Formal Definition of SWIN

Chenglong Wang Jun Li Yingfei Xiong
Zhenjiang Hu
{chenglongwang, lij, xiongyf}@pku.edu.cn, hu@nii.ac.jp

1 Featherweight Java

1.1 Syntax

Class Declaration

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

Constructor Declaration

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

Method Declaration

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

Term

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

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

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

$$\frac{CT(\texttt{C}) = \texttt{class C extends D}\ \{\overline{\texttt{C}}\ \overline{\texttt{f}};\ \texttt{K}\ \overline{\texttt{M}}\} \quad \texttt{fields(D)} = \overline{\texttt{D}}\ \overline{\texttt{g}}}{\texttt{fields(C)} = \overline{\texttt{D}}\ \overline{\texttt{g}},\ \overline{\texttt{C}}\ \overline{\texttt{f}}}\ \text{(FIELD-LOOKUP)}$$

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

$$\frac{CT(\texttt{C}) = \texttt{class C extends D}\ \{\overline{\texttt{C}}\ \overline{\texttt{f}};\ \texttt{K}\ \overline{\texttt{M}}\} \quad \texttt{m is not defined in}\ \overline{\texttt{M}}}{\texttt{mtype(m, C)} = \texttt{mtype(m, D)}}\ \text{(METHOD-LOOKUP2)}$$

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

1.2.3 Typing

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

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

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

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

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

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

$$\frac{\Gamma \vdash \mathtt{t}_0 : \mathtt{D} \qquad \mathtt{C} \not<: \mathtt{D} \qquad \mathtt{D} \not<: \mathtt{C} \qquad \textit{stupid warning}}{\Gamma \vdash (\mathtt{C})\mathtt{t}_0 : \mathtt{C}} \text{ (FJ-SCAST)}$$

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

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

2 SWIN

2.1 Syntax

Π	::=	$\{\bar{\pi}\}$	SWIN program
π	::=	$(\bar{\mathtt{d}})\ [\mathtt{l} : \mathtt{C}_1 \ \to\ \mathtt{r} : \mathtt{C}_r]$	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 API definition

$$\texttt{API} \;::=\; \{\ \overline{\texttt{CL}}\ \} \qquad \text{(API definition)}$$

2.3 Evaluation Rules

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

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

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

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

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

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

$$\frac{\begin{array}{c}(\texttt{x} : \texttt{C}_1 \hookrightarrow \texttt{C}_2)[\ \texttt{x.f} : \texttt{C}\ \rightarrow\ \texttt{r} : \texttt{D}\ ] \in \Pi \qquad \mathbf{Type}(\texttt{t}) <: \texttt{C}_1 \\ \nexists\ (\texttt{x} : \texttt{C}_3 \hookrightarrow \texttt{C}_4)[\ \texttt{x.f} : \texttt{C}\ \rightarrow\ \texttt{r} : \texttt{D}\ ] \in \Pi.(\mathbf{Type}(\texttt{t}) <: \texttt{C}_3 <: \texttt{C}_1 \wedge \texttt{C}_3 \neq \texttt{C}_1)\end{array}}{\Pi(\texttt{t.f}) = [\ \texttt{x} \mapsto \Pi(\texttt{t})\ ]\texttt{r}}\ \text{(E-T-FIELD)}$$

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

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

$$\frac{\begin{array}{c}(\bar{\texttt{y}} : \overline{\texttt{C}_1 \hookrightarrow \texttt{C}_2},\ \texttt{x}_0 : \texttt{C}_3 \hookrightarrow \texttt{C}_4)[\ \texttt{x}_0.\texttt{m}_0(\ \bar{\texttt{y}}\ ) : \texttt{C}\ \rightarrow\ \texttt{r} : \texttt{D}] \in \Pi \\ \mathbf{Type}(\texttt{t}_0) <: \texttt{C}_3 \qquad \mathbf{Type}(\bar{\texttt{t}}_\texttt{u}) <: \bar{\texttt{C}}_1 \\ \nexists\ (\bar{\texttt{y}} : \overline{\texttt{C}_1 \hookrightarrow \texttt{C}_2},\ \texttt{x}_0 : \texttt{C}_5 \hookrightarrow \texttt{C}_6)[\ \texttt{x}_0.\texttt{m}_0(\ \bar{\texttt{y}}\ ) : \texttt{C}\ \rightarrow\ \texttt{r} : \texttt{D}] \in \Pi.(\mathbf{Type}(\texttt{t}_0) <: \texttt{C}_5 <: \texttt{C}_3 \wedge \texttt{C}_5 \neq \texttt{C}_3)\end{array}}{\Pi(\texttt{t}_0.\texttt{m}_0(\bar{\texttt{t}}_\texttt{u})) = [\ \texttt{x}_0 \mapsto \Pi(\texttt{t}_0),\ \bar{\texttt{y}} \mapsto \overline{\Pi(\texttt{t}_\texttt{u})}\ ](\texttt{r})}\ \text{(E-T-INVOKE)}$$

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

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

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

2.4 Auxiliary Functions

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

$\mathbf{TypeMapping}(\{\bar{\pi}\}) = \bigcup_{\pi}\ (\mathbf{TypeMapping}(\pi))$ (Extract type migration information)

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

2.5 Type Checking Rules

$$\frac{\{\ \bar{\mathtt{x}} : \bar{\mathtt{C}}\ \} \vdash_{\mathtt{l}}^{\mathtt{API_s}} \mathtt{l} : \mathtt{C_l} \qquad \{\ \bar{\mathtt{x}} : \bar{\mathtt{D}}\ \} \vdash_{\mathtt{FJ}}^{\mathtt{API_d}} \mathtt{r} : \mathtt{C_2}}{(\bar{\mathtt{x}} : \overline{\mathtt{C} \hookrightarrow \mathtt{D}})[\mathtt{l} : \mathtt{C_l} \rightarrow \mathtt{r} : \mathtt{C_2}]\ \mathit{ok}}\ \text{(T-}\pi\text{)}$$

$$\frac{\Gamma \vdash_{\mathtt{FJ}}^{\mathtt{API}} \mathtt{x} : \mathtt{C_0} \quad \mathtt{mtype}(\mathtt{m}, \mathtt{C_0}) = \bar{\mathtt{D}} \rightarrow \mathtt{C} \quad \Gamma \vdash_{\mathtt{FJ}}^{\mathtt{API}} \bar{\mathtt{y}} : \bar{\mathtt{D}}}{\Gamma \vdash_{\mathtt{l}}^{\mathtt{API}} \mathtt{x.m}(\bar{\mathtt{y}}) : \mathtt{C}}\ \text{(T-L1)}$$

$$\frac{\mathtt{fields}(\mathtt{C}) = \bar{\mathtt{D}}\ \bar{\mathtt{f}} \quad \Gamma \vdash_{\mathtt{FJ}}^{\mathtt{API}} \bar{\mathtt{x}} : \bar{\mathtt{D}}}{\Gamma \vdash_{\mathtt{l}}^{\mathtt{API}} \mathtt{new\ C}(\bar{\mathtt{x}}) : \mathtt{C}}\ \text{(T-L2)}$$

$$\frac{\mathtt{fields}(\mathtt{C}) = \bar{\mathtt{D}}\ \bar{\mathtt{f}} \quad \Gamma \vdash_{\mathtt{FJ}}^{\mathtt{API}} \mathtt{x} : \mathtt{C}}{\Gamma \vdash_{\mathtt{l}}^{\mathtt{API}} \mathtt{x.f_i} : \mathtt{D_i}}\ \text{(T-L3)}$$

$\mathbf{RuleOK}(\Pi) = \forall\ \pi.(\pi \in \Pi \Rightarrow \pi\ \mathit{ok})$

$\mathbf{ConstrCover}(\Pi, \mathtt{API_s}, \mathtt{API_d}) =$
$\forall\ \mathtt{C_1}, \bar{\mathtt{C}}.(\mathtt{class\ C_1\ extends\ \_}\ \{\mathtt{C_1}(\bar{\mathtt{C}}\ \_)\ ...\ \} \in (\mathtt{API_s} - \mathtt{API_d})$
$\Rightarrow\ \exists\ \mathtt{C_2}, \bar{\mathtt{C}}', \bar{\mathtt{x}}, \mathtt{r}.((\ \bar{\mathtt{x}} : \overline{\mathtt{C} \hookrightarrow \mathtt{C}'}\ )[\mathtt{new\ C_1}(\bar{\mathtt{x}}) : \mathtt{C_1} \rightarrow \mathtt{r} : \mathtt{C_2}] \in \Pi))$

$\mathbf{MethCover}(\Pi, \mathtt{API_s}, \mathtt{API_d}) =$
$\forall\ \mathtt{C_1}, \mathtt{C_2}, \mathtt{m}, \bar{\mathtt{C}}.(\mathtt{class\ C_1\ extends\ \_}\ \{\ \mathtt{C_2\ m}(\ \bar{\mathtt{C}}\ \_)\{...\}\ ...\ \} \in (\mathtt{API_s} - \mathtt{API_d})$
$\Rightarrow\ \exists\ \mathtt{x}, \bar{\mathtt{y}}, \mathtt{C_1'}, \mathtt{C_2'}, \bar{\mathtt{C}}', \mathtt{r}.((\mathtt{x} : \mathtt{C_1} \hookrightarrow \mathtt{C_1'},\ \ \bar{\mathtt{y}} : \overline{\mathtt{C} \hookrightarrow \mathtt{C}'}\ )[\mathtt{x.m}(\bar{\mathtt{y}}) : \mathtt{C_2} \rightarrow \mathtt{r} : \mathtt{C_2'}] \in \Pi))$

$\mathbf{FieldCover}(\Pi, \mathtt{API_s}, \mathtt{API_d}) =$
$\forall\ \mathtt{C_1}, \mathtt{C_2}, \mathtt{f}.(\mathtt{class\ C_1\ extends\ \_}\ \{\mathtt{C_2\ f}; ...\} \in (\mathtt{API_s} - \mathtt{API_d})$
$\Rightarrow\ \exists\ \mathtt{x}, \mathtt{C_1'}, \mathtt{C_2'}.((\mathtt{x} : \mathtt{C_1} \hookrightarrow \mathtt{C_1'}\ )[\mathtt{x.f} : \mathtt{C_2} \rightarrow \mathtt{r} : \mathtt{C_2'}] \in \Pi))$

$\mathbf{MapChecking}(\Pi, \mathtt{API_s}, \mathtt{API_d}) =$
$\forall\ \mathtt{C}, \mathtt{D}.(\mathtt{C} \hookrightarrow \mathtt{D} \in \mathbf{TypeMapping}(\Pi)$
$\Rightarrow\ (\exists\ \mathtt{CL} \in \mathtt{API_s} \cap \mathtt{API_d}.(\mathbf{Decl}(\mathtt{CL}) = \mathtt{C} \wedge \mathtt{D} = \mathtt{C}))$
$\vee\ (\exists\ \mathtt{CL} \in \mathtt{API_s} - \mathtt{API_d}.(\mathbf{Decl}(\mathtt{CL}) = \mathtt{C})))$

$\mathbf{Subtyping}(\Pi, \mathtt{API_s}, \mathtt{API_d}) =$
$\forall\ \mathtt{C_i}, \mathtt{D_i}, \mathtt{C_j}, \mathtt{D_j}.(\mathtt{C_i} \hookrightarrow \mathtt{D_i}, \mathtt{C_j} \hookrightarrow \mathtt{D_j} \in \mathbf{TypeMapping}(\Pi)\ \ \Rightarrow\ \ (\mathtt{C_i} <: \mathtt{C_j} \Rightarrow \mathtt{D_i} <: \mathtt{D_j}))$

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* $\Gamma_\mathtt{s} = \bar{\mathtt{x}} : \bar{\mathtt{C}}$ *, then the typing context for* $\Pi(\mathtt{t})$ *is* $\Gamma_\mathtt{d} = \bar{\mathtt{x}} : \overline{\Pi(\mathtt{C})}$*. (For simplicity, we use* $\Pi(\Gamma_\mathtt{s})$ *to represent* $\bar{\mathtt{x}} : \overline{\Pi(\mathtt{C})}$*, i.e.* Π *will act on all variable types in a typing context.)*

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 `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 $\mathtt{C} \notin \mathbf{TypeMapping}(\Pi)$. 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* $\implies$ $\Pi(\mathtt{C}_1) <: \Pi(\mathtt{C}_2)$ *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 $\mathtt{C}_1 \hookrightarrow \mathtt{D}_1, \mathtt{C}_2 \hookrightarrow \mathtt{D}_2 \in \mathbf{TypeMapping}(\Pi)$. 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 $\bar{\mathtt{t}}_u$ for variables $\bar{\mathtt{x}}$, with the property that $\Gamma_1 \vdash_{\mathtt{FJ}} \bar{\mathtt{t}}_u : \bar{\mathtt{C}}_u$ and $\bar{\mathtt{C}}_u <: \bar{\mathtt{C}}_x$, $\mathtt{t}$ can be typed to $\mathtt{C}_t$ or a sub-class of $\mathtt{C}_t$. Namely,*

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

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

- case-1: $\mathtt{t} = \mathtt{x}$ and $\mathtt{x}$ is a variable.

  In this case, we substitute an FJ term $\mathtt{t}_u$ for $\mathtt{x}$, where $\Gamma_1 \vdash_{\mathtt{FJ}} \mathtt{t}_u : \mathtt{C}_u$ and $\mathtt{C}_u <: \mathtt{C}_t$, then the substitution will be $[\mathtt{x} \mapsto \mathtt{t}_u]\mathtt{x} \Rightarrow \mathtt{t}_u$. As $\Gamma_1 \vdash_{\mathtt{FJ}} \mathtt{t}_u : \mathtt{C}_u <: \mathtt{C}_t$, we have $\Gamma_1 \vdash_{\mathtt{FJ}} [\mathtt{x} \mapsto \mathtt{t}_u]\mathtt{x} : \mathtt{C}_u <: \mathtt{C}_t$. Then we have the case proved.

- case-2: $\mathtt{t} = (\mathtt{C})\mathtt{t}_1$ and $\Gamma_1, \{\bar{\mathtt{x}} : \bar{\mathtt{C}}_x\} \vdash_{\mathtt{FJ}} \mathtt{t}_1 : \mathtt{C}_1$.

  In this case, by induction, we have $\Gamma_1 \vdash_{\mathtt{FJ}} [\bar{\mathtt{x}} \mapsto \bar{\mathtt{t}}_u]\mathtt{t}_1 : \mathtt{C}'_1$ and $\mathtt{C}'_1 <: \mathtt{C}_1$. For the term $\mathtt{t}$, after substitution, we have $\Gamma_1 \vdash_{\mathtt{FJ}} (\mathtt{C})\mathtt{t}_1 : \mathtt{C}$ and $\mathtt{C} <: \mathtt{C}$. Then we have the case proved.

- case-3: $\mathtt{t} = \mathtt{t}_1.\mathtt{f}$ and $\Gamma_1, \{\bar{\mathtt{x}} : \bar{\mathtt{C}}_x\} \vdash_{\mathtt{FJ}} \mathtt{t}_1 : \mathtt{C}_1$.

  By induction, we have $\Gamma_1 \vdash_{\mathtt{FJ}} [\bar{\mathtt{x}} \mapsto \bar{\mathtt{t}}_u]\mathtt{t}_1 : \mathtt{C}'_1$ and $\mathtt{C}'_1 <: \mathtt{C}_1$. Then for field access, it will still access the same field $\mathtt{f}$ as it did before. Thus we have $\Gamma_1 \vdash_{\mathtt{FJ}} [\bar{\mathtt{x}} \mapsto \bar{\mathtt{t}}_u]\mathtt{t}_1.\mathtt{f} : \mathtt{C}_t$ and $\mathtt{C}_t <: \mathtt{C}_t$, and the case is proved.

- case-4: $\mathtt{t} = \mathtt{new}\ \mathtt{C_0}(\bar{\mathtt{t}}_1)$ and $\Gamma_1, \{\bar{\mathtt{x}} : \bar{\mathtt{C}}_\mathtt{x}\} \vdash_{\mathtt{FJ}} \bar{\mathtt{t}}_1 : \bar{\mathtt{C}}_1$.

  The substitution $[\bar{\mathtt{x}} \mapsto \bar{\mathtt{t}}_\mathtt{u}]\mathtt{t}$ equals to $\mathtt{new}\ \mathtt{C_0}([\bar{\mathtt{x}} \mapsto \bar{\mathtt{t}}_\mathtt{u}]\bar{\mathtt{t}}_1)$.

  As the term $\mathtt{t}$ is well typed, we have:

$$\frac{\mathtt{fields}(\mathtt{C_0}) = \bar{\mathtt{D}}\ \bar{\mathtt{f}} \quad \Gamma \vdash_{\mathtt{FJ}} \bar{\mathtt{t}}_1 : \bar{\mathtt{C}}_1 \quad \bar{\mathtt{C}}_1 <: \bar{\mathtt{D}}}{\Gamma \vdash_{\mathtt{FJ}} \mathtt{new}\ \mathtt{C}(\bar{\mathtt{t}}_1) : \mathtt{C_0}}$$

  By induction, we have $\Gamma_1 \vdash_{\mathtt{FJ}} [\bar{\mathtt{x}} \mapsto \bar{\mathtt{t}}_\mathtt{u}]\bar{\mathtt{t}}_1 : \bar{\mathtt{C}}'_1$ and $\bar{\mathtt{C}}'_1 <: \bar{\mathtt{C}}_1$, then we have the following derivation:

$$\frac{\mathtt{fields}(\mathtt{C_0}) = \bar{\mathtt{D}}\ \bar{\mathtt{f}} \quad \Gamma \vdash_{\mathtt{FJ}} [\bar{\mathtt{x}} \mapsto \bar{\mathtt{t}}_\mathtt{u}]\bar{\mathtt{t}}_1 : \bar{\mathtt{C}}'_1 \quad \bar{\mathtt{C}}'_1 <: \bar{\mathtt{C}}_1 <: \bar{\mathtt{D}}}{\Gamma \vdash_{\mathtt{FJ}} \mathtt{new}\ \mathtt{C}([\bar{\mathtt{x}} \mapsto \bar{\mathtt{t}}_\mathtt{u}]\bar{\mathtt{t}}_1) : \mathtt{C_0}}$$

  Thus we still have $\Gamma_1 \vdash_{\mathtt{FJ}} \mathtt{new}\ \mathtt{C}([\bar{\mathtt{x}} \mapsto \bar{\mathtt{t}}_\mathtt{u}]\bar{\mathtt{t}}_1) : \mathtt{C_0}$ and $\mathtt{C_0} <: \mathtt{C_0}$. This case is proved.

- case-5: $\mathtt{t} = \mathtt{t_0}.\mathtt{m}(\bar{\mathtt{t}}_1)$ and $\Gamma_1, \{\bar{\mathtt{x}} : \bar{\mathtt{C}}_\mathtt{x}\} \vdash_{\mathtt{FJ}} \bar{\mathtt{t}}_1 : \bar{\mathtt{C}}_1, \mathtt{t_0} : \mathtt{C_0}$.

  In this case, the substitution $[\bar{\mathtt{x}} \mapsto \bar{\mathtt{t}}_\mathtt{u}]\mathtt{t}$ equals to $([\bar{\mathtt{x}} \mapsto \bar{\mathtt{t}}_\mathtt{u}]\mathtt{t_0}).\mathtt{m}([\bar{\mathtt{x}} \mapsto \bar{\mathtt{t}}_\mathtt{u}]\bar{\mathtt{t}}_1)$.

  As the term $\mathtt{t}$ is well typed in FJ type system, we have:

$$\frac{\Gamma \vdash_{\mathtt{FJ}} \mathtt{t_0} : \mathtt{C_0} \quad \mathsf{mtype}(\mathtt{m}, \mathtt{C_0}) = \bar{\mathtt{D}} \to \mathtt{C} \quad \Gamma \vdash_{\mathtt{FJ}} \bar{\mathtt{t}}_1 : \bar{\mathtt{C}}_1 \quad \bar{\mathtt{C}}_1 <: \bar{\mathtt{D}}}{\Gamma \vdash_{\mathtt{FJ}} \mathtt{t_0}.\mathtt{m}(\bar{\mathtt{t}}_1) : \mathtt{C}}$$

  By induction, we have $\Gamma_1 \vdash_{\mathtt{FJ}} [\bar{\mathtt{x}} \mapsto \bar{\mathtt{t}}_\mathtt{u}]\bar{\mathtt{t}}_1 : \mathtt{C}'_1, \mathtt{t_0} : \mathtt{C}'_0$ and $\mathtt{C}'_0 <: \mathtt{C_0}, \bar{\mathtt{C}}'_1 <: \bar{\mathtt{C}}_1$.

  For the condition that $\mathtt{C}'_0 <: \mathtt{C_0}$, we have $\mathsf{mtype}(\mathtt{m}, \mathtt{C}'_0) = \mathsf{mtypem}(\mathtt{m}, \mathtt{C_0})$ according to the rule METHOD-LOOKUP2.

  With these conditions, we have the following derivation for the new term after substitution:

$$\frac{\Gamma_1 \vdash_{\mathtt{FJ}} [\bar{\mathtt{x}} \mapsto \bar{\mathtt{t}}_\mathtt{u}]\mathtt{t_0} : \mathtt{C}'_0 \quad \mathsf{mtype}(\mathtt{m}, \mathtt{C}'_0) = \bar{\mathtt{D}} \to \mathtt{C} \quad \Gamma_1 \vdash_{\mathtt{FJ}} [\bar{\mathtt{x}} \mapsto \bar{\mathtt{t}}_\mathtt{u}]\bar{\mathtt{t}}_1 : \bar{\mathtt{C}}'_\mathtt{t} \quad \bar{\mathtt{C}}'_1 <: \bar{\mathtt{C}}_1 <: \bar{\mathtt{D}}}{\Gamma_1 \vdash_{\mathtt{FJ}} [\bar{\mathtt{x}} \mapsto \bar{\mathtt{t}}_\mathtt{u}]\mathtt{t_0}.\mathtt{m}([\bar{\mathtt{x}} \mapsto \bar{\mathtt{t}}_\mathtt{u}]\bar{\mathtt{t}}_1) : \mathtt{C}}$$

  Thus we prove the case that: $\mathtt{Gamma}_1 \vdash_{\mathtt{FJ}} [\bar{\mathtt{x}} \mapsto \bar{\mathtt{t}}_\mathtt{u}]\mathtt{t} : \mathtt{C}$ and $\mathtt{C} <: \mathtt{C}$.

With these five cases proved, we have the lemma proved. □

Lemma 4 (Term Formation)**.** *Given a well-typed SWIN program* Π*, if a term* $\mathtt{t}$ *in the original typing context can be typed to* $\mathtt{C}$*, then after transformation by* Π*, the term is well-typed and its type is a subtype of* $\Pi(\mathtt{C})$*. i.e.*

$$\Gamma_\mathtt{s} \vdash^{\mathtt{API_s}}_{\mathtt{FJ}} \mathtt{t} : \mathtt{C} \Longrightarrow \Gamma_\mathtt{d} \vdash^{\mathtt{API_d}}_{\mathtt{FJ}} \Pi(\mathtt{t}) : \mathtt{C}',\ \mathtt{where}\ \mathtt{C}' <: \Pi(\mathtt{C})$$

Proof. By induction on a derivation of a term $\mathtt{t}$. At each step of the induction, we assume that the desired property holds for all sub-derivations. We give our proof based on the last step of the derivation, which can only be one of the following five cases:

Before we move on to the cases analysis, we should note that according to Lemma 1, we have the relationship that $\Gamma_\mathtt{d} = \Pi(\Gamma_\mathtt{s})$.

- case-1: $\mathtt{t} = \mathtt{x}$ and $\Gamma_\mathtt{s} \vdash^{\mathtt{API_s}}_{\mathtt{FJ}} \mathtt{x} : \mathtt{C}$.

  In this case, we have that $\Gamma_\mathtt{d} \vdash^{\mathtt{API_d}}_{\mathtt{FJ}} \mathtt{x} : \Pi(\mathtt{C})$ according to Lemma 1. Then we have $\Gamma_\mathtt{d} \vdash^{\mathtt{API_d}}_{\mathtt{FJ}} \mathtt{t} : \Pi(\mathtt{C})$ and $\Pi(\mathtt{C}) <: \Pi(\mathtt{C})$, and the case is proved.

- case-2: $\mathtt{t} = (\mathtt{C})\mathtt{t}_1$ and $\Gamma_\mathtt{s} \vdash^{\mathtt{API_s}}_{\mathtt{FJ}} \mathtt{t}_1 : \mathtt{C}_1$.

  According to the rule E-T-CAST, $\Pi(\mathtt{t}) = \Pi(\mathtt{C})\ \Pi(\mathtt{t}_1)$.

  By induction, we have $\Gamma_\mathtt{d} \vdash^{\mathtt{API_d}}_{\mathtt{FJ}} \mathtt{t}_1 : \mathtt{C}'_1$ and $\mathtt{C}'_1 <: \Pi(\mathtt{C}_1)$.

  By the rule FJ-*CAST (represent one of the three cast rules), we have $\Gamma_\mathtt{d} \vdash^{\mathtt{API_d}}_{\mathtt{FJ}} (\Pi(\mathtt{C}))\ \Pi(\mathtt{t}_1) : \Pi(\mathtt{C})$, and $\Pi(\mathtt{C}) <: \Pi(\mathtt{C})$. Thus the case is proved.

- case-3: $\mathtt{t} = \mathtt{t}_1.\mathtt{f}$ and $\Gamma_\mathtt{s} \vdash^{\mathtt{API_s}}_{\mathtt{FJ}} \mathtt{t}_1 : \mathtt{C}_1$. In this case, there are further two subcases:
  - subcase-1: $\mathtt{C}_1$ is declared in $\mathtt{API_s}$, i.e. $\exists\, \mathtt{CL} \in \mathtt{API_s}$ and $\mathbf{Decl}(\mathtt{CL}) = \mathtt{C}_1$. Then we have $\Pi(\mathtt{C}_1)$ is declared in $\mathtt{API_d}$
    According to the checking rule **FieldCover**, we have a transformation rule
    $$\pi = (\mathtt{x} : \mathtt{D}_1 \hookrightarrow \mathtt{D}'_1)[\mathtt{x.f} : \mathtt{C} \hookrightarrow \mathtt{r} : \mathtt{C}'] \in \Pi$$
    to transform the field access expressions. i.e. $\nexists(\mathtt{x} : \mathtt{E}_1 \hookrightarrow \mathtt{E}'_1)[\mathtt{x.f} : \mathtt{C} \hookrightarrow \mathtt{r} : \mathtt{C}'] \in \Pi.(\mathtt{C}_1 <: \mathtt{E}_1 <: \mathtt{D}_1 \wedge \mathtt{E}_1 \neq \mathtt{D}_1)$
    By the evaluation rule E-T-FIELD, we have:
    $$\frac{(\mathtt{x} : \mathtt{D}_1 \hookrightarrow \mathtt{D}'_1)[\mathtt{x.f} : \mathtt{C} \hookrightarrow \mathtt{r} : \mathtt{C}'] \in \Pi \quad \mathtt{C}_1 <: \mathtt{D}_1 \quad \nexists(\mathtt{x} : \mathtt{E}_1 \hookrightarrow \mathtt{E}'_1)[\mathtt{x.f} : \mathtt{C} \hookrightarrow \mathtt{r} : \mathtt{C}'] \in \Pi.(\mathtt{C}_1 <: \mathtt{E}_1 <: \mathtt{D}_1 \wedge \mathtt{E}_1 \neq \mathtt{D}_1)}{\Pi(\mathtt{t}_1.\mathtt{f}) = [\mathtt{x} \mapsto \Pi(\mathtt{t}_1)]\mathtt{r}}$$
    According to Lemma 3, we have $\Gamma_\mathtt{d} \vdash^{\mathtt{API_d}}_{\mathtt{FJ}} [\mathtt{x} \mapsto \Pi(\mathtt{t}_1)]\mathtt{r} : \mathtt{C}''$ and $\mathtt{C}'' <: \mathtt{C}'$.
    And by the definition of **TypeMapping**, $\mathtt{C} \hookrightarrow \mathtt{C}'$ is collected in $\mathbf{TypeMapping}(\Pi)$, which indicates $\Pi(\mathtt{C}) = \mathtt{C}'$.
    With these properties, we have $\Gamma_\mathtt{d} \vdash^{\mathtt{API_d}}_{\mathtt{FJ}} \Gamma_\mathtt{d} \vdash^{\mathtt{API_d}}_{\mathtt{FJ}} [\mathtt{x} \mapsto \mathtt{t}_1]\mathtt{r} : \mathtt{C}''$ and $\mathtt{C}'' <: \mathtt{C}' = \Pi(\mathtt{C})$. And the subcase is proved.
  - subcase-2: $\mathtt{C}_1$ is defined in client class declarations.
    In this subcase, the rule will be evaluated by the rule E-ALTER-FIELD. By induction, we have $\Gamma \vdash^{\mathtt{API_d}}_{\mathtt{FJ}} \Pi(\mathtt{t}_1) : \mathtt{C}'_1$ and $\mathtt{C}'_1 <: \Pi(\mathtt{C}_1)$. Then according to the rule FJ-FIELD and the auxiliary function FIELD-LOOKUP and the evaluation rule E-DECLARATION, we have the following derivation tree:
    $$\frac{\Gamma \vdash^{\mathtt{API_d}}_{\mathtt{FJ}} \Pi(\mathtt{t}_1) : \mathtt{C}'_1 \quad \frac{\mathtt{C}'_1 <: \Pi(\mathtt{C}_1) \quad \Pi(\mathtt{C})\ \mathtt{f} \in \mathsf{field}(\Pi(\mathtt{C}_1))}{\Pi(\mathtt{C})\ \mathtt{f} \in \mathsf{field}(\mathtt{C}'_1)}}{\Gamma \vdash^{\mathtt{API_d}}_{\mathtt{FJ}} \Pi(\mathtt{t}_1).\mathtt{f} : \Pi(\mathtt{C})}$$
    And of course, $\Pi(\mathtt{C}) <: \Pi(\mathtt{C})$, thus we have the subcase proved.

  With these two subcases proved, the case for field access is proved.

- case-4: $\mathtt{t} = \mathtt{new}\ \mathtt{C}(\bar{\mathtt{t}}_1)$ and $\Gamma \vdash^{\mathtt{API_s}}_{\mathtt{FJ}} \bar{\mathtt{t}}_1 : \bar{\mathtt{C}}_1$.
  In this case, we still have two subcases to discuss.
  - subcase-1: The class $\mathtt{C}$ is declared in $\mathtt{API_s}$, i.e. $\exists\, \mathtt{CL} \in \mathtt{API_s}.(\mathbf{Decl}(\mathtt{CL}) = \mathtt{C}_0)$, then $\Pi(\mathtt{C}_0)$ is declared in $\mathtt{API_d}$.
    By the checking rule **ConstrCover**, there exists the following transformation rule to transform this term:
    $$\pi = (\bar{\mathtt{x}} : \overline{\mathtt{C}_2 \hookrightarrow \mathtt{C}'_2})[\mathtt{new}\ \mathtt{C}(\bar{\mathtt{x}}) : \mathtt{C} \hookrightarrow \mathtt{r} : \mathtt{C}']$$
    As the term is well typed, we have $\mathbf{Type}(\bar{\mathtt{t}}_1) <: \bar{\mathtt{C}}_2$
    Then according to the rule E-T-NEW, we have:
    $$\frac{(\bar{\mathtt{x}} : \overline{\mathtt{C}_2 \hookrightarrow \mathtt{C}'_2})[\mathtt{new}\ \mathtt{C}(\bar{\mathtt{x}}) : \mathtt{C} \hookrightarrow \mathtt{r} : \mathtt{C}'] \in \Pi \quad \mathbf{Type}(\bar{\mathtt{t}}_1) <: \bar{\mathtt{C}}_2}{\Pi(\mathtt{new}\ \mathtt{C}(\bar{\mathtt{t}}_1)) = [\bar{\mathtt{x}} \mapsto \overline{\Pi(\mathtt{t}_1)}]\mathtt{r}}$$
    By Lemma 3, we have $\Gamma_\mathtt{d} \vdash^{\mathtt{API_d}}_{\mathtt{FJ}} [\bar{\mathtt{x}} \mapsto \overline{\Pi(\mathtt{t})_1}]\mathtt{r} : \mathtt{C}''$ and $\mathtt{C}'' <: \mathtt{C}'$. And according to the definition of **TypeMapping** and the checking rule **Subtyping**, we have $\Pi(\mathtt{C}) = \mathtt{C}'$.
    Then we have $\Gamma_\mathtt{d} \vdash^{\mathtt{API_d}}_{\mathtt{FJ}} \Pi(\mathtt{t}) : \mathtt{C}''$ and $\mathtt{C}'' <: \Pi(\mathtt{C})$, and the subcase is proved.

– subcase-2: The class $\mathtt{C}$ is declared in client code. Then the transformation of the term $\mathtt{t}$ should be evaluated according to the rule E-ALTER-NEW, thus we have $\Pi(\mathtt{t}) = \mathtt{new}\ \Pi(\mathtt{C})(\overline{\Pi(\mathtt{t}_1)})$.

To finish the proof of the subcase, we need the following facts:

* According to the rule E-DECLARATION, the class definition is transformed to $\Pi(\mathtt{CL}) = \mathtt{class}\ \Pi(\mathtt{C})\ \mathtt{extends}\ \Pi(\mathtt{D})\ \{\overline{\Pi(\mathtt{C}_2}\ \mathtt{f};\ \Pi(\mathtt{K})\ \overline{\Pi(\mathtt{M})}\}$. Thus $\mathsf{fields}(\Pi(\mathtt{C})) = \overline{\Pi(\mathtt{C}_2)}$
* By induction, we have $\Gamma_\mathtt{d} \vdash^{\mathtt{API_d}}_{\mathtt{FJ}} \overline{\Pi(\mathtt{t}_1)} : \bar{\mathtt{C}}'_1$ and $\bar{\mathtt{C}}'_1 <: \overline{\Pi(\mathtt{C}_1)}$.
* By the typing rule FJ-NEW, we have $\bar{\mathtt{C}}_1 <: \bar{\mathtt{C}}_2$. And by Lemma 2, we have $\overline{\Pi(\mathtt{C}_1)} <: \overline{\Pi(\mathtt{C}_2)}$.

With these facts, we have the following judgment:

$$\frac{\mathsf{fields}(\Pi(\mathtt{C})) = \overline{\Pi(\mathtt{C}_2)} \quad \Gamma_\mathtt{d} \vdash^{\mathtt{API_d}}_{\mathtt{FJ}} \overline{\Pi(\mathtt{t}_1)} : \bar{\mathtt{C}}'_1 \quad \bar{\mathtt{C}}'_1 <: \overline{\Pi(\mathtt{C}_1)} <: \overline{\Pi(\mathtt{C}_2)}}{\Gamma_\mathtt{d} \vdash^{\mathtt{API_d}}_{\mathtt{FJ}} \mathtt{new}\ \Pi(\mathtt{C})(\overline{\Pi(\mathtt{t}_1)}) : \Pi(\mathtt{C})}$$

And this proved the subcase.

With these two subcases proved, the case for object creation is proved.

- case-5: $\mathtt{t} = \mathtt{t}_1.\mathtt{m}(\bar{\mathtt{t}}_2)$ and $\Gamma_\mathtt{s} \vdash^{\mathtt{API_s}}_{\mathtt{FJ}} \mathtt{t}_1 : \mathtt{C}_1, \bar{\mathtt{t}}_2 : \bar{\mathtt{C}}_2$.

  Similar to the previous one, there are also two subcases for this case.

  – subcase-1: the class $\mathtt{C}_1$ is declared in $\mathtt{API_s}$. i.e. $\exists\ \mathtt{CL} \in \mathtt{API_s}.(\mathbf{Decl}(\mathtt{CL}) = \mathtt{C}_1)$.
  In this subcase, the term $\mathtt{t}_1.\mathtt{m}(\bar{\mathtt{t}}_2)$ will be transformed by a rule according to E-T-INVOKE as the checking rule **MethCover** guarantees that there is a rule in Π to transform the method (At least a rule exists to transform the method $\mathtt{m}$ declared in a parent class of $\mathtt{C}$, according to Lemma 6).
  Suppose the transformation rule is the following one (And this one is the closest rule to transform):

$$(\bar{\mathtt{y}} : \overline{\mathtt{C}_3 \hookrightarrow \mathtt{C}'_3}, \mathtt{x} : \mathtt{C}_4 \hookrightarrow \mathtt{C}'_4)[\mathtt{x.m}(\bar{\mathtt{y}}) : \mathtt{C} \to \mathtt{r} : \mathtt{D}]$$

  Then by the rule E-T-INVOKE, the transformation will be :

$$\frac{\begin{array}{c}(\bar{\mathtt{y}} : \overline{\mathtt{C}_3 \hookrightarrow \mathtt{C}'_3}, \mathtt{x} : \mathtt{C}_4 \hookrightarrow \mathtt{C}'_4)[\mathtt{x.m}(\bar{\mathtt{y}}) : \mathtt{C} \to \mathtt{r} : \mathtt{D}] \in \Pi \quad \mathbf{Type}(\mathtt{t}_1) <: \mathtt{C}_4 \quad \mathbf{Type}(\bar{\mathtt{t}}_2) <: \bar{\mathtt{C}}_3 \\ \nexists(\bar{\mathtt{y}} : \overline{\mathtt{C}_3 \hookrightarrow \mathtt{C}'_3}, \mathtt{x} : \mathtt{C}_5 \hookrightarrow \mathtt{C}'_5)[\mathtt{x.m}(\bar{\mathtt{y}}) : \mathtt{C} \to \mathtt{r} : \mathtt{D}] \in \Pi.(\mathbf{Type}(\mathtt{t}_1) <: \mathtt{C}_5 <: \mathtt{C}_4 \wedge \mathtt{C}_5 \neq \mathtt{C}_4)\end{array}}{\Pi(\mathtt{t}_1.\mathtt{m}(\bar{\mathtt{t}}_2)) = [\mathtt{x} \mapsto \Pi(\mathtt{t}_1), \bar{\mathtt{y}} \mapsto \overline{\Pi(\mathtt{t}_2)}]\mathtt{r}}$$

  By Lemma 3, we have $\Gamma_\mathtt{d} \vdash^{\mathtt{API_d}}_{\mathtt{FJ}} [\mathtt{x} \mapsto \Pi(\mathtt{t}_1), \bar{\mathtt{y}} \mapsto \overline{\Pi(\mathtt{t}_2)}]\mathtt{r} : \mathtt{C}'$ and $\mathtt{C}' <: \mathtt{D}$. Also, by the definition of **TypeMapping**, we have $\Pi(\mathtt{C}) = \mathtt{D}$. Thus we have $\Gamma \vdash^{\mathtt{API_d}}_{\mathtt{FJ}} \mathtt{t}_1.\mathtt{m}(\bar{\mathtt{t}}_2) : \mathtt{C}'$ and $\mathtt{C}' <: \Pi(\mathtt{C})$. And this subcase is proved.

  – subcase-2: the class $\mathtt{C}_1$ is declared in client code. And in this case, the term $\mathtt{t}$ will be transformed by the rule E-ALTER-INVOKE:

$$\Pi(\mathtt{t}_1.\mathtt{m}(\bar{\mathtt{t}}_2)) = \Pi(\mathtt{t}_1).\mathtt{m}(\overline{\Pi(\mathtt{t}_2)})$$

  To finish the proof of this case, we need the following points:

  * By induction, we have $\Gamma_\mathtt{d} \vdash^{\mathtt{API_d}}_{\mathtt{FJ}} \Pi(\mathtt{t}_1) : \mathtt{C}'_1$ and $\mathtt{C}'_1 <: \Pi(\mathtt{C}_1)$, $\Gamma_\mathtt{d} \vdash^{\mathtt{API_d}}_{\mathtt{FJ}} \overline{\Pi(\mathtt{t}_2)} : \bar{\mathtt{C}}'_2$ and $\bar{\mathtt{C}}'_2 <: \overline{\Pi(\mathtt{C}_2)}$.
  * According to the well-typedness of the original term in $\mathtt{API_s}$, we have the following derivation:

$$\frac{\Gamma_\mathtt{s} \vdash^{\mathtt{API_s}}_{\mathtt{FJ}} \mathtt{t}_1 : \mathtt{C}_1 \quad \mathsf{mtype}(\mathtt{m}, \mathtt{C}_1) = \bar{\mathtt{C}}_\mathtt{u} \to \mathtt{C} \quad \Gamma_\mathtt{s} \vdash^{\mathtt{API_s}}_{\mathtt{FJ}} \bar{\mathtt{t}}_2 : \bar{\mathtt{C}}_2 \quad \bar{\mathtt{C}}_2 <: \bar{\mathtt{C}}_\mathtt{u}}{\Gamma_\mathtt{s} \vdash^{\mathtt{API_s}}_{\mathtt{FJ}} \mathtt{t}_1.\mathtt{m}(\bar{\mathtt{t}}_2) : \mathtt{C}}$$

* According to the rule E-DECLARATION and E-METHOD, $\mathsf{mtype}(\mathtt{m}, \Pi(\mathtt{C}_1)) = \overline{\Pi(\mathtt{C}_u)} \to \Pi(\mathtt{C})$. By the definition of mtype, we have $\mathsf{mtype}(\mathtt{m}, \mathtt{C}'_1) = \mathsf{mtype}(\mathtt{m}, \Pi(\mathtt{C}_1))$.
* By the checking rule **Subtyping**, and the fact $\bar{\mathtt{C}}_2 <: \bar{\mathtt{C}}_u$ we have $\overline{\Pi(\mathtt{C}_2)} <: \overline{\Pi(\mathtt{C}_u)}$.

With these facts, we can derive the type of the transformed term using the following tree:

$$\frac{\Gamma_d \vdash^{\mathtt{API}_d}_{\mathtt{FJ}} \Pi(\mathtt{t}_1) : \mathtt{C}'_1 \quad \mathsf{mtype}(\mathtt{m}, \mathtt{C}'_1) = \mathsf{mtype}(\mathtt{m}, \Pi(\mathtt{C}_1)) = \overline{\Pi(\mathtt{C}_u)} \to \Pi(\mathtt{C}) \quad \Gamma_d \vdash^{\mathtt{API}_d}_{\mathtt{FJ}} \overline{\Pi(\mathtt{t}_2)} : \bar{\mathtt{C}}'_2 \quad \bar{\mathtt{C}}'_2 <: \overline{\Pi(\mathtt{C}_2)} <: \overline{\Pi(\mathtt{C}_u)}}{\Gamma_d \vdash^{\mathtt{API}_d}_{\mathtt{FJ}} \Pi(\mathtt{t}_1).\mathtt{m}(\overline{\Pi(\mathtt{t}_2)}) : \Pi(\mathtt{C})}$$

And thus we have $\Gamma_d \vdash^{\mathtt{API}_d}_{\mathtt{FJ}} \Pi(\mathtt{t}) : \Pi(\mathtt{C})$ in this subcase.

With these two subcases proved, we have the lemma holds for the case 5.

With these five cases for a term proved, by induction, the lemma holds for any FJ term. □

Lemma 5 (Method Formation)**.** *An FJ method declaration is well formed after transformed by a well-typed SWIN program. i.e. For any* M*,*

$$\Pi(\mathtt{M}) = \Pi(\mathtt{C}_1)\ \mathtt{m}(\Pi(\bar{\mathtt{C}}_m)\ \bar{\mathtt{x}})\ \{\mathtt{return}\ \Pi(\mathtt{t});\}$$

is well-formed with new API if Π *is well typed.*

Proof. According to Lemma 4, we have $\{\bar{\mathtt{x}} : \overline{\Pi(\mathtt{C}_m)}, \mathtt{this} : \Pi(\mathtt{C}_1)\} \vdash^{\mathtt{API}_d}_{\mathtt{FJ}} \mathtt{t} : \mathtt{C}'_1$ and $\mathtt{C}'_1 <: \Pi(\mathtt{C}_1)$.

Suppose $\mathtt{CT}(\mathtt{C}) = \mathtt{class}\ \mathtt{C}\ \mathtt{extends}\ \mathtt{D}\ \{...\}$, according to the rule E-METHOD and E-DECLRATION, $\mathsf{override}(\mathtt{m}, \Pi(\mathtt{D}), \overline{\Pi(\mathtt{C})} \to \mathtt{C}_1)$.

Then the formation of the transformed term is proved by the FJ-M-OK derivation on these judgments:

$$\frac{\{\bar{\mathtt{x}} : \overline{\Pi(\mathtt{C}_m)}, \mathtt{this} : \Pi(\mathtt{C}_1)\} \vdash^{\mathtt{API}_d}_{\mathtt{FJ}} \mathtt{t} : \mathtt{C}'_1 \qquad \mathtt{C}'_1 <: \Pi(\mathtt{C}_1) \quad \mathtt{CT}(\Pi(\mathtt{C})) = \mathtt{class}\ \Pi(\mathtt{C}_1)\ \mathtt{extends}\ \Pi(\mathtt{D})\ \{...\} \quad \mathsf{override}(\mathtt{m}, \Pi(\mathtt{D}), \overline{\Pi(\mathtt{C})} \to \mathtt{C}_1)}{\Pi(\mathtt{C}_1)\ \mathtt{m}\ (\bar{\mathtt{C}}\ \bar{\mathtt{x}})\ \{\mathtt{return}\ \mathtt{t}_0;\}\ \mathtt{OK\ in}\ \Pi(\mathtt{C})}\ \text{(FJ-M-OK)}$$

□

Theorem 1 (Type-Safety)**.** *Any FJ program is well-typed after a transformation by a well-typed SWIN program* Π*. i.e. For any* CL*,*

$$\Pi(\mathtt{CL}) = \mathtt{class}\ \Pi(\mathtt{C}_1)\ \mathtt{extends}\ \Pi(\mathtt{C}_2)\ \{\ \Pi(\bar{\mathtt{C}}_i)\ \bar{\mathtt{f}}_i;\ \Pi(\mathtt{K})\ \overline{\Pi(\mathtt{M})}\ \}$$

is well-typed with new API if Π *is well-typed.*

Proof. By Lemma 5, we have all method declarations well formed. And by the rule E-CONSTRUCTOR, we have the following derivation:

$$\frac{\Pi(\mathtt{K}) = \Pi(\mathtt{C})\ (\overline{\Pi(\mathtt{C})}\ \bar{\mathtt{f}})\{...\} \quad \mathsf{fields}(\Pi(\mathtt{D})) = \overline{\Pi(\mathtt{D})}\ \bar{\mathtt{g}} \quad \overline{\Pi(\mathtt{M})}\ ok}{\mathtt{class}\ \Pi(\mathtt{C})\ \mathtt{extends}\ \Pi(\mathtt{D})\ \{\overline{\Pi(\mathtt{C})}\ \bar{\mathtt{f}}; \Pi(\mathtt{K})\ \overline{\Pi(\mathtt{M})}\}\ ok}$$

With this proved, we have the theorem proved, i.e a well-typed SWIN program can transform any FJ program correctly. □

Lemma 6 (Confluent)**.** *For each API (*$\mathtt{API}_s$*) declared class* C*, there is exactly one transformation rule in a well typed SWIN program* Π *to transform an expression of field access/new object/method invocation related to the class* C*, if all left hand side patterns are different in the SWIN program.*

Proof. The proof consists of two parts: existence and uniqueness. And we make the proof by discussing different kinds of form of a term $\mathtt{t}$ related to $\mathtt{C}$.

- $\mathtt{t} = \mathtt{new}\ \mathtt{C}(\bar{\mathtt{t}}_1)$. According to the checking rule **ConstrCover**. There exists a rule $\pi = (\bar{\mathtt{x}} : \overline{\mathtt{C}_1 \hookrightarrow \mathtt{C}'_1})[\mathtt{new}\ \mathtt{C}(\bar{\mathtt{x}}) : \mathtt{C} \to \mathtt{r} : \mathtt{C}']$ in Π to match the term. By the checking rule T-L2 in for π, if there is another rule to match the term $\mathtt{t}$, it is supposed to be $\pi' = (\bar{\mathtt{x}} : \overline{\mathtt{C}_1} \hookrightarrow \mathtt{C}'_1)[\mathtt{l} : \mathtt{C} \hookrightarrow \mathtt{r}'' : \mathtt{C}'']$, which contradicts the fact in our definition that only one same left hand pattern will appear in the rule set Π. Thus there will be only one rule to match the term.

  Then according to the rule E-T-NEW, this transformation rule π can transform the term $\mathtt{t}$.

- $\mathtt{t} = \mathtt{t}_1.\mathtt{f}$, suppose $\Gamma \vdash^{\mathtt{API}_\mathtt{s}}_{\mathtt{FJ}} \mathtt{t}_1 : \mathtt{C}_1$ and the field $\mathtt{f}$ is declared in a super class of $\mathtt{C}$ named $\mathtt{D}$. (As the class order is linear and field declaration cannot be override, it is the only place to declare the field $\mathtt{f}$)

  Suppose there are two rules can match the term $\mathtt{t}$:

  $$\pi_1 = (\mathtt{x} : \mathtt{C}_2 \hookrightarrow \mathtt{C}'_2)[\ \mathtt{x.f} : \mathtt{C}\ \to\ \mathtt{r} : \mathtt{D}\ ]$$
  $$\pi_2 = (\mathtt{x} : \mathtt{C}_3 \hookrightarrow \mathtt{C}'_3)[\ \mathtt{x.f} : \mathtt{C}\ \to\ \mathtt{r} : \mathtt{D}\ ]$$

  As these two patterns can be used to match the term $\mathtt{t}$, then we have $\mathtt{C}_1 <: \mathtt{C}_2$ and $\mathtt{C}_1 <: \mathtt{C}_3$. And by FJ type system, $\mathtt{C}_2$ and $\mathtt{C}_3$ should have subtyping relation. Suppose $\mathtt{C}_2 <: \mathtt{C}_3$, then by the rule E-T-FIELD, we have $\mathtt{C}_1 <: \mathtt{C}_2 <: \mathtt{C}_3$, and the rule itself guarantees that the first rule π_1 will be used to transform the term.

- $\mathtt{t} = \mathtt{t}_1.\mathtt{m}(\bar{\mathtt{t}}_2)$, and $\Gamma_\mathtt{s} \vdash^{\mathtt{API}_\mathtt{s}}_{\mathtt{FJ}} \mathtt{t}_1 : \mathtt{C}_1, \bar{\mathtt{t}}_2 : \bar{\mathtt{C}}_2$.

  There are two subcases for the case:

  - $\mathtt{m}$ is define (or override) in class $\mathtt{C}_1$. According to the rule **MethCover**, we have a rule π to match the term. And according to the rule E-T-INVOKE, there is no closer class to transform the term and argument types are fixed as the case $\mathtt{t} = \mathtt{new}\ \mathtt{C}(\bar{\mathtt{t}}_1)$. Thus the rule π will be used to transform the rule, by the following derivation:

    $$\frac{(\bar{\mathtt{y}} : \overline{\mathtt{C}_3 \hookrightarrow \mathtt{C}'_3}, \mathtt{x} : \mathtt{C}_1 \hookrightarrow \mathtt{C}'_1)[\mathtt{x.m}(\bar{\mathtt{y}}) : \mathtt{C} \to \mathtt{r} : \mathtt{D}] \in \Pi \quad \mathbf{Type}(\mathtt{t}_1) = \mathtt{C}_1 <: \mathtt{C}_1 \quad \mathbf{Type}(\bar{\mathtt{t}}_2) <: \bar{\mathtt{C}}_3 \quad \nexists(\bar{\mathtt{y}} : \overline{\mathtt{C}_3 \hookrightarrow \mathtt{C}'_3}, \mathtt{x} : \mathtt{C}_5 \hookrightarrow \mathtt{C}'_5)[\mathtt{x.m}(\bar{\mathtt{y}}) : \mathtt{C} \to \mathtt{r} : \mathtt{D}] \in \Pi.(\mathtt{C}_1 <: \mathtt{C}_5 <: \mathtt{C}_1 \wedge \mathtt{C}_5 \neq \mathtt{C}_1)}{\Pi(\mathtt{t}_1.\mathtt{m}(\bar{\mathtt{t}}_2)) = [\mathtt{x} \mapsto \Pi(\mathtt{t}_1), \bar{\mathtt{y}} \mapsto \overline{\Pi(\mathtt{t}_2)}]\mathtt{r}}$$

  - $\mathtt{m}$ is not defined in $\mathtt{C}_1$, but in a class $\mathtt{C}_3$, which is a super class of $\mathtt{C}_1$. Then a transformation rule π' exists to transform the method invocation of $\mathtt{C}_3$. Suppose there is no other rule to match the term $\mathtt{t}$, then π' will be used to transform the term, and if there exists another rule π'' to transform the term, then π'' will be used to transform $\mathtt{t}$. As the subtyping relation is finite in FJ, thus there exists a lower bound π_1, and by E-T-INVOKE, it will be used to transform the term.

  With these three cases proved, the lemma is proved.

□

Theorem 2. *Any SWIN program is convergent.*

Proof. The proof includes two parts: termination and confluent.

- Termination: SWIN employs a normal order evaluation semantics. First, the evaluation rules visit a term leftmost and outermost. After performing the transformation on that term, the evaluation rules recursively visit the sub terms of the term, and for each visit, the transformation will be applied on the original sub terms, and produce the transformation result by combining the transformed sub terms. In this way we can ensure each recursive visit will be terminated as the length of the sub terms are always shorter than the term.

- Confluent: By Lemma 6.

With these two cases proved, any well typed SWIN program is convergent. □