \ \text{exception_} _ \\ e_1 \longrightarrow_{p'} e_2 \\ \mathcal{L} \xrightarrow{p} \mathcal{L}' \\ \max \\ \vdash p' : \text{ok}$	(R-USING)	$\frac{\mathcal{E}^p[e_0] \longrightarrow_p \mathcal{E}^p[e_1]}{\text{with } e_0 = \text{using } \pi \ \text{check } m(\overline{x}:\overline{v}) \ e \\ e_1 = \text{plugin}(p, \pi \ m(\overline{x}:\overline{v}) \ e)}$	(R-USING-OUT)	$\frac{\mathcal{E}^p[\text{using } \pi \ \text{check } m(\overline{x}:\overline{v}) \ e] \longrightarrow_p \mathcal{E}^p[e]}{\text{with } \text{plugin}(p, \pi \ m(\overline{x}:\overline{v}) \ e) \text{ undefined} \\ \text{either } e \text{ is a } v \text{ or } \text{throws}_p(e) = q \ v}$
(R-META)	$\frac{p' \otimes; \emptyset; \emptyset; \emptyset \vdash e_1 : \text{Library}}{\mathcal{E}^p[\mathcal{L}] \longrightarrow_p \mathcal{E}^p[\mathcal{L}[C:e_2]]}$ <p>with</p> $\mathcal{L} = \{ _ \prec: _ \overline{\mathcal{M}} C: e _ \}$ $p' = \mathcal{L}'^{\ominus}[C: \dagger] \ p$ $e_1 = \text{norm}_{p'}(e^e)$	(R-META-METHOD)	$\frac{\mathcal{L}_1 \longrightarrow_{p'} \mathcal{L}_2, \mathcal{L} \xrightarrow{p} \mathcal{L}'}{\mathcal{L} \xrightarrow{p} \mathcal{L}'}$ <p>with</p> $\mathcal{L} = \{ _ \prec: _ \overline{\mathcal{M}} e \mathcal{L}_1 _ \}$ $p' = \mathcal{L}' \ p$		
(R-CAPTURE)	$\frac{\mathcal{E}^p[e_1] \longrightarrow_p \mathcal{E}^p[(\overline{dv} \ \mu \ \pi \ z = v \ e)]}{\text{with } e_1 = (\overline{dv} \ \overline{dv}' \ T' \ x = e' \ \overline{d\text{catch}} \ q \ z \ \text{on } T \ e \ \overline{\mathcal{O}} e_0) \\ \text{throws}_p(e') = q \ v \\ \mu \ \pi = \text{norm}_p(T) \\ p \vdash \text{class}(\mathcal{E}^p[(\overline{dv} \square)], v) \leq \pi}$	(R-ONMISS)	$\frac{\mathcal{E}^p[e_1] \longrightarrow_p \mathcal{E}^p[(\overline{dv} \ T' \ x = q \ v \ \text{catch } q \ z \ \overline{\mathcal{O}} e_0)]}{\text{with } e_1 = (\overline{dv} \ \overline{dv}' \ T' \ x = e' \ \overline{d\text{catch}} \ q \ z \ \text{on } T \ e \ \overline{\mathcal{O}} e_0) \\ \text{throws}_p(e') = q' \ v \\ \mu \ \pi = \text{norm}_p(T) \\ \text{either } q \neq q' \text{ or not } p \vdash \text{class}(\mathcal{E}^p[(\overline{dv} \square)], v) \leq \pi}$	(R-KOUT)	$\frac{}{\mathcal{E}^p[(\overline{dv} \mathcal{K}e)] \longrightarrow_p \mathcal{E}^p[(\overline{dv} e)]}$

Desugering and compilation process

With \mathcal{L} as a source in the sugared language $\mathcal{W}[[\mathcal{L}]_{\emptyset; \text{immutable Void}; \emptyset}]$ is the corresponding desugared term. An **execution process** is a sequence $\mathcal{L}_0 \dots \mathcal{L}_n$ such that

$\emptyset | \mathcal{L}_0 \rightarrow \emptyset | \dots \rightarrow \emptyset | \mathcal{L}_n$.

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

Concrete syntax

immutable, **trait**, **exception** \emptyset in mh and $<:\emptyset$ in \mathcal{H} are represented with the empty string.

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

Well formedness

All the well formedness restriction of the core syntax applies here. Moreover in a \mathcal{B} all \bar{d}_i except the first are not empty and only the last \mathcal{K}_i can be omitted (having an empty \mathcal{O}). A \mathcal{B} can not be empty. **with** $\emptyset \emptyset \emptyset$ is not well formed; **with** $\bar{x} \bar{\mathcal{I}} \bar{d} (\bar{\mathcal{O}} \bar{\mathcal{W}} \bar{\mathcal{B}})$ is well formed if the number of types $T_1 \dots T_n$ in each **on** is the same of the sum of the cardinalities of \bar{x} , $\bar{\mathcal{I}}$ and \bar{d} . In a \mathcal{O}^w body, variables whose type have been made more specific can still beeing updated using the more general type, thus a well formed \mathcal{O}^w body can not read a variable after updating it. In a **with**, variables introduces in the $\bar{\mathcal{I}}$ can be updated only if they are declared **var**.

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

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

The **return** keyword can not be used inside any **if**, **case** or **while** condition or inside the expression of a **xine**.

Operator precedence

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

Definition: downloadFromWeb($_$)

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

Definition: terminating($_$)

terminating($q e$) = terminating(loop e) = true
 terminating(**if** $_$ **else** \mathcal{B}) =
 terminating(e) and terminating(\mathcal{B})
 terminating($(\bar{d}_1 \mathcal{K}_1 \dots \bar{d}_n \mathcal{K}_n e)$) = terminating(\mathcal{K}_1)
 and ... and terminating(\mathcal{K}_n) and terminating(e)
 terminating($\{\bar{d}_1 \mathcal{K}_1 \dots \bar{d}_n \mathcal{K}_n\}$) = terminating(\mathcal{K}_1)
 and ... and terminating(\mathcal{K}_n) and terminating(\bar{d}_n)
 terminating(**catch** $q x$ ($doc \bar{\mathcal{O}} default e$)) =
 terminating(**catch** $q x$ ($doc \bar{\mathcal{O}}$) and terminating(e)
 terminating(**catch** $_$ (**on** $e_1 \dots$ **on** e_n)) =
 terminating(e_1) and ... and terminating(e_n)
 terminating(**if** $_$ e) = terminating($\mathcal{W} \bar{x} \bar{d} e$) =
 terminating($\bar{d} e$) = terminating(e)
 terminating($_$) = false otherwise

Atomic Language Terms

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

Complete Language Syntax

e	::= \mathcal{L} x π void <i>num</i> <i>num</i> e e "string" $q e$ $x eqOp e$ <i>unOp</i> e	expression
\mathcal{B}	::= $e_1 binOp e_2$ $e (doc ps)$ if e else e' while e \mathcal{B} \mathcal{W} $e.m(doc ps)$	
\mathcal{K}	::= catch $q x$ ($doc \bar{\mathcal{O}} default e$)	catch-match
d	::= var $\bar{T} x = e$ $e < e \neq x >$ $C : e$	statement
\mathcal{W}	::= with $\bar{x} \bar{\mathcal{I}} \bar{d} (\bar{\mathcal{O}} \bar{\mathcal{W}} \bar{\mathcal{B}})$ with $\bar{\mathcal{I}} \mathcal{B}$	with
\mathcal{O}^w	::= on $\bar{T} e$ on $\bar{T} case e \mathcal{B}$ case $e \mathcal{B}$	type-case
\mathcal{H}	::= $m(\bar{\mathcal{F}})$ interface trait	lib. node h.
π	::= $C :: C$ $Outer^n :: C$ Any Void Library	node path
\mathcal{L}	::= $\{ doc \mathcal{H} <: \pi \bar{\mathcal{M}} \} \{ reuse url EOL \bar{\mathcal{M}} \}$	library literal
ps	::= $\bar{e} \bar{x} : \bar{e}$	parameters
$\bar{\mathcal{I}}$::= var x in e	iterator decl.
\mathcal{O}	::= on $T e$ on $T case e \mathcal{B}$	signal handler

Definition: guessType _{Γ} (e) note: guessType _{Γ} ($\{\bar{d}_1 \mathcal{K}_1 \dots \bar{d}_n \mathcal{K}_n\}$) is correctly undefined

guessType _{Γ} (\mathcal{L}) = **immutable Library**, guessType _{Γ} (x) = $\Gamma(x)$, guessType _{Γ} (π) = **type** π
 guessType _{Γ} (**void**) = guessType _{Γ} (loop e) =

guessType _{Γ} ($x eqOp e$) = guessType _{Γ} ($q e$) = guessType _{Γ} (**if** $e \mathcal{B}_1$ **else** \mathcal{B}_2) =

guessType _{Γ} (**while** $e \mathcal{B}$) = guessType _{Γ} ($\mathcal{W} \bar{\mathcal{B}}$) = **immutable Void**

guessType _{Γ} (*num* e) = guessType _{Γ} ($e.\#numberParser(\{\text{trait}\})$)

guessType _{Γ} (e " ") = guessType _{Γ} ($e.\#stringParser(\{\text{trait}\})$)

guessType _{Γ} (*unOp* e) = guessType _{Γ} ($e.\llbracket 'unOp \rrbracket ()$)

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

guessType _{Γ} ($e (ps)$) = guessType _{Γ} ($e.\#apply(ps)$)

guessType _{Γ} ($e.m(ps)$) = $T m(xsOf(ps))$ iff guessType _{Γ} (e) = $\pi \bar{m} \bar{x} = T$

guessType _{Γ} ($e.m(ps)$) = $\pi.m(xsOf(ps))$ iff guessType _{Γ} (e) = $\mu \pi^{\wedge}$

guessType _{Γ} ($(\bar{d}_1 \mathcal{K}_1 \dots \bar{d}_n \mathcal{K}_n e)$) = guessType _{Γ'} (e) with $\Gamma' =$ guessType _{Γ} ($\bar{d}_1 \dots \bar{d}_n$)
 and guessType _{Γ} ($_$) = Γ guessType _{Γ} ($e \bar{d}$) = guessType _{Γ} ($C : e \bar{d}$) = guessType _{Γ} (\bar{d})

guessType _{Γ} (**var** $T x = e \bar{d}$) = guessType _{$\Gamma, x \mapsto T$} (\bar{d})

guessType _{Γ} (**var** $x = e \bar{d}$) = guessType _{$\Gamma, x \mapsto$} guessType _{$\Gamma(e)$} (\bar{d})

guessType _{Γ} ($e [ps_1; \dots ps_n;]$) = guessType _{Γ} ($e.\#apply().\#add(ps_1) \dots \#add(ps_n)$)

guessType _{Γ} ($e [\mathcal{W} \bar{\mathcal{B}}]$) = guessType _{Γ} ($e.\#apply()$)

guessType _{Γ} ($e doc$) = guessType _{Γ} (**using** π **check** $m(doc ps)$ e) = guessType _{Γ} (e)

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

(1) **type method** $\mu Outer_0 m(\text{toPh}(T_1 x_1 \dots T_n x_n))$ **exception** $\emptyset \in$ c-f-type($\mu m(\text{var}_1 T_1 x_1 \dots \text{var}_n T_n x_n)$)

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

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

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

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

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

(dw) $\llbracket \{\text{reuse } url \bar{\mathcal{M}}\} \rrbracket_{\bar{\mathcal{L}}} = \{\text{downloadFromWeb}(url) \bar{\mathcal{M}}'\} \text{ iff } \llbracket \{\bar{\mathcal{M}}\} \rrbracket_{\emptyset; \bar{\mathcal{L}}} = \{\bar{\mathcal{M}}'\}$
 otherwise $\llbracket \{\mathcal{H} <: \bar{\pi} \bar{\mathcal{M}}\} \rrbracket_{\bar{\mathcal{L}}} = \{\mathcal{M}[\mathcal{H}]_\Theta <: [\bar{\pi}]_\Theta \mathcal{M}[\bar{\mathcal{M}}]_\Theta\}$
 with $\Theta = \emptyset$; immutable Void; $\{\mathcal{H} <: \bar{\pi} \bar{\mathcal{M}}\}, \bar{\mathcal{L}}$
 (cons) $\llbracket \{\mu m(\bar{\mathcal{F}}) \bar{\mathcal{M}}\} \rrbracket_\Theta = \llbracket \{\text{c-f-type}(\mu m(\bar{\mathcal{F}})) \bar{\mathcal{M}}\} \rrbracket_\Theta$
 (lt) $\llbracket \text{nume} \rrbracket_\Theta = \llbracket \text{e.\#numberParser}(\{'@StringEOL'\} \llbracket 'num' \rrbracket_{EOL}) \rrbracket_\Theta$
 $\llbracket \text{e"char"} \rrbracket_\Theta = \llbracket \text{e.\#stringParser}(\{'@StringEOL'\} \llbracket 'char' \rrbracket_{EOL}) \rrbracket_\Theta$
 $\llbracket \text{e" EOLchar EOL"} \rrbracket_\Theta = \llbracket \text{e.\#stringParser}(\{'@StringEOL'\} \llbracket 'char EOL' \rrbracket_{EOL}) \rrbracket_\Theta$
 (cf) $\llbracket \text{while } e \mathcal{B} \rrbracket_\Theta = \llbracket (\text{loop } (\text{e.\#checkTrue}() \mathcal{B}) \text{ catch exception (on Void void) void) } \rrbracket_\Theta$
 $\llbracket \text{if } a \mathcal{B}_1 \text{ else } \mathcal{B}_2 \rrbracket_\Theta = \llbracket (a.\#checkTrue() \text{ catch exception (on Void } \mathcal{B}_2) \mathcal{B}_1 \text{ void) } \rrbracket_\Theta$
 $\llbracket \text{if } e \mathcal{B}_1 \text{ else } \mathcal{B}_2 \rrbracket_\Theta = \llbracket (y \text{ e if } y \mathcal{B}_1 \text{ else } \mathcal{B}_2) \rrbracket_\Theta$ with y fresh and $e \neq a$
 $\llbracket (\bar{d}_1 \mathcal{K}_1 \dots \bar{d}_n \mathcal{K}_n e) \rrbracket_\Theta = \llbracket (\bar{d}_1 \mathcal{K}_1 (\dots (\bar{d}_n \mathcal{K}_n e) \dots)) \rrbracket_\Theta$
 $\llbracket \text{void} \rrbracket_\Theta = \text{void}$ $\llbracket \pi \rrbracket_{\bar{\mathcal{L}}} = \pi[\pi]_{\bar{\mathcal{L}}}$ $\llbracket e \text{ doc} \rrbracket_\Theta = \llbracket (\text{doc } e) \rrbracket_\Theta$
 $\llbracket x \rrbracket_\Theta = x$ $\llbracket x := e \rrbracket_\Theta = x.\text{inner}(\llbracket e \rrbracket_\Theta)$ $\llbracket \varrho e \rrbracket_\Theta = \varrho \llbracket e \rrbracket_\Theta$
 $\llbracket \text{loop } e \rrbracket_{\Gamma, T, \bar{\mathcal{L}}} = \text{loop } \llbracket e \rrbracket_{\Gamma, \text{immutable Void } \bar{\mathcal{L}}}$
 $\llbracket e.m(ps) \rrbracket_{\Gamma, T, \bar{\mathcal{L}}} = \llbracket e \rrbracket_{\Gamma, T, \bar{\mathcal{L}}}.m(\llbracket ps \rrbracket_{\Gamma, \text{guessType}_\Gamma(e)::m, \bar{\mathcal{L}}})$
 $\llbracket \text{using } \pi \text{ check.m}(ps) e \rrbracket_{\Gamma, T, \bar{\mathcal{L}}} = \text{using } [\pi]_\Theta \text{ check.m}(\llbracket ps \rrbracket_{\Gamma, \text{immutable Void } \bar{\mathcal{L}}}) \llbracket e \rrbracket_{\Gamma, T, \bar{\mathcal{L}}}$
 (tr) $\llbracket \{\bar{d}_1 \mathcal{K}_1 \dots \bar{d}_n \mathcal{K}_n\} \rrbracket_{\Gamma, T, \bar{\mathcal{L}}} = \llbracket ((\bar{d}_1 \mathcal{K}_1 \dots \bar{d}_n \mathcal{K}_n \text{ void}) \text{ catch return } y \text{ (on } T y) \text{ error void) } \rrbracket_{\Gamma, T, \bar{\mathcal{L}}}$
 (vd1) $\llbracket e_0 \rrbracket_\Theta = \llbracket (C:\{\text{mut (var } T \text{ inner) } \}) \rrbracket_\Theta$
 $\bar{d}_1 T x = \bar{e}_2 d' (\bar{d}_3 \bar{\mathcal{K}} e_1) [x \text{ eqOp} := z \text{ eqOp}] [x := z.\#inner()] \rrbracket_\Theta$
 with $e_0 = (\bar{d}_1 \text{var } T x = \bar{e}_2 d_3 \bar{\mathcal{K}} e_1)$ $d' = \text{mut Outer}_0::C z = \text{Outer}_0::C (\text{inner}:x)$
 not $x := _$ inside \bar{d}_2 and either $\bar{d}_3 = \emptyset$ or $\bar{d}_3 = d_$ and $x := _$ inside d
 $\llbracket (\bar{d} \text{var } x = \bar{e} d' \bar{\mathcal{K}} e_1) \rrbracket_{\Gamma, T', \bar{\mathcal{L}}} = \llbracket (\bar{d} \text{var } T x = \bar{e} d' \bar{\mathcal{K}} e_1) \rrbracket_{\Gamma, T', \bar{\mathcal{L}}}$ with $T = \text{guessType}_{\Gamma, \Gamma \text{ of } \bar{a}}(e)$
 $\llbracket (\bar{d}_1 e \bar{d}_2 \bar{\mathcal{K}} e_1) \rrbracket_\Theta = \llbracket (\bar{d}_1 \text{immutable void } x = \bar{e}_2 \bar{\mathcal{K}} e_1) \rrbracket_\Theta$
 $\llbracket (\bar{d}_1 \dots \bar{d}_n \bar{\mathcal{K}} e) \rrbracket_{\Gamma, T, \bar{\mathcal{L}}} = (d[\bar{d}_1]_\Theta \dots d[\bar{d}_n]_\Theta \mathcal{K}[\bar{\mathcal{K}}]_{\Gamma, T, \bar{\mathcal{L}}} \llbracket e \rrbracket_\Theta)$
 with $\Theta = \Gamma, \Gamma \text{ of } (d_1 \dots d_n); T; \bar{\mathcal{L}}$ $d[C:e] = C:e$ $d[T x = e]_{\Gamma, \bar{\mathcal{L}}} = T x = \llbracket e \rrbracket_{\Gamma, T, \bar{\mathcal{L}}}$
 $\mathcal{K}[\text{catch } \varrho \bar{\mathcal{O}} \text{ default } e]_\Theta = \mathcal{K}[\text{catch } \varrho x \text{ on } \bar{\mathcal{O}} \text{ default } e]_\Theta$
 $\mathcal{K}[\text{catch } \varrho x \text{ on } \bar{\mathcal{O}}]_\Theta = \llbracket \text{catch } \varrho x \text{ on Any (with } x (\bar{\mathcal{O}} \text{ default } \varrho x)) \rrbracket_\Theta$ iff on T if $e \mathcal{B} \in \bar{\mathcal{O}}$
 $\mathcal{K}[\text{catch } \varrho x \bar{\mathcal{O}} \text{ default } e]_\Theta = \llbracket \text{catch } \varrho x \text{ on Any (with } x (\bar{\mathcal{O}} \text{ default } e)) \rrbracket_\Theta$ iff on T if $e \mathcal{B} \in \bar{\mathcal{O}}$
 otherwise $\mathcal{K}[\text{catch } \varrho x \mathcal{O}_1 \dots \mathcal{O}_n]_\Theta = \text{catch } \varrho x \mathcal{K}[\mathcal{O}_1]_{x, \Theta} \dots \mathcal{K}[\mathcal{O}_n]_{x, \Theta}$ and
 $\mathcal{K}[\text{catch } \varrho x \mathcal{O}_1 \dots \mathcal{O}_n \text{ default } e]_\Theta = \text{catch } \varrho x \mathcal{K}[\mathcal{O}_1]_{x, \Theta} \dots \mathcal{K}[\mathcal{O}_n]_{x, \Theta} \text{ default } \llbracket e \rrbracket_{x: \text{immutable Any}, \Theta}$
 $\mathcal{K}[\text{on } T \mathcal{B}]_{x, \Theta} = \text{on } [T]_\Theta [\mathcal{B}]_{x: T, \Theta}$ (note: on T case $e \mathcal{B}$ is managed in the catch)

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

(init) $\llbracket e [ps_1; \dots; ps_n;] \rrbracket_\Theta = \llbracket \text{e.\#begin}() \text{.\#add}(ps_1) \dots \text{.\#add}(ps_n) \text{.\#end}() \rrbracket_\Theta$
 (op) $\llbracket e_1 \text{ binOp } e_2 \rrbracket_{\bar{\mathcal{L}}} = \llbracket e_1. \llbracket 'binOp' \rrbracket (e_2) \rrbracket_{\bar{\mathcal{L}}}$
 $\llbracket x \text{ eqOp } e \rrbracket_\Theta = \llbracket x := x.\#inner(). \llbracket 'eqOp' \rrbracket (e) \rrbracket_\Theta$
 $\llbracket \text{unOp } e \rrbracket_{\bar{\mathcal{L}}} = \llbracket e \rrbracket_{\bar{\mathcal{L}}}. \llbracket 'unOp' \rrbracket ()$
 $\llbracket e (ps) \rrbracket_{\bar{\mathcal{L}}} = \llbracket \text{e.\#apply}(ps) \rrbracket_{\bar{\mathcal{L}}}$

Definition: $\llbracket \text{doc} \rrbracket_p$

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

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

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

(a) $\mathcal{W} \llbracket \text{with } \bar{x} \bar{\mathcal{I}} \bar{d} (\bar{\mathcal{O}}^w) \rrbracket = \mathcal{W} \llbracket \text{with } \bar{x} \bar{\mathcal{I}} \bar{d} (\bar{\mathcal{O}}^w \text{ default void) } \rrbracket$
 (b) $\mathcal{W} \llbracket \text{with } \bar{x} \bar{d} (\bar{\mathcal{O}}^w \text{ default } e_2) \rrbracket = \mathcal{W} \llbracket (\bar{d} \text{ with } x_1 \dots x_n (\bar{\mathcal{O}}^w \text{ default } e_2)) \rrbracket$
 with $\text{with } \bar{x} \bar{d} = \text{with } x_1 \dots x_n _$
 (c) $\mathcal{W} \llbracket \text{with } \bar{x} (\bar{\mathcal{O}}^w \text{ default } e) \rrbracket = \mathcal{W} \llbracket \llbracket \bar{\mathcal{O}}^w \text{ default } e \rrbracket_{\bar{x}} \rrbracket$
 (ca) $\mathcal{W} \llbracket \text{on } T_1 \dots T_n \text{ case } \bar{e}_0 e_1 \bar{\mathcal{O}}^w \text{ default } e \rrbracket_{x_1 \dots x_n} = (\bar{e} \text{ cast } T^1 (y_1 \leftarrow x_1) \dots \text{cast } T^n (y_n \leftarrow x_n)$
 $\text{catch exception Void } \mathcal{W} \llbracket \bar{\mathcal{O}}^w \text{ default } e \rrbracket_{x_1 \dots x_n} (e_1 [x_1 T_1 := y_1] \dots [x_n T_n := y_n]) \text{ void})$
 with $y_1 \dots y_n$ fresh and either $\text{case } \bar{e}_0 = \bar{e} = \emptyset$
 or $\text{case } \bar{e}_0 = \text{case } e_0$ and $\bar{e} = \text{with } x_1 \dots x_n (\text{on } T_1 \dots T_n (\text{if } e_0 (\text{void}) \text{ else } (\text{exception void})))$
 (cb) $\mathcal{W} \llbracket \text{case } e_0 e_1 \bar{\mathcal{O}} \text{ default } e \rrbracket_{x_1 \dots x_n} = \text{if } e_0 e_1 \text{ else } (\mathcal{W} \llbracket \bar{\mathcal{O}} \text{ default } e \rrbracket_{x_1 \dots x_n})$
 (cc) $\mathcal{W} \llbracket \text{default } e \rrbracket = e$
 (d) $\mathcal{W} \llbracket \text{with } \bar{x} \bar{\mathcal{I}} \bar{d} (\bar{\mathcal{O}}^w \text{ default } e_2) \rrbracket = \mathcal{W} \llbracket \text{with } \bar{\mathcal{I}} (\bar{d} \text{ with } x_1 \dots x_n (\bar{\mathcal{O}}^w \text{ default } e_2)) \rrbracket$
 with $\text{with } \bar{x} \bar{\mathcal{I}} \bar{d} = \text{with } x_1 \dots x_n _$
 (e) $\mathcal{W} \llbracket \text{with } \bar{\mathcal{I}} \mathcal{B} \rrbracket = \mathcal{W} \llbracket \text{declarelts}(\bar{\mathcal{I}}, (\text{loop } (d_1 \mathcal{K}_1 \dots d_n \mathcal{K}_n \mathcal{B} [x_1 := x_1.\#inner()] \dots [x_n := x_n.\#inner()])$
 $\text{catch exception (on Void void) void) } \rrbracket$
 with $\bar{\mathcal{I}} = _ x_1 \text{ in } \dots x_n \text{ in } _, d_i \mathcal{K}_i = \text{next}_i(x_1 \dots x_n)$
 (initw) $\mathcal{W} \llbracket e [\text{with } \bar{x} \bar{\mathcal{I}} \bar{d} (\bar{\mathcal{O}}^w_1 \dots \bar{\mathcal{O}}^w_n \text{ default } e)] \rrbracket =$
 $\mathcal{W} \llbracket (\text{var } x = \text{e.\#begin}() \text{ with } \bar{x} \bar{\mathcal{I}} \bar{d} (\bar{\mathcal{O}}^w_1 \dots \bar{\mathcal{O}}^w_n \text{ default } e') x.\#end()) \rrbracket$
 with $\bar{\mathcal{O}}^w_i = \text{on } \bar{T} \text{ if } e, \bar{\mathcal{O}}^w_i = \text{on } \bar{T} \text{ if } e: x.\#add(e)$ and either $\text{default } e = \text{default } e' = \emptyset$
 or $\text{default } e = \text{default } e$ and $\text{default } e' = \text{default } x: x.\#add(e)$

Definition: $e[x := y]$

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

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

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

Definition: $\llbracket e \rrbracket_p$ auxiliary definitions

$ps \llbracket e_0 \bar{x}: \bar{e} \rrbracket_{\Gamma, T::m, \bar{\mathcal{L}}} = ps \llbracket \text{that: } e_0 \bar{x}: \bar{e} \rrbracket_{\Gamma, T::m, \bar{\mathcal{L}}}$
 $ps \llbracket x_1: e_1 \dots x_n: e_n \rrbracket_{\Gamma, T::m, \bar{\mathcal{L}}} =$
 $x_1: \llbracket e_1 \rrbracket_{\Gamma, T::m, (x_1 \dots x_n) :: x_1, \bar{\mathcal{L}}}$
 $x_n: \llbracket e_n \rrbracket_{\Gamma, T::m, (x_1 \dots x_n) :: x_n, \bar{\mathcal{L}}}$
 $ps \llbracket x_1: e_1 \dots x_n: e_n \rrbracket_{\Gamma, T, \bar{\mathcal{L}}} =$
 $x_1: \llbracket e_1 \rrbracket_{\Gamma, T, \bar{\mathcal{L}}} \dots x_n: \llbracket e_n \rrbracket_{\Gamma, T, \bar{\mathcal{L}}}$
 $\mathcal{M} \llbracket \mathcal{M}_1 \dots \mathcal{M}_n \rrbracket_p = \mathcal{M} \llbracket \mathcal{M}_1 \rrbracket_p \dots \mathcal{M} \llbracket \mathcal{M}_n \rrbracket_p$
 $\mathcal{M} \llbracket mhe \rrbracket_p = \mathcal{M} \llbracket \mathcal{M} \llbracket mh \rrbracket_p \llbracket e \rrbracket_{\emptyset; \text{Tof}(mh); p} \rrbracket$
 $\mathcal{M} \llbracket mh \mathcal{E}^* [(\bar{d} \text{ catch exception } x (\bar{\mathcal{O}} \text{ default } e) e_0)] \rrbracket =$
 $\mathcal{M} \llbracket mh \mathcal{E}^* [(\bar{d} \text{ catch exception } x (\bar{\mathcal{O}} \text{ on Any } e) e_0)] \rrbracket$
 $\mathcal{M} \llbracket mh \mathcal{E}^* [(\bar{d} \text{ catch return } x (\bar{\mathcal{O}} \text{ default } e) e_0)] \rrbracket =$
 $\mathcal{M} \llbracket mh \mathcal{E}^* [(\bar{d} \text{ catch return } x (\bar{\mathcal{O}} \text{ on } T e) e_0)] \rrbracket$
 where T is obtained using \mathcal{E}^* as the innermost

$\text{catch_return (on } T _)$ or Any if no such \mathcal{E}^* exists
 $\mathcal{M} \llbracket mh \mathcal{E}^* [(\bar{d}_1 C: \bar{e}_2 \bar{\mathcal{K}} e_0)] \rrbracket =$
 $\mathcal{M} \llbracket C: e \rrbracket \mathcal{M} \llbracket mh \mathcal{E}^* [(\bar{d}_1 \bar{d}_2 \bar{\mathcal{K}} e_0)] \rrbracket$
 otherwise $\mathcal{M} \llbracket mhe \rrbracket = mhe$
 $\mathcal{M} \llbracket C: \mathcal{E}^* [\dots] \rrbracket_p = \mathcal{M} \llbracket C: \mathcal{E}^* [e] \rrbracket_p$ Where e is found
 on the local system depending on the original
 position of such \dots symbol in the source and C
 $\mathcal{M} \llbracket \mu \text{method } T m(\bar{T} x) \text{ exception } \bar{\pi} \rrbracket_p =$
 $\mu \text{method } \pi[\bar{T}]_p [\llbracket m' \rrbracket (\pi[\bar{T} x]_p) \text{ exception } \pi[\bar{\pi}]_p]$
 $\mathcal{M} \llbracket \text{method } m(\bar{x}) \rrbracket_p = \text{method } \llbracket m' \rrbracket (\bar{x})$
 $\mathcal{M} \llbracket C: \rrbracket_p = C:$

$\pi \llbracket C:: \bar{C} \rrbracket_{\mathcal{L}_0 \dots \mathcal{L}_n} = \text{Outer}_{k::C::\bar{C}}$

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

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

$\pi \llbracket C:: \bar{C} \rrbracket_{\mathcal{L}_0 \dots \mathcal{L}_n} = \text{Outer}_{n::C::\bar{C}}$

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

Definition: Tof

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

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

$\text{Tof}(\text{method } m(\bar{x})) = \text{Outer}_0.m(\bar{x})$

Definition: declarelts($\bar{\mathcal{I}}, e_0$)

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

$\text{declarelts}(\text{var } x_0 \text{ in } e_0 \bar{\mathcal{I}}, e) = ($
 $x_0 = e_0 \quad ((\text{declarelts}(\bar{\mathcal{I}}, e)$
 $\text{catch exception } y (\text{default } (x_0.\#close() \text{ exception } y))$
 $\text{void})$
 $\text{catch return } y' (\text{default } (x_0.\#close() \text{ exception } y'))$
 $x_0.\#close())$
 $)$

Definition: next_i(\bar{x})

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

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

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

Definition: xsOf(ps)

$e_0 x_1: e_1 \dots x_n: e_n = \text{that } x_1 \dots x_n$

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

Plugin auxiliary definitions

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

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

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

$$\text{interface} \oplus \text{interface} = \text{interface}$$

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

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

$\mathcal{M} \oplus \bar{\mathcal{M}} = \mathcal{M}\bar{\mathcal{M}}$ if $\{\mathcal{M}\bar{\mathcal{M}}\}$ is well formed, otherwise

$$C: \text{doc}_1 \mathcal{L}_1 \oplus C: \text{doc}_2 \mathcal{L}_2 \bar{\mathcal{M}} = C: (\text{doc}_1 \oplus \text{doc}_2 \mathcal{L}_1 \oplus \mathcal{L}_2) \bar{\mathcal{M}}$$

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

where h_1, h_2 differs only for the documentation

$$\text{method doc m x e} \oplus h \bar{\mathcal{M}} = h \oplus \text{method doc m x e} \bar{\mathcal{M}} = \text{method}(\text{docsOf}(h) \oplus \text{doc}) \text{m x e} \bar{\mathcal{M}} \text{ fix in the code}$$

Definition: $p \vdash \mathcal{L}_1 \rtimes \mathcal{L}_2 = \mathcal{L}$

$$\mathcal{L}_1 \rtimes \mathcal{L}_2 = \mathcal{L}_1[p \text{ mapMx}(\mathcal{L}_2)][p \text{ mapC}(\mathcal{L}_2)]$$

$$\pi_1 \mapsto \pi_2 [\text{from } \pi_1] \in \text{mapC}(\mathcal{L}) \text{ iff } \mathcal{L}(\pi_1) = \{(\pi_2 \text{ that}) _ \}$$

$$\pi m (x_1 \dots x_n) \mapsto m' (x'_1 \dots x'_n) \in \text{mapMx}(\mathcal{L}) \text{ iff } \mathcal{L}(\pi) = \{\mathcal{H} \bar{\mathcal{M}}_1 \text{ method Void } m (\text{Void } x_1 \dots \text{Void } x_n) (\text{this.m}'(x'_1 : x_1 \dots x'_n : x_n)) \bar{\mathcal{M}}_2\}$$

Definition: $\mathcal{L}[p \pi . m x \mapsto m x'] = \mathcal{L}'$

$$\mathcal{L}[p \pi_1 . m x_1 \mapsto m x'_1 \dots \pi_n . m x_n \mapsto m x'_n] = \mathcal{L}_0 \oplus \dots \oplus \mathcal{L}_n$$

where:

$$p' = \text{removeTopLevel} \dagger(p)$$

$$\mathcal{L}' = \mathcal{L}[\text{renUsage}_{p'} \pi . m x \mapsto m x']$$

$$\mathcal{L}_0 = \mathcal{L}'[\text{remove } \pi_1 . m x_1] \dots [\text{remove } \pi_n . m x_n]$$

$$\mathcal{L}_i = \mathcal{L}'[\text{retainOnly } \pi_i . m x_i \mapsto m x'_i]$$

on purpose not put \dagger when composing $\mathcal{L} p$ to stop normalization scope

Definition: $\mathcal{L}[\text{renUsage}_{p'} \pi . m x \mapsto m x'] = \mathcal{L}'$

$\mathcal{L}[\text{renUsage}_{p'} \pi . m x \mapsto m x']$ and $e[\text{renUsage}_{\Gamma, p'} \pi . m x \mapsto m x']$ propagate on the subterms, but

$$(C: e)[\text{renUsage}_{\hat{\mathcal{L}} p} \pi . m x \mapsto m x'] =$$

$$C: (e[\text{renUsage}_{\hat{\mathcal{L}}[C: \dagger] p} \pi . m x \mapsto m x']),$$

$$\{\mathcal{H} \bar{\mathcal{M}}\}[\text{renUsage}_{p'} \pi . m x \mapsto m x'] = \{\mathcal{H}[\text{renUsage}_{\hat{\mathcal{L}} p} \text{Outer}_1 :: \pi . m x \mapsto m x'] \bar{\mathcal{M}}[\text{renUsage}_{\hat{\mathcal{L}} p} \text{Outer}_1 :: \pi . m x \mapsto m x']\} \text{ (first time not add outer1)}$$

with $p; \emptyset \vdash \{\mathcal{H} \bar{\mathcal{M}}\} \rightarrow \hat{\mathcal{L}}$ and

$$e.m(x_1 : e_1 \dots x_n : e_n) [\text{renUsage}_{\Gamma, p} \pi . m x \mapsto m x'] =$$

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

iff $\pi.m(x_1 \dots x_n) \mapsto m'(x'_1 \dots x'_n) \in \pi.m x \mapsto m x'$

norm_p(guessType_Γ(e)) = $_ \pi' _$ and norm_p(π) = π'

Actually, smarter way for block is used, looking in catches

Definition: $\mathcal{L}[p \pi \mapsto \pi'] = \mathcal{L}'$

$$\mathcal{L}[p \pi \mapsto \pi'] = \mathcal{L}_0 \oplus \dots \oplus \mathcal{L}_n$$

where:

$$p' = \text{removeTopLevel} \dagger(p)$$

$$\mathcal{L}' = \mathcal{L}[\text{renUsage}_{p'} \pi \mapsto \pi']$$

$$\mathcal{L}_0 = \mathcal{L}'[\text{remove } \pi_1] \dots [\text{remove } \pi_n]$$

$$\mathcal{L}_i = \mathcal{L}'[\text{redirectDefinition } \pi_i \mapsto \pi'_i]$$

Definition: $\mathcal{L}[\text{redirectDefinition } \pi \mapsto \pi'] = \mathcal{L}'$

$$\mathcal{L}[\text{redirectDefinition } \pi \mapsto \text{Library}] = \mathcal{L}[\text{redirectDefinition } \pi \mapsto \text{Void}] =$$

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

$$\mathcal{L}[\text{redirectDefinition } \text{Outer}_0 :: \bar{C}_0 \mapsto \text{Outer}_0 :: \bar{C}_1] =$$

$$\mathcal{L}(\bar{C}_0)[\text{from } \bar{C}_0 \pi'] [\text{encapsulateIn } \bar{C}_1] \text{ iff } [\text{to } \bar{C}_1] = \text{Outer}_1 :: \pi' : C'$$

otherwise $\mathcal{L}[\text{redirectDefinition } \text{Outer}_0 :: \bar{C}_0 \mapsto \text{Outer}_0 :: \bar{C}_1] =$

$$\mathcal{L}(\bar{C}_0)[\text{remove n outers}] [\text{encapsulateIn } \bar{C}_1] \text{ if } [\text{to } \bar{C}_1] = \text{Outer}_0 : C_1 :: \dots C_n$$

Definition: $\bar{C}_0[\text{to } \bar{C}_1] = \pi$

$$\bar{C}_0[\text{to } \emptyset] = \text{Outer}_0 :: \bar{C}_0$$

$$C :: \bar{C}_0[\text{to } C :: \bar{C}_0] = \bar{C}_0[\text{to } \bar{C}_0]$$

otherwise $\bar{C}_0[\text{to } C_1 :: \dots C_n] = \text{Outer}_n :: \bar{C}_0$

Definition: $\mathcal{L}[\text{renUsage}_{p'} \pi_1] \pi'_1 = \mathcal{L}'$

$$\mathcal{L}[\text{renUsage}_{p'} \pi] \pi' \text{ and } e[\text{renUsage}_{p'} \pi] \pi'$$

propagate on the subterms, but

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

$$C: (e[\text{renUsage}_{\hat{\mathcal{L}}[C: \dagger] p} \pi] \pi'),$$

$$\{\mathcal{H} \bar{\mathcal{M}}\}[\text{renUsage}_{p'} \pi] \pi' = \{\mathcal{H}[\text{renUsage}_{\hat{\mathcal{L}} p} \text{Outer}_1 :: \pi] \text{Outer}_1 :: \pi' \bar{\mathcal{M}}[\text{renUsage}_{\hat{\mathcal{L}} p} \text{Outer}_1 :: \pi] \text{Outer}_1 :: \pi'\} \text{ (first time not add outer1)}$$

with $p; \emptyset \vdash \{\mathcal{H} \bar{\mathcal{M}}\} \rightarrow \hat{\mathcal{L}}$

and $\pi_0[\text{renUsage}_{p'} \pi] \pi' = \text{norm}(\pi') \text{ iff } \text{norm}(\pi_0) = \text{norm}(\pi)$

Plugin auxiliary definitions

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

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

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

$$\text{interface} \oplus \text{interface} = \text{interface}$$

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

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

$\mathcal{M} \oplus \bar{\mathcal{M}} = \mathcal{M}\bar{\mathcal{M}}$ if $\{\mathcal{M}\bar{\mathcal{M}}\}$ is well formed, otherwise

$$C: \text{doc}_1 \mathcal{L}_1 \oplus C: \text{doc}_2 \mathcal{L}_2 \bar{\mathcal{M}} = C: (\text{doc}_1 \oplus \text{doc}_2 \mathcal{L}_1 \oplus \mathcal{L}_2) \bar{\mathcal{M}}$$

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

where h_1, h_2 differs only for the documentation

$$\text{method doc m x e} \oplus h \bar{\mathcal{M}} = h \oplus \text{method doc m x e} \bar{\mathcal{M}} = \text{method}(\text{docsOf}(h) \oplus \text{doc}) \text{m x e} \bar{\mathcal{M}} \text{ fix in the code}$$

Definition: $p \vdash \mathcal{L}_1 \rtimes \mathcal{L}_2 = \mathcal{L}$

$$\mathcal{L}_1 \rtimes \mathcal{L}_2 = \mathcal{L}_1[p \text{ mapMx}(\mathcal{L}_2)][p \text{ mapC}(\mathcal{L}_2)]$$

$$\pi_1 \mapsto \pi_2 [\text{from } \pi_1] \in \text{mapC}(\mathcal{L}) \text{ iff } \mathcal{L}(\pi_1) = \{(\pi_2 \text{ that}) _ \}$$

$$\pi m (x_1 \dots x_n) \mapsto m' (x'_1 \dots x'_n) \in \text{mapMx}(\mathcal{L}) \text{ iff } \mathcal{L}(\pi) = \{\mathcal{H} \bar{\mathcal{M}}_1 \text{ method Void } m (\text{Void } x_1 \dots \text{Void } x_n) (\text{this.m}'(x'_1 : x_1 \dots x'_n : x_n)) \bar{\mathcal{M}}_2\}$$

Definition: $\mathcal{L}[p \pi . m x \mapsto m x'] = \mathcal{L}'$

$$\mathcal{L}[p \pi_1 . m x_1 \mapsto m x'_1 \dots \pi_n . m x_n \mapsto m x'_n] = \mathcal{L}_0 \oplus \dots \oplus \mathcal{L}_n$$

where:

$$p' = \text{removeTopLevel} \dagger(p)$$

$$\mathcal{L}' = \mathcal{L}[\text{renUsage}_{p'} \pi . m x \mapsto m x']$$

$$\mathcal{L}_0 = \mathcal{L}'[\text{remove } \pi_1 . m x_1] \dots [\text{remove } \pi_n . m x_n]$$

$$\mathcal{L}_i = \mathcal{L}'[\text{retainOnly } \pi_i . m x_i \mapsto m x'_i]$$

on purpose not put \dagger when composing $\mathcal{L} p$ to stop normalization scope

Definition: $\mathcal{L}[\text{renUsage}_{p'} \pi . m x \mapsto m x'] = \mathcal{L}'$

$\mathcal{L}[\text{renUsage}_{p'} \pi . m x \mapsto m x']$ and $e[\text{renUsage}_{\Gamma, p'} \pi . m x \mapsto m x']$ propagate on the subterms, but

$$(C: e)[\text{renUsage}_{\hat{\mathcal{L}} p} \pi . m x \mapsto m x'] =$$

$$C: (e[\text{renUsage}_{\hat{\mathcal{L}}[C: \dagger] p} \pi . m x \mapsto m x']),$$

$$\{\mathcal{H} \bar{\mathcal{M}}\}[\text{renUsage}_{p'} \pi . m x \mapsto m x'] = \{\mathcal{H}[\text{renUsage}_{\hat{\mathcal{L}} p} \text{Outer}_1 :: \pi . m x \mapsto m x'] \bar{\mathcal{M}}[\text{renUsage}_{\hat{\mathcal{L}} p} \text{Outer}_1 :: \pi . m x \mapsto m x']\} \text{ (first time not add outer1)}$$

with $p; \emptyset \vdash \{\mathcal{H} \bar{\mathcal{M}}\} \rightarrow \hat{\mathcal{L}}$ and

$$e.m(x_1 : e_1 \dots x_n : e_n) [\text{renUsage}_{\Gamma, p} \pi . m x \mapsto m x'] =$$

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

iff $\pi.m(x_1 \dots x_n) \mapsto m'(x'_1 \dots x'_n) \in \pi.m x \mapsto m x'$

norm_p(guessType_Γ(e)) = $_ \pi' _$ and norm_p(π) = π'

Actually, smarter way for block is used, looking in catches

Definition: $\mathcal{L}[p \pi \mapsto \pi'] = \mathcal{L}'$

$$\mathcal{L}[p \pi \mapsto \pi'] = \mathcal{L}_0 \oplus \dots \oplus \mathcal{L}_n$$

where:

$$p' = \text{removeTopLevel} \dagger(p)$$

$$\mathcal{L}' = \mathcal{L}[\text{renUsage}_{p'} \pi \mapsto \pi']$$

$$\mathcal{L}_0 = \mathcal{L}'[\text{remove } \pi_1] \dots [\text{remove } \pi_n]$$

$$\mathcal{L}_i = \mathcal{L}'[\text{redirectDefinition } \pi_i \mapsto \pi'_i]$$

Definition: $\mathcal{L}[\text{redirectDefinition } \pi \mapsto \pi'] = \mathcal{L}'$

$$\mathcal{L}[\text{redirectDefinition } \pi \mapsto \text{Library}] = \mathcal{L}[\text{redirectDefinition } \pi \mapsto \text{Void}] =$$

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

$$\mathcal{L}[\text{redirectDefinition } \text{Outer}_0 :: \bar{C}_0 \mapsto \text{Outer}_0 :: \bar{C}_1] =$$

$$\mathcal{L}(\bar{C}_0)[\text{from } \bar{C}_0 \pi'] [\text{encapsulateIn } \bar{C}_1] \text{ iff } [\text{to } \bar{C}_1] = \text{Outer}_1 :: \pi' : C'$$

otherwise $\mathcal{L}[\text{redirectDefinition } \text{Outer}_0 :: \bar{C}_0 \mapsto \text{Outer}_0 :: \bar{C}_1] =$

$$\mathcal{L}(\bar{C}_0)[\text{remove n outers}] [\text{encapsulateIn } \bar{C}_1] \text{ if } [\text{to } \bar{C}_1] = \text{Outer}_0 : C_1 :: \dots C_n$$

Definition: $\bar{C}_0[\text{to } \bar{C}_1] = \pi$

$$\bar{C}_0[\text{to } \emptyset] = \text{Outer}_0 :: \bar{C}_0$$

$$C :: \bar{C}_0[\text{to } C :: \bar{C}_0] = \bar{C}_0[\text{to } \bar{C}_0]$$

otherwise $\bar{C}_0[\text{to } C_1 :: \dots C_n] = \text{Outer}_n :: \bar{C}_0$

Definition: $\mathcal{L}[\text{renUsage}_{p'} \pi_1] \pi'_1 = \mathcal{L}'$

$$\mathcal{L}[\text{renUsage}_{p'} \pi] \pi' \text{ and } e[\text{renUsage}_{p'} \pi] \pi'$$

propagate on the subterms, but

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

$$C: (e[\text{renUsage}_{\hat{\mathcal{L}}[C: \dagger] p} \pi] \pi'),$$

$$\{\mathcal{H} \bar{\mathcal{M}}\}[\text{renUsage}_{p'} \pi] \pi' = \{\mathcal{H}[\text{renUsage}_{\hat{\mathcal{L}} p} \text{Outer}_1 :: \pi] \text{Outer}_1 :: \pi' \bar{\mathcal{M}}[\text{renUsage}_{\hat{\mathcal{L}} p} \text{Outer}_1 :: \pi] \text{Outer}_1 :: \pi'\} \text{ (first time not add outer1)}$$

with $p; \emptyset \vdash \{\mathcal{H} \bar{\mathcal{M}}\} \rightarrow \hat{\mathcal{L}}$

and $\pi_0[\text{renUsage}_{p'} \pi] \pi' = \text{norm}(\pi') \text{ iff } \text{norm}(\pi_0) = \text{norm}(\pi)$