

# 1 Syntax

## 1.1 Syntax

Interfaces	$IL$	$::=$	<code>interface I extends <math>\bar{I}</math> {<math>\bar{M}</math>}</code>
Methods	$M$	$::=$	<code>I m(<math>\bar{I}</math> x) override J {return e;}</code>
Expressions	$e$	$::=$	<code>x   e.m(<math>\bar{e}</math>)   new I()   e.I :: m(<math>\bar{e}</math>)   super.I :: m(e)</code>
Context	$\Gamma$	$::=$	<code><math>x_1 : I_1 \dots x_n : I_n</math></code>
Values	$v$	$::=$	<code>&lt; I &gt; new C()</code>

## 1.2 Subtyping

$$I <: I$$

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

$$\frac{\text{interface } I \text{ extends } \bar{I} \{ \bar{M} \}}{\forall I_i \in \bar{I}, I <: I_i}$$

## 1.3 Typing Rules

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

$$(T\text{-INVK}) \frac{\Gamma \vdash e_0 : I_0 \quad \text{mtype}(m, I_0) = \bar{J} \rightarrow I \quad \Gamma \vdash \bar{e} : \bar{I} \quad \bar{I} <: \bar{J}}{\Gamma \vdash e_0.m(\bar{e}) : I}$$

$$(T\text{-PATHINVK}) \frac{\Gamma \vdash e_0 : I_0 \quad I_0 <: J_0 \quad \text{mtype}(m, J_0) = \bar{J} \rightarrow I \quad \Gamma \vdash \bar{e} : \bar{I} \quad \bar{I} <: \bar{J}}{\Gamma \vdash e_0.J_0 :: m(\bar{e}) : I}$$

$$(T\text{-SUPERINVK}) \frac{\Gamma \vdash \text{this} : I_0 \quad \text{ext}(I_0, J_0) \quad \text{mtype}(m, J_0) = \bar{J} \rightarrow I \quad \Gamma \vdash \bar{e} : \bar{I} \quad \bar{I} <: \bar{J}}{\Gamma \vdash I_0.\text{super}.J_0 :: m(\bar{e}) : I}$$

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

$$(T\text{-METHOD}) \frac{\text{ext}(I, J) \quad \text{mtype}(m, J) = \bar{I} \rightarrow I_0 \quad \text{If } I = J \text{ then only}(m, I) = \text{true}}{I_0 \text{ m}(\bar{I} \text{ x}) \text{ override } J \{ \text{return } e_0; \} \text{ OK IN } I}$$

$$(T\text{-INTF}) \frac{\bar{I} \text{ OK} \quad \forall m \in \text{collectMethods}(I), \text{mbody}(m, I) \neq \text{Undefined}}{\text{interface } I \text{ extends } \bar{I} \{ \bar{M} \} \text{ OK}}$$

## 1.4 Small-step Semantics

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

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

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

## 1.5 Congruence

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

$$(C\text{-RECEIVER}) \frac{f \rightarrow f'}{f.m(\bar{e}) \rightarrow f'.m(\bar{e})}$$

$$(C\text{-ARGS}) \frac{e_1 \rightarrow f}{< J > \text{new } I().m(< \bar{E} > \bar{v}, e_1, \bar{e}_2) \rightarrow < J > \text{new } I().m(< \bar{E} > \bar{v}, f, \bar{e}_2)}$$

$$(C\text{-STATIC TYPE}) \text{new } I() \rightarrow < I > \text{new } I()$$

$$(C\text{-FREDUCE}) \frac{f \rightarrow f' \quad f \neq \text{new } I()}{< I > f \rightarrow < I > f'}$$

$$(C\text{-ANNOREDUCE}) < I > (< J > e) \rightarrow < I > e$$

## 1.6 Auxiliary Definitions

### 1.6.1 mbody

$$\frac{C\{m() \text{ override } C...\}}{mbody(m, C, A) = (\bar{X} \bar{x}, E \ e_0) \text{ IN } C} \quad \frac{C\{m() \text{ override } A...\}}{mbody(m, C, A) = (\bar{X} \bar{x}, E \ e_0) \text{ IN } C} \quad \frac{mbody(m, C) = \{A.m(), B.m(), \dots\} \quad \nexists C.m()}{mbody(m, C, A) = (\bar{X} \bar{x}, E \ e_0) \text{ IN } A}$$

$$\text{interface } I \text{ extends } \bar{I} \{\bar{M}\}$$

mbody(m, I) algorithm:

- If m is defined in I directly, then return I.m()
- Else, let  $\bar{I}' = mdefined(fathers(I))$ , all ancestors of I that has directly defined m().
- $\bar{I}'' = needed(\bar{I}')$ , keep only interfaces that are needed, which are not super-interface of others.
- If  $\bar{I}''$  is unique, then return this unique one. Else if any two I1, I2 in  $\bar{I}''$  share a parent in  $\bar{I}'$ , 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,  $I_0 \ m(\bar{I} \ x) \text{ override } J \ \{\text{return } e_0;\}$ , then  $mtype(m, C) = \bar{I} \rightarrow I_0$
- Else (Undefined or multiple methods returned),  $mtype(m, C) = \text{Error}$

### 1.6.3 ext

ext(I, J) means interface I (directly) extends J.

$$\frac{\text{interface } I \text{ extends } \bar{I} \{\bar{M}\} \quad J \in \bar{I}}{\text{ext}(I, J) = \text{true}}$$

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

### 1.6.4 collectMethods

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

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

### 1.6.5 needed

### 1.6.6 only

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