Abstract

Notes on a formal semantics for GFX.

1. Judgements

In the rest of this paper we will assume give some fixed but unknown constraint system $\mathcal{D}$. We will assume that the program P is written using constraints from $\mathcal{D}$, and further that classes defined in P do not have a cyclic inheritance struture.

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

From P and $\mathcal{D}$ we also generate a new constraint system $\mathcal{O}$, the constraint system of *objects* over P and $\mathcal{D}$ as follows. Let $\mathtt{C},\mathtt{D}$ range over names of classes in P, $\mathtt{f}$ over field names, $\mathtt{m}$ over method names, $\mathtt{S},\mathtt{T}$ over types, $\mathtt{c}$ over constraints in the underlying data constraint system $\mathcal{D}$. The constraints of $\mathcal{O}$ are given by:

$$\begin{array}{llll} \text{(Const.)} & \mathtt{c},\mathtt{d} & ::= & \mathtt{class(C)} \mid \mathtt{S} \trianglelefteq \mathtt{T} \\ & & & \mid \mathtt{fields(x)} = \overline{\mathtt{f}} : \overline{\mathtt{T}} \\ & & & \mid \mathtt{mtype(x,m,}\overline{\mathtt{x}}\mathtt{)} = (\overline{T}, \mathtt{c} \to \mathtt{T})) \end{array}$$

The constraint system satisfies the following axioms and inference rules.

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

$$\frac{\overline{X}, \overline{\mathtt{f}} : \overline{\mathtt{C\{e\}}} \text{ type and value fields defined or inherited at class } D}{\mathtt{z}:\mathtt{D\{d\}} \vdash_{\mathcal{O}} \mathtt{fields(z)} = \overline{\mathtt{X}}, \overline{\mathtt{f}} : \overline{\mathtt{C\{e[z/this], self{==}\ z.f_i, d[z/self]\}}}} \quad \text{(FIELDS)}$$

$$\vdash_{\mathcal{O}} D[\overline{\mathtt{T}}](\overline{\mathtt{t}}).\mathtt{f_i} = \mathtt{t_i} \quad \text{(SEL-V)}$$

$$\vdash_{\mathcal{O}} D[\overline{\mathtt{T}}](\overline{\mathtt{t}}).\mathtt{X_i} = \mathtt{t_i} \quad \text{(SEL-T)}$$

$$\frac{\begin{array}{c}\mathtt{m}(\overline{\mathtt{x}}:\overline{\mathtt{E}})\{\mathtt{c}\}:\mathtt{F} = \{\ldots\} \text{ defined or inherited at } \mathtt{D} \\ \theta = [\mathtt{z},\overline{\mathtt{z}}/\mathtt{this},\overline{\mathtt{x}}]\end{array}}{\mathtt{z}:\mathtt{D\{d\}} \vdash_{\mathcal{O}} \mathtt{mtype(z,m,}\overline{\mathtt{z}}\mathtt{)} = (\overline{\mathtt{E}}, \mathtt{c} \to \mathtt{F\{d[z/self]\}})\theta} \quad \text{(MTYPE)}$$

The constraint system $\mathcal{C}$ is the (disjoint) conjunction $\mathcal{D},\mathcal{O}$ of the constraint systems $\mathcal{D}$ and $\mathcal{O}$. (This requires the assumption that $\mathcal{D}$ does not have any constraints in common with $\mathcal{O}$.)

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

2. Rules

2.1 Judgements

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

1. for any formula ϕ in the sequence all variables $\mathtt{x}$ ($\mathtt{X}$) occuring in ϕ are defined by a declaration $\mathtt{x}:\mathtt{T}$ ($\mathtt{X}$ $\mathtt{type}$) in the sequence to the left of ϕ.
2. for any variable $\mathtt{x}$ ($\mathtt{X}$), there is at most one formula $\mathtt{x}:\mathtt{T}$ ($\mathtt{X}$ $\mathtt{type}$) in Γ.

The judgements of interest are:

Type well-formedness $\Gamma \vdash \mathtt{T}\ \mathtt{type}$

Subtyping $\Gamma \vdash \mathtt{S} <: \mathtt{T}$

Typing $\Gamma \vdash \mathtt{e}:\mathtt{T}$

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

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

Class ok $\Gamma \vdash \mathtt{Cl}$ *OK* (class definition $\mathtt{Cl}$ is ok).

In defining these judgements we will use $\Gamma \vdash_{\mathcal{C}} c$, the judgement corresponding to the underlying constraint system. For simplicity, we define $\Gamma \vdash \mathtt{c}$ to mean $\sigma(\Gamma) \vdash_{\mathcal{C}} \mathtt{c}$, 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, $\sigma(\Gamma)$

2.2 Well formedness rules

We posit a special type $\mathtt{o}$ (traditionally the type of propositions), and regard constraints as expressions of type $\mathtt{o}$. See Figure **??**.

3. Type inference rules

3.1 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 $\mathtt{m}$ with signature $\overline{z}:\overline{Z}, c \to U$. If there is no method named $\mathtt{m}$ for the class $\mathtt{C}$ 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 $\mathtt{this}$ 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.

FX **productions:**

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

FX **well-formedness rules:**

$$\Gamma \vdash \mathtt{true} : \mathtt{o} \quad (\mathtt{true}) \qquad \frac{\Gamma \vdash \mathtt{c}_0 : \mathtt{o} \qquad \Gamma \vdash \mathtt{c}_1 : \mathtt{o}}{\Gamma \vdash (\mathtt{c}_0, \mathtt{c}_1) : \mathtt{o}} \ (\textsc{And}) \qquad \frac{\Gamma \vdash \mathtt{t} : \mathtt{T} \qquad \Gamma \vdash \mathtt{c}[\mathtt{t}/\mathtt{x}] : \mathtt{o}}{\Gamma \vdash \mathtt{x} : \mathtt{T}; \mathtt{c} : \mathtt{o}} \ (\textsc{Exists})$$

$$\frac{\Gamma \vdash \mathtt{class}(\mathtt{C})}{\Gamma \vdash \mathtt{C}\ \mathtt{type}} \ (\textsc{Class}) \qquad \frac{\Gamma \vdash \mathtt{T}\ \mathtt{type} \qquad \Gamma, \mathtt{self} : \mathtt{T} \vdash \mathtt{c} : \mathtt{o}}{\Gamma \vdash \mathtt{T}\{\mathtt{c}\}\ \mathtt{type}} \ (\textsc{Dep})$$

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

$$\frac{\Gamma, \mathtt{c} \vdash \mathtt{S} \trianglelefteq \mathtt{T}}{\Gamma \vdash \mathtt{S}\{\mathtt{c}\} \trianglelefteq \mathtt{T}} \ (\text{S-}\textsc{Const-L}) \qquad \frac{\Gamma, \mathtt{self} : \mathtt{S} \vdash \mathtt{c} \qquad \Gamma \vdash \mathtt{S} \trianglelefteq \mathtt{T}}{\Gamma \vdash \mathtt{S} \trianglelefteq \mathtt{T}\{\mathtt{c}\}} \ (\text{S-}\textsc{Const-R})$$

$$\frac{\mathtt{Gamma} \vdash \mathtt{U}\ \mathtt{type} \qquad \Gamma \vdash \mathtt{S} \trianglelefteq \mathtt{T} \qquad (\mathtt{x}\ \text{fresh})}{\Gamma \vdash \mathtt{x} : \mathtt{U}; \mathtt{S} \trianglelefteq \mathtt{T}} \ (\text{S-}\textsc{Exists-L}) \qquad \frac{\mathtt{Gamma} \vdash \mathtt{t} : \mathtt{U} \qquad \Gamma \vdash \mathtt{S} \trianglelefteq \mathtt{T}[\mathtt{t}/\mathtt{x}]}{\Gamma \vdash \mathtt{S} \trianglelefteq \mathtt{x}; \mathtt{U} : \mathtt{T}} \ (\text{S-}\textsc{Exists-R}) \qquad \frac{\Gamma \vdash \mathtt{S} \trianglelefteq \mathtt{T} \qquad \Gamma \vdash \mathtt{T} \trianglelefteq \mathtt{S}}{\Gamma \vdash \mathtt{S} \equiv \mathtt{T}} \ (\textsc{Type-equiv})$$

Type judgement rules:

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

$$\frac{\begin{array}{c}\Gamma \vdash \mathtt{e} : \mathtt{T}, \overline{\mathtt{e}} : \overline{\mathtt{T}} \\ \Gamma, \mathtt{v} : \mathtt{T}, \overline{\mathtt{v}} : \overline{\mathtt{T}} \vdash \mathtt{mtype}(\mathtt{v}, \mathtt{m}, \overline{\mathtt{v}}) = (\overline{\mathtt{V}}, \mathtt{c} \rightarrow \mathtt{U}), \overline{\mathtt{T}} <: \overline{\mathtt{V}}, \mathtt{c} \qquad (\mathtt{v}, \overline{\mathtt{v}}\ \text{fresh})\end{array}}{\Gamma \vdash \mathtt{e.m}(\overline{\mathtt{e}}) : (\mathtt{v} : \mathtt{T}; \overline{\mathtt{v}} : \overline{T}; U)} \ (\text{T-INVK})$$

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

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

Transition Rules:

$$\frac{\mathtt{fields}(\mathtt{C}) = \overline{\mathtt{C}}\ \overline{\mathtt{f}}}{(\mathtt{new}\ \mathtt{C}(\overline{\mathtt{e}})).\mathtt{f}_i \rightarrow \mathtt{e}_i} \ (\text{R-}\textsc{Field}) \qquad \frac{\mathit{mbody}(\mathtt{new}\ \mathtt{C}(\overline{\mathtt{e}}), \mathtt{m}, \overline{\mathtt{d}}) = \mathtt{e}}{(\mathtt{new}\ \mathtt{C}(\overline{\mathtt{e}})).\mathtt{m}(\overline{\mathtt{d}}) \rightarrow \mathtt{e}} \ (\text{R-}\textsc{Invk})$$

$$\frac{\vdash \mathtt{C} <: \mathtt{T}[\mathtt{new}\ \mathtt{C}(\overline{\mathtt{d}})/\mathtt{self}]}{(\mathtt{T})(\mathtt{new}\ \mathtt{C}(\overline{\mathtt{d}})) \rightarrow \mathtt{new}\ C(\overline{\mathtt{d}})} \ (\text{R-}\textsc{Cast}) \qquad \frac{\mathtt{e} \rightarrow \mathtt{e}'}{\mathtt{e.m}(\overline{\mathtt{e}}) \rightarrow \mathtt{e}'.\mathtt{m}(\overline{\mathtt{e}})} \ (\text{RC-}\textsc{Invk-Recv})$$

$$\frac{\mathtt{e} \rightarrow \mathtt{e}'}{\mathtt{e.f}_i \rightarrow \mathtt{e}'.\mathtt{f}_i} \ (\text{RC-}\textsc{Field}) \qquad \frac{\mathtt{e}_i \rightarrow {\mathtt{e}_i}'}{\mathtt{e.m}(\ldots, \mathtt{e}_i, \ldots) \rightarrow \mathtt{e.m}(\ldots, \mathtt{e}_i', \ldots)} \ (\text{RC-}\textsc{Invk-Arg})$$

$$\frac{\mathtt{e} \rightarrow \mathtt{e}'}{(\mathtt{T})\mathtt{e} \rightarrow (\mathtt{T})\mathtt{e}'} \ (\text{RC-}\textsc{Cast}) \qquad \frac{\mathtt{e}_i \rightarrow {\mathtt{e}_i}'}{\mathtt{new}\ \mathtt{C}(\ldots, \mathtt{e}_i, \ldots) \rightarrow \mathtt{new}\ \mathtt{C}(\ldots, \mathtt{e}_i', \ldots)} \ (\text{RC-}\textsc{New-Arg})$$

Figure 1. Semantics of FX

Additional rules for FX(G): No additional rules for sub-typing, type-equivalence, expression typing or dynamic semantics.

FX(G)=FX + *these productions:*

(Par Type)	V	::=	type
(Type)	T	::=	X
(Const.)	c	::=	$X \trianglelefteq N$

Additional FX(G) well-formedness rule:

$\Gamma, X : \texttt{type} \vdash X\ \texttt{type}$ (Type Var)

Additional rules for FX(D($\mathcal{C}$)): Only the following rules are needed; no additional rules are needed for sub-typing, type-equivalence, expression typing or dynamic semantics.

FX(D($\mathcal{C}$))=FX + *these productions:*

$\mathcal{C}$ specifies predicates q and functions f.

(Type)	T	::=	new base types, e.g. int, boolean
(Const.)	c	::=	$t{==}t \mid q(\overline{t})$
(C Term)	t	::=	$x \mid f(\overline{t}) \mid t.f \mid \texttt{new}\ C(\overline{t})$

Additional FX(D($\mathcal{C}$)) well-formedness rule:

$$\frac{p(\overline{T}) : o \in \mathcal{C} \qquad \Gamma \vdash \overline{t} : \overline{T}}{\Gamma \vdash p(\overline{T}) : o}\ (\textsc{Pred}) \qquad \frac{f(\overline{T}) : T \in \mathcal{C} \qquad \Gamma \vdash \overline{t} : \overline{T}}{\Gamma \vdash f(\overline{T}) : T}\ (\textsc{Fun}) \qquad \frac{\begin{array}{c}\Gamma \vdash t_0 : T_0 \qquad \Gamma \vdash t_1 : T_1 \\ (\Gamma \vdash T_0 <: T_1 \vee \Gamma \vdash T_1 <: T_0)\end{array}}{\Gamma \vdash t_0 = t_1 : o}\ (\textsc{Equals})$$

Additional rules for FX(G): None.

Additional rules for FX(G,D($\mathcal{C}$),P): Only the following rules are needed; no additional rules are needed for sub-typing, type-equivalence, expression typing or dynamic semantics.

(Path)	p	::=	$x \mid p.f$
(Type)	T	::=	p

$$\frac{\Gamma \vdash p : T \qquad \Gamma, x : T \vdash x\ \texttt{has}\ X : \texttt{type}}{\Gamma \vdash p.X\ \texttt{type}}\ (\textsc{Path})$$

Figure 2. Semantics for FX(G),FX(D($\mathcal{C}$)),FX(G,D($\mathcal{C}$)),FX(G,D($\mathcal{C}$),P)