1 Syntax

1.1 Syntax

$$
\begin{array}{llll}
\text{Interfaces} & \text{IL} & ::= & \texttt{interface I extends } \overline{\text{I}}\ \{\overline{\text{M}}\} \\
\text{Methods} & \text{M} & ::= & \text{I } \text{m}(\overline{\text{I}}\ \text{x}) \texttt{ override J } \{\texttt{return } e;\} \\
\text{Expressions} & e & ::= & \text{x} \mid e.\text{m}(\overline{e}) \mid \texttt{new I}() \mid e.\text{I} :: \text{m}(\overline{e}) \mid \texttt{super.I} :: \text{m}(e) \\
\text{Context} & \Gamma & ::= & \text{x}_1 : \text{I}_1 ... \text{x}_n : \text{I}_n
\end{array}
$$

1.2 Subtyping

$$\text{I} <: \text{I}$$

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

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

1.3 Typing Rules

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

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

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

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

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

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

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

1.4 Small-step Semantics

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

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

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

1.5 Congruence

$$\text{Annotated Expressions} \quad \text{f} ::= e \mid < \text{I} > e$$

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

$$\text{(C-Args)}\ \frac{e_1 \to f}{< J > \texttt{new}\ I().m(< \overline{E} > \overline{e_0}, e_1, \overline{e_2}) \to < J > \texttt{new}\ I().m(< \overline{E} > \overline{e_0}, f, \overline{e_2})}$$

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

$$\text{(C-FReduce)}\ \frac{f \to f'\ f \neq \texttt{new}\ I()}{< I > f \to < I > f'}$$

$$\text{(C-AnnoReduce)} < I > (< J > e) \to < I > e$$

1.6 Auxilary Definitions

1.6.1 mbody

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

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

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

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

1.6.2 mtype

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

- If the result of $\texttt{mbody}(m, C, A)$ is a unique method, $I_0\ m(\overline{I}\ x)\ \texttt{override}\ J\ \{\texttt{return}\ e_0;\}$, then $\texttt{mtype}(m, C) = \overline{I} \to I_0$
- Else (`Undefined` or multiple methods returned), $\texttt{mtype}(m, C) = \texttt{Error}$

1.6.3 ext

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

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

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

1.6.4 collectMethods

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

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

1.6.5 needed

1.6.6 only

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