1 Syntax

1.1 Syntax

Interfaces	IL	::=	$\texttt{interface I extends } \overline{\texttt{I}}\ \{\overline{\texttt{M}}\}$
Methods	M	::=	$\texttt{I } m(\overline{\texttt{I}}\ x)\ \texttt{override J } \{\texttt{return } e;\}$
Expressions	e	::=	$x \mid e.m(\overline{e}) \mid \texttt{new I}() \mid e.\texttt{I} :: m(\overline{e}) \mid \texttt{super.I} :: m(e)$
Context	Γ	::=	$x_1 : \texttt{I}_1 ... x_n : \texttt{I}_n$

1.2 Subtyping

$$\texttt{I} <: \texttt{I}$$

$$\frac{\texttt{I} <: \texttt{J} \qquad \texttt{J} <: \texttt{K}}{\texttt{I} <: \texttt{K}}$$

$$\frac{\texttt{interface I extends } \overline{\texttt{I}}\ \{\overline{\texttt{M}}\}}{\forall \texttt{I}_i \in \overline{\texttt{I}}, \texttt{I} <: \texttt{I}_i}$$

1.3 Typing Rules

$$\text{(T-Var)}\ \Gamma \vdash x : \Gamma(x)$$

$$\text{(T-Invk)}\ \frac{\Gamma \vdash e_0 : \texttt{I}_0 \qquad \texttt{mtype}(m, \texttt{I}_0) = \overline{\texttt{J}} \to \texttt{I} \qquad \Gamma \vdash \overline{e} : \overline{\texttt{I}} \qquad \overline{\texttt{I}} <: \overline{\texttt{J}}}{\Gamma \vdash e_0.m(\overline{e}) : \texttt{I}}$$

$$\text{(T-PathInvk)}\ \frac{\Gamma \vdash e_0 : \texttt{I}_0 \qquad \texttt{I}_0 <: \texttt{J}_0 \qquad \texttt{mtype}(m, \texttt{J}_0) = \overline{\texttt{J}} \to \texttt{I} \qquad \Gamma \vdash \overline{e} : \overline{\texttt{I}} \qquad \overline{\texttt{I}} <: \overline{\texttt{J}}}{\Gamma \vdash e_0.\texttt{J}_0 :: m(\overline{e}) : \texttt{I}}$$

$$\text{(T-SuperInvk)}\ \frac{\Gamma \vdash \texttt{this} : \texttt{I}_0 \qquad \texttt{ext}(\texttt{I}_0, \texttt{J}_0) \qquad \texttt{mtype}(m, \texttt{J}_0) = \overline{\texttt{J}} \to \texttt{I} \qquad \Gamma \vdash \overline{e} : \overline{\texttt{I}} \qquad \overline{\texttt{I}} <: \overline{\texttt{J}}}{\Gamma \vdash {}_{\texttt{I}_0}\texttt{super.J}_0 :: m(\overline{e}) : \texttt{I}}$$

$$\text{(T-New)}\ \Gamma \vdash \texttt{new I}() : \texttt{I}$$

$$\text{(T-Method)}\ \frac{\texttt{ext}(\texttt{I}, \texttt{J}) \qquad \texttt{mtype}(m, \texttt{J}) = \overline{\texttt{I}} \to \texttt{I}_0 \qquad \text{If I} = \text{J then } \texttt{only}(m, \texttt{I}) = \texttt{true}}{\texttt{I}_0\ m(\overline{\texttt{I}}\ x)\ \texttt{override J } \{\texttt{return } e_0;\}\ \text{OK IN I}}$$

$$\text{(T-Intf)}\ \frac{\overline{\texttt{I}}\ \text{OK} \qquad \forall m \in \texttt{collectMethods}(\texttt{I}), \texttt{mbody}(m, \texttt{I}) \neq \texttt{Undefined}}{\texttt{interface I extends } \overline{\texttt{I}}\ \{\overline{\texttt{M}}\}\ \text{OK}}$$

1.4 Small-step Semantics

$$\text{(S-Invk)}\ \frac{\texttt{mbody}(m, \texttt{I}, \texttt{J}) = (\overline{\texttt{X}}\ \overline{x}, \texttt{E}'\ e_0)}{< \texttt{J} > \texttt{new I}().m(< \overline{\texttt{E}} > \overline{e}) \to [< \overline{\texttt{X}} > \overline{e}/\overline{x}, < \texttt{J} > \texttt{new I}()/\texttt{this}]e_0}$$

$$\text{(S-PathInvk)}\ \frac{\texttt{mbody}(m, \texttt{I}, \texttt{K}) = (\overline{\texttt{X}}\ \overline{x}, \texttt{E}'\ e_0)}{< \texttt{J} > \texttt{new I}().\texttt{K} :: m(< \overline{\texttt{E}} > \overline{e}) \to [< \overline{\texttt{X}} > \overline{e}/\overline{x}, < \texttt{J} > \texttt{new I}()/\texttt{this}]e_0}$$

$$\text{(S-SuperInvk)}\ \frac{\texttt{mbody}(m, \texttt{K}, \texttt{K}) = (\overline{\texttt{X}}\ \overline{x}, \texttt{E}'\ e_0)}{\texttt{super.K} :: m(< \overline{\texttt{E}} > \overline{e}) \to [< \overline{\texttt{X}} > \overline{e}/\overline{x}, < \texttt{J} > \texttt{new I}()/\texttt{this}]e_0}$$

1.5 Congruence

$$\text{(C-Receiver)}\ \frac{e \to e'}{e.m(\overline{e}) \to e'.m(\overline{e})}$$

$$\text{(C-Args)}\ \frac{e_i \to e_i'}{e.m(..., e_i, ...) \to e.m(..., e_i', ...)}$$

$$\text{(C-StaticType)}\ \texttt{new I}() \to < \texttt{I} > \texttt{new I}()$$

1.6 Auxilary Definitions

1.6.1 mbody

$$\frac{\mathsf{C}\{\mathsf{m}()\ \texttt{override}\ \mathsf{C}...\}}{\texttt{mbody}(\mathsf{m},\mathsf{C},\mathsf{A}) = (\overline{\mathsf{X}}\ \overline{\mathsf{x}},\mathsf{E}\ e_0)\ \texttt{IN}\ \mathsf{C}} \qquad \frac{\mathsf{C}\{\mathsf{m}()\ \texttt{override}\ \mathsf{A}...\}}{\texttt{mbody}(\mathsf{m},\mathsf{C},\mathsf{A}) = (\overline{\mathsf{X}}\ \overline{\mathsf{x}},\mathsf{E}\ e_0)\ \texttt{IN}\ \mathsf{C}} \qquad \frac{\texttt{mbody}(\mathsf{m},\mathsf{C}) = \{\mathsf{A}.\mathsf{m}(),\mathsf{B}.\mathsf{m}(),...\} \qquad \nexists\ \mathsf{C}.\mathsf{m}()}{\texttt{mbody}(\mathsf{m},\mathsf{C},\mathsf{A}) = (\overline{\mathsf{X}}\ \overline{\mathsf{x}},\mathsf{E}\ e_0)\ \texttt{IN}\ \mathsf{A}}$$

$$\texttt{interface}\ \mathsf{I}\ \texttt{extends}\ \overline{\mathsf{I}}\ \{\overline{\mathsf{M}}\}$$

$\texttt{mbody}(\mathsf{m},\mathsf{I})$ algorithm:

- If m is defined in I directly, then return I.m()
- Else, let $\overline{\mathsf{I}'} = \texttt{mdefined}(\texttt{fathers}(\mathsf{I}))$, all ancestors of I that has directly defined $\mathsf{m}()$.
- $\overline{\mathsf{I}''} = \texttt{needed}(\overline{\mathsf{I}'})$, keep only interfaces that are needed, which are not super-interface of others.
- If $\overline{\mathsf{I}''}$ is unique, then return this unique one. Else if any two I1,I2 in $\overline{\mathsf{I}''}$ share a parent in $\overline{\mathsf{I}'}$, then diamond conflict is detected, report error. Else return multiple $\mathsf{m}()$s.

1.6.2 mtype

$\texttt{mtype}(\mathsf{m},\mathsf{C})$ algorithm:

- If the result of $\texttt{mbody}(\mathsf{m},\mathsf{C},\mathsf{A})$ is a unique method, $\mathsf{I}_0\ \mathsf{m}(\overline{\mathsf{I}}\ \mathsf{x})\ \texttt{override}\ \mathsf{J}\ \{\texttt{return}\ e_0;\}$, then $\texttt{mtype}(\mathsf{m},\mathsf{C}) = \overline{\mathsf{I}} \to \mathsf{I}_0$
- Else (`Undefined` or multiple methods returned), $\texttt{mtype}(\mathsf{m},\mathsf{C}) = \texttt{Error}$

1.6.3 ext

$\texttt{ext}(\mathsf{I},\mathsf{J})$ means interface I (directly) extends J.

$$\frac{\texttt{interface}\ \mathsf{I}\ \texttt{extends}\ \overline{\mathsf{I}}\ \{\overline{\mathsf{M}}\} \qquad \mathsf{J} \in \overline{\mathsf{I}}}{\texttt{ext}(\mathsf{I},\mathsf{J}) = \texttt{true}}$$

$$\texttt{ext}(\mathsf{I},\mathsf{J}) = \texttt{false}$$

1.6.4 collectMethods

$$\texttt{collectMethods}(\mathsf{I}) = \left(\bigcup_{\mathsf{I}_i \in \overline{\mathsf{I}}} \texttt{methods}(\mathsf{I}_i)\right) \bigcup \texttt{methods}(\mathsf{I})$$

$$\texttt{methods}(\mathsf{I}) = \overline{\mathsf{M}}, \text{where } \mathrm{IT}(\mathsf{I}) = \texttt{interface}\ \mathsf{I}\ \texttt{extends}\ \overline{\mathsf{I}}\ \{\overline{\mathsf{M}}\}$$

1.6.5 needed

1.6.6 only

$\texttt{only}(\mathsf{m},\mathsf{I})$ is true iff inside I there is only one (direct) methoed m definition.