1 Syntax

1.1 Syntax

$$
\begin{array}{lllll}
\text{Interfaces} & IL & ::= & \texttt{interface I extends } \overline{I} \ \{\overline{M}\} \\
\text{Methods} & M & ::= & I\ m(\overline{I}\ x)\ \texttt{override}\ J\ \{\texttt{return}\ e;\} \\
\text{Expressions} & e & ::= & x \mid e.m(\overline{e}) \mid \texttt{new}\ I() \mid e.I :: m(\overline{e}) \mid \texttt{super.I} :: m(e) \\
\text{Context} & \Gamma & ::= & x_1 : I_1 ... x_n : I_n \\
\text{Values} & v & ::= & < I > \texttt{new}\ C()
\end{array}
$$

1.2 Subtyping

$$I <: I$$

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

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

1.3 Typing Rules

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

$$\text{(T-Invk)}\ \frac{\Gamma \vdash e_0 : I_0 \qquad \texttt{mtype}(m, I_0) = \overline{J} \to I \qquad \Gamma \vdash \overline{e} : \overline{I} \qquad \overline{I} <: \overline{J}}{\Gamma \vdash e_0.m(\overline{e}) : I}$$

$$\text{(T-PathInvk)}\ \frac{\Gamma \vdash e_0 : I_0 \qquad I_0 <: J_0 \qquad \texttt{mtype}(m, J_0) = \overline{J} \to I \qquad \Gamma \vdash \overline{e} : \overline{I} \qquad \overline{I} <: \overline{J}}{\Gamma \vdash e_0.J_0 :: m(\overline{e}) : I}$$

$$\text{(T-SuperInvk)}\ \frac{\Gamma \vdash \text{this} : I_0 \qquad \texttt{ext}(I_0, J_0) \qquad \texttt{mtype}(m, J_0) = \overline{J} \to I \qquad \Gamma \vdash \overline{e} : \overline{I} \qquad \overline{I} <: \overline{J}}{\Gamma \vdash {}_{I_0}\texttt{super.}J_0 :: m(\overline{e}) : I}$$

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

$$\text{(T-Method)}\ \frac{\texttt{ext}(I, J) \qquad \texttt{mtype}(m, J) = \overline{I} \to I_0 \qquad \text{If } I = J \text{ then } \texttt{only}(m, I) = \texttt{true}}{I_0\ m(\overline{I}\ x)\ \texttt{override}\ J\ \{\texttt{return}\ e_0;\}\ \text{OK IN I}}$$

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

1.4 Small-step Semantics

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

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

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

1.5 Congruence

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

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

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

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

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

$$\text{(C-SuperArgs)}\ \frac{e_1 \to f}{\texttt{super}.K :: m(< \overline{E} > \overline{v}, e_1, \overline{e_2}) \to \texttt{super}.K :: m(< \overline{E} > \overline{v}, f, \overline{e_2})}$$

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

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

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

1.6 Auxilary Definitions

1.6.1 mbody

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

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

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

mtype(m, C) algorithm:

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

1.6.3 ext

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

only(m, I) is true iff inside I there is only one (direct) methoed m definition.