Formal Definition of SWIN language

Chenglong Wang Jun Li Yingfei Xiong Zhenjiang Hu

September 12, 2014

1 Featherweight Java

1.1 Syntax

Class Declaration

$$\texttt{CL} ::= \texttt{class C extends C}\{\bar{\texttt{C}}\ \bar{\texttt{f}}; \texttt{K}\ \bar{\texttt{M}}\}$$

Constructor Declaration

$$\texttt{K} ::= \texttt{C}\ (\bar{\texttt{C}}\ \bar{\texttt{f}})\ \{\texttt{super}(\bar{\texttt{f}}); \texttt{this}.\bar{\texttt{f}} = \bar{\texttt{f}}\}$$

Method Declaration

$$\texttt{M} ::= \texttt{C}\ \texttt{m}(\bar{\texttt{C}}\ \bar{\texttt{x}})\ \{\texttt{return t};\}$$

Term

$$\texttt{t} ::= \texttt{x} \mid \texttt{t.f} \mid \texttt{t.m}(\bar{\texttt{t}}) \mid \texttt{new C}(\bar{\texttt{t}}) \mid (\texttt{C})\ \texttt{t}$$

1.2 Type System

1.2.1 Subtyping

$$\frac{}{\texttt{C} <: \texttt{C}}\ \text{(S-SELF)} \qquad \frac{\texttt{C} <: \texttt{D} \quad \texttt{D} <: \texttt{E}}{\texttt{C} <: \texttt{E}}\ \text{(S-TRANS)}$$

$$\frac{CT(\texttt{C}) = \texttt{class C extends D}\ \{...\}}{\texttt{C} <: \texttt{D}}\ \text{(S-DEF)}$$

1.2.2 Auxiliary Functions

$$\frac{}{\texttt{fields}(\texttt{Object}) = \{\}}\ \text{FIELD-OBJECT}$$

$$\frac{\begin{array}{c}CT(\texttt{C}) = \texttt{class C extends D}\ \{\bar{\texttt{C}}\ \bar{\texttt{f}};\ \texttt{K}\ \bar{\texttt{M}}\}\\ \texttt{fields}(\texttt{D}) = \bar{\texttt{D}}\ \bar{\texttt{g}}\end{array}}{\texttt{fields}(\texttt{C}) = \bar{\texttt{D}}\ \bar{\texttt{g}},\ \bar{\texttt{C}}\ \bar{\texttt{f}}}\ \text{(FIELD-LOOKUP)}$$

$$\frac{\begin{array}{c}CT(\texttt{C}) = \texttt{class C extends D}\ \{\bar{\texttt{C}}\ \bar{\texttt{f}};\ \texttt{K}\ \bar{\texttt{M}}\}\\ \texttt{B m}\ (\bar{\texttt{B}}\ \bar{\texttt{x}})\ \{\texttt{return t};\} \in \bar{\texttt{M}}\end{array}}{\texttt{mtype}(\texttt{m}, \texttt{C}) = \bar{\texttt{B}} \to \texttt{B}}\ \text{(METHOD-LOOKUP1)}$$

$$\frac{\begin{array}{c}\texttt{CT}(\texttt{C}) = \texttt{class C extends D}\ \{\bar{\texttt{C}}\ \bar{\texttt{f}};\ \texttt{K}\ \bar{\texttt{M}}\}\\ \texttt{m}\ \text{is not defined in}\ \bar{\texttt{M}}\end{array}}{\texttt{mtype}(\texttt{m}, \texttt{C}) = \texttt{mtype}(\texttt{m}, \texttt{D})}\ \text{(METHOD-LOOPUP2)}$$

$$\frac{\texttt{mtype}(\texttt{m}, \texttt{D}) = \bar{\texttt{D}} \to \texttt{D}_0\ \texttt{implies}\ \bar{\texttt{C}} = \bar{\texttt{D}}\ \texttt{and}\ \texttt{C}_0 = \texttt{D}_0}{\texttt{override}(\texttt{m}, \texttt{D}, \bar{\texttt{C}} \to \texttt{C}_0)}\ \text{(OVERRIDE)}$$

1.2.3 Typing

$$\frac{\mathtt{x} : \mathtt{C} \in \Gamma}{\Gamma \vdash \mathtt{x} : \mathtt{C}} \quad \text{(FJ-VAR)}$$

$$\frac{\Gamma \vdash \mathtt{t}_0 : \mathtt{C}_0 \qquad \mathtt{fields}(\mathtt{C}_0) = \bar{\mathtt{C}}\ \bar{\mathtt{f}}}{\Gamma \vdash \mathtt{t}_0.\mathtt{f}_\mathtt{i} : \mathtt{C}_\mathtt{i}} \quad \text{(FJ-FIELD)}$$

$$\frac{\begin{array}{c}\Gamma \vdash \mathtt{t}_0 : \mathtt{C}_0 \qquad \mathtt{mtype}(\mathtt{m}, \mathtt{C}_0) = \bar{\mathtt{D}} \to \mathtt{C} \\ \Gamma \vdash \bar{\mathtt{t}} : \bar{\mathtt{C}} \qquad \bar{\mathtt{C}} <: \bar{\mathtt{D}}\end{array}}{\Gamma \vdash \mathtt{t}_0.\mathtt{m}(\bar{\mathtt{t}}) : \mathtt{C}} \quad \text{(FJ-INVK)}$$

$$\frac{\mathtt{fields}(\mathtt{C}) = \bar{\mathtt{D}}\ \bar{\mathtt{f}} \qquad \Gamma \vdash \bar{\mathtt{t}} : \bar{\mathtt{C}} \qquad \bar{\mathtt{C}} <: \bar{\mathtt{D}}}{\Gamma \vdash \mathtt{new}\ \mathtt{C}(\bar{\mathtt{t}}) : \mathtt{C}} \quad \text{(FJ-NEW)}$$

$$\frac{\Gamma \vdash \mathtt{t}_0 : \mathtt{D} \qquad \mathtt{D} <: \mathtt{C}}{\Gamma \vdash (\mathtt{C})\mathtt{t}_0 : \mathtt{C}} \quad \text{(FJ-UCAST)}$$

$$\frac{\Gamma \vdash \mathtt{t}_0 : \mathtt{D} \qquad \mathtt{C} <: \mathtt{D} \qquad \mathtt{C} \neq \mathtt{D}}{\Gamma \vdash (\mathtt{C})\mathtt{t}_0 : \mathtt{C}} \quad \text{(FJ-DCAST)}$$

$$\frac{\begin{array}{c}\Gamma \vdash \mathtt{t}_0 : \mathtt{D} \qquad \mathtt{C} \not<: \mathtt{D} \qquad \mathtt{D} \not<: \mathtt{C} \\ \mathrm{s}\mathit{tupid\ warning}\end{array}}{\Gamma \vdash (\mathtt{C})\mathtt{t}_0 : \mathtt{C}} \quad \text{(FJ-SCAST)}$$

$$\frac{\begin{array}{c}\bar{\mathtt{x}} : \bar{\mathtt{C}}, \mathtt{this} : \mathtt{C} \vdash \mathtt{t}_0 : \mathtt{E}_0 \qquad \mathtt{E}_0 <: \mathtt{C}_0 \\ \mathtt{CT}(\mathtt{C}) = \mathtt{class}\ \mathtt{C}\ \mathtt{extends}\ \mathtt{D}\ \{...\} \\ \mathtt{override}(\mathtt{m}, \mathtt{D}, \bar{\mathtt{C}} \to \mathtt{C}_0)\end{array}}{\mathtt{C}_0\ \mathtt{m}\ (\bar{\mathtt{C}}\ \bar{\mathtt{x}})\ \{\mathtt{return}\ \mathtt{t}_0;\}\ \mathtt{OK}\ \mathtt{in}\ \mathtt{C}} \quad \text{(FJ-M-OK)}$$

$$\frac{\begin{array}{c}\mathtt{K} = \mathtt{C}\ (\bar{\mathtt{C}}\ \bar{\mathtt{f}})\{\mathtt{super}(\bar{\mathtt{f}});\ \mathtt{this}.\bar{\mathtt{f}} = \bar{\mathtt{f}}\} \\ \mathtt{fields}(\mathtt{D}) = \bar{\mathtt{D}}\ \bar{\mathtt{g}} \qquad \bar{\mathtt{M}}\ \mathtt{OK}\ \mathtt{in}\ \mathtt{C}\end{array}}{\mathtt{class}\ \mathtt{C}\ \mathtt{extends}\ \mathtt{D}\ \{\bar{\mathtt{C}}\ \bar{\mathtt{f}}; \mathtt{K}\ \bar{\mathtt{M}}\}\ \mathtt{OK}} \quad \text{(FJ-C-OK)}$$

2 SWIN

2.1 Syntax

Π	$::=$	$\{\bar{\pi}\}$	SWIN program
π	$::=$	$(\bar{\mathtt{d}})\ [\mathtt{l} : \mathtt{C}_\mathtt{l} \ \to\ \mathtt{r} : \mathtt{C}_\mathtt{r}]$	rule
$\mathtt{d}$	$::=$	$\mathtt{x} : \mathtt{C}_1 \hookrightarrow \mathtt{C}_2$	variable declaration
$\mathtt{l}$	$::=$	$\mathtt{x.f} \mid \mathtt{new}\ \mathtt{C}(\bar{\mathtt{x}}) \mid \mathtt{x.m}(\bar{\mathtt{x}})$	code pattern
$\mathtt{r}$	$::=$	$\mathtt{t}$	FJ term

2.2 Environment

$$\begin{array}{rll} \texttt{API} & ::= \{\ \overline{\texttt{CL}}\ \} & \text{API definition} \\ \texttt{E} & ::= \{\ \overline{\texttt{x} : \texttt{C}_1 \hookrightarrow \texttt{C}_2}\ \} & \text{SWIN typing context} \end{array}$$

2.3 Evaluation Rules

$$\frac{\texttt{CL} = \texttt{class}\ \texttt{C}_1\ \texttt{extends}\ \texttt{C}_2\ \{\ \bar{\texttt{C}}_\texttt{i}\ \bar{\texttt{f}}_\texttt{i};\ \texttt{K}\ \bar{\texttt{M}}\ \}}{\Pi(\texttt{CL}) = \texttt{class}\ \Pi(\texttt{C}_1)\ \texttt{extends}\ \Pi(\texttt{C}_2)\ \{\ \Pi(\bar{\texttt{C}}_\texttt{i})\ \bar{\texttt{f}}_\texttt{i};\ \Pi(\texttt{K})\ \overline{\Pi(\texttt{M})}\ \}}\ \text{(E-DECLARATION)}$$

$$\frac{\texttt{K} = \texttt{C}_1\ (\bar{\texttt{C}}_2\ \bar{\texttt{f}}_2)\ \{\texttt{super}(\bar{\texttt{f}}_3);\ \texttt{this}.\bar{\texttt{f}}_\texttt{i} = \bar{\texttt{f}}_\texttt{j}\}}{\Pi(\texttt{K}) = \Pi(\texttt{C}_1)\ (\Pi(\bar{\texttt{C}}_2)\ \bar{\texttt{f}}_2)\ \{\texttt{super}(\bar{\texttt{f}}_3);\ \texttt{this}.\bar{\texttt{f}}_\texttt{i} = \bar{\texttt{f}}_\texttt{j}\}}\ \text{(E-CONSTRUCTOR)}$$

$$\frac{\texttt{M} = \texttt{C}_1\ \texttt{m}(\bar{\texttt{C}}_\texttt{m}\ \bar{\texttt{x}})\ \{\texttt{return}\ \texttt{t};\}}{\Pi(\texttt{M}) = \Pi(\texttt{C}_1)\ \texttt{m}(\Pi(\bar{\texttt{C}}_\texttt{m})\ \bar{\texttt{x}})\ \{\texttt{return}\ \Pi(\texttt{t});\}}\ \text{(E-METHOD)}$$

$$\frac{\texttt{C}_0 \hookrightarrow \texttt{C}_1 \in \textbf{TypeMapping}(\Pi)}{\Pi(\texttt{C}_0) = \texttt{C}_1}\ \text{(E-CLASS)}$$

$$\frac{\forall \texttt{C}.\ \texttt{C}_0 \hookrightarrow \texttt{C} \notin \textbf{TypeMapping}(\Pi)}{\Pi(\texttt{C}_0) = \texttt{C}_0}\ \text{(E-ALTER-CLASS)}$$

$$\frac{}{\Pi(\texttt{x}) = \texttt{x}}\ \text{(E-T-VALUE)}$$

$$\frac{(\texttt{x} : \texttt{C}_1 \hookrightarrow \texttt{C}_2)[\ \texttt{x.f} : \texttt{C}_1\ \rightarrow\ \texttt{r} : \texttt{C}_\texttt{r}\ ] \in \Pi \qquad \mathsf{Type}(\texttt{t}) < \texttt{C}_1}{\Pi(\texttt{t.f}) = [\ \texttt{x} \rightarrow \Pi(\texttt{t})\ ]\texttt{r}}\ \text{(E-T-FIELD)}$$

$$\frac{}{\Pi((\texttt{C})\ \texttt{t}) = (\Pi(\texttt{C}))\ \Pi(\texttt{t})}\ \text{(E-T-CAST)}$$

$$\frac{\begin{array}{c}(\bar{\texttt{d}})[\ \texttt{new}\ \texttt{C}_0(\ \bar{\texttt{x}}\ ) : \texttt{C}_1\ \rightarrow\ \texttt{r} : \texttt{C}_\texttt{r}] \in \Pi \\ \{\ \bar{\texttt{x}} : \overline{\texttt{C}_1 \hookrightarrow \texttt{C}_2}\ \} \subseteq \bar{\texttt{d}} \qquad \mathsf{Type}(\bar{\texttt{t}}_\texttt{u}) <: \bar{\texttt{C}}_1\end{array}}{\Pi(\texttt{new}\ \texttt{C}_0(\bar{\texttt{t}}_\texttt{u})) = [\ \bar{\texttt{x}} \rightarrow \overline{\Pi(\texttt{t}_\texttt{u})}\ ](\texttt{r})}\ \text{(E-T-NEW)}$$

$$\frac{\begin{array}{c}(\bar{\texttt{d}})[\ \texttt{x}_0.\texttt{m}_0(\ \bar{\texttt{y}}\ ) : \texttt{C}_1\ \rightarrow\ \texttt{r} : \texttt{C}_\texttt{r}] \in \Pi \\ \{\bar{\texttt{y}} : \overline{\texttt{C}_1 \hookrightarrow \texttt{C}_2},\ \texttt{x}_0 : \texttt{C}_3 \hookrightarrow \texttt{C}_4\} \subseteq \bar{\texttt{d}} \qquad \mathsf{Type}(\texttt{t}_0) <: \texttt{C}_3 \qquad \mathsf{Type}(\bar{\texttt{t}}_\texttt{u}) <: \bar{\texttt{C}}_1\end{array}}{\Pi(\texttt{t}_0.\texttt{m}_0(\bar{\texttt{t}}_\texttt{u})) = [\ \texttt{x}_0 \rightarrow \Pi(\texttt{t}_0),\ \bar{\texttt{y}} \rightarrow \overline{\Pi(\texttt{t}_\texttt{u})}\ ](\texttt{r})}\ \text{(E-T-INVOKE)}$$

$$\frac{\texttt{no other inference rule can be applied}}{\Pi(\texttt{new}\ \texttt{C}_0(\bar{\texttt{t}}_\texttt{u})) = \texttt{new}\ \texttt{C}_0(\ \overline{\Pi(\texttt{t}_\texttt{u})}\ )}\ \text{(E-ALTER-NEW)}$$

$$\frac{\texttt{no other inference rule can be applied}}{\Pi(\texttt{t}_0.\texttt{m}_0(\bar{\texttt{t}}_\texttt{u})) = \Pi(\texttt{t}_0).\texttt{m}(\ \overline{\Pi(\texttt{t}_\texttt{u})}\ )}\ \text{(E-ALTER-INVOKE)}$$

$$\frac{\texttt{no other inference rule can be applied}}{\Pi(\texttt{t.f}) = \Pi(\texttt{t}).\texttt{f}}\ \text{(E-ALTER-FIELD)}$$

2.4 Auxiliary Functions

$$\textbf{TypeMapping}([(\ \bar{\mathtt{x}} : \overline{\mathtt{C_1} \hookrightarrow \mathtt{C_2}}\ )\ [\mathtt{l} : \mathtt{C_1}\ \rightarrow\ \mathtt{r} : \mathtt{C_r}]]) = \{\mathtt{C_1} \hookrightarrow \mathtt{C_r}\} \cup \{\ \overline{\mathtt{C_1} \hookrightarrow \mathtt{C_2}}\ \}$$

$$\textbf{TypeMapping}(\{\bar{\pi}\}) = \bigcup_{\pi} (\textbf{TypeMapping}(\pi)) \qquad \text{(Extract type migration information)}$$

$$\textbf{Decl}(\mathtt{class\ C\ extends\ D}\ \{...\}) = \mathtt{C} \qquad \text{(Extract the declared class name)}$$

2.5 Type Checking Rules

$$\frac{\mathtt{x} : \mathtt{C_1} \hookrightarrow \mathtt{C_1'},\ \bar{\mathtt{y}} : \overline{\mathtt{C_2} \hookrightarrow \mathtt{C_2'}}\ \in \mathtt{E} \qquad \mathtt{class\ C_1\ extends\ D}\{\bar{\mathtt{C}}\ \bar{\mathtt{f}}; \mathtt{K}\ \bar{\mathtt{M}}\} \in \mathtt{API_s} \qquad \mathtt{C_d\ m}(\bar{\mathtt{C}}_\mathtt{s}\ \bar{\mathtt{u}})\{...\} \in \bar{\mathtt{M}} \qquad \bar{\mathtt{C}}_2 <: \bar{\mathtt{C}}_\mathtt{s}}{\mathtt{E} \vdash_\mathtt{l} \mathtt{x.m}(\bar{\mathtt{y}}) : \mathtt{C_d}} \text{ (T-L1)}$$

$$\frac{\mathtt{class\ C}\ \{\ \bar{\mathtt{C}}\ \bar{\mathtt{f}}; \mathtt{C}(\ \bar{\mathtt{C}}_\mathtt{s}\ \bar{\mathtt{u}}\ )\{...\}\ \bar{\mathtt{M}}\} \in \mathtt{API_s} \qquad \bar{\mathtt{x}} : \overline{\mathtt{C_1} \hookrightarrow \mathtt{C_1'}} \in \mathtt{E} \qquad \bar{\mathtt{C}}_1 <: \bar{\mathtt{C}}_\mathtt{s}}{\mathtt{E} \vdash_\mathtt{l} \mathtt{new\ C}(\bar{\mathtt{x}}) : \mathtt{C}} \text{ (T-L2)}$$

$$\frac{\mathtt{E} = \{\bar{\mathtt{x}} : \overline{\mathtt{C} \hookrightarrow \mathtt{D}}\} \qquad \{\ \bar{\mathtt{x}} : \bar{\mathtt{D}}\ \} \vdash_{\mathtt{FJ}}^{\mathtt{API_d}} \mathtt{t} : \mathtt{C_d}}{\mathtt{E} \vdash_\mathtt{r} \mathtt{t} : \mathtt{C_d}} \text{ (T-R)}$$

$$\frac{\{\ \bar{\mathtt{x}} : \overline{\mathtt{C} \hookrightarrow \mathtt{D}}\ \} \vdash_\mathtt{l} \mathtt{l} : \mathtt{C_1},\ \{\ \bar{\mathtt{x}} : \overline{\mathtt{C} \hookrightarrow \mathtt{D}}\ \} \vdash_\mathtt{r} \mathtt{r} : \mathtt{C_2}}{[\{\ \bar{\mathtt{x}} : \overline{\mathtt{C} \hookrightarrow \mathtt{D}}\ \}\ \mathtt{l} : \mathtt{C_1} \rightarrow \mathtt{r} : \mathtt{C_2}]\ ok} \text{ (T-}\pi\text{)}$$

$$
\begin{aligned}
\textbf{RuleOK}(\Pi) &= \forall\, \pi.(\pi \in \Pi \Rightarrow \pi\ ok) \\
\textbf{ConstrCover}(\Pi, \mathtt{API_s}, \mathtt{API_d}) &= \\
&\forall\, \mathtt{C_1}, \bar{\mathtt{C}}.(\mathtt{class\ C_1\ extends\ \_}\ \{\mathtt{C_1}(\bar{\mathtt{C}}\ \_)\ ...\ \} \in (\mathtt{API_s} - \mathtt{API_d}) \\
&\quad \Rightarrow\ \exists\, \mathtt{C_2}, \bar{\mathtt{C}}', \bar{\mathtt{x}}, \mathtt{r}.((\ \bar{\mathtt{x}} : \overline{\mathtt{C} \hookrightarrow \mathtt{C}'}\ )[\mathtt{new\ C_1}(\bar{\mathtt{x}}) : \mathtt{C_1} \rightarrow \mathtt{r} : \mathtt{C_2}] \in \Pi)) \\
\textbf{MethCover}(\Pi, \mathtt{API_s}, \mathtt{API_d}) &= \\
&\forall\, \mathtt{C_1}, \mathtt{C_2}, \mathtt{m}, \bar{\mathtt{C}}.(\mathtt{class\ C_1\ extends\ \_}\ \{\ \mathtt{C_2\ m}(\ \bar{\mathtt{C}}\ \_\ )\{...\}\ ...\ \} \in (\mathtt{API_s} - \mathtt{API_d}) \\
&\quad \Rightarrow\ \exists\, \mathtt{x}, \bar{\mathtt{y}}, \mathtt{C_1'}, \mathtt{C_2'}, \bar{\mathtt{C}}', \mathtt{r}.((\mathtt{x} : \mathtt{C_1} \hookrightarrow \mathtt{C_1'},\ \ \bar{\mathtt{y}} : \overline{\mathtt{C} \hookrightarrow \mathtt{C}'}\ )[\mathtt{x.m}(\bar{\mathtt{y}}) : \mathtt{C_2} \rightarrow \mathtt{r} : \mathtt{C_2'}] \in \Pi)) \\
\textbf{FieldCover}(\Pi, \mathtt{API_s}, \mathtt{API_d}) &= \\
&\forall\, \mathtt{C_1}, \mathtt{C_2}, \mathtt{f}.(\mathtt{class\ C_1\ extends\ \_}\ \{\mathtt{C\ f}; ...\} \in (\mathtt{API_s} - \mathtt{API_d}) \\
&\quad \Rightarrow\ \exists\, \mathtt{x}, \mathtt{C_1'}, \mathtt{C_2'}.((\mathtt{x} : \mathtt{C_1} \hookrightarrow \mathtt{C_1'}\ )[\mathtt{x.f} : \mathtt{C_2} \rightarrow \mathtt{r} : \mathtt{C_2'}] \in \Pi)) \\
\textbf{MapChecking}(\Pi, \mathtt{API_s}, \mathtt{API_d}) &= \\
&\forall\, \mathtt{C}, \mathtt{D}.(\mathtt{C} \hookrightarrow \mathtt{D} \in \textbf{TypeMapping}(\Pi) \\
&\quad \Rightarrow\ (\exists\, \mathtt{CL} \in \mathtt{API_s} \cap \mathtt{API_d}.(\textbf{Decl}(\mathtt{CL}) = \mathtt{C} \wedge \mathtt{D} = \mathtt{C})) \\
&\qquad \vee (\exists\, \mathtt{CL} \in \mathtt{API_s} - \mathtt{API_d}.(\textbf{Decl}(\mathtt{CL}) = \mathtt{C}))) \\
\textbf{Subtyping}(\Pi, \mathtt{API_s}, \mathtt{API_d}) &= \\
&\forall\, \mathtt{C_i}, \mathtt{D_i}, \mathtt{C_j}, \mathtt{D_j}.(\mathtt{C_i} \hookrightarrow \mathtt{D_i}, \mathtt{C_j} \hookrightarrow \mathtt{D_j} \in \textbf{TypeMapping}(\Pi)\ \ \Rightarrow\ \ (\mathtt{C_i} <: \mathtt{C_j} \Rightarrow \mathtt{D_i} <: \mathtt{D_j})) \\
\textbf{TypeSafe}(\Pi, \mathtt{API_s}, \mathtt{API_d}) &= \\
&\textbf{RuleOK}(\Pi) \wedge \textbf{ConstrCover}(\Pi, \mathtt{API_s}, \mathtt{API_d}) \wedge \textbf{MethCover}(\Pi, \mathtt{API_s}, \mathtt{API_d}) \\
&\wedge \textbf{FieldCover}(\Pi, \mathtt{API_s}, \mathtt{API_d}) \wedge \textbf{MapChecking}(\Pi, \mathtt{API_s}, \mathtt{API_d}) \wedge \textbf{Subtyping}(\Pi, \mathtt{API_s}, \mathtt{API_d})
\end{aligned}
$$

3 Metatheory

Lemma 1 (Typing Context)**.** *Given a SWIN program* Π *acting on* $\mathtt{API_s}$ *to* $\mathtt{API_d}$*, suppose the typing context for a term* $\mathtt{t}$ *is* $\Gamma_s = \bar{\mathtt{x}} : \bar{\mathtt{C}}$ *, then the typing context for* $\Pi(\mathtt{t})$ *is* $\Gamma_d = \bar{\mathtt{x}} : \overline{\Pi(\mathtt{C})}$.

Proof. According to the FJ typing rules, the typing context will not change once it is created in the rule FJ-M-OK. For the typing context Γ, except the variable `this`, all other variables in the typing context are bounded in the definition of a method $\mathtt{M}$.

Induction on Γ. Suppose $\Gamma = \bar{\mathtt{y}} : \bar{\mathtt{D}}, \mathtt{x} : \mathtt{C} = \Gamma_1, \mathtt{x} : \mathtt{C}$, then $\Pi(\Gamma_1) = \bar{\mathtt{y}} : \overline{\Pi(\mathtt{D})}$.

There are two cases for $\mathtt{x} : \mathtt{C}$

- $\mathtt{x} = \mathtt{this}$ in Γ. The type $\mathtt{C}$ is a client defined class type, so $\mathtt{C} \notin$ **TypeMapping**(Π). According to the rule E-ALTER-CLASS, $\Pi(\mathtt{C}) = \mathtt{C}$, then we have $\Pi(\Gamma) = \Pi(\Gamma_0), \mathtt{x} : \Pi(\mathtt{C}) = \bar{\mathtt{y}} : \overline{\Pi(\mathtt{D})}, \mathtt{this} : \mathtt{C}$.
- $\mathtt{x}$ is an argument in method declaration. According to the rule E-METHOD, after transformation, the type of $\mathtt{x}$ in the definition is $\Pi(\mathtt{C})$, thus $\Gamma = \Pi(\Gamma_0), \mathtt{x} : \Pi(\mathtt{C}) = \bar{\mathtt{y}} : \overline{\Pi(\mathtt{D})}, \mathtt{x} : \Pi(\mathtt{C})$.

With these two cases proved, the lemma is proved. □

Lemma 2 (Subtyping)**.** *Suppose* Π *passes SWIN type checking rules, and it transforms an FJ program with* $\mathtt{API_s}$ *to a new program with* $\mathtt{API_d}$*, then:*

$\mathtt{C_1} <: \mathtt{C_2}$ *in old program* $\Longrightarrow$ $\Pi(\mathtt{C_1}) <: \Pi(\mathtt{C_2})$ *in the transformed program.*

Proof. First, we suppose $\mathtt{C_1} <: \mathtt{C_2}$, in which $\mathtt{C_1} \neq \mathtt{C_2}$ and $\nexists \mathtt{C}$, s.t. $\mathtt{C_1} <: \mathtt{C'} <: \mathtt{C_2}$ and $\mathtt{C'} \neq \mathtt{C_1}, \mathtt{C'} \neq \mathtt{C_2}$.

Consider the two possibilities for $\mathtt{C_1}$:

- case-1: class $\mathtt{C_1}$ is defined in client code.

  In this case, the declaration of $\mathtt{C_1}$ should be $\mathtt{CL} = \mathtt{class}\ \mathtt{C_1}\ \mathtt{extends}\ \mathtt{C_2}\ \{...\}$. According to the rule $\mathtt{E-DECLARATION}$, we have $\Pi(\mathtt{CL}) = \mathtt{class}\ \Pi(\mathtt{C_1})\ \mathtt{extends}\ \Pi(\mathtt{C_2})\ \{...\}$. Thus we have $\Pi(\mathtt{C_1}) <: \Pi(\mathtt{C_2})$.

- case-2: class $\mathtt{C_1}$ is defined in API.

  In this case we have $\mathtt{C_2}$ is also a API defined class according to the definition of API in FJ. According to the checking rule **ConstrCover**, these exists $\mathtt{C_1} \hookrightarrow \mathtt{D_1}, \mathtt{C_2} \hookrightarrow \mathtt{D_2} \in$ **TypeMapping**(Π). By the checking rule **Subtyping** and the fact that $\mathtt{C_1} <: \mathtt{C_2}$, we have $\mathtt{D_1} = \Pi(\mathtt{C_1}) <: \mathtt{D_2} = \Pi(\mathtt{C_2})$.

With this case proved, for any $\mathtt{C_1} <: \mathtt{C_2}$, it can be split into $\mathtt{C_1} <: \mathtt{C'} <: ... <: \mathtt{C_2}$. Applying the proof on each step by induction, the lemma is proved. □

Lemma 3 (Variable Substitution)**.** *Suppose that an FJ term* $\mathtt{t}$ *is well typed under context* $\Gamma = \Gamma_1, \{\bar{\mathtt{x}} : \bar{\mathtt{C}}_x\}$*, i.e.* $\Gamma \vdash_{\mathtt{FJ}} \mathtt{t} : \mathtt{C_t}$*, then after substituting terms* $\bar{\mathtt{t}}_v$ *for variables* $\bar{\mathtt{x}}$ *, with the property that* $\Gamma_1 \vdash_{\mathtt{FJ}} \bar{\mathtt{t}}_v : \bar{\mathtt{C}}_v$ *and* $\bar{\mathtt{C}}_v <: \bar{\mathtt{C}}_x$*,* $\mathtt{t}$ *can be typed to* $\mathtt{C_t}$ *or a sub-class of* $\mathtt{C_t}$*. Namely,*

$$\Gamma_1, \{\bar{\mathtt{x}} : \bar{\mathtt{C}}_x\} \vdash_{\mathtt{FJ}} \mathtt{t} : \mathtt{C_t} \Longrightarrow \Gamma_1 \vdash_{\mathtt{FJ}} [\bar{\mathtt{x}} \to \bar{\mathtt{t}}_u]\mathtt{t} : \mathtt{C'_t},\ \mathtt{C'_t} <: \mathtt{C_t}$$

Proof. By induction on the derivition on an FJ term $\mathtt{t}$, there are five cases to discuss:

- case-1 $\mathtt{t} = \mathtt{x}, \Gamma_{\mathtt{FJ}}\mathtt{t} : \mathtt{C_t}, \mathtt{x} : \mathtt{C_t}$.

  In this case, we substitue an FJ term $\mathtt{t_u}$ for $\mathtt{x}$, where $\Gamma_1 \vdash_{\mathtt{FJ}} \mathtt{t_u} : \mathtt{C_u}$ and $\mathtt{C_u} <: \mathtt{C_t}$

□