Abstract

Notes on a formal semantics for GFX.

1. Semantics

We now describe the semantics of languages in the FX family.

Each language $\mathcal{L}$ in the family is defined over a given input constraint system $\mathcal{X}$. Given a program P, we now show how to build a larger constraint system $\mathcal{O}(\mathcal{X})$ on top of $\mathcal{X}$ which captures constraints related to the object-oriented structure of P. $\mathcal{O}$ is sensitive to $\mathcal{X}$ only in that $\mathcal{O}$ depends on the types defined by $\mathcal{L}$, and these may depend on $\mathcal{X}$.

The static and dynamic semantics of $\mathcal{L}$ rests on $\mathcal{O}(\mathcal{X})$.

1.1 The Object constraint system, $\mathcal{O}$

Given $\mathcal{L}$, its input constraint system $\mathcal{X}$ we now show how to define $\mathcal{O}$. The inference relation for $\mathcal{O}$ depends on the object-oriented structure of the input program P in $\mathcal{L}$. For some members of $\mathcal{L}$, viz. the generic languages, $\mathcal{X}$ itself may use some of the constraints defined by $\mathcal{O}$. Thus we should think of $\mathcal{X}$ and $\mathcal{O}$ as being defined simultaneously and recursively.

The constraints of $\mathcal{O}$ are given by:

$$\begin{array}{rlll} \text{(Member)} & \mathtt{M} & ::= & \mathtt{m}(\overline{\mathtt{x}}:\overline{\mathtt{V}})\{\mathtt{c}\}:\mathtt{T}=\mathtt{e} \mid \mathtt{f}:\mathtt{V} \\ \text{(Const.)} & \mathtt{c,d} & ::= & \mathtt{S} \trianglelefteq \mathtt{T} \mid \mathtt{S} <: \mathtt{T} \\ & & & \mid \mathtt{fields(x)} = \overline{\mathtt{f}}:\overline{\mathtt{V}} \\ & & & \mid \mathtt{x\ has\ M} \end{array}$$

Intuitively, $\mathtt{S} \trianglelefteq \mathtt{T}$ is intended to hold if it can be established that S is extends T. For a variable x, fields(x) is intended to specify the (complete) set of typed fields available to x. x has M is intended to specify that the member M (field or method) is availabele to x.

$\mathcal{O}$ satisfies the following axioms and inference rules:

$$\frac{\mathtt{class\ C}(\overline{\mathtt{f}}:\overline{\mathtt{V}})\ \mathtt{extends\ D}\ \ldots \in \mathtt{P}}{\vdash_{\mathcal{O}} \mathtt{class(C)}, \mathtt{C} \trianglelefteq \mathtt{D}} \quad \text{(CLASS)}$$

$$\vdash_{\mathcal{O}} T \trianglelefteq T \quad \text{(V-ID)}$$

$$\frac{\Gamma \vdash_{\mathcal{O}} \mathtt{T}_1 <: \mathtt{T}_2, \mathtt{T}_2 <: \mathtt{T}_3}{\Gamma \vdash_{\mathcal{O}} \mathtt{T}_1 <: \mathtt{T}_3} \quad \text{(S-TRANS)}$$

$$\frac{\Gamma \vdash_{\mathcal{O}} \mathtt{t} \trianglelefteq \mathtt{T}}{\Gamma \vdash_{\mathcal{O}} \mathtt{t} <: \mathtt{T}} \quad \text{(Sub-X)}$$

$$x:\mathtt{Object} \vdash_{\mathcal{O}} \mathtt{fields(x)} = \bullet \quad \text{(FIELDS-B)}$$

$$\frac{\Gamma, \mathtt{x}:\mathtt{D} \vdash_{\mathcal{O}} \mathtt{fields(x)} = \overline{\mathtt{g}}:\overline{\mathtt{V}} \qquad \mathtt{class\ C}(\overline{\mathtt{f}}:\overline{\mathtt{U}})\{\mathtt{c}\}\ \mathtt{extends\ D}\{\overline{\mathtt{M}}\} \in \mathtt{C}}{\Gamma, \mathtt{x}:\mathtt{C} \vdash \mathtt{fields(x)} = \overline{\mathtt{g}}:\overline{\mathtt{V}}, \overline{\mathtt{f}}:\overline{\mathtt{U}}[\mathtt{x/this}]} \quad \text{(FIELDS-I)}$$

$$\frac{\Gamma, \mathtt{x}:\mathtt{S} \vdash_{\mathcal{O}} \mathtt{fields(x)} = \overline{\mathtt{f}}:\overline{\mathtt{V}}}{\begin{array}{c}\Gamma, \mathtt{x}:\mathtt{S}\{\mathtt{d}\} \vdash_{\mathcal{O}} \mathtt{fields(x)} = \overline{\mathtt{f}}:\overline{\mathtt{V}\{\mathtt{d}[\mathtt{x/self}]\}} \\ \Gamma, \mathtt{x}:(\mathtt{y}:\mathtt{U};\mathtt{S}) \vdash_{\mathcal{O}} \mathtt{fields(x)} = \overline{\mathtt{f}}:(\mathtt{y}:\mathtt{U};\mathtt{V})\end{array}} \quad \text{(FIELDS-C,E)}$$

$$\frac{\Gamma \vdash_{\mathcal{O}} \mathtt{fields(x)} = \overline{\mathtt{f}}:\overline{\mathtt{V}}}{\Gamma \vdash_{\mathcal{O}} \mathtt{x\ has\ f_i}:\mathtt{V_i}} \quad \text{(has-F)}$$

$$\vdash_{\mathcal{O}} \mathtt{new\ D}(\overline{\mathtt{t}}).\mathtt{f_i} = \mathtt{t_i} \quad \text{(SEL)}$$

$$\frac{\Gamma, \mathtt{x}:\mathtt{C} \vdash_{\mathcal{O}} \mathtt{class(C)} \qquad \theta = [\mathtt{x/this}] \qquad \mathtt{def\ m}(\overline{\mathtt{z}}:\overline{\mathtt{V}})\{\mathtt{c}\}:\mathtt{T} = \mathtt{e} \in P}{\Gamma, \mathtt{x}:\mathtt{C} \vdash_{\mathcal{O}} \mathtt{x\ has}\ (\mathtt{m}(\overline{\mathtt{z}}:\overline{\mathtt{V}\theta})\{\mathtt{c}\theta\}:\mathtt{T}\theta = \mathtt{e})} \quad \text{(METHOD-B)}$$

$$\frac{\Gamma, \mathtt{x}:\mathtt{D} \vdash_{\mathcal{O}} \mathtt{x\ has}\ m(\overline{\mathtt{z}}:\overline{\mathtt{V}})\{c\}:T = e \qquad \mathtt{class\ C}(\ldots)\ \mathtt{extends\ D}\{\overline{\mathtt{M}}\} \qquad \mathtt{m} \notin \overline{\mathtt{M}}}{\Gamma, \mathtt{x}:\mathtt{C} \vdash_{\mathcal{O}} \mathtt{x\ has}\ m(\overline{\mathtt{z}}:\overline{\mathtt{V}})\{c\}:T = e} \quad \text{(METHOD-I)}$$

$$\frac{\Gamma, \mathtt{x}:\mathtt{S} \vdash_{\mathcal{O}} \mathtt{x\ has\ m}(\overline{\mathtt{z}}:\overline{\mathtt{V}})\{\mathtt{c}\}:\mathtt{T} = \mathtt{e}}{\begin{array}{c}\Gamma, \mathtt{x}:\mathtt{S}\{\mathtt{d}\} \vdash_{\mathcal{O}} \mathtt{x\ has\ m}(\overline{\mathtt{z}}:\overline{\mathtt{V}})\{\mathtt{c}\}:\mathtt{T}\{\mathtt{d}[\mathtt{x/self}]\} = \mathtt{e} \\ \Gamma, \mathtt{x}:(\mathtt{y}:\mathtt{U};\mathtt{S}) \vdash_{\mathcal{O}} \mathtt{x\ has\ m}(\overline{\mathtt{z}}:\overline{\mathtt{V}})\{\mathtt{c}\}:(\mathtt{y}:\mathtt{U};\mathtt{T}) = \mathtt{e}\end{array}} \quad \text{(METHOD-C,E)}$$

Note: Figure out whether consistency checks need to be added.

1.2 Extensions of FX

1.2.1 FX(G)

Generics as in FGJ are supported by adding the following productions:

$$\begin{array}{rlll} \text{(Par Type)} & \mathtt{V} & ::= & \mathtt{type} \\ \text{(Path)} & \mathtt{p} & ::= & \mathtt{x} \mid \mathtt{p.f} \\ \text{(Type)} & \mathtt{T} & ::= & \mathtt{X} \mid \mathtt{p} \\ \text{(C Term)} & \mathtt{t} & ::= & \mathtt{T} \\ \text{(Const.)} & \mathtt{c} & ::= & \mathtt{t} \trianglelefteq \mathtt{N} \mid \mathtt{t{=}{=}t} \end{array}$$

and the following rule:

$$\frac{\Gamma \vdash \mathtt{p}:\mathtt{T} \qquad \Gamma, \mathtt{x}:\mathtt{T} \vdash \mathtt{x\ has\ X:\ type}}{\Gamma \vdash \mathtt{p.X\ type}} \quad \text{(PATH)}$$

$$\Gamma, \mathtt{X}:\mathtt{type} \vdash \mathtt{X\ type} \quad \text{(Type Var)}$$

$$\frac{\Gamma \vdash_{\mathcal{O}} \mathtt{p} \trianglelefteq \mathtt{T} \qquad \Gamma, \mathtt{x}:\mathtt{T} \vdash_{\mathcal{O}} \mathtt{x\ has\ M}}{\Gamma, \mathtt{x}:\mathtt{p} \vdash_{\mathcal{O}} \mathtt{x\ has\ M}} \quad \text{(Inh-p)}$$

$$\frac{\Gamma \vdash_{\mathcal{O}} \mathtt{X} \trianglelefteq \mathtt{T} \qquad \Gamma, \mathtt{x}:\mathtt{T} \vdash_{\mathcal{O}} \mathtt{x\ has\ M}}{\Gamma, \mathtt{x}:\mathtt{X} \vdash_{\mathcal{O}} \mathtt{x\ has\ M}} \quad \text{(Inh-X)}$$

1.2.2 FX(D($\mathcal{A}$))

Only the following rules are needed; no additional rules are needed for sub-typing, type-equivalence, expression typing or dynamic semantics. Below, q (f) ranges over all predicates (functions) specified by $\mathcal{C}$:

$$\begin{array}{rlll} \text{(Type)} & \mathtt{T} & ::= & \text{new base types, e.g. } \mathtt{int}, \mathtt{boolean} \\ \text{(C Term)} & \mathtt{t} & ::= & \mathtt{x} \mid \mathtt{f}(\overline{\mathtt{t}}) \mid \mathrm{t.f} \mid \mathtt{new\ C}(\overline{\mathtt{t}}) \\ \text{(Const.)} & \mathtt{c} & ::= & \mathtt{t{=}{=}t} \mid \mathtt{q}(\overline{\mathtt{t}}) \end{array}$$

We need the following rules:

$$\frac{\mathtt{p}(\overline{\mathtt{T}}):\mathtt{o} \in \mathcal{C} \qquad \Gamma \vdash \overline{\mathtt{t}}:\overline{\mathtt{T}}}{\Gamma \vdash \mathtt{p}(\overline{\mathtt{T}}):\mathtt{o}} \quad \text{(PRED)}$$

$$\frac{\mathtt{f}(\overline{\mathtt{T}}):\mathtt{T} \in \mathcal{C} \qquad \Gamma \vdash \overline{\mathtt{t}}:\overline{\mathtt{T}}}{\Gamma \vdash \mathtt{f}(\overline{\mathtt{T}}):\mathtt{T}} \quad \text{(FUN)}$$

$$\frac{\begin{array}{c}\Gamma \vdash \mathtt{t_0}:\mathtt{T_0} \qquad \Gamma \vdash \mathtt{t_1}:\mathtt{T_1} \\ (\Gamma \vdash \mathtt{T_0} <: \mathtt{T_1} \vee \Gamma \vdash \mathtt{T_1} <: \mathtt{T_0})\end{array}}{\Gamma \vdash \mathtt{t_0} = \mathtt{t_1}:\mathtt{o}} \quad \text{(EQUALS)}$$

1.2.3 FX(G,D($\mathcal{A}$),P)

Only the following rules are needed beyond those of FX(G) and FX(D($\mathcal{A}$)).

1.3 Judgements

In the following Γ is a *well-typed context*, i.e. a (finite, possibly empty) sequence of formulas $\mathtt{x}:\mathtt{T}$, T type and constraints c satisfying:

1. for any formula ϕ in the sequence all variables x (X) occuring in ϕ are defined by a declaration $\mathtt{x}:\mathtt{T}$ (X type) in the sequence to the left of ϕ.

FX **productions:**

$$
\begin{array}{rlll}
\text{(Class)} & \mathtt{L} & ::= & \mathtt{class}\ \mathtt{C}(\overline{\mathtt{f}}{:}\overline{\mathtt{V}})\{\mathtt{c}\}\ \mathtt{extends}\ \mathtt{N}\ \{\overline{\mathtt{M}}\} \\
\text{(Method)} & \mathtt{M} & ::= & \mathtt{def}\ \mathtt{m}(\overline{\mathtt{x}} : \overline{\mathtt{V}})\{\mathtt{c}\}\text{:V=e;} \\
\text{(Exp.)} & \mathtt{e} & ::= & \mathtt{x}\ \mid \mathtt{this} \mid \mathtt{e.f}\ \mid \mathtt{e.m}(\overline{\mathtt{e}})\ \mid \mathtt{new}\ \mathtt{C}(\overline{\mathtt{e}})\ \mid \mathtt{e}\ \mathtt{as}\ \mathtt{T} \\
\text{(Par Type)} & \mathtt{U,V} & ::= & \mathtt{T}
\end{array}
\qquad
\begin{array}{rlll}
\text{(Type)} & \mathtt{S,T} & ::= & \mathtt{N}\ \mid \mathtt{T}\{\mathtt{c}\} \\
\text{(N Type)} & \mathtt{N} & ::= & \mathtt{C}\ \mid \mathtt{N}\{\mathtt{c}\} \\
\text{(C Term)} & \mathtt{t} & ::= & \mathtt{self} \\
\text{(Const.)} & \mathtt{c,d} & ::= & \mathtt{true} \mid \mathtt{c,c}\ \mid \mathtt{x{:}V;c}
\end{array}
$$

FX **well-formedness rules:**

$$
\Gamma \vdash \mathtt{true} : \mathtt{o} \quad \text{(true)} \qquad
\frac{\Gamma \vdash \mathtt{c_0} : \mathtt{o} \qquad \Gamma \vdash \mathtt{c_1} : \mathtt{o}}{\Gamma \vdash (\mathtt{c_0}, \mathtt{c_1}) : \mathtt{o}}\ \text{(AND)} \qquad
\frac{\Gamma \vdash \mathtt{t} : \mathtt{T} \qquad \Gamma \vdash \mathtt{c}[\mathtt{t}/\mathtt{x}] : \mathtt{o}}{\Gamma \vdash \mathtt{x} : \mathtt{T}; \mathtt{c} : \mathtt{o}}\ \text{(EXISTS)}
$$

$$
\frac{\Gamma \vdash \mathtt{class}(\mathtt{C})}{\Gamma \vdash \mathtt{C}\ \mathtt{type}}\ \text{(CLASS)} \qquad
\frac{\Gamma \vdash \mathtt{T}\ \mathtt{type} \qquad \Gamma, \mathtt{self} : \mathtt{T} \vdash \mathtt{c} : \mathtt{o}}{\Gamma \vdash \mathtt{T}\{\mathtt{c}\}\ \mathtt{type}}\ \text{(DEP)}
$$

FX **sub-typing and type-equivalence rules:**

$$
\frac{\mathtt{class}\ \mathtt{C}(\ldots)\ \mathtt{extends}\ \mathtt{D}\{\ldots\} \in \mathtt{P}}{\vdash \mathtt{C} <: \mathtt{D}}\ \text{(S-EXTENDS)} \qquad
\frac{\Gamma, \mathtt{c} \vdash \mathtt{S} <: \mathtt{T}}{\Gamma \vdash \mathtt{S}\{\mathtt{c}\} <: \mathtt{T}}\ \text{(S-CONST-L)} \qquad
\frac{\Gamma, \mathtt{self} : \mathtt{S} \vdash \mathtt{c} \qquad \Gamma \vdash \mathtt{S} <: \mathtt{T}}{\Gamma \vdash \mathtt{S} <: \mathtt{T}\{\mathtt{c}\}}\ \text{(S-CONST-R)}
$$

$$
\frac{\mathtt{Gamma} \vdash \mathtt{U}\ \mathtt{type} \qquad \Gamma \vdash \mathtt{S} <: \mathtt{T} \qquad (\mathtt{x}\ \text{fresh})}{\Gamma \vdash \mathtt{x} : \mathtt{U}; \mathtt{S} <: \mathtt{T}}\ \text{(S-EXISTS-L)} \qquad
\frac{\mathtt{Gamma} \vdash \mathtt{t} : \mathtt{U} \qquad \Gamma \vdash \mathtt{S} <: \mathtt{T}[\mathtt{t}/\mathtt{x}]}{\Gamma \vdash \mathtt{S} <: \mathtt{x}; \mathtt{U} : \mathtt{T}}\ \text{(S-EXISTS-R)} \qquad
\frac{\Gamma \vdash \mathtt{S} <: \mathtt{T} \qquad \Gamma \vdash \mathtt{T} <: \mathtt{S}}{\Gamma \vdash \mathtt{S} \equiv \mathtt{T}}\ \text{(TYPE-EQUIV)}
$$

Type judgement rules:

$$
\Gamma, \mathtt{x} : \mathtt{T} \vdash \mathtt{x} : \mathtt{T}\{\mathtt{self} == \mathtt{x}\}\ \text{(T-VAR)} \qquad
\frac{\Gamma \vdash \mathtt{e} : \mathtt{U} \qquad \Gamma \vdash \mathtt{T}\ \mathtt{type}}{\Gamma \vdash \mathtt{e}\ \mathtt{as}\ \mathtt{T} : \mathtt{T}}\ \text{(T-CAST)} \qquad
\frac{\Gamma \vdash \mathtt{e} : \mathtt{S} \qquad \Gamma, \mathtt{z} : \mathtt{S} \vdash \mathtt{z}\ \mathtt{has}\ \mathtt{f} : \mathtt{T} \qquad (\mathtt{z}\ \text{fresh})}{\Gamma \vdash \mathtt{e.f} : (\mathtt{z} : \mathtt{S}; \mathtt{T})}\ \text{(T-FIELD)}
$$

$$
\frac{\begin{array}{c}\Gamma \vdash \mathtt{e} : \mathtt{T}, \overline{\mathtt{e}} : \overline{\mathtt{T}} \\ \Gamma, \mathtt{v} : \mathtt{T}, \overline{\mathtt{v}} : \overline{\mathtt{T}} \vdash \mathtt{v}\ \mathtt{has}\ (\mathtt{m}(\overline{\mathtt{v}} : \overline{\mathtt{V}}), \mathtt{c} \rightarrow \mathtt{U}), \overline{\mathtt{T}} <: \overline{\mathtt{V}}, \mathtt{c} \qquad (\mathtt{v}, \overline{\mathtt{v}}\ \text{fresh})\end{array}}{\Gamma \vdash \mathtt{e.m}(\overline{\mathtt{e}}) : (\mathtt{v} : \mathtt{T}; \overline{\mathtt{v}} : \overline{T}; U)}\ \text{(T-INVK)} \qquad
\frac{\begin{array}{c}\Gamma \vdash \overline{\mathtt{e}} : \overline{\mathtt{T}} \\ \Gamma, \mathtt{v} : \mathtt{C} \vdash \mathtt{fields}(\mathtt{v}) = \overline{\mathtt{f}} : \overline{\mathtt{V}} \qquad (\mathtt{v}, \overline{\mathtt{v}}\ \text{fresh}) \\ \Gamma, \mathtt{v} : \mathtt{C}, \overline{\mathtt{v}} : \overline{\mathtt{T}}, \mathtt{v}.\overline{\mathtt{f}} = \overline{\mathtt{v}} \vdash \overline{\mathtt{T}} <: \overline{\mathtt{V}}, \mathtt{inv}(\mathtt{C}, \mathtt{v})\end{array}}{\Gamma \vdash \mathtt{new}\ \mathtt{C}(\overline{\mathtt{e}}) : C\{\overline{\mathtt{v}} : \overline{\mathtt{T}}; \mathtt{self}.\overline{\mathtt{f}} = \overline{\mathtt{v}}, \mathtt{inv}(\mathtt{C}, \mathtt{self})\}}\ \text{(T-NEW)}
$$

$$
\frac{\mathtt{this} : \mathtt{C}, \overline{\mathtt{x}} : \overline{\mathtt{V}}, \mathtt{c} \vdash \mathtt{T}\ \mathtt{type}, \overline{\mathtt{V}}\ \mathtt{type}, \mathtt{e} : \mathtt{U}, \mathtt{U} <: \mathtt{T}}{\mathtt{def}\ \mathtt{m}(\overline{\mathtt{x}} : \overline{\mathtt{V}})\{\mathtt{c}\} : \mathtt{T} = \mathtt{e};\ \text{OK in}\ C}\ \text{(METHOD OK)} \qquad
\frac{\overline{\mathtt{M}}\ \text{OK in}\ \mathtt{C} \qquad \mathtt{this} : \mathtt{C}, \mathtt{c} \vdash \overline{\mathtt{V}}\ \mathtt{type}, \mathtt{N}\ \mathtt{type}}{\mathtt{class}\ \mathtt{C}(\overline{\mathtt{f}} : \overline{\mathtt{V}})\{\mathtt{c}\}\ \mathtt{extends}\ \mathtt{N}\{\overline{\mathtt{M}}\}\ \text{OK}}\ \text{(CLASS OK)}
$$

Transition Rules:

$$
\frac{\mathtt{x} : \mathtt{C}\} \vdash \mathtt{fields}(\mathtt{x}) = \overline{\mathtt{f}} : \overline{\mathtt{V}}}{(\mathtt{new}\ \mathtt{C}(\overline{\mathtt{e}})).\mathtt{f}_i \rightarrow \mathtt{e}_i}\ \text{(R-FIELD)} \qquad
\frac{\mathtt{x} : \mathtt{C} \vdash \mathtt{x}\mathtt{has} m(\overline{\mathtt{x}} : \overline{T})\{c\} : T = e}{(\mathtt{new}\ \mathtt{C}(\overline{\mathtt{e}})).\mathtt{m}(\overline{\mathtt{d}}) \rightarrow \mathtt{e}[\mathtt{new}\ \mathtt{C}(\overline{\mathtt{e}}), \overline{\mathtt{d}}/\mathtt{this}, \overline{\mathtt{x}}]}\ \text{(R-INVK)}
$$

$$
\frac{\mathtt{e} \rightarrow \mathtt{e}'}{\mathtt{e.f}_i \rightarrow \mathtt{e}'.\mathtt{f}_i}\ \text{(RC-FIELD)} \qquad
\frac{\mathtt{e} \rightarrow \mathtt{e}'}{\mathtt{e.m}(\overline{\mathtt{e}}) \rightarrow \mathtt{e}'.\mathtt{m}(\overline{\mathtt{e}})}\ \text{(RC-INVK-RECV)}
$$

$$
\frac{\vdash \mathtt{C} <: \mathtt{T}[\mathtt{new}\ \mathtt{C}(\overline{\mathtt{d}})/\mathtt{self}]}{(\mathtt{T})(\mathtt{new}\ \mathtt{C}(\overline{\mathtt{d}})) \rightarrow \mathtt{new}\ C(\overline{\mathtt{d}})}\ \text{(R-CAST)} \qquad
\frac{\mathtt{e}_i \rightarrow \mathtt{e}_i{}'}{\mathtt{e.m}(\ldots, \mathtt{e}_i, \ldots) \rightarrow \mathtt{e.m}(\ldots, \mathtt{e}'_i, \ldots)}\ \text{(RC-INVK-ARG)}
$$

$$
\frac{\mathtt{e} \rightarrow \mathtt{e}'}{(\mathtt{T})\mathtt{e} \rightarrow (\mathtt{T})\mathtt{e}'}\ \text{(RC-CAST)} \qquad
\frac{\mathtt{e}_i \rightarrow \mathtt{e}_i{}'}{\mathtt{new}\ \mathtt{C}(\ldots, \mathtt{e}_i, \ldots) \rightarrow \mathtt{new}\ \mathtt{C}(\ldots, \mathtt{e}'_i, \ldots)}\ \text{(RC-NEW-ARG)}
$$

Figure 1. Semantics of FX

2. for any variable `x` (`X`), there is at most one formula `x : T` (`X type`) in Γ.

The judgements of interest are:

Type well-formedness $\Gamma \vdash$ `T type`

Subtyping $\Gamma \vdash$ `S<:T`

Typing $\Gamma \vdash$ `e : T`

Method ok $\Gamma \vdash M$ OK in C (method `M` is well-defined for the class `C`).

Field ok $\Gamma \vdash f : T$ OK in C (field `f : T` is well-defined for the class `C`).

Class ok $\Gamma \vdash$ `Cl` *OK* (class definition `Cl` is ok).

In defining these judgements we will use $\Gamma \vdash_C c$, the judgement corresponding to the underlying constraint system. For simplicity, we define $\Gamma \vdash \mathtt{c}$ to mean $\sigma(\Gamma) \vdash_C \mathtt{c}$, where the *constraint projection*, $\sigma(\Gamma)$ is defined thus:

$\sigma(\varepsilon) = \mathtt{true}$
$\sigma(x : C\{c\}, \Gamma)$=c[x/`self`], $\sigma(\Gamma)$
$\sigma(c, \Gamma)$ = c, $\sigma(\Gamma)$

1.4 Well formedness rules

We posit a special type `o` (traditionally the type of propositions), and regard constraints as expressions of type `o`. See Figure **??**.

2. Type inference rules

2.1 Expression typing judgement

Now we consider the rule for method invocation. Assume that in a type environment Γ the expressions $\overline{e}$ have the types $\overline{T}$. Since the actual values of these expressions are not known, we shall assume that they take on some fixed but unknown values $\overline{z}$ of type $\overline{T}$. Now for z as receiver, let us assume that the type $T \equiv C\{d\}$ has a method named `m` with signature $\overline{z} : \overline{Z}, c \rightarrow U$. If there is no method named `m` for the class `C` then this method invocation cannot be type-checked. Without loss of generality we may assume that the parameters of this method are named $\overline{z}$, since we are free to choose variable names as we wish because of α-equivalence. Now in order for the method to be invocable, it must be the case that the types $\overline{T}$ are subtypes of $\overline{Z}$. (Note that there are no occurrences of `this` in $\overline{Z}$; they have been replaced by z – see Section 1.1) Further, it must be the case that for these parameter values, the constraint c is entailed. Given all these assumptions it must be the case that the return type is U — with all the parameters $\overline{z}$ existentially quantified.