Type-Preserving Compilation of (most of) FGJ into DOT

Guillaume Martres

Initial Upload: April 14, 2022
Last Revision: May 28, 2022

Abstract

The Dependent Object Type (DOT) calculus was designed to put Scala on a sound basis, but while DOT relies on structural subtyping, Scala is a fundamentally class-based language. This impedance mismatch means that a proof of DOT soundness by itself is not enough to declare a particular subset of the language as sound. While a few examples of Scala snippets have been manually translated into DOT, no systematic compilation scheme has been presented so far.

In this report we take advantage of the fact that the Featherweight Generic Java (FGJ) calculus can be seen as a subset of Scala (you just have to squint a little bit!) and present a compilation scheme from cast-less FGJ into DOT as well as a proof that a well-typed cast-less FGJ program compiles down to a well-typed DOT term. Due to limitations in DOT subtyping rules, this requires imposing one limitation on our source program: the type parameters of the base types of a class can never refer back to the class itself (this excludes `class C extends B[C]` as well as `class D extends B[E]; class E extends D`).

While this result is not especially interesting by itself given that FGJ is already known to be sound, it is a first step towards establishing soundness for larger subsets of Scala like the Pathless Scala calculus, the author plans to pursue this in his thesis (currently under preparation) which will also include the content of this report. It also clearly illustrates limitations in DOT that future work may wish to focus on.

Contents

1 Dependent Object Types **3**
1.1 Syntax and semantics 3
1.2 Syntactic sugar 7
1.3 Meta-theory 8

2 Type-preserving compilation of FJ to DOT **12**
2.1 Featherweight Java (FJ) 12
2.2 Translation 15
2.2.1 Meta-theory 17

3 Type-preserving compilation of FGJ to DOT **22**
3.1 Featherweight Generic Java (FGJ) 22
3.1.1 Meta-theory 27
3.2 Translation 27
3.2.1 Meta-theory 30

Bibliography **42**

1 Dependent Object Types

1.1 Syntax and semantics

Figure 1.1: DOT: Syntax

x, y, z	Variable
L	Type label
m	Method label
$s, t, u ::=$	Term
$\quad x$	variable reference
$\quad \{z \Rightarrow \overline{d}\}$	new instance
$\quad s.m(t)$	method invocation
$\Gamma ::= \overline{x : T}$	Context
$\sigma, \tau ::= [\overline{S/T}]$	Type substitution

$d ::=$	Declaration
$\quad L = T$	type member
$\quad m(x : S) : U^x = t$	method member
$S, T, U ::=$	Type
$\quad \top$	top type
$\quad \bot$	bottom type
$\quad L : S..U$	type member
$\quad m(x : S) : U^x$	method member
$\quad x.L$	type selection
$\quad \{z \Rightarrow T^z\}$	recursive self type
$\quad T \wedge T$	intersection type
$\quad T \vee T$	union type

This version of DOT is adapted from [Rompf and Amin 2016] [1]. The notation T^x emphasizes that x can appear free in T and $\Gamma_{[x]}$ means that all bindings in Γ to the right of x are dropped. Terms and types that are equal up to renaming of bound variables are identified. The set of free variables of a type T or term t is written $\mathsf{fv}(T)$ or $\mathsf{fv}(t)$ respectively. Unlike the original presentation, ours allows optional type ascriptions on method arguments and result types (the paper notes that their mechanized proof supports both variants). We denote optional syntax elements with wavy underlines. As in the original presentation:

> "For readability, we omit well-formedness requirements from the rules, and assume all types to be syntactically well-formed in the given environment."

[1] Other versions of DOT such as [Amin et al. 2016; Rapoport and Lhoták 2019] lack the recursive subtyping rules Bind1 and BindX which our type-preservation proofs critically rely on.

Figure 1.2: DOT: Well-formedness

Well-formed type $\boxed{\Gamma \vdash T \ \mathsf{wf}}$

$$\frac{\mathsf{fv}(T) \subseteq \mathsf{dom}(\Gamma)}{\Gamma \vdash T \ \mathsf{wf}}$$

Well-formed term $\boxed{\Gamma \vdash t \ \mathsf{wf}}$

$$\frac{\mathsf{fv}(t) \subseteq \mathsf{dom}(\Gamma)}{\Gamma \vdash t \ \mathsf{wf}}$$

Well-formed environment $\boxed{\Gamma \ \mathsf{wf}}$

$$\varnothing \ \mathsf{wf}$$

$$\frac{\Gamma, x : T^x \vdash T^x \ \mathsf{wf}}{\Gamma, x : T^x \ \mathsf{wf}}$$

Figure 1.3: DOT: Subtyping rules

$\boxed{\Gamma \vdash S <: U}$

Lattice structure

$$\Gamma \vdash \bot <: T \quad (\textsc{Bot}) \qquad \Gamma \vdash T <: \top \quad (\textsc{Top})$$

$$\frac{\Gamma \vdash T_1 <: T}{\Gamma \vdash T_1 \wedge T_2 <: T} \quad (\textsc{And11}) \qquad \frac{\Gamma \vdash T <: T_1}{\Gamma \vdash T <: T_1 \vee T_2} \quad (\textsc{Or21})$$

$$\frac{\Gamma \vdash T_2 <: T}{\Gamma \vdash T_1 \wedge T_2 <: T} \quad (\textsc{And12}) \qquad \frac{\Gamma \vdash T <: T_2}{\Gamma \vdash T <: T_1 \vee T_2} \quad (\textsc{Or22})$$

$$\frac{\Gamma \vdash T <: T_1,\ T <: T_2}{\Gamma \vdash T <: T_1 \wedge T_2} \quad (\textsc{And2}) \qquad \frac{\Gamma \vdash T_1 <: T,\ T_2 <: T}{\Gamma \vdash T_1 \vee T_2 <: T} \quad (\textsc{Or1})$$

Type and method members

$$\frac{\Gamma \vdash S_2 <: S_1,\ U_1 <: U_2}{\Gamma \vdash L : S_1..U_1 <: L : S_2..U_2} \quad (\textsc{Typ}) \qquad \frac{\Gamma \vdash S_2 <: S_1 \quad \Gamma, x : S_2 \vdash U_1^x <: U_2^x}{\Gamma \vdash m(x : S_1) : U_1^x <: m(x : S_2) : U_2^x} \quad (\textsc{Fun})$$

Path selections

$$\Gamma \vdash x.L <: x.L \quad (\textsc{SelX}) \qquad \frac{\Gamma_{[x]} \vdash x :_! (L : \bot..T)}{\Gamma \vdash x.L <: T} \quad (\textsc{Sel1})$$

$$\frac{\Gamma_{[x]} \vdash x :_! (L : T..\top)}{\Gamma \vdash T <: x.L} \quad (\textsc{Sel2})$$

Recursive self types

$$\frac{\Gamma, z : T_1^z \vdash T_1^z <: T_2^z}{\Gamma \vdash \{z \Rightarrow T_1^z\} <: \{z \Rightarrow T_2^z\}} \quad (\textsc{BindX}) \qquad \frac{\Gamma, z : T_1^z \vdash T_1^z <: T_2}{\Gamma \vdash \{z \Rightarrow T_1^z\} <: T_2} \quad (\textsc{Bind1})$$

Transitivity

$$\frac{\Gamma \vdash T_1 <: T_2,\ T_2 <: T_3}{\Gamma \vdash T_1 <: T_3} \quad (\textsc{Trans})$$

Figure 1.4: DOT: Typing rules

Type assignment $\boxed{\Gamma \vdash t :_{(!)} T}$

Variables, self packing/unpacking

$$\frac{\Gamma(x) = T^x}{\Gamma \vdash x :_{(!)} T^x} \quad (\textsc{Var})$$

$$\frac{\Gamma \vdash x : T^x}{\Gamma \vdash x : \{z \Rightarrow T^z\}} \quad (\textsc{VarPack})$$

$$\frac{\Gamma \vdash x :_{(!)} \{z \Rightarrow T^z\}}{\Gamma \vdash x :_{(!)} T^x} \quad (\textsc{VarUnpack})$$

Subsumption

$$\frac{\Gamma \vdash t :_{(!)} T_1,\ T_1 <: T_2}{\Gamma \vdash t :_{(!)} T_2} \quad (\textsc{Sub})$$

Method invocation

$$\frac{\Gamma \vdash t : (m(x : T_1) : T_2^x),\ y : T_1}{\Gamma \vdash t.m(y) : T_2^y} \quad (\textsc{TAppVar})$$

$$\frac{\Gamma \vdash t : (m(x : T_1) : T_2),\ t_2 : T_1}{\Gamma \vdash t.m(t_2) : T_2} \quad (\textsc{TApp})$$

Object creation

$$\frac{\overset{\text{(labels disjoint)}}{\Gamma,\ z : T_1^z \wedge \ldots \wedge T_n^z \vdash d_i : T_i^z \quad \forall i.1 \leq i \leq n}}{\Gamma \vdash \{z \Rightarrow d_1 \ldots d_n\} : \{z \Rightarrow T_1^z \wedge \ldots \wedge T_n^z\}} \quad (\textsc{TNew})$$

Member initialization $\boxed{\Gamma \vdash d : T}$

$$\frac{\Gamma \vdash T <: T}{\Gamma \vdash (L = T) : (L : T..T)} \quad (\textsc{DTyp})$$

$$\frac{\Gamma,\ x : T_1 \vdash t : T_2}{\Gamma \vdash (m(x : T_1) : T_2 = t) : (m(x : T_1) : T_2)} \quad (\textsc{DFun})$$

1.2 Syntactic sugar

The following derived syntactic forms will come in handy in our translations.

Definition 1.2.1: Type alias

$$(X = T) \rightsquigarrow (X : T .. T)$$

Definition 1.2.2: Anonymous function

We write _ to denote a fresh variable we never refer to.

Derived type

$$(x : S) \Rightarrow U^x \rightsquigarrow \{_ \Rightarrow \mathsf{apply}(x : S) : U^x\}$$

Derived term

$$\lambda x.\, u \rightsquigarrow \{_ \Rightarrow \mathsf{apply}(x) = u\}$$

Definition 1.2.3: Let bindings

Derived terms

$$\textbf{let } x = s \textbf{ in } u \rightsquigarrow (\lambda x.\, u).\mathsf{apply}(s)$$

$$\textbf{let } x = s,\ \overline{y = t} \textbf{ in } u \rightsquigarrow \textbf{let } x = s \textbf{ in } (\textbf{let } \overline{y = t} \textbf{ in } u)$$

Lemma 1.2.4

$$\frac{\Gamma \vdash t : T \quad \Gamma, x : T \vdash s : S}{\Gamma \vdash \textbf{let } x = t \textbf{ in } s : S} \qquad (\textsc{Let})$$

Definition 1.2.5: Methods with variable number of parameters

We write $\overline{x : S^*}$ to denote a list of parameters where each parameter type can refer to all previous parameters.

Derived types

$$m() : U_0 \rightsquigarrow m(_ : \top) : U_0$$

$$m(\overline{x : S^*}, y : T^{\overline{x}}) : U_0^{\overline{x}, y} \rightsquigarrow m(\overline{x : S^*}) : ((y : T^{\overline{x}}) \Rightarrow U_0^{\overline{x}, y})$$

Derived declarations (all type ascriptions are optional)

$$m() : U_0 = t \rightsquigarrow m(_ : \top) : U_0 = t$$

$$m(\overline{x : S^*}, y : T^{\overline{x}}) : U_0^{\overline{x}, y} = t \rightsquigarrow m(\overline{x : S^*}) : ((y : T^{\overline{x}}) \Rightarrow U_0^{\overline{x}, y}) = t$$

Derived terms

$$t.m() \rightsquigarrow t.m(\{_ \Rightarrow\})$$

$$t.m(\overline{x}, y) \rightsquigarrow t.m(\overline{x}).\mathsf{apply}(y)$$

Lemma 1.2.6

We can generalize DFun, Fun, TApp and TAppVar to methods with variable number of parameters. Note that TApp' generalizes both TApp and TAppVar since it lets the result type depend on a subset of the method arguments.

$$\frac{\Delta_1 = \Gamma \quad \Delta_{i+1} = \Gamma, x_1 : S_1, \ldots, x_i : S_i \quad \overline{\Delta \vdash T <: S} \quad \Delta_n \vdash U_1^{\overline{x}} <: U_2^{\overline{x}}}{\Gamma \vdash m(\overline{x : S^*}) : U_1^{\overline{x}} <: m(\overline{x : T^*}) : U_2^{\overline{x}}} \tag{Fun'}$$

$$\frac{\Gamma, \overline{x : S^*} \vdash u : U^{\overline{x}}}{\Gamma \vdash (m(\overline{x : S^*}) : U^{\overline{x}} = u) : (m(\overline{x : S^*}) : U^{\overline{x}})} \tag{DFun'}$$

$$\frac{\Gamma \vdash t : (m(\overline{x : S^*}, \overline{y : T^{\overline{x}}}) : U^{\overline{x}}) \quad \Delta_1 = \Gamma \quad \Delta_{i+1} = \Gamma, x_1 : S_1, \ldots, x_i : S_i \quad \overline{\Delta \vdash x : S^*} \quad \Delta_n \vdash \overline{t : T^{\overline{x}}}}{\Gamma \vdash t.m(\overline{x}, \overline{t}) : U^{\overline{x}}} \tag{TApp'}$$

1.3 Meta-theory

Definition 1.3.1: Type Equality

We write $\Gamma \vdash T =:= S$ as a short-hand for $\Gamma \vdash T <: S$, $S <: T$.

The following derived subtyping rules are defined in [Rompf and Amin 2016]:

$$\Gamma \vdash T <: T \tag{Refl}$$

$$\frac{\Gamma_1 \vdash T_1 <: T_2 \quad \Gamma_2(x) = T_2 \quad \Gamma_1 = \Gamma_2(x \to T_1) \quad \Gamma_2 \vdash S <: U}{\Gamma_1 \vdash S <: U} \tag{Narrow}$$

where $\Gamma_1 = \Gamma_2(x \to T_1)$ means that Γ_1 is like Γ_2 except the binding for x maps to T_1.

Lemma 1.3.2: Weakening

$$\frac{\Gamma_1, \Gamma_2 \vdash T_1 <: T_2 \quad y \notin \mathsf{dom}(\Gamma_1)}{\Gamma_1, y : U, \Gamma_2 \vdash T_1 <: T_2} \tag{Weaken}$$

$$\frac{\Gamma_1, \Gamma_2 \vdash t :_{(!)} T \quad y \notin \mathsf{dom}(\Gamma_1)}{\Gamma_1, y : U, \Gamma_2 \vdash t :_{(!)} T} \tag{WeakenTp}$$

Proof. Both rules are proved together by mutual induction on the subtyping and typing derivations, we only show a few representative cases:

Case $\dfrac{(\Gamma_1, \Gamma_2)_{[x]} \vdash x :_! (L : T..\top)}{\Gamma_1, \Gamma_2 \vdash T <: x.L}$ (Sel2)

> We can distinguish two sub-cases:
>
> - If $x \in \Gamma_1$, then $(\Gamma_1, y : U, \Gamma_2)_{[x]} = (\Gamma_1, \Gamma_2)_{[x]}$ and Sel2 finishes the case.
> - If $x \in \Gamma_2$, then $(\Gamma_1, \Gamma_2)_{[x]} = \Gamma_1, \Gamma_{2_{[x]}}$ and $(\Gamma_1, y : U, \Gamma_2)_{[x]} = \Gamma_1, y : U, \Gamma_{2_{[x]}}$, therefore by the IH we have $(\Gamma_1, y : U, \Gamma_2)_{[x]} \vdash x :_! (L : T..\top)$ and Sel2 finishes the case again.

Case $\dfrac{\Gamma_1, \Gamma_2, z : T_1^z \vdash T_1^z <: T_2^z}{\Gamma_1, \Gamma_2 \vdash \{z \Rightarrow T_1^z\} <: \{z \Rightarrow T_2^z\}}$ (BindX)

> By the IH we have $\Gamma_1, y : U, \Gamma_2, z : T_1^z \vdash T_1^z <: T_2^z$ and BindX finishes the case.

■

Lemma 1.3.3: Narrowing of types

$$\frac{\Gamma_1 \vdash T_1 <: T_2 \quad \Gamma_2(x) = T_2 \quad \Gamma_1 = \Gamma_2(x \to T_1) \qquad \Gamma_2 \vdash s :_{(!)} S}{\Gamma_1 \vdash s :_{(!)} S} \quad \text{(NarrowTp)}$$

Proof. By induction on the derivation of $\Gamma_2 \vdash s :_{(!)} S$, with a case analysis on the final rule. We only show Var and TNew as all other cases follow directly from the IH and Narrow.

Case $\dfrac{\Gamma_2(s) = S}{\Gamma_2 \vdash s :_{(!)} S}$ (Var)

> We can distinguish two sub-cases:
>
> - If $s = x$, then $S = T_2$, $\Gamma_1(s) = T_1$ and Sub finishes the case.
> - Otherwise, $\Gamma_1(s) = S$ and Var finishes the case.

Case $\dfrac{U = U_1^z \wedge \ldots \wedge U_n^z \Gamma_2, z : U \vdash d_i : U_i^z \quad \forall i.\, 1 \leq i \leq n}{\Gamma_2 \vdash \{z \Rightarrow d_1 \ldots d_n\} : \{z \Rightarrow U\}}$ (TNew)

> By Weaken we have $\Gamma_1, z : U \vdash T_1 <: T_2$ so by the IH $\Gamma_1, z : U \vdash d_i : U_i^z$ and TNew finishes the case.

■

Lemma 1.3.4

$$\frac{\Gamma_2(x) = T^x \quad \Gamma_1 = \Gamma_2(x \to \{z \Rightarrow T^z\}) \qquad \Gamma_2 \vdash S <: U}{\Gamma_1 \vdash S <: U} \quad \text{(EnvPack)}$$

$$\frac{\Gamma_2(x) = T^x \quad \Gamma_1 = \Gamma_2(x \to \{z \Rightarrow T^z\}) \qquad \Gamma_2 \vdash s :_{(!)} S}{\Gamma_1 \vdash s :_{(!)} S} \quad \text{(EnvPackTp)}$$

Proof. By mutual induction on the subtyping and typing derivations, we only show the Var case as all others follow by the IH:

Case $\dfrac{\Gamma_2(s) = S}{\Gamma_2 \vdash s :_{(!)} S}$ (Var)

We can distinguish two sub-cases:

- If $s = x$, then $S = T^x$ and $\Gamma_1 \vdash s :_{(!)} \{z \Rightarrow T^z\}$ by Var. VarUnpack finishes the case.
- Otherwise, $\Gamma_1(s) = S$ and Var finishes the case.

■

Lemma 1.3.5: Commutativity and associativity of intersection

$\Gamma \vdash T_1 \wedge T_2 <: T_2 \wedge T_1$, $\Gamma \vdash T_1 \wedge (T_2 \wedge T_3) <: (T_1 \wedge T_2) \wedge T_3$ and $\Gamma \vdash (T_1 \wedge T_2) \wedge T_3 <: T_1 \wedge (T_2 \wedge T_3)$

Proof. By And2, And11, And12 and Refl. ■

Lemma 1.3.6: Width and depth subtyping

1. $\Gamma \vdash T_0 \wedge \ldots \wedge T_1 \wedge \ldots \wedge T_2 <: T_1$
2. $\Gamma \vdash \{z \Rightarrow \overline{T_0}, \overline{T_1}, \overline{T_2}\} <: \{z \Rightarrow \overline{T_1}\}$
3. If $\Gamma \vdash \overline{S <: T}$ then $\Gamma \vdash \bigwedge \overline{S} <: \bigwedge \overline{T}$
4. If $\Gamma, z : \bigwedge \overline{S} \vdash \overline{S <: T}$ then $\Gamma \vdash \{z \Rightarrow \overline{S}\} <: \{z \Rightarrow \overline{T}\}$

Lemma 1.3.7: Depth subtyping

1. If $\Gamma \vdash \overline{S <: T}$ then $\Gamma \vdash \bigwedge \overline{S} <: \bigwedge \overline{T}$
2. If $\Gamma, z : \bigwedge \overline{S} \vdash \overline{S <: T}$ then $\Gamma \vdash \{z \Rightarrow \overline{S}\} <: \{z \Rightarrow \overline{T}\}$

Lemma 1.3.8: Substituting equal types preserves type equality

Given $\sigma = [\overline{T/S}]$ and $\Gamma \vdash \overline{T =:= S}$, if $\Gamma \vdash U$ wf then $\Gamma \vdash \sigma U =:= U$

Proof. By structural induction on U.

Case $U = \bot$ or $U = \top$ or $U = x.L$

If $U \notin \mathsf{dom}(\sigma)$ then $\sigma U = U$, otherwise $\sigma U = T_i$ for some i and $\Gamma \vdash T_i =:= U$.

Case $U = (L : U_1..U_2)$

We have $\sigma U = (L : \sigma U_1..\sigma U_2)$. By the IH, $\Gamma \vdash \sigma U_1 =:= U_1$ and $\Gamma \vdash \sigma U_2 =:= U_2$. Typ finishes the case.

Case $U = m(x : U_1) : U_2^x$

We have $\sigma U = m(x : \sigma U_1) : \sigma U_2^x$.

$$\dfrac{\dfrac{}{\Gamma \vdash U_1 <: \sigma U_1}\,(\text{IH}) \qquad \dfrac{\dfrac{}{\Gamma, x : \sigma U_1 \vdash \overline{T =:= S}}\,(\textsc{Weaken})}{\Gamma, x : \sigma U_1 \vdash U_2^x <: \sigma U_2^x}\,(\text{IH})}{\Gamma \vdash m(x : U_1) : U_2^x <: m(x : \sigma U_1) : \sigma U_2^x}\,(\textsc{Fun})$$

$$\dfrac{\dfrac{}{\Gamma \vdash \sigma U_1 <: U_1}\,(\text{IH}) \qquad \dfrac{\dfrac{}{\Gamma, x : U_1 \vdash \overline{T =:= S}}\,(\textsc{Weaken})}{\Gamma, x : U_1 \vdash \sigma U_2^x <: U_2^x}\,(\text{IH})}{\Gamma \vdash m(x : \sigma U_1) : \sigma U_2^x <: m(x : U_1) : U_2^x}\,(\textsc{Fun})$$

Case $U = \{z \Rightarrow U_1^z\}$

We have $\sigma U = \{z \Rightarrow \sigma U_1^z\}$, we only show one direction since the other proceeds similarly.

$$\dfrac{\dfrac{\dfrac{}{\Gamma, z : U_1^z \vdash \overline{T =:= S}}\,(\textsc{Weaken})}{\Gamma, z : U_1^z \vdash U_1^z <: \sigma U_1^z}\,(\text{IH})}{\Gamma \vdash \{z \Rightarrow U_1^z\} <: \{z \Rightarrow \sigma U_1^z\}}\,(\textsc{BindX})$$

Case $U = U_1 \wedge U_2$

We have $\sigma U = \sigma U_1 \wedge \sigma U_2$ and again we only show one direction.

$$\dfrac{\dfrac{\dfrac{}{\Gamma \vdash U_1 <: \sigma U_1}\,(\text{IH})}{\Gamma \vdash U_1 \wedge U_2 <: \sigma U_1}\,(\textsc{And12}) \qquad \dfrac{\dfrac{}{\Gamma \vdash U_2 <: \sigma U_2}\,(\text{IH})}{\Gamma \vdash U_1 \wedge U_2 <: \sigma U_2}\,(\textsc{And12})}{\Gamma \vdash U_1 \wedge U_2 <: \sigma U_1 \wedge \sigma U_2}\,(\textsc{And2})$$

Case $U = U_1 \vee U_2$

We have $\sigma U = \sigma U_1 \vee \sigma U_2$ and we only show one direction here too.

$$\dfrac{\dfrac{\dfrac{}{\Gamma \vdash U_1 <: \sigma U_1}\,(\text{IH})}{\Gamma \vdash U_1 <: \sigma U_1 \vee \sigma U_2}\,(\textsc{Or21}) \qquad \dfrac{\dfrac{}{\Gamma \vdash U_2 <: \sigma U_2}\,(\text{IH})}{\Gamma \vdash U_2 <: \sigma U_1 \vee \sigma U_2}\,(\textsc{Or22})}{\Gamma \vdash U_1 \vee U_2 <: \sigma U_1 \vee \sigma U_2}\,(\textsc{Or1})$$

■

2 Type-preserving compilation of FJ to DOT

2.1 Featherweight Java (FJ)

Figure 2.1: FJ: Syntax

		$L ::=$	Class declaration
x, y, z	Variable	$\textbf{class}\, C(\overline{f : D}) \lhd B(\overline{g})\{\overline{M}\}$	
B, C, D, E	Class type	$M ::=$	Method declaration
f, g	Class parameter	$\textbf{def}\, m(\overline{x : D}) : D_0 = e_0$	
m	Method name	$e ::=$	Expression
		x	variable
$\Gamma ::=$	Context	$e.f$	getter call
$\varnothing \mid \Gamma, x : C$		$e_0.m(\overline{e})$	method call
		$\textbf{new}\, C(\overline{e})$	object

The version of FJ we present here is adapted from [Igarashi, Pierce, and Wadler 2001] but excludes casts and uses Scala-like syntax as in [Martres 2021]. As in the original presentation we implicitly operate with a fixed class table CT such that $\vdash CT\ \mathsf{ok}$, all classes we reference are assumed to be defined in CT.

Figure 2.2: FJ: Subtyping rules and lookup functions

Subtyping $\boxed{C <: D}$

$$C <: C \qquad \text{(S-Refl)}$$

$$\frac{C <: D \quad D <: B}{C <: B} \qquad \text{(S-Trans)}$$

$$\frac{\textbf{class}\, C \ldots \lhd B \ldots}{C <: B} \qquad \text{(S-Class)}$$

Getters lookup $\boxed{\texttt{getters}(C) := \overline{f : D}}$

$$\texttt{getters}(\mathsf{Object}) := \varnothing$$

$$\frac{\textbf{class}\, C(\overline{f : D}) \ldots}{\texttt{getters}(C) := \overline{f : D}}$$

Method names lookup $\boxed{\texttt{mnames}(C) := \overline{m}}$

$$\texttt{mnames}(\mathsf{Object}) := \varnothing$$

$$\frac{\begin{array}{c}\textbf{class}\, C \ldots \lhd B\, \{\overline{\textbf{def}\, m_C \ldots}\} \\ \texttt{mnames}(B) = \overline{m_B} \\ \overline{n} = \left[m \in \overline{m_C} \;\middle|\; m \notin \overline{n}\right]\end{array}}{\texttt{mnames}(C) := \overline{m_B}, \overline{n}}$$

Method type and body lookup $\boxed{\begin{array}{l}\texttt{mtype}(m, C) := (\overline{x : D}) \to D_0 \\ \texttt{mbody}(m, C) := e_0\end{array}}$

$$\frac{\begin{array}{c}\textbf{class}\, C \ldots \{\overline{M}\} \\ \textbf{def}\, m(\overline{x : D}) : D_0 = e_0 \in \overline{M}\end{array}}{\begin{array}{l}\texttt{mtype}(m, C) := (\overline{x : D}) \to D_0 \\ \texttt{mbody}(m, C) := e_0\end{array}}$$

$$\frac{\textbf{class}\, C \ldots \lhd B\, \{\overline{M}\} \quad m \ldots \notin \overline{M}}{\begin{array}{l}\texttt{mtype}(m, C) := \texttt{mtype}(m, B) \\ \texttt{mbody}(m, C) := \texttt{mbody}(m, B)\end{array}}$$

Figure 2.3: FJ: Typing rules

Expression typing $\boxed{\Gamma \vdash e : C}$

$$\frac{\Gamma(x) = C}{\Gamma \vdash x : C} \quad \text{(T-Var)}$$

$$\frac{\Gamma \vdash e_0 : C \quad \texttt{getters}(C) = \overline{f : D}}{\Gamma \vdash e_0.f_i : D_i} \quad \text{(T-Getter)}$$

$$\frac{\Gamma \vdash e_0 : C \quad \texttt{mtype}(m, C) = (\overline{x : D}) \to D_0 \quad \Gamma \vdash \overline{e : E} \quad \overline{E <: D}}{\Gamma \vdash e_0.m(\overline{e}) : D_0} \quad \text{(T-Invk)}$$

$$\frac{\textbf{class}\, C(\overline{f : D}) \quad \Gamma \vdash \overline{e : E} \quad \overline{E <: D}}{\Gamma \vdash \textbf{new}\, C(\overline{e}) : C} \quad \text{(T-New)}$$

Method typing $\boxed{\Gamma \vdash m \text{ ok}}$

$$\frac{\begin{array}{c} C = \Gamma(\text{this}) \quad \textbf{class}\, C \lhd B \ldots \\ \texttt{mtype}(m, C) = (\overline{x : D}) \to D_0 \\ \texttt{mbody}(m, C) = e_0 \\ \Gamma, \overline{x : D} \vdash e_0 : E_0 \quad E_0 <: D_0 \\ \texttt{mtype}(m, B) \text{ defined implies } \texttt{mtype}(m, B) = \texttt{mtype}(m, C) \end{array}}{\Gamma \vdash m \text{ ok}} \quad \text{(T-Method)}$$

Class typing $\boxed{\vdash C \text{ ok}}$

$$\frac{\begin{array}{c} \textbf{class}\, C(\overline{g : E}, \overline{f : D}) \lhd B(\overline{g})\, \{\overline{\textbf{def}\, m \ldots}\} \\ \texttt{getters}(B) = \overline{g : E} \quad \overline{\text{this} : C \vdash m \text{ ok}} \end{array}}{\vdash C \text{ ok}} \quad \text{(T-Class)}$$

Class table typing $\boxed{\vdash CT \text{ ok}}$

$$\frac{\begin{array}{c} \text{Object} \notin \text{dom}(CT) \quad \forall C \in \text{dom}(CT).\, \vdash C \text{ ok} \\ \text{If class name } C \text{ (except Object) appears anywhere in } CT\text{, then } C \in CT \\ \text{No inheritance cycle between the classes in } CT \end{array}}{\vdash CT \text{ ok}} \quad \text{(T-CT)}$$

2.2 Translation

Our translation scheme is defined using three operators:

- $|\cdot|$ translates FJ types into DOT types and FJ terms into DOT terms.
- $(\!|\cdot|\!)$ translates lists of FJ declarations into lists of Church-style DOT declarations.
- $[\![\cdot]\!]$ returns the intersection of the types of the DOT declarations obtained using $(\!|\cdot|\!)$. This mapping is not defined explicitly below since it is easily inferred from the definition of $(\!|\cdot|\!)$. For convenience, we additionally define $[\![\mathsf{Object}]\!] \coloneqq \top$.

Figure 2.4: Translating FJ types and expressions to DOT

Type Translation $\boxed{|C| \coloneqq T_{\text{DOT}}}$

$$|\mathsf{Object}| \coloneqq \top$$

$$\frac{\textbf{class}\, C \ldots \lhd D \ldots}{|C| \coloneqq |D| \wedge \mathsf{ct}.C}$$

Expression Translation $\boxed{|e| \coloneqq t_{\text{DOT}}}$

$$|x| \coloneqq x$$

$$|e.f| \coloneqq |e|.f()$$

$$|e_0.m(\overline{e})| \coloneqq |e_0|.m(\overline{|e|})$$

$$|\textbf{new}\, C(\overline{e})| \coloneqq \mathsf{ct}.\,\mathsf{new}_C(\overline{|e|})$$

Figure 2.5: Translating FJ definitions to DOT

Getter Translation $\boxed{(\!|\overline{f : D}|\!) \coloneqq d_{\text{DOT}}}$

$$(\!|f : D|\!) \coloneqq f() : |D| = f_{\mathsf{param}}$$

$$(\!|\overline{f : D}|\!) \coloneqq \overline{(\!|f : D|\!)}$$

Method Translation $\boxed{(\!|\overline{m}|\!)_C \coloneqq d_{\text{DOT}}}$

$$\frac{\begin{array}{l}\texttt{mtype}(m, C) = (\overline{x : D}) \to D_0 \\ \texttt{mbody}(m, C) = e_0\end{array}}{(\!|m|\!)_C \coloneqq m(\overline{x : |D|}) : |D_0| = |e_0|}$$

$$(\!|\overline{m}|\!)_C \coloneqq \overline{(\!|m|\!)_C}$$

Class Translation $\boxed{(\!|C|\!) \coloneqq \overline{d_{\text{DOT}}}}$

$$(\!|C|\!) \coloneqq (\!|\texttt{getters}(C)|\!), (\!|\texttt{mnames}(C)|\!)_C$$

Class Table Translation $\boxed{(\!|CT|\!) \coloneqq \overline{d_{\text{DOT}}}}$

$$(\!|\varnothing|\!) \coloneqq \varnothing$$

$$(\!|\overline{L}, \textbf{class}\, C(\overline{f : D}) \ldots|\!) \coloneqq$$

$$(\!|\overline{L}|\!), C = \{\mathsf{this} \Rightarrow [\![C]\!]\},\ \mathsf{new}_C(\overline{f_{\mathsf{param}} : |D|}) : |C| = \{\mathsf{this} \Rightarrow (\!|C|\!)\}$$

Environment Translation $\boxed{|\Gamma| \coloneqq \Gamma_{\text{DOT}}}$

$$|\varnothing| \coloneqq \mathsf{ct} : [\![CT]\!]$$

$$\frac{\texttt{getters}(C) = \overline{f : D}}{|\Gamma, \mathsf{this} : C| \coloneqq |\Gamma|, \overline{f_{\mathsf{param}} : |D|}, \mathsf{this} : [\![C]\!]}$$

$$\frac{x \neq \mathsf{this}}{|\Gamma, x : C| \coloneqq |\Gamma|, x : |C|}$$

2.2.1 Meta-theory

Lemma 2.2.1: Well-formed translation

1. $|\Gamma| \vdash |C|, [\![C]\!]$ wf
2. If $(\mathsf{this} : C) \in \Gamma$ then $|\Gamma| \vdash (\!| C |\!)$ wf

Proof.

1. The only free variable that can appear in $|C|$ or $[\![C]\!]$ is ct which is always present in $|\Gamma|$.
2. Let $\texttt{getters}(C) = \overline{f : D}$, then we have $\{\overline{f_{\mathsf{param}}}, \mathsf{this}\} \subseteq \mathsf{dom}(|\Gamma|)$ which covers all additional free variables that can appear in $(\!| C |\!)$ by inspection.

■

Theorem 2.2.2: Subtyping preservation

If $C <: B$ then $|\Gamma| \vdash |C| <: |B|$.

Proof. By induction on the derivation of $C <: B$.

Case $C <: C$ (S-Refl)

By Refl.

Case $\dfrac{\textbf{class}\, C \ldots \lhd B \ldots}{C <: B}$ (S-Class)

By Refl we have $|\Gamma| \vdash |B| <: |B|$ and since $|C| = |B| \wedge \mathsf{ct}.C$, And12 finishes the case.

Case $\dfrac{C <: D \quad D <: B}{C <: B}$ (S-Trans)

By the IH, $|\Gamma| \vdash |C| <: |D|$ and $|\Gamma| \vdash |D| <: |B|$, Trans finishes the case.

■

Lemma 2.2.3

$|\Gamma| \vdash |C| <: [\![C]\!]$

Proof. $C = \mathsf{Object}$ is easy, otherwise we have $|C| = B \wedge \mathsf{ct}.C$ for some type B, $|\Gamma|_{[\mathsf{ct}]} = \mathsf{ct} : [\![CT]\!]$ and $[\![CT]\!] = \ldots \wedge (C : \{\mathsf{this} \Rightarrow [\![C]\!]\}..\{\mathsf{this} \Rightarrow [\![C]\!]\}) \wedge \ldots$ Hence:

$$\dfrac{\dfrac{\dfrac{}{|\Gamma|_{[\mathsf{ct}]} \vdash \mathsf{ct} :_! [\![CT]\!]}\,(\textsc{Var}) \qquad \dfrac{\dfrac{}{|\Gamma|_{[\mathsf{ct}]} \vdash \{\mathsf{this} \Rightarrow [\![C]\!]\} <: [\![C]\!]}\,(\textsc{Bind1})}{|\Gamma|_{[\mathsf{ct}]} \vdash [\![CT]\!] <: (C : \bot..[\![C]\!])}\,(1.3.6,\ \textsc{Typ})}{|\Gamma|_{[\mathsf{ct}]} \vdash \mathsf{ct} :_! (C : \bot..[\![C]\!])}\,(\textsc{Sub})}{|\Gamma| \vdash |C| <: [\![C]\!]}\,(\textsc{Sel1},\ \textsc{And12})$$

■

Corollary 2.2.4: Class translation preserves getters and methods

- If $\texttt{getters}(C) = \overline{f : D}$ then $|\Gamma| \vdash \overline{|C| <: (f() : |D|)}$.
- If $\texttt{mtype}(m, C) = (\overline{x : D}) \to D_0$ then $|\Gamma| \vdash |C| <: (m(\overline{x : |D|}) : |D_0|)$.

Proof. By definition, $[\![C]\!] = [\![\texttt{getters}(C)]\!] \wedge [\![\texttt{mnames}(C)]\!]_C$ so this follows from the previous lemma, transitivity and width subtyping. ■

Lemma 2.2.5

Given **class** $C \ldots \lhd B \ldots$ then $|\Gamma| \vdash [\![C]\!] <: [\![B]\!]$.

Proof. By definition, we want to show:

$$|\Gamma| \vdash [\![\texttt{getters}(C)]\!] \wedge [\![\texttt{mnames}(C)]\!]_C <: [\![\texttt{getters}(B)]\!] \wedge [\![\texttt{mnames}(B)]\!]_B$$

After proving the following claims, we can finish the case by depth subtyping.

Claim 1: $|\Gamma| \vdash [\![\texttt{getters}(C)]\!] <: [\![\texttt{getters}(B)]\!]$

$\vdash C$ ok implies that $\texttt{getters}(C) = (\texttt{getters}(B), \ldots)$ so the claim follows by width subtyping.

Claim 2: $|\Gamma| \vdash [\![\texttt{mnames}(C)]\!]_C <: [\![\texttt{mnames}(B)]\!]_B$

By definition, $\texttt{mnames}(C) = (\texttt{mnames}(B), \ldots)$ so $[\![\texttt{mnames}(C)]\!]_C = [\![\texttt{mnames}(B)]\!]_C \wedge T$ for some T and we only need to prove that $|\Gamma| \vdash [\![m]\!]_C <: [\![m]\!]_B$ for all $m \in \texttt{mnames}(B)$. If $m \in \texttt{mnames}(B)$ then either $m \notin M$ or $\Gamma \vdash m$ ok, in both cases this implies $[\![m]\!]_C = [\![m]\!]_B$. ■

Lemma 2.2.6

$|\Gamma| \vdash \{\mathsf{this} \Rightarrow [\![C]\!]\} <: |C|$

Proof. Since $\vdash C$ ok, we can have a sequence of class $\overline{D}$ such that $D_1 = C$, $D_n = \mathsf{Object}$ and $D_i <: D_{i+1}$ derived by the rules S-Refl and S-Class for any i. We prove by induction on the length n (≥ 1) of the sequence.

Case $(n = 1)$

By Top.

Case **class** $C \ldots \lhd B \ldots \quad (n \geq 2)$

In this case, $|C| = |B| \wedge \mathsf{ct}.C$. After proving the following claims, And2 finishes the case.

Claim 1: $|\Gamma| \vdash \{\mathsf{this} \Rightarrow [\![C]\!]\} <: |B|$

$$\dfrac{\dfrac{\dfrac{}{|\Gamma|, \mathsf{this} : [\![C]\!] \vdash [\![C]\!] <: [\![B]\!]}\,(2.2.5)}{|\Gamma| \vdash \{\mathsf{this} \Rightarrow [\![C]\!]\} <: \{\mathsf{this} \Rightarrow [\![B]\!]\}}\,(\textsc{BindX}) \quad \dfrac{}{|\Gamma| \vdash \{\mathsf{this} \Rightarrow [\![B]\!]\} <: |B|}\,(\text{IH})}{|\Gamma| \vdash \{\mathsf{this} \Rightarrow [\![C]\!]\} <: |B|}\,(\textsc{Trans})$$

Claim 2: $|\Gamma| \vdash \{\mathsf{this} \Rightarrow [\![C]\!]\} <: \mathsf{ct}.C$

$$\dfrac{\dfrac{}{|\Gamma|_{[\mathsf{ct}]} \vdash \mathsf{ct} :_! [\![CT]\!]}\,(\textsc{Var}) \qquad \dfrac{}{|\Gamma|_{[\mathsf{ct}]} \vdash [\![CT]\!] <: (C : (\{\mathsf{this} \Rightarrow [\![C]\!]\})..\top)}\,(1.3.6, \textsc{Typ})}{\dfrac{|\Gamma|_{[\mathsf{ct}]} \vdash \mathsf{ct} :_! (C : (\{\mathsf{this} \Rightarrow [\![C]\!]\})..\top)}{|\Gamma| \vdash \{\mathsf{this} \Rightarrow [\![C]\!]\} <: \mathsf{ct}.C}\,(\textsc{Sel2})}\,(\textsc{Sub})$$

■

Conjecture 2.2.7

$|\Gamma| \vdash [\![C]\!] <: |C|$

This would follow from Lemma 2.2.6 and Trans if we had a Bind2 rule symmetric to the existing Bind1 to prove $|\Gamma| \vdash [\![C]\!] <: \{\mathsf{this} \Rightarrow [\![C]\!]\}$, but this rule is missing from [Rompf and Amin 2016] (unlike gDOT [Giarrusso et al. 2020]):

> "We conjecture that these contractiveness restrictions could be lifted without breaking soundness, since we can always construct explicit conversion functions that use rules VarPack and VarUnpack on proper term bindings. However, removing these contractiveness restrictions would likely require different and harder to mechanize proof techniques such as a coinductive interpretation of subtyping."

For our purposes, VarPack is indeed enough:

Lemma 2.2.8: this **translation is type-preserving**

If $(\mathsf{this} : C) \in \Gamma$ then $|\Gamma| \vdash \mathsf{this} : |C|$

Proof.

$$\dfrac{\dfrac{\dfrac{}{|\Gamma| \vdash \mathsf{this} : [\![C]\!]}\,(\textsc{Var})}{|\Gamma| \vdash \mathsf{this} : \{\mathsf{this} \Rightarrow [\![C]\!]\}}\,(\textsc{VarPack}) \qquad \dfrac{}{|\Gamma| \vdash \{\mathsf{this} \Rightarrow [\![C]\!]\} <: |C|}\,(2.2.6)}{|\Gamma| \vdash \mathsf{this} : |C|}\,(\textsc{Sub})$$

■

Theorem 2.2.9: Typing translation is type-preserving

If $\Gamma \vdash e : C$ then $|\Gamma| \vdash |e| : |C|$.

Proof. By induction on the derivation of $\Gamma \vdash e : C$.

Case $\dfrac{\Gamma(x) = C}{\Gamma \vdash x : C}\,(\text{T-}\textsc{Var})$

By definition, $|\Gamma|(|x|) = |\Gamma|(x)$, and we can distinguish two sub-cases:

- If $x = \mathsf{this}$, then $|\Gamma|(\mathsf{this}) = [\![C]\!]$ by definition and Lemma 2.2.8 finishes the case.
- Otherwise, $|\Gamma|(x) = |C|$ and Var finishes the case.

Case $\dfrac{\Gamma \vdash e_0 : C \quad \texttt{getters}(C) = \overline{f : D}}{\Gamma \vdash e_0.f_i : D_i}$ (T-Getter)

By the IH, $|\Gamma| \vdash |e_0| : |C|$. By Corollary 2.2.4 and Sub, $|\Gamma| \vdash |e_0| : (f_i() : |D_i|)$. TApp finishes the case.

Case $\dfrac{\Gamma \vdash e_0 : C \quad \texttt{mtype}(m, C) = (\overline{x : D}) \to D_0 \quad \Gamma \vdash \overline{e : E} \quad \overline{E <: D}}{\Gamma \vdash e_0.m(\overline{e}) : D_0}$ (T-Invk)

By the IH, $|\Gamma| \vdash |e_0| : |C|$ and $|\Gamma| \vdash \overline{|e| : |E|}$. By Corollary 2.2.4 and Sub, $|\Gamma| \vdash |e_0| : (m(\overline{x : |D|}) : |D_0|)$. TApp' finishes the case since by Theorem 2.2.2, $|\Gamma| \vdash \overline{|E| <: |D|}$ and so by Sub, $|\Gamma| \vdash \overline{|e| : |D|}$.

Case $\dfrac{\textbf{class}\, C(\overline{f : D}) \quad \Gamma \vdash \overline{e : E} \quad \overline{E <: D}}{\Gamma \vdash \textbf{new}\, C(\overline{e}) : C}$ (T-New)

By the IH, $|\Gamma| \vdash \overline{|e| : |E|}$. By Lemma 1.3.6, Sub and Var, $|\Gamma| \vdash \mathsf{ct} : (\mathsf{new}_C(\overline{f_{\mathsf{param}} : |D|}) : |C|)$. We can finish using TApp' like in the previous case.

■

Lemma 2.2.10: Class translation is type-preserving

Given $\textbf{class}\, C(\overline{f : D})$ ... then $|\varnothing|, \overline{f_{\mathsf{param}} : |D|} \vdash \{\mathsf{this} \Rightarrow (\!| C |\!)\} : \{\mathsf{this} \Rightarrow [\![C]\!]\}$

Proof. By TNew, this is true if the following claims are all true.

Claim 1: $|\mathsf{this} : C| \vdash (\!| f : D |\!) : [\![f : D]\!] \quad \forall (f : D) \in \texttt{getters}(C)$

By Var.

Claim 2: $|\mathsf{this} : C| \vdash (\!| m |\!)_C : [\![m]\!]_C \quad \forall m \in \texttt{mnames}(C)$

Let $\texttt{mtype}(m, C) = (\overline{x : D}) \to D_0$ and $\texttt{mbody}(m, C) = e_0$. Then, by inversion of T-Method we have

$$\mathsf{this} : C, \overline{x : D} \vdash e_0 : E_0$$
$$E_0 <: D_0$$

Therefore

$$\dfrac{\dfrac{\dfrac{\mathsf{this} : C, \overline{x : D} \vdash e_0 : E_0}{|\mathsf{this} : C, \overline{x : D}| \vdash |e_0| : |E_0|}\,(2.2.9)}{|\mathsf{this} : C, \overline{x : D}| \vdash |e_0| : |D_0|}\,(\text{Sub, } 2.2.2)}{|\mathsf{this} : C| \vdash (\!| m |\!)_C : [\![m]\!]_C}\,(\text{DFun'})$$

■

Lemma 2.2.11: Class table translation is type-preserving

$\varnothing \vdash \{\mathsf{ct} \Rightarrow (\!| CT |\!)\} : \{\mathsf{ct} \Rightarrow [\![CT]\!]\}$.

Proof. After proving the following claims for each **class** $C(\overline{f : D})$ in CT, we can finish the proof by TNew.

Claim 1: $|\varnothing| \vdash (C = \{\mathsf{this} \Rightarrow [\![C]\!]\}) : (C : \{\mathsf{this} \Rightarrow [\![C]\!]\}..\{\mathsf{this} \Rightarrow [\![C]\!]\})$

By DTyp.

Claim 2: $|\varnothing| \vdash (\mathsf{new}_C(\overline{f_{\mathsf{param}} : |D|}) : |C| = \{\mathsf{this} \Rightarrow (\!| C |\!)\}) : (\mathsf{new}_C(\overline{f_{\mathsf{param}} : |D|}) : |C|)$

Let $\Gamma_0 = |\varnothing|, f_{\mathsf{param}} : |D|$. Then,

$$\dfrac{\dfrac{}{\Gamma_0 \vdash \{\mathsf{this} \Rightarrow (\!| C |\!)\} : \{\mathsf{this} \Rightarrow [\![C]\!]\}}\,(2.2.10) \qquad \dfrac{\dfrac{}{|\varnothing| \vdash \{\mathsf{this} \Rightarrow [\![C]\!]\} <: |C|}\,(2.2.6)}{\Gamma_0 \vdash \{\mathsf{this} \Rightarrow [\![C]\!]\} <: |C|}\,(\textsc{WeakenTp})}{\Gamma_0 \vdash \{\mathsf{this} \Rightarrow (\!| C |\!)\} : |C|}\,(\textsc{Sub})$$

and DFun' finishes the claim.

■

Theorem 2.2.12: Program translation is type-preserving

If $\varnothing \vdash_{\text{FJ}} e : C$ then $\varnothing \vdash_{\text{DOT}} \textbf{let}\ \mathsf{ct} = \{\mathsf{ct} \Rightarrow (\!| CT |\!)\}\ \textbf{in}\ |e| : |C|$.

Proof.

$$\dfrac{\dfrac{}{\varnothing \vdash \{\mathsf{ct} \Rightarrow (\!| CT |\!)\} : \{\mathsf{ct} \Rightarrow [\![CT]\!]\}}\,(2.2.11) \qquad \dfrac{\dfrac{\varnothing \vdash e : C}{\mathsf{ct} : [\![CT]\!] \vdash |e| : |C|}\,(2.2.9)}{\mathsf{ct} : \{\mathsf{ct} \Rightarrow [\![CT]\!]\} \vdash |e| : |C|}\,(\textsc{EnvPackTp})}{\varnothing \vdash \textbf{let}\ \mathsf{ct} = \{\mathsf{ct} \Rightarrow (\!| CT |\!)\}\ \textbf{in}\ |e| : |C|}\,(\textsc{Let})$$

■

3 Type-preserving compilation of FGJ to DOT

3.1 Featherweight Generic Java (FGJ)

Figure 3.1: FGJ: Syntax

x, y, z	Variable
B, C, D, E	Class name
f, g	Class parameter
m	Method name
X_C	Class variable
X_m	Method variable
$X, Y, Z ::= X_C \mid X_m$	Type variable
$N, P, Q ::= C[\overline{T}]$	Non-variable
$S, T, U ::= X \mid N$	Type
$\Gamma ::=$	Context
$\varnothing \mid \Gamma, x : T \mid \Gamma, X <: N$	

$L ::=$	Class declaration
$\textbf{class}\, C\, [\overline{X_C <: N}]\, (\overline{f : T}) \lhd P(\overline{f})\, \{\overline{M}\}$	
$M ::=$	Method declaration
$\textbf{def}\, m\, [\overline{X_m <: N}]\, (\overline{x : T}) : T_0 = e_0$	
$e ::=$	Expression
x	variable
$e.f$	getter call
$e_0.m\, [\overline{T}]\, (\overline{e})$	method call
$\textbf{new}\, C\, [\overline{T}]\, (\overline{e})$	object
$\sigma, \tau ::= [\overline{T/X}]$	Type substitution

As in Section 2.1, we implicitly operate with a fixed class table CT such that $\vdash CT$ ok. The version of FGJ we present here differs from [Igarashi, Pierce, and Wadler 2001] in a few ways:

- Like in Section 2.1, we drop casts and use Scala-like syntax.
- We distinguish between class type variables X_C and method type variables X_m in the syntax so that we can translate them differently (Section 3.2).
- Our definition of `override` is *more* expressive than the original one as it take into account the environment Γ containing the class type parameters. This is needed to be able to typecheck:

```
class A
class Base { def foo(): A = … }
class Sub[S <: A] ◁ Base { def foo(): S = … }
```

The equivalent Java code is valid and yet `Sub` is not well-formed in [Igarashi, Pierce, and

Wadler 2001, Figure 6] because the type parameter `S <: A` is not part of the environment when the override check is done.

- Our definition of type well-formedness is *less* expressive than the original one. Our version of WF-Class additionally requires that the parent type is well-formed after substituting the type arguments:

$$\frac{\textbf{class}\, C[\overline{X <: N}] \lhd P \ldots \quad \sigma = [\overline{T/X}] \qquad \Gamma \vdash \overline{T}\ \mathsf{wf} \quad \boxed{\Gamma \vdash \sigma P\ \mathsf{wf}} \quad \Gamma \vdash \overline{T <: \sigma N}}{\Gamma \vdash C[\overline{T}]\ \mathsf{wf}} \qquad \text{(WF-Class)}$$

This effectively disallows class hierarchies where a type parameter of the base type of a class refers back to the class itself as in

```
class C ◁ B[C] // cannot derive Γ ⊢ C wf
```

or

```
class B[X] ◁ Object
class D ◁ B[E]
class E ◁ D // cannot derive Γ ⊢ E wf
```

This restriction is motivated by our type translation scheme (Section 3.2):

$$\frac{\textbf{class}\, C[\overline{X <: \ldots}] \ldots \lhd P \ldots \quad \sigma = [\overline{T/X}]}{|C[\overline{T}]| \coloneqq |\sigma P| \wedge \mathsf{ct}.C \wedge \bigwedge \overline{X = |T|}}$$

This encoding makes it possible to prove that $\Gamma \vdash C[\overline{T}] <: \sigma P$ implies $|\Gamma| \vdash |C[\overline{T}]| <: |\sigma P|$ which is key to subtyping preservation (Theorem 3.2.3 and Corollary 3.2.4) but it leads to non-terminating expansions when applied to the classes defined above:

$$\boxed{|C|} = |B[C]| \wedge \mathsf{ct}.C$$
$$|B[C]| = |\mathsf{Object}| \wedge \mathsf{ct}.B \wedge \left(X_B = \boxed{|C|}\right)$$

$$\boxed{|E|} = |D| \wedge \mathsf{ct}.E$$
$$|D| = |B[E]| \wedge \mathsf{ct}.D$$
$$|B[E]| = |\mathsf{Object}| \wedge \mathsf{ct}.B \wedge \left(X_B = \boxed{|E|}\right)$$

Could we come up with an alternative translation scheme that does not have this limitation? Not according to Conjecture 3.2.1.

Figure 3.2: FGJ: Subtyping

$\boxed{\Gamma \vdash S <: T}$

$$\Gamma \vdash S <: S \qquad \text{(GS-Refl)}$$

$$\frac{\Gamma(X) = N}{\Gamma \vdash X <: N} \qquad \text{(GS-Var)}$$

$$\frac{\textbf{class}\, C[\overline{X <: N}] \ldots \lhd P \ldots}{\Gamma \vdash C[\overline{T}] <: [\overline{T/X}]P} \qquad \text{(GS-Class)}$$

$$\frac{\Gamma \vdash S <: U \quad \Gamma \vdash U <: T}{\Gamma \vdash S <: T} \qquad \text{(GS-Trans)}$$

Figure 3.3: FGJ: Well-formedness

Well-formed type $\boxed{\Gamma \vdash T \text{ wf}}$

$$\Gamma \vdash \text{Object wf} \qquad \text{(WF-Object)}$$

$$\frac{X \in \text{dom}(\Gamma)}{\Gamma \vdash X \text{ wf}} \qquad \text{(WF-Var)}$$

$$\frac{\begin{array}{c}\textbf{class}\, C[\overline{X <: N}] \lhd P \ldots \quad \sigma = [\overline{T/X}] \\ \Gamma \vdash \overline{T} \text{ wf} \quad \boxed{\Gamma \vdash \sigma P \text{ wf}} \quad \Gamma \vdash \overline{T <: \sigma N}\end{array}}{\Gamma \vdash C[\overline{T}] \text{ wf}} \qquad \text{(WF-Class)}$$

Well-formed environment $\boxed{\Gamma \text{ wf}}$

$$\varnothing \text{ wf}$$

$$\frac{\Gamma, \overline{X <: N} \vdash \overline{N} \text{ wf}}{\Gamma, \overline{X <: N} \text{ wf}}$$

$$\frac{\Gamma \vdash T \text{ wf}}{\Gamma, x : T \text{ wf}}$$

Figure 3.4: FGJ: Lookup functions

Getters lookup $\boxed{\texttt{getters}_\Gamma(T) := \overline{f : T}}$

$$\texttt{getters}(\mathsf{Object}) := \varnothing \qquad \text{(G-Object)}$$

$$\texttt{getters}_\Gamma(X) := \texttt{getters}(\Gamma(X)) \qquad \text{(G-Var)}$$

$$\frac{\textbf{class}\, C[\overline{X <: N}](\overline{f : U}) \ldots \quad \sigma = [\overline{S/X}]}{\texttt{getters}(C[\overline{T}]) := \overline{f : \sigma U}} \qquad \text{(G-Class)}$$

Method names lookup $\boxed{\texttt{mnames}(C) := \overline{m}}$

$$\texttt{mnames}(\mathsf{Object}) := \varnothing$$

$$\frac{\begin{array}{c}\textbf{class}\, C \ldots \lhd B\, \{\overline{\textbf{def}\, m_C \ldots}\} \\ \texttt{mnames}(B) = \overline{m_B} \\ \overline{n} = \left[m \in \overline{m_C} \;\middle|\; m \notin \overline{n}\right]\end{array}}{\texttt{mnames}(C) := \overline{m_B}, \overline{n}}$$

Method type and body lookup

$$\boxed{\begin{array}{l}\texttt{mtype}_\Gamma(m, S) := [\overline{Y <: P}] \to (\overline{x : T}) \to T_0 \\ \texttt{mbody}_\Gamma(m, S) := e_0\end{array}}$$

$$\begin{array}{l}\texttt{mtype}_\Gamma(m, X) := \texttt{mtype}(m, \Gamma(X)) \\ \texttt{mbody}_\Gamma(m, X) := \texttt{mbody}(m, \Gamma(X))\end{array} \qquad \text{(M-Var)}$$

$$\frac{\begin{array}{c}\textbf{class}\, C[\overline{X <: N}] \ldots \{\overline{M}\} \quad \sigma = [\overline{T/X}] \\ (\textbf{def}\, m[\overline{Y <: P}](\overline{x : T}) : T_0 = e_0) \in \overline{M}\end{array}}{\begin{array}{l}\texttt{mtype}(m, C[\overline{T}]) := [\overline{Y <: \sigma P}] \to (\overline{x : \sigma T}) \to \sigma T_0) \\ \texttt{mbody}(m, C[\overline{T}]) := \sigma e_0\end{array}} \qquad \text{(M-Class)}$$

$$\frac{\begin{array}{c}\textbf{class}\, C[\overline{X <: N}] \ldots \lhd P\, \{\overline{M}\} \quad \sigma = [\overline{T/X}] \\ (\textbf{def}\, m \ldots) \notin \overline{M}\end{array}}{\begin{array}{c}\texttt{mtype}(m, C[\overline{T}]) := \texttt{mtype}(m, \sigma P) \\ \texttt{mbody}(m, C[\overline{T}]) := \texttt{mbody}(m, \sigma P)\end{array}} \qquad \text{(M-Super)}$$

Valid method overriding $\boxed{\texttt{override}_\Gamma(m, N_i, N_o)}$

$$\frac{\begin{array}{l}\texttt{mtype}(m, N_o) \text{ defined implies:} \\ \bullet\ \texttt{mtype}(m, N_i) = [\overline{Y <: P}] \to (\overline{x : U}) \to U_i \\ \bullet\ \texttt{mtype}(m, N_o) = [\overline{Y <: P}] \to (\overline{x : U}) \to U_o \\ \bullet\ \Gamma, \overline{Y <: P} \vdash U_i <: U_o\end{array}}{\texttt{override}_\Gamma(m, N_i, N_o)}$$

Figure 3.5: FGJ: Typing rules

Expression typing $\boxed{\Gamma \vdash e : T}$

$$\frac{\Gamma(x) = T}{\Gamma \vdash x : T} \quad \text{(GT-Var)}$$

$$\frac{\Gamma \vdash e_0 : T_0 \quad \texttt{getters}_\Gamma(T_0) = \overline{f : T}}{\Gamma \vdash e_0.f_i : T_i} \quad \text{(GT-Getter)}$$

$$\frac{\begin{array}{c}\Gamma \vdash e_0 : T_0 \quad \texttt{mtype}_\Gamma(m, T_0) = [\overline{Y <: P}] \to (\overline{x : U}) \to U_0 \\ \sigma = [\overline{V/Y}] \quad \Gamma \vdash \overline{V} \text{ wf}, \overline{V <: \sigma P}, \overline{e : S}, \overline{S <: \sigma U}\end{array}}{\Gamma \vdash e_0.m[\overline{V}](\overline{e}) : T_0} \quad \text{(GT-Invk)}$$

$$\frac{\Gamma \vdash N \text{ wf} \quad \texttt{getters}(N) = \overline{f : U} \quad \Gamma \vdash \overline{e : S}, \ \overline{S <: U}}{\Gamma \vdash \textbf{new}\, N(\overline{e}) : N} \quad \text{(GT-New)}$$

Method typing $\boxed{\Gamma \vdash m \text{ ok}}$

$$\frac{\begin{array}{c}C[\overline{X}] = \Gamma(\text{this}) \quad \textbf{class}\, C[\overline{X <: \ldots}](\ldots) \lhd N \ldots \\ \texttt{mtype}(m, C[\overline{X}]) = [\overline{Y <: P}] \to (\overline{x : U}) \to U_0 \quad \texttt{mbody}(m, C[\overline{X}]) = e_0 \\ \Gamma, \overline{Y <: P} \vdash \overline{U}, U_0, \overline{P} \text{ wf} \\ \Gamma, \overline{Y <: P}, \overline{x : U} \vdash e_0 : E_0, \ E_0 <: U_0 \\ \texttt{override}_\Gamma(m, C[\overline{X}], N)\end{array}}{\Gamma \vdash m \text{ ok}} \quad \text{(GT-Method)}$$

Class typing $\boxed{\vdash C \text{ ok}}$

$$\frac{\begin{array}{c}\textbf{class}\, C[\overline{X <: N}](\overline{g : U}, \overline{f : T}) \lhd P(\overline{g})\ \{\overline{\textbf{def}\ m \ldots}\} \\ \Gamma = \overline{X <: N}, \text{this} : C[\overline{X}] \\ \Gamma \vdash \overline{N}, \overline{U}, \overline{T}, P \text{ ok} \quad \texttt{getters}(P) = \overline{g : U} \quad \Gamma \vdash \overline{m \text{ ok}}\end{array}}{\vdash C \text{ ok}} \quad \text{(GT-Class)}$$

Class table typing $\boxed{\vdash CT \text{ ok}}$

$$\frac{\begin{array}{c}\text{Object} \notin \text{dom}(CT) \quad \forall C \in \text{dom}(CT).\ \vdash C \text{ ok} \\ \text{If class name } C \text{ (except Object) appears anywhere in } CT\text{, then } C \in \text{dom}(CT) \\ \text{No inheritance cycle between the classes in } CT\end{array}}{\vdash CT \text{ ok}} \quad \text{(GT-CT)}$$

3.1.1 Meta-theory

Lemma 3.1.1

If Γ wf, $\Gamma \vdash S$ wf and $\Gamma \vdash S <: T$, then $\Gamma \vdash T$ wf and only well-formed types appear in the derivation tree of $\Gamma \vdash S <: T$.

Proof. By induction on $\Gamma \vdash S <: T$, case GS-Refl is trivial.

Case $\dfrac{\Gamma(X) = N}{\Gamma \vdash X <: N}$ (GS-Var)

$\Gamma \vdash N$ wf because Γ wf

Case $\dfrac{\textbf{class}\, C[\overline{X <: N}] \ldots \lhd P \ldots}{\Gamma \vdash C[\overline{T}] <: [\overline{T/X}]P}$ (GS-Class)

$\Gamma \vdash [\overline{T/X}]P$ wf by WF-Class.

Case $\dfrac{\Gamma \vdash S <: U \quad \Gamma \vdash U <: T}{\Gamma \vdash S <: T}$ (GS-Trans)

By the IH, $\Gamma \vdash U$ wf so by the IH again $\Gamma \vdash T$ wf.

■

Definition 3.1.2: Free variables in a type

$$\mathsf{fv}(X) \coloneqq \{X\}$$
$$\mathsf{fv}(C[\overline{T}]) \coloneqq \bigcup \overline{\mathsf{fv}(T)}$$

Lemma 3.1.3: Associativity of substitutions

If $\mathsf{fv}(T) \subseteq \mathsf{dom}(\sigma)$ then $(\tau\sigma)T = \tau(\sigma T)$

Proof. By structural induction on T.

Case $T = X$

In this case $\mathsf{fv}(T) = \{X\} \subseteq \mathsf{dom}(\sigma)$, therefore $\sigma = [\ldots, S/X, \ \ldots]$ for some S and by definition $\tau\sigma = [\ldots, (\tau S)/X, \ \ldots]$. Hence, $(\tau\sigma)X = \tau S = \tau(\sigma X)$.

Case $T = C[\overline{S}]$

We have $(\tau\sigma)C[\overline{S}] = C[\overline{(\tau\sigma)S}]$ and $\tau(\sigma C[\overline{S}]) = C[\overline{\tau(\sigma S)}]$. The IH finishes the case.

■

3.2 Translation

As in Section 2.2, our translation scheme is defined using $|\cdot|$, $(\!|\cdot|\!)$ and $[\![\cdot]\!]$.

Figure 3.6: Translating FGJ types and expressions to DOT

Type Translation $\boxed{|T| \coloneqq T_{\text{DOT}}}$

$$|\mathsf{Object}| \coloneqq \top$$

$$|X_C| \coloneqq \mathsf{this}.X_C$$

$$|X_m| \coloneqq \mathsf{mtag}.X_m$$

$$\frac{\textbf{class}\, C[\overline{X <: \ldots}] \ldots \lhd P \ldots \quad \sigma = [\overline{T/X}]}{|C[\overline{T}]| \coloneqq |\sigma P| \wedge \mathsf{ct}.C \wedge \bigwedge \overline{X = |T|}}$$

Type Parameter Clause Translation $\boxed{|X <: N| \coloneqq T_{\text{DOT}}}$

$$|\overline{X_C <: N}| \coloneqq \{\mathsf{this} \Rightarrow \overline{X_C : \bot..|N|}\}$$

$$|\overline{X_m <: N}| \coloneqq \{\mathsf{mtag} \Rightarrow \overline{X_m : \bot..|N|}\}$$

Expression Translation $\boxed{|e|_\Gamma \coloneqq t_{\text{DOT}}}$

$$|x|_\Gamma \coloneqq x$$

$$\frac{x \text{ is fresh} \quad \Gamma \vdash e_0.f : T_0}{|e_0.f|_\Gamma \coloneqq \textbf{let}\ x = |e_0|_\Gamma\ \textbf{in}\ x.f}$$

$$\frac{\begin{array}{c} x \text{ is fresh} \quad x_{\mathsf{mtag}} \text{ is fresh} \quad \Gamma \vdash e_0 : T_0 \\ \mathtt{mtype}_\Gamma(m, T_0) = [\overline{Y <: P}] \rightarrow \ldots \end{array}}{|e_0.m[\overline{V}](\overline{e})|_\Gamma \coloneqq \textbf{let}\ x = |e_0|_\Gamma, x_{\mathsf{mtag}} = \{_ \Rightarrow \overline{Y = |V|}\}\ \textbf{in}\ x.m(x_{\mathsf{mtag}}, \overline{|e|_\Gamma})}$$

$$\frac{x_{\mathsf{ctag}} \text{ is fresh} \quad \textbf{class}\, C[\overline{X <: \ldots}] \ldots}{|\textbf{new}\, C[\overline{V}](\overline{e})|_\Gamma \coloneqq \textbf{let}\ x_{\mathsf{ctag}} = \{_ \Rightarrow \overline{X = |V|}\}\ \textbf{in}\ \mathsf{ct.new}_C(x_{\mathsf{ctag}}, \overline{|e|_\Gamma})}$$

Figure 3.7: Translating FGJ definitions to DOT

Getter Translation $\boxed{(\!| f : T |\!) \coloneqq d_{\text{DOT}}}$

$$(\!| f : T |\!) \coloneqq f() : |T| = f_{\mathsf{param}}$$

Method Translation $\boxed{(\!| m |\!)_C \coloneqq d_{\text{DOT}}}$

$$\frac{\begin{array}{l}\textbf{class}\, C[\overline{X <: N}] \ldots \quad \Gamma = \overline{X <: N}, \mathsf{this} : C[\overline{X}] \\ \mathtt{mtype}(m, C[\overline{X}]) = [\overline{Y <: P}] \to (\overline{x : U}) \to U_0 \\ \mathtt{mbody}(m, C[\overline{X}]) = e_0\end{array}}{(\!| m |\!)_C \coloneqq m(\mathsf{mtag} : |\overline{Y <: P}|, \overline{x : |U|}) : |U_0| = |e_0|_{\Gamma, \overline{Y <: P}, \overline{x : U}}}$$

Class Translation $\boxed{(\!| C |\!) \coloneqq \overline{d_{\text{DOT}}}}$

$$\frac{\textbf{class}\, C[\overline{X <: N}] \ldots \quad \mathtt{baseArgs}(C) = \bigwedge \overline{Z = S}}{\begin{array}{l}(\!| C |\!) \coloneqq (\!| \mathtt{getters}(C[\overline{X}]) |\!), (\!| \mathtt{mnames}(C) |\!)_C, \overline{Z = S} \\ (\!| C |\!)^{\overline{T}} \coloneqq (\!| C |\!), \overline{X = T}\end{array}}$$

Class Table Translation $\boxed{(\!| CT |\!) \coloneqq \overline{d_{\text{DOT}}}}$

$$(\!| \varnothing |\!) \coloneqq \varnothing$$

$$\frac{L_C = \textbf{class}\, C[\overline{X <: N}](\overline{f : U}) \ldots \quad \tau = [\overline{\mathsf{ctag}.X / |X|}]}{\begin{array}{c}(\!| \overline{L}, L_C |\!) \coloneqq (\!| \overline{L} |\!), C = \{\mathsf{this} \Rightarrow [\![C]\!]^{\overline{\bot..|N|}}\}, \\ \mathsf{new}_C(\mathsf{ctag} : |\overline{X <: N}|, \overline{f_{\mathsf{param}} : \tau|U|}) : \tau|C[\overline{X}]| = \{\mathsf{this} \Rightarrow (\!| C |\!)^{\overline{\tau|X|}}\}\end{array}}$$

Environment Translation $\boxed{|\Gamma| \coloneqq \Gamma_{\text{DOT}}}$

$$|\varnothing| \coloneqq \mathsf{ct} : [\![CT]\!]$$

$$|\Gamma, \overline{X_m <: N}| \coloneqq |\Gamma|, \mathsf{mtag} : |\overline{X_m <: N}|$$

$$|\Gamma, \overline{X_C <: N}| \coloneqq |\Gamma|, \mathsf{ctag} : |\overline{X_C <: N}|$$

$$\frac{\Gamma = \overline{X <: N} \quad \mathtt{getters}(C[\overline{T}]) = \overline{f : U} \quad \tau = [\overline{\mathsf{ctag}.X / |X|}]}{|\Gamma, \mathsf{this} : C[\overline{T}]| \coloneqq |\Gamma|, \overline{f_{\mathsf{param}} : \tau|U|}, \mathsf{this} : [\![C]\!]^{\overline{\tau|T|}}}$$

$$\frac{x \neq \mathsf{this}}{|\Gamma, x : T| \coloneqq |\Gamma|, x : |T|}$$

Arguments of Base Types $\boxed{\mathtt{baseArgs}(C) \coloneqq T_{\text{DOT}}}$

$$\mathtt{baseArgs}(\mathsf{Object}) \coloneqq \top$$

$$\frac{\textbf{class}\, C \ldots \lhd B[\overline{S}] \ldots \quad \textbf{class}\, B[\overline{X <: \ldots}] \ldots}{\mathtt{baseArgs}(C) \coloneqq \left(\bigwedge \overline{X = |S|}\right) \wedge \mathtt{baseArgs}(B)}$$

3.2.1 Meta-theory

Conjecture 3.2.1

Type-preserving compilation of FGJ to DOT with the original WF-Class definition from [Igarashi, Pierce, and Wadler 2001] requires at least one additional rule. A promising candidate is

$$\Gamma \vdash \{z \Rightarrow S\} \wedge \{z \Rightarrow T\} <: \{z \Rightarrow S \wedge T\} \qquad \text{(And-Bind)}$$

This rule is admissible in gDOT [1] so we conjecture that there exists a sound generalization of [Rompf and Amin 2016] which includes it.

Recall from Section 3.1 that the original definition of WF-Class allows class hierarchies involving cyclic references through type arguments. A well-founded translation scheme that can handle these hierarchies cannot blindly recurse on the type arguments of parent classes. Intuitively, it seems like we could get away with not translating these type arguments at all since the class translation $[\![C]\!]$ already handles them via `baseArgs`, so we might try using the following translation:

$$\frac{\textbf{class}\, C[\overline{X <: \ldots}] \ldots \lhd B[\overline{U}] \ldots}{|C[\overline{T}]| \;\textbf{:=}\; \boxed{\mathsf{ct}.B} \wedge \mathsf{ct}.C \wedge \{\mathsf{this} \Rightarrow \overline{X = |T|}\}}$$

But does subtyping preservation still hold? Consider the following class table:

```
class C[X] extends Object
class D[Y] extends C[Y]
```

Then the environment translation (as defined by Section 3.2) will be

$$\begin{aligned} &|\varnothing| = \mathsf{ct} : \\ &\quad (C = \{\mathsf{this} \Rightarrow X : \bot..\top\}) \wedge \\ &\quad (D = \{\mathsf{this} \Rightarrow X = \mathsf{this}.Y,\ Y : \bot..\top\}) \ \wedge \ldots \end{aligned}$$

And we should be able to establish that $|\varnothing| \vdash |D[\mathsf{Object}]| <: |C[\mathsf{Object}]|$:

$$\begin{array}{rl}
\dfrac{}{|\varnothing|, \mathsf{this} : \ldots \wedge (Y = \top) \vdash \mathsf{this} :_! (Y = \top)} & (\textsc{Sub}, \textsc{Var}) \\[2ex]
\dfrac{|\varnothing|, \mathsf{this} : \ldots \wedge (Y = \top) \vdash \mathsf{this}.Y >: \top}{|\varnothing|, \mathsf{this} : \ldots \wedge (Y = \top) \vdash (X = \mathsf{this}.Y) \wedge \ldots <: (X = \top)} & (\textsc{Sel2}) \\[2ex]
\dfrac{|\varnothing| \vdash \{\mathsf{this} \Rightarrow X = \mathsf{this}.Y,\ Y = \top\} <: \{\mathsf{this} \Rightarrow X = \top\}}{|\varnothing|, \mathsf{this} : \ldots \wedge (Y = \top) \vdash (X = \mathsf{this}.Y) \wedge \ldots <: (X = \top)} & (\textsc{Trans}, \textsc{Typ}) \\[2ex]
\dfrac{|\varnothing| \vdash \{\mathsf{this} \Rightarrow X = \mathsf{this}.Y\} \wedge \{\mathsf{this} \Rightarrow Y = \top\} <: \{\mathsf{this} \Rightarrow X = \top\}}{|\varnothing| \vdash \{\mathsf{this} \Rightarrow X = \mathsf{this}.Y,\ Y = \top\} <: \{\mathsf{this} \Rightarrow X = \top\}} & (\textsc{BindX}) \\[2ex]
\dfrac{|\varnothing| \vdash \mathsf{ct}.D \wedge \{\mathsf{this} \Rightarrow Y = \top\} <: \{\mathsf{this} \Rightarrow X = \top\}}{|\varnothing| \vdash \{\mathsf{this} \Rightarrow X = \mathsf{this}.Y\} \wedge \{\mathsf{this} \Rightarrow Y = \top\} <: \{\mathsf{this} \Rightarrow X = \top\}} & (\textsc{Trans}, \textsc{And-Bind}) \\[2ex]
\dfrac{|\varnothing| \vdash \mathsf{ct}.C \wedge \mathsf{ct}.D \wedge \{\mathsf{this} \Rightarrow Y = \top\} <: \mathsf{ct}.C \wedge \{\mathsf{this} \Rightarrow X = \top\}}{|\varnothing| \vdash \mathsf{ct}.D \wedge \{\mathsf{this} \Rightarrow Y = \top\} <: \{\mathsf{this} \Rightarrow X = \top\}} & (\textsc{Trans}, \textsc{Sel1}) \\[2ex]
 & (1.3.6)
\end{array}$$

[1] https://github.com/Blaisorblade/dot-iris/pull/397

Notice that without And-Bind we would not be able to unpack the recursive types via BindX since it only works for recursive types at the top-level. The type translation scheme defined in Section 3.2 side-steps this issue by emitting $|D[\top]| = \ldots \wedge (X = \top) \wedge \ldots$ which works but requires restricting WF-Class as discussed previously.

Theorem 3.2.2: Well-formedness preservation up to substitution

Suppose $\mathsf{ct} \in \mathsf{dom}(\Delta)$. If $\Gamma_1(X) = N$ implies $\Delta \vdash \tau|X| <: \tau|N|$, then $\Gamma_1 \vdash S$ wf implies $\Delta \vdash \tau|S|$ wf.

Proof. By induction on the derivation of $\Gamma_1 \vdash S$ wf.

Case $\Gamma \vdash \mathsf{Object}$ wf (WF-Object)

$|\mathsf{Object}| = \top \notin \mathsf{dom}(\tau)$ so $\tau|\mathsf{Object}| = \top$ which is always well-formed.

Case $\dfrac{Z \in \mathsf{dom}(\Gamma_1)}{\Gamma_1 \vdash Z \text{ wf}}$ (WF-Var)

We have $\Gamma_1(Z) = N$ for some N, therefore $\Delta \vdash \tau|Z| <: \tau|N|$. This implies that $\Delta \vdash \tau|Z|$ wf since DOT subtyping rules only apply to well-formed types.

Case $\dfrac{\textbf{class}\, C[\overline{X_C <: N}] \lhd P \ldots \quad \sigma = [\overline{T/X_C}] \quad \Gamma_1 \vdash \overline{T}, \sigma P \text{ wf} \quad \Gamma_1 \vdash \overline{T <: \sigma N}}{\Gamma_1 \vdash C[\overline{T}] \text{ wf}}$ (WF-Class)

By definition, $\tau|C[\overline{T}]| = \tau|\sigma P| \wedge \mathsf{ct}.C \wedge \bigwedge \overline{X_C = \tau|T|}$. By the IH, $\Delta \vdash \tau|\sigma P|$, $\overline{\tau|T|}$ wf and we know that $\mathsf{ct} \in \mathsf{dom}(\Delta)$, therefore $\Delta \vdash \mathsf{ct}.C$ wf which completes the case.

■

Theorem 3.2.3: Subtyping preservation up to substitution

Suppose $\mathsf{ct} \in \mathsf{dom}(\Delta)$ and Γ_1 wf. If $\Gamma_1(X) = N$ implies $\Delta \vdash \tau|X| <: \tau|N|$, then $\Gamma_1 \vdash S$ wf, $S <: T$ implies $\Delta \vdash \tau|S| <: \tau|T|$.

Proof. By induction on the derivation of $\Gamma_1 \vdash S <: T$.

Case $\Gamma_1 \vdash S <: S$ (GS-Refl)

By Theorem 3.2.2, $\Delta \vdash \tau|S|$ wf and Refl finishes the case.

Case $\dfrac{\Gamma_1(Z) = P}{\Gamma_1 \vdash Z <: P}$ (GS-Var)

Immediate.

Case $\dfrac{\textbf{class}\, C[\overline{X_C <: N}] \ldots \lhd P \ldots \quad \sigma = [\overline{T/X_C}]}{\Gamma_1 \vdash C[\overline{T}] <: \sigma P}$ (GS-Class)

By Theorem 3.2.2, $\Delta \vdash \tau|C[\overline{T}]|$ wf and by definition, $\tau|C[\overline{T}]| = \tau|\sigma P| \wedge \mathsf{ct}.C \wedge \bigwedge\overline{X_C = \tau|T|}$ so width subtyping finishes the case.

Case $\dfrac{\Gamma_1 \vdash S <: U \quad \Gamma_1 \vdash U <: T}{\Gamma_1 \vdash S <: T}$ (GS-Trans)

By the IH, $\Delta \vdash \tau|S| <: \tau|U|$. By Lemma 3.1.1, $\Gamma_1 \vdash U$ wf, so by the IH again $\Delta \vdash \tau|U| <: \tau|T|$ and Trans finishes the case.

■

Corollary 3.2.4: Well-formedness and subtyping preservation

Given $\textbf{class}\, C[X <: N] \ldots$, $\Gamma = (\overline{X <: N}, \mathsf{this} : C[\overline{X}], \overline{Y <: P}, \overline{x : U})$, Γ wf then

1. $\Gamma \vdash S$ wf implies $|\Gamma| \vdash |S|$ wf.
2. $\Gamma \vdash S$ wf, $S <: T$ implies $|\Gamma| \vdash |S| <: |T|$ wf.

Proof. This follows by Theorems 3.2.2 and 3.2.3 if we can show that $\Gamma(Z) = Q$ implies $|\Gamma| \vdash |Z| <: |Q|$. We need to distinguish two cases:

Case $(Z = X_i \in \overline{X})$

Let $\Delta_1 = |\Gamma|_{[\mathsf{ctag}]}$, $\Delta_2 = |\Gamma|_{[\mathsf{this}]}$, $\tau = [\overline{\mathsf{ctag}.X_i/|X_i|}]$.

$$
\begin{array}{c}
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Delta_1 \vdash \mathsf{ctag} :_! |\overline{X <: N}|}\,(\textsc{Var})}{\Delta_1 \vdash \mathsf{ctag} :_! (X_i : \bot..\tau|N_i|)}\,(\textsc{Sub, VarUnpack})}{\Delta_2 \vdash \mathsf{ctag}.X_i <: \tau|N_i|}\,(\textsc{Sel1}) \quad \dfrac{\dfrac{\dfrac{}{\Delta_2 \vdash \mathsf{this} :_! [\![C]\!]^{\overline{\mathsf{ctag}.X}}}\,(\textsc{Var})}{\Delta_2 \vdash \mathsf{this} :_! (X_i : \mathsf{ctag}.X_i)}\,(1.3.6)}{\Delta_2 \vdash \mathsf{ctag}.X_i =:= |X_i|}\,(\textsc{Sel1, Sel2})}{\Delta_2 \vdash \mathsf{ctag}.X_i <: |N_i|}\,(1.3.8)}{\Delta_2 \vdash (X_i = \mathsf{ctag}.X_i) <: (X_i : \bot..|N_i|)}\,(\textsc{Typ})}{\Delta_2 \vdash [\![C]\!]^{\overline{\mathsf{ctag}.X}} <: (X_i : \bot..|N_i|)}\,(\textsc{Trans}, 1.3.6)}{\Delta_2 \vdash \mathsf{this} :_! (X_i : \bot..|N_i|)}\,(\textsc{Sub, Var})}{|\Gamma| \vdash |X_i| <: |N_i|}\,(\textsc{Sel1})
\end{array}
$$

Case $(Z = Y_i \in \overline{Y})$

$$
\begin{array}{c}
\dfrac{\dfrac{\dfrac{}{|\Gamma|_{[\mathsf{mtag}]} \vdash \mathsf{mtag} :_! |\overline{Y <: P}|}\,(\textsc{Var})}{|\Gamma|_{[\mathsf{mtag}]} \vdash \mathsf{mtag} :_! (Y_i : \bot..|P_i|)}\,(\textsc{Sub, VarUnpack})}{|\Gamma| \vdash |Y_i| <: |P_i|}\,(\textsc{Sel1})
\end{array}
$$

■

Theorem 3.2.5: Substitution preservation

If $\Gamma \vdash T$ wf then $|\tau T| = |\tau||T|$

Proof. By induction on the derivation of $\Gamma \vdash T$ wf.

Case $\Gamma \vdash \mathsf{Object}$ wf (WF-Object)

Syntactically $\tau = [\overline{\ldots/X}]$, so $\tau\mathsf{Object} = \mathsf{Object}$ and this is trivial.

Case $\Gamma \vdash X$ wf (WF-Var)

If $X \notin \mathsf{dom}(\tau)$ this is trivial, otherwise $\tau = [\ldots, S/X, \ \ldots]$ and $|\tau| = [\ldots, |S|/|X|, \ \ldots]$ for some S. Hence, $|\tau X| = |S| = |\tau||X|$.

Case $$\frac{\textbf{class}\, C[\overline{X <: N}] \lhd P \ldots \quad \sigma = [\overline{U/X}] \quad \Gamma \vdash \overline{U}, \sigma P \text{ wf} \quad \Gamma \vdash \overline{U <: \sigma N}}{\Gamma \vdash C[\overline{U}] \text{ wf}} \text{ (WF-Class)}$$

By definition,

$$|\tau T| = |C[\overline{\tau U}]| \quad = |(\tau\sigma)P| \wedge \mathsf{ct}.C \wedge \bigwedge \overline{X = |\tau U|}$$
$$|\tau||T| = |\tau||C[\overline{U}]| = |\tau||\sigma P| \wedge ct.C \wedge \bigwedge \overline{X = |\tau||U|}$$

By the IH, $\overline{|\tau||U| = |\tau U|}$ and $|\tau||\sigma P| = |\tau(\sigma P)|$. By GT-Class, $\mathsf{fv}(P) \subseteq \mathsf{dom}(\sigma)$ so by Lemma 3.1.3, $|\tau(\sigma P)| = |(\tau\sigma)P|$ which lets us finish the case.

■

Lemma 3.2.6

If $\Gamma \vdash T$ wf and $\mathsf{fv}(T) \subseteq \mathsf{dom}(\sigma)$ and $\tau = [\overline{\ldots/|Z|}]$ then $(\tau|\sigma|)|T| = \tau|\sigma T|$

Proof. By induction on the derivation of $\Gamma \vdash T$ wf, case WF-Object is trivial.

Case $\Gamma \vdash X$ wf (WF-Var)

In this case $\mathsf{fv}(T) = \{X\} \subseteq \mathsf{dom}(\sigma)$, therefore $\sigma = [\ldots, S/X, \ \ldots]$ for some S and by definition $\tau|\sigma| = [\ldots, (\tau|S|)/|X|, \ \ldots]$. Hence, $(\tau|\sigma|)|X| = \tau|S| = \tau|\sigma X|$.

Case $$\frac{\textbf{class}\, C[\overline{X <: N}] \lhd P \ldots \quad \sigma_2 = [\overline{U/X}] \quad \Gamma \vdash \overline{U}, \sigma_2 P \text{ wf} \quad \Gamma \vdash \overline{U <: \sigma N}}{\Gamma \vdash C[\overline{U}] \text{ wf}} \text{ (WF-Class)}$$

By definition,

$$(\tau|\sigma|)|C[\overline{U}]| = (\tau|\sigma|)|\sigma_2 P| \wedge \mathsf{ct}.C \wedge \bigwedge \overline{X = (\tau|\sigma|)|U|}$$
$$\tau(|\sigma C[\overline{U}]|) = \tau(|C[\sigma\overline{U}]|) = \tau|(\sigma\sigma_2)P| \wedge \mathsf{ct}.C \wedge \bigwedge \overline{X = \tau|\sigma U|}$$

By definition of fv, $\mathsf{fv}(U) \subseteq \mathsf{fv}(T) \subseteq \mathsf{dom}(\sigma)$ so by the IH $\overline{(\tau|\sigma|)|U| = \tau|\sigma U|}$. By GT-Class, $\mathsf{fv}(P) \subseteq \mathsf{dom}(\sigma_2)$ so $\mathsf{fv}(\sigma_2 P) \subseteq \mathsf{fv}(U) \subseteq \mathsf{dom}(\sigma)$ and by the IH $(\tau|\sigma|)|\sigma_2 P| = \tau|\sigma(\sigma_2 P)|$. Finally, by Lemma 3.1.3, $\tau|\sigma(\sigma_2 P)| = \tau|(\sigma\sigma_2)P|$.

■

Lemma 3.2.7: Class translation preserves getters

Given **class** $C[\overline{X_C <: N}] \ldots, \Gamma = (\overline{X_C <: N}, \mathsf{this} : C[\overline{X_C}], x : T), \Gamma$ wf, then $\mathtt{getters}_\Gamma(T) = \overline{f : U}$ implies $|\Gamma| \vdash \overline{x :_{(!)} (f() : |U|)}$.

Proof. By induction on the derivation of $\mathtt{getters}_\Gamma(T)$, case G-Object is trivial.

Case $\mathtt{getters}_\Gamma(X) := \mathtt{getters}(\Gamma(X))$ (G-Var)

Let $\Delta = |\Gamma|(x \to |\Gamma|(X))$, then

$$\dfrac{\dfrac{}{|\Gamma| \vdash X <: \Gamma(X)}\,(\text{GS-Var}) \qquad \dfrac{}{\Delta \vdash \overline{x :_{(!)} (f() : |U|)}}\,(\text{IH})}{|\Gamma| \vdash \overline{x :_{(!)} (f() : |U|)}}\,(\text{NarrowTp})$$

Case $\dfrac{\textbf{class}\, D[\overline{X_D <: N_D}](\overline{f : U}) \ldots \quad \sigma = [\overline{S/X_D}]}{\mathtt{getters}(D[\overline{S}]) := \overline{f : \sigma U}}\,(\text{G-Class})$

Let $V = [\![\mathtt{getters}(D[\overline{X_D}])]\!]$, then

$$\dfrac{\dfrac{\dfrac{\dfrac{|\Gamma| \vdash x :_{(!)} |D[\overline{S}]|}{|\Gamma| \vdash x :_{(!)} \{\mathsf{this} \Rightarrow [\![D]\!]\}}\,(\text{Sub, Sel1})}{|\Gamma| \vdash x :_{(!)} \{\mathsf{this} \Rightarrow V\}}\,(\text{Sub, BindX})}{|\Gamma| \vdash x :_{(!)} [\overline{x.X_D/|X_D|}]V}\,(\text{VarUnpack}) \qquad \dfrac{|\Gamma| \vdash \overline{x :_{(!)} (X_D = |S|)}}{|\Gamma| \vdash \overline{|S| =:= x.X_D}}\,(\text{Sel1, Sel2})}{|\Gamma| \vdash x :_{(!)} \underbrace{[\overline{|S|/x.X_D}]\,[\overline{x.X_D/|X_D|}]}_{|\sigma|}\,V}\,(\text{Sub, 1.3.8})$$

By inspection, $\sigma\,\mathtt{getters}(D[\overline{X_D}]) = \mathtt{getters}(D[\overline{S}])$, so we can conclude by Theorem 3.2.5 that $|\sigma|V = [\![\mathtt{getters}(D[\overline{S}])]\!]$ and subsumption applied with width subtyping finish the case. ∎

Lemma 3.2.8

Given **class** $C[\overline{X_C <: N}] \ldots, \Gamma = (\overline{X_C <: N}, \mathsf{this} : C[\overline{X_C}], x : T), \Gamma$ wf, then $\mathtt{mtype}_\Gamma(m, T) = [\overline{Y <: P}] \to (\overline{y : U}) \to U_0$ implies $|\Gamma| \vdash x :_{(!)} (m(\mathsf{mtag} : |\overline{Y <: P}|, \overline{y : |U|}) : |U_0|)$.

Proof. By induction on the derivation of $\mathtt{mtype}_\Gamma(m, T)$. Cases M-Var and M-Super follow by the IH and NarrowTp as in case G-Var of Lemma 3.2.7.

Case $$\frac{\begin{array}{c}\textbf{class}\, D[\overline{X_D <: N_D}] \ldots \{\overline{M}\} \quad \sigma = [\overline{S/X}] \\ (\textbf{def}\, m[\overline{Y <: P'}](\overline{x : U'}) : U'_0 = \ldots) \in \overline{M}\end{array}}{\texttt{mtype}(m, D[\overline{S}]) \coloneqq [\overline{Y <: \sigma P'}] \to (\overline{x : \sigma U'}) \to \sigma U'_0)} \text{(M-Class)}$$

Let $V = [\![m]\!]_D$, by the same reasoning as case G-Class of Lemma 3.2.7 we find

$$|\Gamma| \vdash x :_{(!)} |\sigma|V$$

which finishes the case since $[\![m]\!]_D = (m(\texttt{mtag} : |\overline{Y <: P'}|, \overline{y : |U'|}) : |U'_0|)$ by definition. ∎

Lemma 3.2.9: Method translation preserves overriding relationship

Given $\textbf{class}\, B[\overline{X_B <: N_B}] \ldots$, $\textbf{class}\, C[\overline{X_C <: N_C}] \ldots \lhd B[\overline{U}]\, \{\overline{M}\}$, $\Gamma = (\overline{X_C <: N_C}, \text{this} : C[\overline{X_C}])$ and $\Delta \vdash \overline{|X_C| <: |N_C|}$, then $m \in \texttt{mnames}(B)$ implies $\Delta \vdash [\![m]\!]_C <: |\sigma|[\![m]\!]_B$ where $\sigma = [\overline{U/X_B}]$

Proof. By inspection, $\texttt{mtype}(m, B[\overline{U}]) = \sigma\, \texttt{mtype}(m, B[\overline{X_B}])$. If $\textbf{def}\, m \ldots \notin \overline{M}$ then by definition $\texttt{mtype}(m, C[\overline{X_C}]) = \texttt{mtype}(m, B[\overline{U}])$ and so $[\![m]\!]_C = |\sigma|[\![m]\!]_B$ by Theorem 3.2.5. Otherwise, let

$$\begin{aligned}\texttt{mtype}(m, C[\overline{X_C}]) &= [\overline{Y <: P}] \to (\overline{x : U}) \to U_0 \\ \texttt{mtype}(m, B[\overline{U}]) &= [\overline{Z <: Q}] \to (\overline{y : V}) \to V_0 \\ \Gamma_1 &= \Gamma, \overline{Y <: P} \\ \Delta_1 &= \Delta, \overline{|Y <: P|}\end{aligned}$$

Then $\vdash C$ ok implies $\Gamma \vdash m$ ok implies $\texttt{override}_\Gamma(m, C[\overline{X_C}], B[\overline{U}])$. By inversion, we must have $Y = Z$, $P = Q$, $\overline{x = y}$, $\overline{U = V}$ and $\Gamma_1 \vdash U_0 <: V_0$, hence

$$\dfrac{\dfrac{\Gamma_1 \vdash U_i <: U_o \quad \dfrac{\Delta \vdash \overline{|X| <: |N|}}{\Delta_1 \vdash \overline{|X| <: |N|}}\text{(Weaken)} \quad \dfrac{}{\Delta_1 \vdash \overline{|Y| <: |P|}}\text{(Sel1)}}{\Delta_1 \vdash |U_0| <: |V_0|}\text{(3.2.3)}}{\Delta \vdash [\![m]\!]_C <: |\sigma|[\![m]\!]_B}\text{(Fun', 3.2.5)}$$

∎

Lemma 3.2.10

Given $\textbf{class}\, B[\overline{X_B <: N_B}] \ldots$, $\textbf{class}\, C[\overline{X_C <: N_C}] \ldots \lhd B[\overline{U}]$ and $\Gamma = (\overline{Z <: Q}, \text{this} : C[\overline{T}])$, Γ wf, then $|\Gamma| \vdash [\![C]\!]^{\tau|\overline{T}|} <: [\![B]\!]^{\tau|\overline{\sigma_C U}|}$ where $\sigma_C = [\overline{T/X_B}]$ and $\tau = [\overline{\text{ctag}.Z/\text{this}.Z}]$.

Proof. By definition, we want to show:

$$\begin{aligned}&|\Gamma| \vdash \\ &[\![\texttt{getters}(C[\overline{X_C}])]\!] \wedge [\![\texttt{mnames}(C)]\!]_C \wedge \texttt{baseArgs}(C) \wedge \bigwedge(\overline{X_C = \tau|T|}) <: \\ &[\![\texttt{getters}(B[\overline{X_B}])]\!] \wedge [\![\texttt{mnames}(B)]\!]_B \wedge \texttt{baseArgs}(B) \wedge \bigwedge(\overline{X_B = \tau|\sigma_C U|})\end{aligned}$$

First, we note that $|\Gamma| \vdash \overline{|X_B| =:= |U|}$:

$$\dfrac{\dfrac{\dfrac{}{|\Gamma|_{[\mathsf{this}]} \vdash \mathsf{this} :_! \mathsf{baseArgs}(C)}\,(\textsc{Sub})}{|\Gamma|_{[\mathsf{this}]} \vdash \overline{\mathsf{this} :_! (X_B = U)}}\,(\textsc{Sub})}{|\Gamma| \vdash \overline{|X_B| =:= |U|}}\,(\textsc{Sel1}, \textsc{Sel2})$$

Let $\sigma_B = [\overline{U/X_B}]$, after proving the following claims, we can finish the case by width and depth subtyping.

Claim 1: $|\Gamma| \vdash [\![\mathtt{getters}(C[\overline{X_C}])]\!] <: [\![\mathtt{getters}(B[\overline{X_B}])]\!]$

> $\vdash C$ ok implies that $\mathtt{getters}(C[\overline{X_C}]) = ((\sigma_B\,\mathtt{getters}(B[\overline{X_B}])),\ \ldots)$ so by Theorem 3.2.5 and observation, $[\![\mathtt{getters}(C[\overline{X_C}])]\!] = |\sigma_B|[\![\mathtt{getters}(B[\overline{X_B}])]\!] \wedge T$ for some T, so the claim follows by width and depth subtyping.

Claim 2: $|\Gamma| \vdash [\![\mathtt{mnames}(C)]\!]_C <: [\![\mathtt{mnames}(B)]\!]_B$

> By definition, $\mathtt{mnames}(C) = (\mathtt{mnames}(B),\ \ldots)$ so $[\![\mathtt{mnames}(C)]\!]_C = [\![\mathtt{mnames}(B)]\!]_C \wedge T$ for some T and we only need to prove that $|\Gamma| \vdash [\![m]\!]_C <: [\![m]\!]_B$ for all $m \in \mathtt{mnames}(B)$. Note that by inversion of $\Gamma \vdash C[\overline{T}]$ wf we have $\Gamma \vdash \overline{T <: \sigma_C N_C}$.
>
> $$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{|\Gamma|_{[\mathsf{ctag}]} \vdash \mathsf{ctag} :_! \overline{|Z <: Q|}}\,(\textsc{Var})}{|\Gamma|_{[\mathsf{ctag}]} \vdash \mathsf{ctag} :_! \overline{\bigwedge(Z : \bot..\tau|Q|)}}\,(\textsc{VarUnpack})}{|\Gamma| \vdash \overline{\tau|Z| <: \tau|Q|}}\,(\textsc{Sel1}, \textsc{Sel2})}{|\Gamma| \vdash \overline{\tau|T| <: \tau|\sigma_C N_C|}}\,(3.2.3)}{|\Gamma| \vdash \overline{\tau|T| <: (\tau|\sigma_C|)|N_C|}}\,(3.2.6) \qquad \dfrac{|\Gamma|_{[\mathsf{this}]} \vdash \overline{\mathsf{this} :_! (X_C = \tau|T|)}}{|\Gamma| \vdash \overline{|X_C| =:= \tau|T|}}\,(\textsc{Sel1}, \textsc{Sel2})}{|\Gamma| \vdash \overline{|X_C| <: |N_C|}}\,(1.3.8)}{|\Gamma| \vdash [\![m]\!]_C <: |\sigma_B|[\![m]\!]_B}\,(3.2.9)}{|\Gamma| \vdash [\![m]\!]_C <: [\![m]\!]_B}\,(1.3.8)$$

Claim 3: $|\Gamma| \vdash \mathtt{baseArgs}(C) <: \mathtt{baseArgs}(B)$

> By width subtyping.

Claim 4: For all i in bounds, $|\Gamma| \vdash \mathtt{baseArgs}(C) <: (X_{Bi} = \tau|\sigma_C U_i|)$

> $\mathtt{baseArgs}(C) = \ldots \wedge \overline{X_{Bi} = |U_i|} \wedge \ldots$, therefore we only need to prove:
>
> $$\dfrac{\dfrac{\dfrac{\dfrac{|\Gamma|_{[\mathsf{this}]} \vdash \mathsf{this} :_! (X : \tau|T|..\tau|T|)}{|\Gamma| \vdash \tau|T| =:= |X|}\,(\textsc{Sel1}, \textsc{Sel2})}{|\Gamma| \vdash |U_i| =:= (\tau|\sigma_C|)|U_i|}\,(1.3.8)}{|\Gamma| \vdash |U_i| =:= \tau|\sigma_C U_i|}\,(3.2.6)}{|\Gamma| \vdash X_{Bi} = |U_i| <: X_{Bi} = \tau|\sigma_C U_i|}\,(\textsc{Typ})$$

■

Lemma 3.2.11

Given **class** $D[X_D <: N_D] \ldots$, $\Gamma = (\overline{X_D <: N_D}, \mathsf{this} : D[\overline{X_D}])$. If $\Gamma \vdash C[\overline{T}]$ wf where C is a class that may be equal to D, then $|\Gamma| \vdash \{\mathsf{this} \Rightarrow [\![C]\!]^{\tau|\overline{T}|}\} <: |C[\overline{T}]|$ where $\tau = [\overline{\mathsf{ctag}.X_D/|X_D|}]$.

Proof. Since $\Gamma \vdash C[\overline{T}]$ wf, we can have a sequence of type $\overline{S}$ such that $S_1 = C[\overline{T}]$, $S_n = \mathsf{Object}$ and $\Gamma \vdash S_i <: S_{i+1}$ derived by the rules GS-Refl and GS-Class for any i. We prove by induction on the length n (≥ 1) of the sequence.

Case $(n = 1)$

By Top.

Case **class** $C[\overline{X_C <: N_C}] \ldots \lhd B[\overline{U}] \ldots \quad (n \geq 2)$

Let $\overline{V} = \overline{\sigma U}$ where $\sigma = [\overline{T/X_C}]$. Then $|C[\overline{T}]| = |B[\overline{V}]| \wedge \mathsf{ct}.C \wedge \bigwedge \overline{X_C = |T|}$. After proving the following claims, And2 finishes the case.

Claim 1: $|\Gamma| \vdash \{\mathsf{this} \Rightarrow [\![C]\!]^{\tau|\overline{T}|}\} <: |B[\overline{V}]|$

By the IH, $|\Gamma| \vdash \{\mathsf{this} \Rightarrow [\![B]\!]^{\tau|\overline{V}|}\} <: |B[\overline{V}]|$, so by transitivity we only need to prove $|\Gamma| \vdash \{\mathsf{this} \Rightarrow [\![C]\!]^{\tau|\overline{T}|}\} <: \{\mathsf{this} \Rightarrow [\![B]\!]^{\tau|\overline{V}|}\}$ Let $\Gamma_1 = (\overline{X_D <: N_D}, \mathsf{this} : C[\overline{T}])$ and $|\Gamma_1| = (\Delta, \mathsf{this} : \ldots)$, it is easy to see that Δ is also a prefix of $|\Gamma|$.

$$\dfrac{\dfrac{\dfrac{}{|\Gamma_1| \vdash [\![C]\!]^{\tau|\overline{T}|} <: [\![B]\!]^{\tau|\overline{V}|}}\,(3.2.10)}{\Delta \vdash \{\mathsf{this} \Rightarrow [\![C]\!]^{\tau|\overline{T}|}\} <: \{\mathsf{this} \Rightarrow [\![B]\!]^{\tau|\overline{V}|}\}}\,(\textsc{BindX})}{|\Gamma| \vdash \{\mathsf{this} \Rightarrow [\![C]\!]^{\tau|\overline{T}|}\} <: \{\mathsf{this} \Rightarrow [\![B]\!]^{\tau|\overline{V}|}\}}\,(\textsc{Weaken})$$

Claim 2: $|\Gamma| \vdash \{\mathsf{this} \Rightarrow [\![C]\!]^{\tau|\overline{T}|}\} <: \mathsf{ct}.C$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{|\Gamma|_{[\mathsf{ct}]} \vdash \mathsf{ct} :_! [\![CT]\!]}\,(\textsc{Var}) \quad \dfrac{}{|\Gamma|_{[\mathsf{ct}]} \vdash [\![CT]\!] <: (C : (\{\mathsf{this} \Rightarrow [\![C]\!]\})..\top)}\,(1.3.6, \textsc{Typ})}{|\Gamma|_{[\mathsf{ct}]} \vdash \mathsf{ct} :_! (C : (\{\mathsf{this} \Rightarrow [\![C]\!]\})..\top)}\,(\textsc{Sub})}{|\Gamma| \vdash \{\mathsf{this} \Rightarrow [\![C]\!]\} <: \mathsf{ct}.C}\,(\textsc{Sel2})}{|\Gamma| \vdash \{\mathsf{this} \Rightarrow [\![C]\!]^{\tau|\overline{T}|}\} <: \mathsf{ct}.C}\,(\textsc{Trans}, \textsc{BindX})$$

Claim 3: $|\Gamma| \vdash \{\mathsf{this} \Rightarrow [\![C]\!]^{\tau|\overline{T}|}\} <: \bigwedge \overline{X_C = |T|}$

$$\dfrac{\dfrac{\dfrac{}{|\Gamma| \vdash \{\mathsf{this} \Rightarrow \overline{X = \tau|T|}\} <: \bigwedge \overline{X_C = \tau|T|}}\,(\textsc{Bind1})}{|\Gamma| \vdash \{\mathsf{this} \Rightarrow [\![C]\!]^{\tau|\overline{T}|}\} <: \bigwedge \overline{X_C = \tau|T|}}\,(\textsc{Trans}, \textsc{BindX}) \qquad |\Gamma| \vdash \overline{\mathsf{ctag}.X_D =:= |X_D|}}{|\Gamma| \vdash \{\mathsf{this} \Rightarrow [\![C]\!]^{\tau|\overline{T}|}\} <: \bigwedge \overline{X_C = |T|}}\,(1.3.8)$$

■

Lemma 3.2.12: this **translation is type-preserving**

Given **class** $C[\overline{X <: N}]$ ..., let $\Gamma = (\overline{X <: N}, \mathsf{this} : C[\overline{X}], \ ...)$ then $|\Gamma| \vdash \mathsf{this} : |C[\overline{X}]|$.

Proof. Let $\Delta = |\overline{X <: N}, \mathsf{this} : C[\overline{X}]|$, $\tau = [\overline{\mathsf{ctag}.X/|X|}]$.

$$\dfrac{\dfrac{\Delta \vdash \mathsf{this} : [\![C]\!]^{\overline{\tau|X|}}}{\Delta \vdash \mathsf{this} : \{\mathsf{this} \Rightarrow [\![C]\!]^{\overline{\tau|X|}}\}}\,(\textsc{VarPack}) \qquad \dfrac{}{\Delta \vdash \{\mathsf{this} \Rightarrow [\![C]\!]^{\overline{\tau|X|}}\} <: |C[\overline{X}]|}\,(3.2.11)}{|\Gamma| \vdash \mathsf{this} : |C[\overline{X}]|}\,(\textsc{Sub}, \textsc{Weaken})$$

■

Theorem 3.2.13: Typing translation is type-preserving

Given **class** $C[\overline{X <: N}]$ ..., $\Gamma = (\overline{X <: N}, \mathsf{this} : C[\overline{X}], \overline{Y <: P}, \overline{x : U})$, Γ wf then $|\Gamma| \vdash |e|_\Gamma : |T|$.

Proof. By induction on the derivation of $\Gamma \vdash e : T$.

Case $\dfrac{\Gamma(x) = T}{\Gamma \vdash x : T}\,(\text{GT-}\textsc{Var})$

By definition, $|\Gamma|(|x|) = |\Gamma|(x)$, and we can distinguish two sub-cases:

- If $x = \mathsf{this}$, then Lemma 3.2.12 finishes the case.
- Otherwise, $|\Gamma|(x) = |T|$ and $\textsc{Var}$ finishes the case.

Case $\dfrac{\Gamma \vdash e_0 : T_0 \quad \mathtt{getters}_\Gamma(T_0) = \overline{f : T}}{\Gamma \vdash e_0.f_i : T_i}\,(\text{GT-}\textsc{Getter})$

We have $|e_0.f_i|_\Gamma = (\textbf{let } x = |e_0|_\Gamma \textbf{ in } x.f_i())$. By the IH, $|\Gamma| \vdash |e_0|_\Gamma : |T_0|$. By Lemma 3.2.7, $|\Gamma|, x : |T_0| \vdash x : (f_i() : |T_i|)$. $\textsc{TApp}$ finishes the case.

Case $$\frac{\Gamma \vdash e_0 : T_0 \quad \mathtt{mtype}_\Gamma(m, T_0) = [\overline{Y <: P}] \to (\overline{y : U}) \to U_0 \qquad \sigma = [\overline{V/Y}] \quad \Gamma \vdash \overline{V}\ \mathsf{wf},\ \overline{V <: \sigma P},\ \overline{e : S},\ \overline{S <: \sigma U}}{\Gamma \vdash e_0.m[\overline{V}](\overline{e}) : \sigma U_0}\ (\text{GT-Invk})$$

We have $|e_0.m[\overline{V}](\overline{e})|_\Gamma = \textbf{let}\ x = |e_0|_\Gamma,\ x_{\mathsf{mtag}} = \{_ \Rightarrow \overline{Y = |V|}\}\ \textbf{in}\ x.m(x_{\mathsf{mtag}}, \overline{|e|_\Gamma})$. By Lemma 3.2.8 we have

$$|\Gamma| \vdash x : (m(\mathsf{mtag} : |\overline{Y <: P}|, \overline{y : |U|}) : |U_0|)$$

Let $\tau = [\overline{x_{\mathsf{mtag}}.Y/|Y|}]$ and $\Delta = |\Gamma|, x : |S|, x_{\mathsf{mtag}} : \{_ \Rightarrow \overline{Y = |V|}\}$. Note that $|\sigma| = [\overline{|V|/x_{\mathsf{mtag}}.Y}]\tau$ and that we can always weaken $|\Gamma|$ to Δ because x and x_{mtag} are fresh names, then

$$\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{}{\Delta \vdash \overline{|V| <: |\sigma P|}}\ (3.2.4)}{\Delta \vdash \overline{|V| <: |\sigma||P|}}\ (3.2.5)}{\Delta \vdash \overline{|V| <: \tau|P|}}\ (1.3.8)}{\Delta \vdash x_{\mathsf{mtag}} : (Y : \bot..\tau|P|)}\ (\text{Typ})}{\Delta \vdash x_{\mathsf{mtag}} : \{\mathsf{mtag} \Rightarrow \overline{Y : \bot..|P|}\}}\ (\text{VarPack})}{\Delta \vdash x.m(x_{\mathsf{mtag}}) : \tau(\mathsf{apply}(\overline{y : |U|}) : |U_0|)}\ (\text{TAppVar})}{\Delta \vdash x.m(x_{\mathsf{mtag}}) : |\sigma|(\mathsf{apply}(\overline{y : |U|}) : |U_0|)}\ (\text{Sub, 1.3.8})}{\Delta \vdash x.m(x_{\mathsf{mtag}}) : (\mathsf{apply}(\overline{y : |\sigma U|}) : |\sigma U_0|)}\ (3.2.5) \qquad \cfrac{}{\Delta \vdash \overline{|e|_\Gamma : |\sigma U|}}\ (\text{Sub, IH})}{\Delta \vdash x.m(x_{\mathsf{mtag}}, \overline{|e|_\Gamma}) : |\sigma U_0|}\ (\text{TApp'})$$

Case $$\frac{\Gamma \vdash C[\overline{T}]\ \mathsf{wf} \quad \mathtt{getters}(C[\overline{T}]) = \overline{f : U} \quad \Gamma \vdash \overline{e : S},\ \overline{S <: U}}{\Gamma \vdash \textbf{new}\, C[\overline{T}](\overline{e}) : C[\overline{T}]}\ (\text{GT-New})$$

We have $|\textbf{new}\, C[\overline{T}](\overline{e})|_\Gamma = (\textbf{let}\ x_{\mathsf{ctag}} = \{_ \Rightarrow \overline{X = |T|}\}\ \textbf{in}\ \mathsf{ct}.\mathsf{new}_C(x_{\mathsf{ctag}}, \overline{|e|_\Gamma}))$. Let $CT(C) = \textbf{class}\, C[\overline{X} <: ...]$ and $\mathtt{getters}(C[\overline{X}]) = \overline{f : U'}$, then by inspection $\overline{U = \sigma U'}$ where $\sigma = [\overline{T/X}]$. It is easy to see that

$$|\Gamma| \vdash \mathsf{ct} : (\mathsf{new}_C(\mathsf{ctag} : |\overline{X <: N}|, \overline{f_{\mathsf{param}} : \tau'|U'|}) : \tau'|C[\overline{X}]|)$$

where $\tau' = [\overline{\mathsf{ctag}.X/|X|}]$, and the rest of the case proceeds similarly to the previous case with $\Delta = |\Gamma|, x_{\mathsf{ctag}} : \{_ \Rightarrow \overline{X = |T|}\}$.

■

Lemma 3.2.14: Class translation is type-preserving

Given $\textbf{class}\, C[\overline{X <: N}]$, $\Gamma = (\overline{X <: N}, \mathsf{this} : C[\overline{X}])$, let $|\Gamma| = (\Delta, \mathsf{this} : ...)$. Then, $\Delta \vdash \{\mathsf{this} \Rightarrow (\!| C |\!)^{\overline{\tau|X|}}\} : \{\mathsf{this} \Rightarrow [\![C]\!]^{\overline{\tau|X|}}\}$ where $\tau = [\overline{\mathsf{ctag}.X/|X|}]$

Proof. By TNew, this is true if the following claims are all true.

Claim 1: $|\Gamma| \vdash (\!| f : U |\!) : [\![f : U]\!] \quad \forall (f : U) \in \texttt{getters}(C[\overline{X}])$

By VAR, we have $|\Gamma| \vdash \overline{f_{\mathsf{param}} : \tau|U|}$. By Lemma 1.3.8, $|\Gamma| \vdash \overline{f_{\mathsf{param}} : |U|}$ and DFUN finishes the claim.

Claim 2: $|\Gamma| \vdash (\!| m |\!)_C : [\![m]\!]_C \quad \forall m \in \texttt{mnames}(C)$

Let

$$\begin{aligned} \texttt{mtype}(m, C[\overline{X}]) &= [\overline{Y <: P}] \to (\overline{x : U}) \to U_0 \\ \texttt{mbody}(m, C[\overline{X}]) &= e_0 \\ \Gamma_m &= \Gamma, \overline{Y <: P}, \overline{x : U} \end{aligned}$$

Then by inversion of GT-METHOD, Γ_m wf, $\Gamma_m \vdash e_0 : E_0$, $E_0 <: U_0$ and

$$\dfrac{\dfrac{\dfrac{\Gamma_m \text{ wf} \quad \Gamma_m \vdash e_0 : E_0}{|\Gamma_m| \vdash |e_0|_{\Gamma_m} : |E_0|}\,(3.2.13)}{|\Gamma_m| \vdash |e_0|_{\Gamma_m} : |U_0|}\,(\text{SUB, } 3.2.4)}{|\Gamma| \vdash (\!| m |\!)_C : [\![m]\!]_C}\,(\text{DFUN'})$$

Claim 3: $(X_i = \mathsf{ctag}.X_i) : (X_i = \mathsf{ctag}.X_i) \quad \forall X_i \in \overline{X}$

By DTYP

■

Lemma 3.2.15: Class table translation is type-preserving

$\varnothing \vdash_{\text{DOT}} \{\mathsf{ct} \Rightarrow (\!| CT |\!)\} : \{\mathsf{ct} \Rightarrow [\![CT]\!]\}$.

Proof. After proving the following claims for each **class** $C[\overline{X <: N}](\overline{f : U})$ in CT, we can finish the proof by TNEW.

Claim 1: Let $T = \{\mathsf{this} \Rightarrow [\![C]\!]^{\overline{\bot..|N|}}\}$, then $|\varnothing| \vdash (C = T) : (C = T)$

By DTYP.

Claim 2: Let $\tau = [\overline{\mathsf{ctag}.X/|X|}]$, then

$$\begin{aligned} |\varnothing| \vdash (\mathsf{new}_C(\mathsf{ctag}, \overline{f_{\mathsf{param}}}) &= (\textbf{let } \mathsf{this} = \{\mathsf{this} \Rightarrow (\!| C[\overline{X}] |\!)\} \textbf{ in } \mathsf{this})) : \\ &(\mathsf{new}_C(\mathsf{ctag} : \{\mathsf{this} \Rightarrow \overline{X : \bot..|N|}\}, f_{\mathsf{param}} : \tau|U|) : \tau|C[\overline{X}]|) \end{aligned}$$

Let

$$\begin{aligned} \Gamma &= [\overline{X <: N}], \mathsf{this} : C[\overline{X}] \\ |\Gamma| &= \Delta_0, \mathsf{this} : [\![C]\!]^{\overline{\tau|X|}} \\ \Delta_1 &= \Delta_0, \mathsf{this} : \{\mathsf{this} \Rightarrow [\![C]\!]^{\overline{\tau|X|}}\} \end{aligned}$$

Then

$$\dfrac{\dfrac{}{\Delta_0 \vdash \{\mathsf{this} \Rightarrow (\!| C |\!)^{\overline{\tau|X|}}\} : \{\mathsf{this} \Rightarrow [\![C]\!]^{\overline{\tau|X|}}\}}\,(3.2.14) \qquad \dfrac{\dfrac{\dfrac{}{|\Gamma| \vdash \mathsf{this} : |C[\overline{X}]|}\,(3.2.12)}{\Delta_1 \vdash \mathsf{this} : |C[\overline{X}]|}\,(\textsc{EnvPackTp})}{\Delta_1 \vdash \mathsf{this} : \tau|C[\overline{X}]|}\,(\textsc{Sub},\ 1.3.8)}{\Delta_0 \vdash (\textbf{let}\ \mathsf{this} = \{\mathsf{this} \Rightarrow (\!| C[\overline{X}] |\!)\}\ \textbf{in}\ \mathsf{this}) : \tau|C[\overline{X}]|}\,(\textsc{Let})$$

and DFun' finishes the claim.

■

Theorem 3.2.16: Program translation is type-preserving

If $\varnothing \vdash_{\text{FGJ}} T\ \mathsf{wf}$ and $\varnothing \vdash_{\text{FGJ}} e : T$ then $\varnothing \vdash_{\text{DOT}} \textbf{let}\ \mathsf{ct} = \{\mathsf{ct} \Rightarrow (\!| CT |\!)\}\ \textbf{in}\ |e|_{\varnothing} : |T|$.

Proof.

$$\dfrac{\dfrac{}{\varnothing \vdash \{\mathsf{ct} \Rightarrow (\!| CT |\!)\} : \{\mathsf{ct} \Rightarrow [\![CT]\!]\}}\,(3.2.15) \qquad \dfrac{\dfrac{\varnothing \vdash T\ \mathsf{wf} \quad \varnothing \vdash e : T}{\mathsf{ct} : [\![CT]\!] \vdash |e|_{\varnothing} : |T|}\,(3.2.13)}{\mathsf{ct} : \{\mathsf{ct} \Rightarrow [\![CT]\!]\} \vdash |e|_{\varnothing} : |T|}\,(\textsc{EnvPackTp})}{\varnothing \vdash \textbf{let}\ \mathsf{ct} = \{\mathsf{ct} \Rightarrow (\!| CT |\!)\}\ \textbf{in}\ |e|_{\varnothing} : |T|}\,(\textsc{Let})$$

■

Bibliography

Amin, Nada, Samuel Grütter, Martin Odersky, Tiark Rompf, and Sandro Stucki (Mar. 2016). "The Essence of Dependent Object Types". In: *A List of Successes That Can Change the World: Essays Dedicated to Philip Wadler on the Occasion of His 60th Birthday*. Cham, Switzerland: Springer, pp. 249–272. ISBN: 978-3-319-30935-4. DOI: 10.1007/978-3-319-30936-1_14.

Giarrusso, Paolo G., Léo Stefanesco, Amin Timany, Lars Birkedal, and Robbert Krebbers (Aug. 2020). "Scala step-by-step: soundness for DOT with step-indexed logical relations in Iris". In: *Proc. ACM Program. Lang.* 4.ICFP, pp. 1–29. ISSN: 2475-1421. DOI: 10.1145/3408996.

Igarashi, Atsushi, Benjamin C. Pierce, and Philip Wadler (May 2001). "Featherweight Java: a minimal core calculus for Java and GJ". In: *ACM Trans. Program. Lang. Syst.* 23.3, pp. 396–450. ISSN: 0164-0925. DOI: 10.1145/503502.503505.

Martres, Guillaume (Oct. 2021). "Pathless Scala: a calculus for the rest of Scala". In: *SCALA 2021: Proceedings of the 12th ACM SIGPLAN International Symposium on Scala*. New York, NY, USA: Association for Computing Machinery, pp. 12–21. ISBN: 978-1-45039113-9. DOI: 10.1145/3486610.3486894.

Rapoport, Marianna and Ondřej Lhoták (Oct. 2019). "A path to DOT: formalizing fully path-dependent types". In: *Proceedings of the ACM on Programming Languages* 3, pp. 1–29. ISSN: 2475-1421. DOI: 10.1145/3360571.

Rompf, Tiark and Nada Amin (Oct. 2016). "Type soundness for dependent object types (DOT)". In: *SIGPLAN Not.* 51.10, pp. 624–641. ISSN: 0362-1340. DOI: 10.1145/3022671.2984008.