Featherweight Java

Piotr Krzemiński

Wrocław, 9th April 2014

Introduction

"Inside every large language is a small language struggling to get out..."

– T. Hoare

Agenda

Provide rigorous calculus to reason about Java's type system.

- completeness vs. compactness
- FJ favors compactness over completeness to focus on just a few key issues
- skip most of language features
- make formal proof of type soundness simple while still capturing essence of soundness for full Java

To achieve simplicity, language is reduced

- no concurrency
- no reflection
- no interfaces
- no method overloading
- no inner classes
- no primitive types
- no messages to `super`
- no null pointers
- no assignment

...so it's more or less functional subset of Java, only little larger than classical Church's λ-calculus

Featherweight Java *key concepts*

Minimal syntax, containing only

- mutually recursive class definitions
- object creation
- field access
- method invocation
- method override
- method recursion through `this`
- subtyping
- casting

...but still legal subset of Java.

```
class A extends Object {
   A() { super(); }
}
class B extends Object {
   B() { super(); }
}

class Pair extends Object {
   Object fst;
   Object snd;
   Pair(Object fst, Object snd) {
      super(); this.fst = fst; this.snd = snd;
   }
   Pair setfst(Object newfst) {
      return new Pair(newfst, this.snd);
   }
}
```

The expression:

```
new Pair(new A(), new B()).setfst(new B())
```

evaluates to expression:

```
new Pair(new B(), new B())
```

Featherweight Java *syntax*

$$
\begin{array}{lcl}
\texttt{L} & ::= & \texttt{class C extends C \{}\overline{\texttt{C}}\ \overline{\texttt{f}}\texttt{; K }\overline{\texttt{M}}\texttt{\}} \\
\texttt{K} & ::= & \texttt{C(}\overline{\texttt{C}}\ \overline{\texttt{f}}\texttt{)\{super(}\overline{\texttt{f}}\texttt{); this.}\overline{\texttt{f}}\texttt{=}\overline{\texttt{f}}\texttt{;\}} \\
\texttt{M} & ::= & \texttt{C m(}\overline{\texttt{C}}\ \overline{\texttt{x}}\texttt{)\{ return e; \}} \\
\texttt{e} & ::= & \texttt{x} \mid \texttt{e.f} \mid \texttt{e.m(}\overline{\texttt{e}}\texttt{)} \mid \texttt{new C(}\overline{\texttt{e}}\texttt{)} \mid \texttt{(C)e}
\end{array}
$$

Syntactical shorthands:

- we write $\overline{f}$ for $f_1, f_2, \ldots, f_n$ (similarly for $\overline{C}, \overline{x}, \overline{e}$, etc.)
- length of $\overline{x}$ is $\#(\overline{x})$
- $\overline{C}\ \overline{f}$ is for $C_1\ f_1, \ldots, C_n\ f_n$
- $\texttt{this}.\overline{f} = \overline{f}$ is for $\texttt{this}.f_1 = f_1, \ldots, \texttt{this}.f_n = f_n$

Featherweight Java *syntax*

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

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

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

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

- `this` is distinguished variable never used as method parameter
- supertype in class definition is always included
- instance variables should have distinct names from those defined in superclasses
- instance variables cannot be redeclared in subclasses
- constructor takes as many parameters as there are instance variables (including those from superclasses)
- casts bind less tightly than other form of expression

Featherweight Java *program definition*

Class table is mapping from class names `C` to class declarations `L`

FJ program = class table + expression

$$\texttt{C} <: \texttt{C} \qquad \frac{\texttt{C} <: \texttt{D} \qquad \texttt{D} <: \texttt{E}}{\texttt{C} <: \texttt{E}}$$

$$\frac{\texttt{class C extends D \{...\}}}{\texttt{C} <: \texttt{D}}$$

We may decide this relation by looking at the class table. Class table *CT* must satisfy following conditions:

- $CT(C) =$ `class C`... for every $C \in dom(CT)$
- `Object` $\notin dom(CT)$
- for every class name *C* (except `Object`) appearing anywhere in *CT* we have $C \in dom(CT)$
- there are no cycles in subtype relation induced by *CT* (i.e. <: is antisymmetric)

$$\mathit{fields}(\texttt{Object}) = \bullet$$

$$\frac{\texttt{class C extends D \{}\overline{\texttt{C}}\ \overline{\texttt{f}}\texttt{; K }\overline{\texttt{M}}\texttt{\}} \qquad \mathit{fields}(\texttt{D}) = \overline{\texttt{D}}\ \overline{\texttt{g}}}{\mathit{fields}(\texttt{C}) = \overline{\texttt{D}}\ \overline{\texttt{g}}, \overline{\texttt{C}}\ \overline{\texttt{f}}}$$

Featherweight Java *method type lookup*

$$\frac{\texttt{class C extends D } \{\overline{\texttt{C}}\ \overline{\texttt{f}};\ \texttt{K}\ \overline{\texttt{M}}\} \qquad \texttt{B m}(\overline{\texttt{B}}\ \overline{\texttt{x}})\{\ \texttt{return e;}\ \} \in \overline{\texttt{M}}}{\mathit{mtype}(\texttt{m}, \texttt{C}) = \overline{\texttt{B}} \longrightarrow \texttt{B}}$$

$$\frac{\texttt{class C extends D } \{\overline{\texttt{C}}\ \overline{\texttt{f}};\ \texttt{K}\ \overline{\texttt{M}}\} \qquad \texttt{m} \notin \overline{\texttt{M}}}{\mathit{mtype}(\texttt{m}, \texttt{C}) = \mathit{mtype}(\texttt{m}, \texttt{D})}$$

Featherweight Java *method body lookup*

$$\frac{\texttt{class C extends D \{}\overline{\texttt{C}}\ \overline{\texttt{f}}\texttt{; K }\overline{\texttt{M}}\texttt{\}} \qquad \texttt{B m(}\overline{\texttt{B}}\ \overline{\texttt{x}}\texttt{)\{ return e; \}} \in \overline{\texttt{M}}}{\mathit{mbody}(\texttt{m}, \texttt{C}) = \overline{\texttt{x}}\texttt{.e}}$$

$$\frac{\texttt{class C extends D \{}\overline{\texttt{C}}\ \overline{\texttt{f}}\texttt{; K }\overline{\texttt{M}}\texttt{\}} \qquad \texttt{m} \notin \overline{\texttt{M}}}{\mathit{mbody}(\texttt{m}, \texttt{C}) = \mathit{mbody}(\texttt{m}, \texttt{D})}$$

$$\Gamma \vdash \mathtt{x} : \Gamma(\mathtt{x}) \qquad \text{(T-VAR)}$$

$$\frac{\Gamma \vdash \mathtt{e}_0 : \mathtt{C}_0 \qquad \mathit{fields}(\mathtt{C}_0) = \overline{\mathtt{C}}\ \overline{\mathtt{f}}}{\Gamma \vdash \mathtt{e}_0\mathtt{.f}_i : \mathtt{C}_i} \qquad \text{(T-FIELD)}$$

$$\frac{\Gamma \vdash \mathtt{e}_0 : \mathtt{C}_0 \qquad \mathit{mtype}(\mathtt{m}, \mathtt{C}_0) = \overline{\mathtt{D}}{\rightarrow}\mathtt{C} \qquad \Gamma \vdash \overline{\mathtt{e}} : \overline{\mathtt{C}} \qquad \overline{\mathtt{C}} <: \overline{\mathtt{D}}}{\Gamma \vdash \mathtt{e}_0\mathtt{.m}(\overline{\mathtt{e}}) : \mathtt{C}} \qquad \text{(T-INVK)}$$

$$\frac{\mathit{fields}(\mathtt{C}) = \overline{\mathtt{D}}\ \overline{\mathtt{f}} \qquad \Gamma \vdash \overline{\mathtt{e}} : \overline{\mathtt{C}} \qquad \overline{\mathtt{C}} <: \overline{\mathtt{D}}}{\Gamma \vdash \mathtt{new}\ \mathtt{C}(\overline{\mathtt{e}}) : \mathtt{C}} \qquad \text{(T-NEW)}$$

- an environment Γ is finite mapping from variables to types, written $\overline{x} : \overline{C}$
- obvious shorthands $\Gamma \vdash \overline{e} : \overline{C}$ and $\overline{C} <: \overline{D}$

Featherweight Java *typing casts*

$$\frac{\Gamma \vdash \mathtt{e}_0 : \mathtt{D} \qquad \mathtt{D} <: \mathtt{C}}{\Gamma \vdash (\mathtt{C})\mathtt{e}_0 : \mathtt{C}} \qquad \text{(T-UCast)}$$

$$\frac{\Gamma \vdash \mathtt{e}_0 : \mathtt{D} \qquad \mathtt{C} <: \mathtt{D} \qquad \mathtt{C} \neq \mathtt{D}}{\Gamma \vdash (\mathtt{C})\mathtt{e}_0 : \mathtt{C}} \qquad \text{(T-DCast)}$$

$$\frac{\Gamma \vdash \mathtt{e}_0 : \mathtt{D} \qquad \mathtt{C} \not<: \mathtt{D} \qquad \mathtt{D} \not<: \mathtt{C} \qquad \textit{stupid warning}}{\Gamma \vdash (\mathtt{C})\mathtt{e}_0 : \mathtt{C}} \qquad \text{(T-SCast)}$$

- Java compiler rejects stupid casts
- in FJ stupid casts are present to formulate type soundness in small-step semantics
- its special nature indicated by *stupid warning* hypothesis
- FJ typing corresponds to a legal Java typing only if it does not contain this rule

Featherweight Java *typing methods and classes*

$$\frac{\overline{\texttt{x}} : \overline{\texttt{C}}, \texttt{this} : \texttt{C} \vdash \texttt{e}_0 : \texttt{E}_0 \qquad \texttt{E}_0 \;\texttt{<:}\; \texttt{C}_0 \qquad \texttt{class C extends D \{...\}} \qquad \text{if } \mathit{mtype}(\texttt{m}, \texttt{D}) = \overline{\texttt{D}} \to \texttt{D}_0, \text{ then } \overline{\texttt{C}} = \overline{\texttt{D}} \text{ and } \texttt{C}_0 = \texttt{D}_0}{\texttt{C}_0 \;\texttt{m}(\overline{\texttt{C}}\;\overline{\texttt{x}})\texttt{\{ return } \texttt{e}_0\texttt{; \} OK IN C}} \quad \text{(T-Method)}$$

$$\frac{\texttt{K} = \texttt{C}(\overline{\texttt{D}}\;\overline{\texttt{g}},\;\overline{\texttt{C}}\;\overline{\texttt{f}})\texttt{\{super}(\overline{\texttt{g}})\texttt{; this.}\overline{\texttt{f}}\texttt{=}\overline{\texttt{f}}\texttt{;\}} \qquad \mathit{fields}(\texttt{D}) = \overline{\texttt{D}}\;\overline{\texttt{g}} \qquad \overline{\texttt{M}}\;\texttt{OK IN C}}{\texttt{class C extends D \{}\overline{\texttt{C}}\;\overline{\texttt{f}}\texttt{; K }\overline{\texttt{M}}\texttt{\} OK}} \quad \text{(T-Class)}$$

Featherweight Java *operational semantics*

$$\frac{\mathit{fields}(\texttt{C}) = \overline{\texttt{C}}\ \overline{\texttt{f}}}{\texttt{(new C(}\overline{\texttt{e}}\texttt{)).f}_i \longrightarrow \texttt{e}_i} \qquad \text{(R-Field)}$$

$$\frac{\mathit{mbody}(\texttt{m}, \texttt{C}) = \overline{\texttt{x}}\texttt{.e}_0}{\texttt{(new C(}\overline{\texttt{e}}\texttt{)).m(}\overline{\texttt{d}}\texttt{)} \longrightarrow [\overline{\texttt{d}}/\overline{\texttt{x}}, \texttt{new C(}\overline{\texttt{e}}\texttt{)}/\texttt{this}]\texttt{e}_0} \qquad \text{(R-Invk)}$$

$$\frac{\texttt{C <: D}}{\texttt{(D)(new C(}\overline{\texttt{e}}\texttt{))} \longrightarrow \texttt{new C(}\overline{\texttt{e}}\texttt{)}} \qquad \text{(R-Cast)}$$

Featherweight Java *operational semantics*

$$\frac{\texttt{e}_0 \longrightarrow \texttt{e}_0{}'}{\texttt{e}_0\texttt{.f} \longrightarrow \texttt{e}_0{}'\texttt{.f}} \qquad \text{(RC-Field)}$$

$$\frac{\texttt{e}_0 \longrightarrow \texttt{e}_0{}'}{\texttt{e}_0\texttt{.m(}\overline{\texttt{e}}\texttt{)} \longrightarrow \texttt{e}_0{}'\texttt{.m(}\overline{\texttt{e}}\texttt{)}} \qquad \text{(RC-Invk-Recv)}$$

$$\frac{\texttt{e}_i \longrightarrow \texttt{e}_i{}'}{\texttt{e}_0\texttt{.m(}\ldots\texttt{,e}_i\texttt{,}\ldots\texttt{)} \longrightarrow \texttt{e}_0\texttt{.m(}\ldots\texttt{,e}_i{}'\texttt{,}\ldots\texttt{)}} \qquad \text{(RC-Invk-Arg)}$$

$$\frac{\texttt{e}_i \longrightarrow \texttt{e}_i{}'}{\texttt{new C(}\ldots\texttt{,e}_i\texttt{,}\ldots\texttt{)} \longrightarrow \texttt{new C(}\ldots\texttt{,e}_i{}'\texttt{,}\ldots\texttt{)}} \qquad \text{(RC-New-Arg)}$$

$$\frac{\texttt{e}_0 \longrightarrow \texttt{e}_0{}'}{\texttt{(C)e}_0 \longrightarrow \texttt{(C)e}_0{}'} \qquad \text{(RC-Cast)}$$

Theorem 1 (Subject Reduction)

If $\Gamma \vdash e : C$ *and* $e \longrightarrow e'$, *then* $\Gamma \vdash e' : C'$ *for some* $C' <: C$.

Theorem 2 (Progress)

Suppose e is a well-typed expression.

- *If e includes* $\texttt{new}\ \texttt{C}_0(\overline{\texttt{e}}).\texttt{f}$ *as a subexpression, then* $fields(\mathrm{C}_0) = \overline{\mathrm{C}}\ \overline{f}$ *and* $f \in \overline{f}$ *for some* $\overline{\mathrm{C}}$ *and* $\overline{f}$.
- *If e includes* $\texttt{new}\ \texttt{C}_0(\overline{\texttt{e}}).\texttt{m}(\overline{\texttt{d}})$ *as a subexpression, then* $mbody(m, \mathrm{C}_0) = \overline{x}.e_0$ *and* $\#(\overline{x}) = \#(\overline{d})$ *for some* $\overline{x}$ *and* e_0.

To state type soundness formally, we give the definition of values, given by the following syntax:

$$\mathtt{v} ::= \mathtt{new}\ \mathtt{C}(\overline{\mathtt{v}})$$

Theorem 3 (FJ Type Soundness)

If $\emptyset \vdash e : C$ and $e \longrightarrow^{*} e'$ with e' a normal form, then e' is either a value v with $\emptyset \vdash v : D$ and $D <: C$, or an expression containing $(\mathtt{D})\ \mathtt{new}\ \mathtt{C}(\overline{\mathtt{e}})$ where $C \not<: D$.

Expression e is cast-safe in Γ if the type derivations of the underlying class table and $\Gamma \vdash e : C$ contain no *downcasts* or *stupid casts*.

Theorem 4 (Reduction Preserves Cast-Safety)

If e is cast-safe in Γ and $e \rightarrow e'$, then e' is cast-safe in Γ.

Theorem 5 (Progress of Cast-Safety)

Suppose e is cast-safe in Γ. If e has $\mathtt{(C)\ new\ C_0(\overline{e})}$ as a subexpression, then $C_0 <: C$.

Corollary 6 (No Typecast Errors in Cast-Safe programs)

If e is cast-safe in $\emptyset$ and $e \rightarrow^* e'$ with e' a normal form, then e' is a value v.

Featherweight Generic Java

Extension of Featherweight Java:

- FGJ = FJ + generic types
- type-parametrizable classes and methods
- two possible implementations:
  - type passing
  - type erasure

```
class A extends Object {
   A() { super(); }
}
class B extends Object {
   B() { super(); }
}

class Pair<X extends Object, Y extends Object> extends Object {
   X fst;
   Y snd;
   Pair(X fst, Y snd) {
      super(); this.fst = fst; this.snd = snd;
   }
   <Z extends Object> Pair<Z, Y> setfst(Z newfst) {
      return new Pair<Z, Y>(newfst, this.snd);
   }
}
```

The expression:

```
new Pair<A, B>(new A(), new B()).setfst<B>(new B())
```

evaluates to expression:

```
new Pair<B, B>(new B(), new B())
```

- while FJ is subset of Java, FGJ is not quite a subset of GJ
- in GJ type inference of type parameters in generic method invocation is obligatory
- for example `e.m<A,B>(x)` parses to two expressions `e.m<A` and `B>(x)` separated by a comma

Featherweight Generic Java *syntax*

$$
\begin{array}{lcl}
\texttt{T} & ::= & \texttt{X} \mid \texttt{N} \\
\texttt{N} & ::= & \texttt{C<}\overline{\texttt{T}}\texttt{>} \\
\texttt{L} & ::= & \texttt{class C<}\overline{\texttt{X}} \triangleleft \overline{\texttt{N}}\texttt{>} \triangleleft \texttt{N \{}\overline{\texttt{T}}\ \overline{\texttt{f}}\texttt{; K }\overline{\texttt{M}}\texttt{\}} \\
\texttt{K} & ::= & \texttt{C(}\overline{\texttt{T}}\ \overline{\texttt{f}}\texttt{)\{super(}\overline{\texttt{f}}\texttt{); this.}\overline{\texttt{f}}\texttt{=}\overline{\texttt{f}}\texttt{;\}} \\
\texttt{M} & ::= & \texttt{<}\overline{\texttt{X}} \triangleleft \overline{\texttt{N}}\texttt{> T m(}\overline{\texttt{T}}\ \overline{\texttt{x}}\texttt{)\{ return e; \}} \\
\texttt{e} & ::= & \texttt{x} \mid \texttt{e.f} \mid \texttt{e.m<}\overline{\texttt{T}}\texttt{>(}\overline{\texttt{e}}\texttt{)} \mid \texttt{new N(}\overline{\texttt{e}}\texttt{)} \mid \texttt{(N)e}
\end{array}
$$

- $\triangleleft$ is abbreviation of keyword `extends`
- we allow `C<>` and `m<>` to be abbreviated as `C` and `m`
- the bound of type variable may be type expression, even recursive! (`<X extends C<X>>`)
- ...or mutually recursive if there are several bounds (`<X extends C<Y>, Y extends D<X>>`)

$$\texttt{C} \trianglelefteq \texttt{C} \qquad \frac{\texttt{C} \trianglelefteq \texttt{D} \qquad \texttt{D} \trianglelefteq \texttt{E}}{\texttt{C} \trianglelefteq \texttt{E}}$$

$$\frac{\texttt{class C<}\overline{\texttt{X}} \triangleleft \overline{\texttt{N}}\texttt{>} \triangleleft \texttt{D<}\overline{\texttt{T}}\texttt{> \{...\}}}{\texttt{C} \trianglelefteq \texttt{D}}$$

- subclassing $\neq$ subtyping
- subclassing is relation between class names

$$fields(\texttt{Object}) = \bullet \qquad \text{(F-OBJECT)}$$

$$\frac{\texttt{class C<}\overline{\texttt{X}} \triangleleft \overline{\texttt{N}}\texttt{>} \triangleleft \texttt{N \{}\overline{\texttt{S}}\ \overline{\texttt{f}}\texttt{; K}\ \overline{\texttt{M}}\texttt{\}} \qquad fields([\overline{\texttt{T}}/\overline{\texttt{X}}]\texttt{N}) = \overline{\texttt{U}}\ \overline{\texttt{g}}}{fields(\texttt{C<}\overline{\texttt{T}}\texttt{>}) = \overline{\texttt{U}}\ \overline{\texttt{g}},\ [\overline{\texttt{T}}/\overline{\texttt{X}}]\overline{\texttt{S}}\ \overline{\texttt{f}}} \qquad \text{(F-CLASS)}$$

Featherweight Generic Java *method type lookup*

$$\frac{\begin{array}{c}\texttt{class C<}\overline{\texttt{X}} \triangleleft \overline{\texttt{N}}\texttt{>} \triangleleft \texttt{N \{}\overline{\texttt{S}}\ \overline{\texttt{f}}\texttt{; K }\overline{\texttt{M}}\texttt{\}} \\ \texttt{<}\overline{\texttt{Y}} \triangleleft \overline{\texttt{P}}\texttt{> U m(}\overline{\texttt{U}}\ \overline{\texttt{x}}\texttt{)\{ return e; \}} \in \overline{\texttt{M}}\end{array}}{\mathit{mtype}(\texttt{m}, \texttt{C<}\overline{\texttt{T}}\texttt{>}) = [\overline{\texttt{T}}/\overline{\texttt{X}}](\texttt{<}\overline{\texttt{Y}} \triangleleft \overline{\texttt{P}}\texttt{>}\overline{\texttt{U}} \rightarrow \texttt{U})} \quad \text{(MT-Class)}$$

$$\frac{\texttt{class C<}\overline{\texttt{X}} \triangleleft \overline{\texttt{N}}\texttt{>} \triangleleft \texttt{N \{}\overline{\texttt{S}}\ \overline{\texttt{f}}\texttt{; K }\overline{\texttt{M}}\texttt{\}} \qquad \texttt{m} \notin \overline{\texttt{M}}}{\mathit{mtype}(\texttt{m}, \texttt{C<}\overline{\texttt{T}}\texttt{>}) = \mathit{mtype}(\texttt{m}, [\overline{\texttt{T}}/\overline{\texttt{X}}]\texttt{N})} \quad \text{(MT-Super)}$$

$$\frac{\texttt{class C<}\overline{\texttt{X}} \triangleleft \overline{\texttt{N}}\texttt{>} \triangleleft \texttt{N \{}\overline{\texttt{S}}\ \overline{\texttt{f}}\texttt{; K }\overline{\texttt{M}}\texttt{\}} \qquad \texttt{<}\overline{\texttt{Y}} \triangleleft \overline{\texttt{P}}\texttt{> U m(}\overline{\texttt{U}}\ \overline{\texttt{x}}\texttt{)\{ return e}_0\texttt{; \}} \in \overline{\texttt{M}}}{\mathit{mbody}(\texttt{m<}\overline{\texttt{V}}\texttt{>, C<}\overline{\texttt{T}}\texttt{>}) = \overline{\texttt{x}}\texttt{.}[\overline{\texttt{T}}/\overline{\texttt{X}}, \overline{\texttt{V}}/\overline{\texttt{Y}}]\texttt{e}_0} \quad \text{(MB-Class)}$$

$$\frac{\texttt{class C<}\overline{\texttt{X}} \triangleleft \overline{\texttt{N}}\texttt{>} \triangleleft \texttt{N \{}\overline{\texttt{S}}\ \overline{\texttt{f}}\texttt{; K }\overline{\texttt{M}}\texttt{\}} \qquad \texttt{m} \notin \overline{\texttt{M}}}{\mathit{mbody}(\texttt{m<}\overline{\texttt{V}}\texttt{>, C<}\overline{\texttt{T}}\texttt{>}) = \mathit{mbody}(\texttt{m<}\overline{\texttt{V}}\texttt{>}, [\overline{\texttt{T}}/\overline{\texttt{X}}]\texttt{N})} \quad \text{(MB-Super)}$$

$$\begin{aligned} bound_{\Delta}(\mathtt{X}) &= \Delta(\mathtt{X}) \\ bound_{\Delta}(\mathtt{N}) &= \mathtt{N} \end{aligned}$$

- Δ is finite mapping from type variables to nonvariable types, written $\overline{X} <: \overline{N}$
- we write $bound_{\Delta}(T)$ for upper bound of T in Δ

$$\Delta \vdash \mathtt{T} <: \mathtt{T} \qquad \text{(S-Refl)}$$

$$\frac{\Delta \vdash \mathtt{S} <: \mathtt{T} \qquad \Delta \vdash \mathtt{T} <: \mathtt{U}}{\Delta \vdash \mathtt{S} <: \mathtt{U}} \qquad \text{(S-Trans)}$$

$$\Delta \vdash \mathtt{X} <: \Delta(\mathtt{X}) \qquad \text{(S-Var)}$$

$$\frac{\mathtt{class}\ \mathtt{C}\mathtt{<}\overline{\mathtt{X}} \triangleleft \overline{\mathtt{N}}\mathtt{>} \triangleleft \mathtt{N}\ \{\ldots\}}{\Delta \vdash \mathtt{C}\mathtt{<}\overline{\mathtt{T}}\mathtt{>} <: [\overline{\mathtt{T}}/\overline{\mathtt{X}}]\mathtt{N}} \qquad \text{(S-Class)}$$

- type parameters are invariant with regard to subtyping
- type parameter can be both argument and result type of method
- $\Delta \vdash \overline{\mathtt{T}} <: \overline{\mathtt{U}}$ does not imply $\Delta \vdash \mathtt{C} < \overline{\mathtt{T}} > \ <: \ \mathtt{C} < \overline{\mathtt{U}} >$

$$\Delta \vdash \texttt{Object ok} \qquad \text{(WF-OBJECT)}$$

$$\frac{\texttt{X} \in dom(\Delta)}{\Delta \vdash \texttt{X ok}} \qquad \text{(WF-VAR)}$$

$$\frac{\texttt{class C<}\overline{\texttt{X}} \triangleleft \overline{\texttt{N}}\texttt{>} \triangleleft \texttt{N \{...\}} \qquad \Delta \vdash \overline{\texttt{T}}\ \text{ok} \qquad \Delta \vdash \overline{\texttt{T}} <: [\overline{\texttt{T}}/\overline{\texttt{X}}]\overline{\texttt{N}}}{\Delta \vdash \texttt{C<}\overline{\texttt{T}}\texttt{> ok}} \qquad \text{(WF-CLASS)}$$

- we perform a simultaneous substitution, permitting recursion and mutual recursion between variables and bounds
- type environment Δ is well formed if $\Delta \vdash \Delta(X)$ *ok* for all $X \in dom(\Delta)$
- environment Γ is well formed with respect to Δ, written $\Delta \vdash \Gamma$ *ok*, if $\Delta \vdash \Gamma(x)$ *ok* for all $x \in dom(\Gamma)$

$$\Delta; \Gamma \vdash \mathtt{x} : \Gamma(\mathtt{x}) \qquad \text{(GT-VAR)}$$

$$\frac{\Delta; \Gamma \vdash \mathtt{e}_0 : \mathtt{T}_0 \qquad \mathit{fields}(\mathit{bound}_\Delta(\mathtt{T}_0)) = \overline{\mathtt{T}}\ \overline{\mathtt{f}}}{\Delta; \Gamma \vdash \mathtt{e}_0.\mathtt{f}_i : \mathtt{T}_i} \qquad \text{(GT-FIELD)}$$

$$\frac{\begin{array}{c}\Delta; \Gamma \vdash \mathtt{e}_0 : \mathtt{T}_0 \qquad \mathit{mtype}(\mathtt{m}, \mathit{bound}_\Delta(\mathtt{T}_0)) = \mathtt{<}\overline{\mathtt{Y}} \lhd \overline{\mathtt{P}}\mathtt{>}\overline{\mathtt{U}} \rightarrow \mathtt{U} \\ \Delta \vdash \overline{\mathtt{V}}\ \text{ok} \qquad \Delta \vdash \overline{\mathtt{V}} \mathrel{<:} [\overline{\mathtt{V}}/\overline{\mathtt{Y}}]\overline{\mathtt{P}} \qquad \Delta; \Gamma \vdash \overline{\mathtt{e}} : \overline{\mathtt{S}} \qquad \Delta \vdash \overline{\mathtt{S}} \mathrel{<:} [\overline{\mathtt{V}}/\overline{\mathtt{Y}}]\overline{\mathtt{U}}\end{array}}{\Delta; \Gamma \vdash \mathtt{e}_0.\mathtt{m}\mathtt{<}\overline{\mathtt{V}}\mathtt{>}(\overline{\mathtt{e}}) : [\overline{\mathtt{V}}/\overline{\mathtt{Y}}]\mathtt{U}} \qquad \text{(GT-INVK)}$$

$$\frac{\Delta \vdash \mathtt{N}\ \text{ok} \qquad \mathit{fields}(\mathtt{N}) = \overline{\mathtt{T}}\ \overline{\mathtt{f}} \qquad \Delta; \Gamma \vdash \overline{\mathtt{e}} : \overline{\mathtt{S}} \qquad \Delta \vdash \overline{\mathtt{S}} \mathrel{<:} \overline{\mathtt{T}}}{\Delta; \Gamma \vdash \mathtt{new}\ \mathtt{N}(\overline{\mathtt{e}}) : \mathtt{N}} \qquad \text{(GT-NEW)}$$

Featherweight Generic Java *typing casts*

$$\frac{dcast(\mathtt{C},\ \mathtt{D}) \qquad dcast(\mathtt{D},\ \mathtt{E})}{dcast(\mathtt{C},\ \mathtt{E})}$$

$$\frac{\mathtt{class\ C}\texttt{<}\overline{\mathtt{X}} \triangleleft \overline{\mathtt{N}}\texttt{>} \triangleleft \mathtt{D}\texttt{<}\overline{\mathtt{T}}\texttt{>}\ \{\ldots\} \qquad \overline{\mathtt{X}} = FV(\overline{\mathtt{T}})}{dcast(\mathtt{C}, \mathtt{D})}$$

($FV(\overline{\mathtt{T}})$ denotes the set of type variables in $\overline{\mathtt{T}}$.)

$$\frac{\Delta;\Gamma \vdash \mathtt{e}_0 : \mathtt{T}_0 \qquad \Delta \vdash bound_\Delta(\mathtt{T}_0) \texttt{<:}\ \mathtt{N}}{\Delta;\Gamma \vdash (\mathtt{N})\mathtt{e}_0 : \mathtt{N}} \qquad \text{(GT-UCast)}$$

$$\frac{\begin{array}{ccc}\Delta;\Gamma \vdash \mathtt{e}_0 : \mathtt{T}_0 & \Delta \vdash \mathtt{N}\ \mathrm{ok} & \Delta \vdash \mathtt{N} \texttt{<:}\ bound_\Delta(\mathtt{T}_0) \\ \mathtt{N} = \mathtt{C}\texttt{<}\overline{\mathtt{T}}\texttt{>} & bound_\Delta(\mathtt{T}_0) = \mathtt{D}\texttt{<}\overline{\mathtt{U}}\texttt{>} & dcast(\mathtt{C},\ \mathtt{D})\end{array}}{\Delta;\Gamma \vdash (\mathtt{N})\mathtt{e}_0 : \mathtt{N}} \qquad \text{(GT-DCast)}$$

$$\frac{\begin{array}{cccc}\Delta;\Gamma \vdash \mathtt{e}_0 : \mathtt{T}_0 & \Delta \vdash \mathtt{N}\ \mathrm{ok} & \mathtt{N} = \mathtt{C}\texttt{<}\overline{\mathtt{T}}\texttt{>} & bound_\Delta(\mathtt{T}_0) = \mathtt{D}\texttt{<}\overline{\mathtt{U}}\texttt{>} \\ \mathtt{C} \not\trianglelefteq \mathtt{D} & \mathtt{D} \not\trianglelefteq \mathtt{C} & stupid\ warning & \end{array}}{\Delta;\Gamma \vdash (\mathtt{N})\mathtt{e}_0 : \mathtt{N}} \qquad \text{(GT-SCast)}$$

dcast ensures us that the result of the cast does not depend on reduction semantics used (type passing or type erasure).

```
class List<X extends Object> extends Object { ... }
class LinkedList<X extends Object> extends List<X> { ... }
```

- if *o* has type `Object`, then `(List<C>)o` is not permitted
- if at runtime *o* is bound to `new List<D>()`, then cast would fail in the type-passing semantics, but succeed in the erasure semantics, since `(List<C>)o` erases to `(List)o`, while both `new List<C>()` and `new List<D>()` erases to `new List()`
- if *cl* has type `List<C>`, then cast `(LinkedList<C>)cl` is permitted, since the type-passing and erased versions of the cast are guaranteed to either both succeed or both fail

Featherweight Generic Java *typing methods and classes*

$$\frac{mtype(\mathtt{m},\mathtt{N}) = \mathtt{<}\overline{\mathtt{Z}} \triangleleft \overline{\mathtt{Q}}\mathtt{>}\overline{\mathtt{U}}\to\mathtt{U}_0 \text{ implies } \overline{\mathtt{P}},\overline{\mathtt{T}} = [\overline{\mathtt{Y}}/\overline{\mathtt{Z}}](\overline{\mathtt{Q}},\overline{\mathtt{U}}) \text{ and } \overline{\mathtt{Y}}\mathtt{<:}\overline{\mathtt{P}} \vdash \mathtt{T}_0 \mathtt{<:} [\overline{\mathtt{Y}}/\overline{\mathtt{Z}}]\mathtt{U}_0}{override(\mathtt{m},\mathtt{N},\mathtt{<}\overline{\mathtt{Y}} \triangleleft \overline{\mathtt{P}}\mathtt{>}\overline{\mathtt{T}}\to\mathtt{T}_0)}$$

$$\frac{\begin{array}{c}\Delta = \overline{\mathtt{X}}\mathtt{<:}\overline{\mathtt{N}},\ \overline{\mathtt{Y}}\mathtt{<:}\overline{\mathtt{P}} \qquad \Delta \vdash \overline{\mathtt{T}},\ \mathtt{T},\ \overline{\mathtt{P}}\ \text{ok} \\ \Delta; \overline{\mathtt{x}} : \overline{\mathtt{T}},\ \mathtt{this} : \mathtt{C<}\overline{\mathtt{X}}\mathtt{>} \vdash \mathtt{e}_0 : \mathtt{S} \qquad \Delta \vdash \mathtt{S} \mathtt{<:} \mathtt{T} \\ \mathtt{class\ C<}\overline{\mathtt{X}} \triangleleft \overline{\mathtt{N}}\mathtt{>} \triangleleft \mathtt{N}\ \{\ldots\} \qquad override(\mathtt{m},\mathtt{N},\mathtt{<}\overline{\mathtt{Y}} \triangleleft \overline{\mathtt{P}}\mathtt{>}\overline{\mathtt{T}}\to\mathtt{T})\end{array}}{\mathtt{<}\overline{\mathtt{Y}} \triangleleft \overline{\mathtt{P}}\mathtt{>}\ \mathtt{T\ m(}\overline{\mathtt{T}}\ \overline{\mathtt{x}}\mathtt{)\{\ return\ e_0;\ \}\ OK\ IN\ C<}\overline{\mathtt{X}} \triangleleft \overline{\mathtt{N}}\mathtt{>}} \quad \text{(GT-METHOD)}$$

$$\frac{\begin{array}{c}\overline{\mathtt{X}}\mathtt{<:}\overline{\mathtt{N}} \vdash \overline{\mathtt{N}},\ \mathtt{N},\ \overline{\mathtt{T}}\ \text{ok} \qquad fields(\mathtt{N}) = \overline{\mathtt{U}}\ \overline{\mathtt{g}} \qquad \overline{\mathtt{M}}\ \mathtt{OK\ IN\ C<}\overline{\mathtt{X}} \triangleleft \overline{\mathtt{N}}\mathtt{>} \\ \mathtt{K} = \mathtt{C(}\overline{\mathtt{U}}\ \overline{\mathtt{g}},\ \overline{\mathtt{T}}\ \overline{\mathtt{f}}\mathtt{)\{super(}\overline{\mathtt{g}}\mathtt{);\ this.}\overline{\mathtt{f}}\ \mathtt{=}\ \overline{\mathtt{f}}\mathtt{;\}}\end{array}}{\mathtt{class\ C<}\overline{\mathtt{X}} \triangleleft \overline{\mathtt{N}}\mathtt{>} \triangleleft \mathtt{N}\ \{\overline{\mathtt{T}}\ \overline{\mathtt{f}};\ \mathtt{K}\ \overline{\mathtt{M}}\}\ \mathtt{OK}} \quad \text{(GT-CLASS)}$$

- in FGJ unlike to FJ, covariant overriding on the method result type is allowed

Featherweight Generic Java *operational semantics*

$$\frac{\mathit{fields}(\mathtt{N}) = \overline{\mathtt{T}}\ \overline{\mathtt{f}}}{(\mathtt{new\ N}(\overline{\mathtt{e}})).\mathtt{f}_i \longrightarrow \mathtt{e}_i} \qquad \text{(GR-Field)}$$

$$\frac{\mathit{mbody}(\mathtt{m}\texttt{<}\overline{\mathtt{V}}\texttt{>}, \mathtt{N}) = \overline{\mathtt{x}}.\mathtt{e}_0}{(\mathtt{new\ N}(\overline{\mathtt{e}})).\mathtt{m}\texttt{<}\overline{\mathtt{V}}\texttt{>}(\overline{\mathtt{d}}) \longrightarrow [\overline{\mathtt{d}}/\overline{\mathtt{x}}, \mathtt{new\ N}(\overline{\mathtt{e}})/\mathtt{this}]\mathtt{e}_0} \qquad \text{(GR-Invk)}$$

$$\frac{\emptyset \vdash \mathtt{N} <: \mathtt{P}}{(\mathtt{P})(\mathtt{new\ N}(\overline{\mathtt{e}})) \longrightarrow \mathtt{new\ N}(\overline{\mathtt{e}})} \qquad \text{(GR-Cast)}$$

Featherweight Generic Java *operational semantics*

$$\frac{\mathtt{e}_0 \longrightarrow \mathtt{e}_0{}'}{\mathtt{e}_0\mathtt{.f} \longrightarrow \mathtt{e}_0{}'\mathtt{.f}} \qquad \text{(GRC-Field)}$$

$$\frac{\mathtt{e}_0 \longrightarrow \mathtt{e}_0{}'}{\mathtt{e}_0\mathtt{.m<}\overline{\mathtt{T}}\mathtt{>(}\overline{\mathtt{e}}\mathtt{)} \longrightarrow \mathtt{e}_0{}'\mathtt{.m<}\overline{\mathtt{T}}\mathtt{>(}\overline{\mathtt{e}}\mathtt{)}} \qquad \text{(GRC-Inv-Recv)}$$

$$\frac{\mathtt{e}_i \longrightarrow \mathtt{e}_i{}'}{\mathtt{e}_0\mathtt{.m<}\overline{\mathtt{T}}\mathtt{>(}\ldots\mathtt{,e}_i\mathtt{,}\ldots\mathtt{)} \longrightarrow \mathtt{e}_0\mathtt{.m<}\overline{\mathtt{T}}\mathtt{>(}\ldots\mathtt{e}_i{}'\mathtt{,}\ldots\mathtt{)}} \qquad \text{(GRC-Inv-Arg)}$$

$$\frac{\mathtt{e}_i \longrightarrow \mathtt{e}_i{}'}{\mathtt{new\ N(}\ldots\mathtt{,e}_i\mathtt{,}\ldots\mathtt{)} \longrightarrow \mathtt{new\ N(}\ldots\mathtt{e}_i{}'\mathtt{,}\ldots\mathtt{)}} \qquad \text{(GRC-New-Arg)}$$

$$\frac{\mathtt{e}_0 \longrightarrow \mathtt{e}_0{}'}{\mathtt{(N)e}_0 \longrightarrow \mathtt{(N)e}_0{}'} \qquad \text{(GRC-Cast)}$$

Theorem 7 (Subject Reduction)

If $\Delta;\Gamma \vdash e : T$ and $e \longrightarrow e'$, then $\Delta;\Gamma \vdash e' : T'$ for some T' such that $\Delta \vdash T' <: T$.

Theorem 8 (Progress)

Suppose e is a well-typed expression.

- *If e includes* $\texttt{new N}_0(\overline{\texttt{e}}).\texttt{f}$ *as a subexpression, then* $\mathit{fields}(N_0) = \overline{T}\ \overline{f}$ *and* $f \in \overline{f}$ *for some* $\overline{T}$ *and* $\overline{f}$.
- *If e includes* $\texttt{new N}_0(\overline{\texttt{e}}).\texttt{m} < \overline{\texttt{V}} > (\overline{\texttt{d}})$ *as a subexpression, then* $\mathit{mbody}(m < \overline{V} >, N_0) = \overline{x}.e_0$ *and* $\#(\overline{x}) = \#(\overline{d})$ *for some* $\overline{x}$ *and* $\overline{e_0}$.

As we did for FJ, we will give the definition of FGJ values:

$$\mathtt{w} ::= \mathtt{new\ N}(\overline{\mathtt{w}})$$

Theorem 9 (FGJ Type Soundness)

If $\emptyset;\emptyset \vdash e : T$ *and* $e \longrightarrow^* e'$ *with* e' *a normal form, then* e' *is either an FGJ value* w *with* $\emptyset;\emptyset \vdash w : S$ *and* $\emptyset \vdash S <: T$*, or an expression containing* $(\mathtt{P})\ \mathtt{new\ N}(\overline{\mathtt{e}})$ *where* $\emptyset \vdash N \not<: P$*.*

Intuitively, FGJ can be used to typecheck and execute FJ programs without changing their meanings.

Lemma 10

If CT is an FJ class table, then $fields_{FJ}(C) = fields_{FGJ}(C)$ *for all* $C \in dom(CT)$.

Lemma 11

Suppose CT is an FJ class table. Then $mtype_{FJ}(m, C) = \overline{C} \to C$ *if and only if* $mtype_{FGJ}(m, C) = \overline{C} \to C$. *Similarly,* $mbody_{FJ}(m, C) = \overline{x}.e$ *if and only if* $mbody_{FGJ}(m, C) = \overline{x}.e$

Theorem 12 (Backward Compatibility)

If an FJ program (e, CT) is well typed under FJ, then is is also well typed under FGJ. Moreover, for all FJ programs e and e' (whether well typed or not) $e \longrightarrow_{FJ} e'$ if and only if $e \longrightarrow_{FGJ} e'$.

Type erasure *idea*

Idea is to translate FGJ expression to FJ with erasing all information about type parameters.
That's the way how generics are implemented in JVM.

Type erasure *example*

Class `Pair<X, Y>` from previous example erases to the following:

```
class Pair extends Object {
    Object fst;
    Object snd;
    Pair(Object fst, Object snd) {
        super(); this.fst = fst; this.snd = snd;
    }
    Pair setfst(Object newfst) {
        return new Pair(newfst, this.snd);
    }
}
```

Type erasure *example*

Similarly, the field selection:

```
new Pair<A, B>(new A(), new B()).snd
```

erases to:

```
(B) new Pair(new A(), new B()).snd
```

- downcast `(B)` is inserted to recover type information from the original program
- we call such downcasts inserted by erasure *syntetic*
- we would want them not to fail at runtime ;)

Type erasure *erasure of types*

$$|T|_\Delta = C$$

where

$$bound_\Delta(T) = C < \overline{T} >$$

Type erasure *auxilliary functions*

$$\mathit{fieldsmax}(\texttt{Object}) = \bullet$$

$$\frac{\texttt{class C<}\overline{\texttt{X}} \triangleleft \overline{\texttt{N}}\texttt{>} \triangleleft \texttt{D<}\overline{\texttt{U}}\texttt{> \{}\overline{\texttt{T}}\ \overline{\texttt{f}}\texttt{; ... \}} \qquad \Delta = \overline{\texttt{X}}\texttt{<:}\overline{\texttt{N}} \qquad \overline{\texttt{C}}\ \overline{\texttt{g}} = \mathit{fieldsmax}(\texttt{D})}{\mathit{fieldsmax}(\texttt{C}) = \overline{\texttt{C}}\ \overline{\texttt{g}}, |\overline{\texttt{T}}|_\Delta\ \overline{\texttt{f}}}$$

If $\mathit{fieldsmax}(C) = \overline{D}\,\overline{f}$, then $\mathit{fieldsmax}(C)(f_i) = D_i$.

Type erasure *auxilliary functions*

$$\frac{\texttt{class C<}\overline{\texttt{X}} \triangleleft \overline{\texttt{N}}\texttt{>} \triangleleft \texttt{D<}\overline{\texttt{U}}\texttt{> \{...\}} \qquad \texttt{<}\overline{\texttt{Y}} \triangleleft \overline{\texttt{P}}\texttt{>}\overline{\texttt{T}}\to\texttt{T} = \mathit{mtype}(\texttt{m}, \texttt{D<}\overline{\texttt{U}}\texttt{>})}{\mathit{mtypemax}(\texttt{m}, \texttt{C}) = \mathit{mtypemax}(\texttt{m}, \texttt{D})}$$

$$\frac{\begin{array}{c}\texttt{class C<}\overline{\texttt{X}} \triangleleft \overline{\texttt{N}}\texttt{>} \triangleleft \texttt{D<}\overline{\texttt{U}}\texttt{> \{... } \overline{\texttt{M}}\texttt{ \}} \qquad \mathit{mtype}(\texttt{m}, \texttt{D<}\overline{\texttt{U}}\texttt{>}) \text{ undefined} \\ \texttt{<}\overline{\texttt{Y}} \triangleleft \overline{\texttt{P}}\texttt{> T m(}\overline{\texttt{T}}\ \overline{\texttt{x}}\texttt{)\{ return e; \}} \in \overline{\texttt{M}} \qquad \Delta = \overline{\texttt{X}}\texttt{<:}\overline{\texttt{N}}, \overline{\texttt{Y}}\texttt{<:}\overline{\texttt{P}}\end{array}}{\mathit{mtypemax}(\texttt{m}, \texttt{C}) = |\overline{\texttt{T}}|_\Delta \to |\texttt{T}|_\Delta}$$

Type erasure *erasure of expressions*

$$|\mathtt{x}|_{\Delta,\Gamma} = \mathtt{x} \qquad \text{(E-VAR)}$$

$$\frac{\Delta;\Gamma \vdash \mathtt{e}_0\mathtt{.f} : \mathtt{T} \qquad \Delta;\Gamma \vdash \mathtt{e}_0 : \mathtt{T}_0 \qquad \mathit{fieldsmax}(|\mathtt{T}_0|_\Delta)(\mathtt{f}) = |\mathtt{T}|_\Delta}{|\mathtt{e}_0\mathtt{.f}|_{\Delta,\Gamma} = |\mathtt{e}_0|_{\Delta,\Gamma}\mathtt{.f}} \qquad \text{(E-FIELD)}$$

$$\frac{\Delta;\Gamma \vdash \mathtt{e}_0\mathtt{.f} : \mathtt{T} \qquad \Delta;\Gamma \vdash \mathtt{e}_0 : \mathtt{T}_0 \qquad \mathit{fieldsmax}(|\mathtt{T}_0|_\Delta)(\mathtt{f}) \neq |\mathtt{T}|_\Delta}{|\mathtt{e}_0\mathtt{.f}|_{\Delta,\Gamma} = \mathtt{(}|\mathtt{T}|_\Delta\mathtt{)}^s|\mathtt{e}_0|_{\Delta,\Gamma}\mathtt{.f}} \qquad \text{(E-FIELD-CAST)}$$

$$\frac{\Delta;\Gamma \vdash \mathtt{e}_0\mathtt{.m<}\overline{\mathtt{V}}\mathtt{>(}\overline{\mathtt{e}}\mathtt{)} : \mathtt{T} \qquad \Delta;\Gamma \vdash \mathtt{e}_0 : \mathtt{T}_0 \qquad \mathit{mtypemax}(\mathtt{m}, |\mathtt{T}_0|_\Delta) = \overline{\mathtt{C}}\rightarrow\mathtt{D} \qquad \mathtt{D} = |\mathtt{T}|_\Delta}{|\mathtt{e}_0\mathtt{.m<}\overline{\mathtt{V}}\mathtt{>(}\overline{\mathtt{e}}\mathtt{)}|_{\Delta,\Gamma} = |\mathtt{e}_0|_{\Delta,\Gamma}\mathtt{.m(}|\overline{\mathtt{e}}|_{\Delta,\Gamma}\mathtt{)}} \qquad \text{(E-INVK)}$$

Type erasure *erasure of expressions*

$$\frac{\Delta;\Gamma \vdash \texttt{e}_0\texttt{.m<}\overline{\texttt{V}}\texttt{>(}\overline{\texttt{e}}\texttt{)} : \texttt{T} \qquad \Delta;\Gamma \vdash \texttt{e}_0 : \texttt{T}_0 \\ \mathit{mtypemax}(\texttt{m}, |\texttt{T}_0|_\Delta) = \overline{\texttt{C}}\rightarrow\texttt{D} \qquad \texttt{D} \neq |\texttt{T}|_\Delta}{|\texttt{e}_0\texttt{.m<}\overline{\texttt{V}}\texttt{>(}\overline{\texttt{e}}\texttt{)}|_{\Delta,\Gamma} = \texttt{(}|\texttt{T}|_\Delta\texttt{)}^s|\texttt{e}_0|_{\Delta,\Gamma}\texttt{.m(}|\overline{\texttt{e}}|_{\Delta,\Gamma}\texttt{)}} \quad \text{(E-INVK-CAST)}$$

$$|\texttt{new N(}\overline{\texttt{e}}\texttt{)}|_{\Delta,\Gamma} = \texttt{new } |\texttt{N}|_\Delta\texttt{(}|\overline{\texttt{e}}|_{\Delta,\Gamma}\texttt{)} \quad \text{(E-NEW)}$$

$$|\texttt{(N)e}_0|_{\Delta,\Gamma} = \texttt{(}|\texttt{N}|_\Delta\texttt{)}\ |\texttt{e}_0|_{\Delta,\Gamma} \quad \text{(E-CAST)}$$

Type erasure *erasure of methods and classes*

$$\dfrac{\Gamma = \overline{\mathtt{x}}:\overline{\mathtt{T}}, \mathtt{this} : \mathtt{C}\texttt{<}\overline{\mathtt{X}}\texttt{>} \qquad \Delta = \overline{\mathtt{X}}\texttt{<:}\overline{\mathtt{N}}, \overline{\mathtt{Y}}\texttt{<:}\overline{\mathtt{P}} \\ \mathit{mtypemax}(\mathtt{m}, \mathtt{C}) = \overline{\mathtt{D}}\to\mathtt{D} \qquad \mathtt{e}_i = \begin{cases} \mathtt{x}_i{}' & \text{if } \mathtt{D}_i = |\mathtt{T}_i|_\Delta \\ (|\mathtt{T}_i|_\Delta)^s\mathtt{x}_i{}' & \text{otherwise} \end{cases}}{|\texttt{<}\overline{\mathtt{Y}} \triangleleft \overline{\mathtt{P}}\texttt{>}\ \mathtt{T}\ \mathtt{m}(\overline{\mathtt{T}}\ \overline{\mathtt{x}})\{\ \mathtt{return}\ \mathtt{e}_0;\ \}|_{\overline{\mathtt{X}}\texttt{<:}\overline{\mathtt{N}},\mathtt{C}} = \mathtt{D}\ \mathtt{m}(\overline{\mathtt{D}}\ \overline{\mathtt{x}}')\{\ \mathtt{return}\ [\overline{\mathtt{e}}/\overline{\mathtt{x}}]|\mathtt{e}_0|_{\Delta,\Gamma};\ \}}$$

(E-Method)

$$\begin{aligned} &|\mathtt{C}(\overline{\mathtt{U}}\ \overline{\mathtt{g}},\ \overline{\mathtt{T}}\ \overline{\mathtt{f}})\ \{\mathtt{super}(\overline{\mathtt{g}});\ \mathtt{this}.\overline{\mathtt{f}}\ =\ \overline{\mathtt{f}};\}|_\mathtt{C} \\ = &\ \mathtt{C}(\mathit{fieldsmax}(\mathtt{C}))\ \{\mathtt{super}(\overline{\mathtt{g}});\ \mathtt{this}.\overline{\mathtt{f}}\ =\ \overline{\mathtt{f}};\} \end{aligned}$$

(E-Constructor)

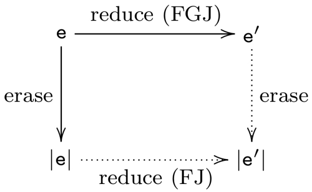
$$\dfrac{\Delta = \overline{\mathtt{X}}\texttt{<:}\overline{\mathtt{N}}}{\begin{aligned} &|\mathtt{class}\ \mathtt{C}\texttt{<}\overline{\mathtt{X}}\ \mathtt{extends}\ \overline{\mathtt{N}}\texttt{>}\ \mathtt{extends}\ \mathtt{N}\ \{\overline{\mathtt{T}}\ \overline{\mathtt{f}};\ \mathtt{K}\ \overline{\mathtt{M}}\}| \\ = &\ \mathtt{class}\ \mathtt{C}\ \mathtt{extends}\ |\mathtt{N}|_\Delta\{|\overline{\mathtt{T}}|_\Delta\ \overline{\mathtt{f}};\ |\mathtt{K}|_\mathtt{C}\ |\overline{\mathtt{M}}|_{\Delta,\mathtt{C}}\} \end{aligned}}$$

(E-Class)

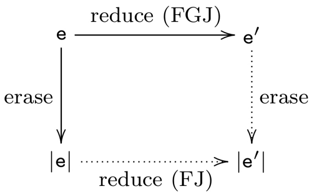
Theorem 13 (Erasure Preserves Typing)

If an FGJ class table CT is ok and $\Delta;\Gamma \vdash_{FGJ} e : T$*, then* $|CT|$ *is ok using the FJ typing rules and* $|\Gamma|_{\Delta} \vdash_{FJ} |e|_{\Delta;\Gamma} : |T|_{\Delta}$*. Moreover, every synthetic cast in* $|CT|$ *and* $|e|_{\Delta;\Gamma}$ *does not involve a stupid warning.*

We would intuitively expect that erasure from FGJ to FJ preserves reduction behaviour of FGJ programs, as in the diagram below.



Type erasure *properties*



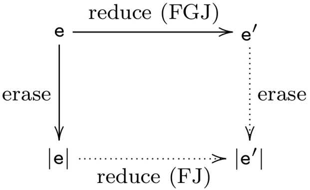
This is actually not quite true. Let's pick:

$\texttt{e} = \texttt{new Pair} < \texttt{A}, \texttt{B} > (\texttt{a}, \texttt{b}).\texttt{fst}$
$|\texttt{e}| = (\texttt{A})^{s}\ \texttt{new Pair}(|\texttt{a}|, |\texttt{b}|).\texttt{fst}$

$e \rightarrow^{*}_{FGJ} \texttt{a}$, but $|e| \rightarrow^{*}_{FJ} (A)^{s}\ |\texttt{a}|$

In this example, FJ expression reduced from $|e|$ has more synthetic casts than $|e'|$. This is not always the case.

Type erasure *properties*



In the example below:

$$\mathtt{e} = \mathtt{new\ Pair} < \mathtt{A}, \mathtt{B} > (\mathtt{a}, \mathtt{b}).\mathtt{setfst} < \mathtt{B} > (\mathtt{b}')$$
$$|\mathtt{e}| = \mathtt{new\ Pair}(|\mathtt{a}|, |\mathtt{b}|).\mathtt{setfst}(|\mathtt{b}'|)$$

$$e \longrightarrow^{*}_{FGJ} \mathtt{new\ Pair} < \mathtt{B}, \mathtt{B} > (\mathtt{b}', \mathtt{new\ Pair} < \mathtt{A}, \mathtt{B} > (\mathtt{a}, \mathtt{b}).\mathtt{snd})$$
$$|e| \longrightarrow^{*}_{FJ} \mathtt{new\ Pair}(|\mathtt{b}'|, \mathtt{new\ Pair}(|\mathtt{a}|, |\mathtt{b}|).\mathtt{snd}$$

which has fewer synthetic casts than

$$\mathtt{new\ Pair}(|\mathtt{b}'|, (\mathtt{B})^{s}\ \mathtt{new\ Pair}(|\mathtt{a}|, |\mathtt{b}|).\mathtt{snd})$$

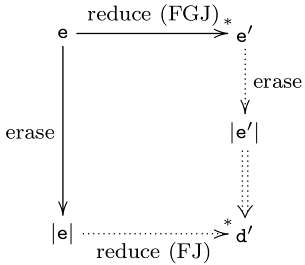
Definition 14 (Expression expansion)

Suppose $\Gamma \vdash_{FJ} e : C$. Let us call d an *expansion* of e under Γ, written $\Gamma \vdash e \overset{exp}{\Longrightarrow} d$ if d is obtained from e by some combination of:

- addition zero or more synthetic upcasts,
- replacement of some synthetic casts $(\mathtt{D})^s$ with $(\mathtt{C})^s$ where $D <: C$,
- removal of some synthetic casts

...and $\Gamma \vdash_{FJ} d : D$ for some D.

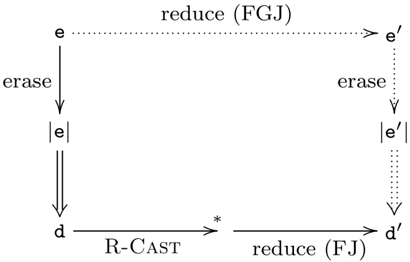
Type erasure *properties*



Theorem 15 (Erasure Preserves Reduction Modulo Expansion)

If $\Delta; \Gamma \vdash e : T$ *and* $e \longrightarrow^{*}_{FGJ} e'$*, then there exists some expression* d' *such that* $|\Gamma|_{\Delta} \vdash |e'|_{\Delta,\Gamma} \overset{exp}{\Longrightarrow} d'$ *and* $|e|_{\Delta,\Gamma} \longrightarrow^{*}_{FJ} d'$*.*

Type erasure *properties*



Theorem 16 (Erased Program Reflects FGJ Execution)

Suppose that $\Delta; \Gamma \vdash e : T$ *and* $|\Gamma|_\Delta \vdash |e|_{\Delta,\Gamma} \overset{exp}{\Longrightarrow} d$. *If* d *reduces to* d' *with zero or more steps by removing synthetic casts, followed by one step by other kinds of reduction, then* $e \rightarrow_{FGJ} e'$ *for some* e' *and* $|\Gamma|_\Delta \vdash |e'|_{\Delta,\Gamma} \overset{exp}{\Longrightarrow} d'$.

Type erasure *properties*

Corollary 17 (Erasure Preserves Execution Results)

If $\Delta;\Gamma \vdash e : T$ and $e \longrightarrow^*_{FGJ} w$, then $|e|_{\Delta,\Gamma} \longrightarrow^*_{FJ} |w|_{\Delta,\Gamma}$. Similarly, if $\Delta,\Gamma \vdash e : T$ and $|e|_{\Delta,\Gamma} \longrightarrow^*_{FJ} v$, then there exists an FGJ value w such that $e \longrightarrow^*_{FGJ} w$ and $|w|_{\Delta,\Gamma} = v$.

Corollary 18 (Erasure Preserves Typecast Errors)

If $\Delta;\Gamma \vdash e : T$ *and* $e \longrightarrow^*_{FGJ} e'$, *where* e' *has a stuck subexpression* $(\mathtt{C} < \overline{\mathtt{S}} >)\ \mathtt{new}\ \mathtt{D} < \overline{\mathtt{T}} > (\overline{\mathtt{e}})$, *then* $|e|_{\Delta,\Gamma} \longrightarrow^*_{FJ} d'$ *such that* d' *has a stuck subexpression* $(\mathtt{C})\ \mathtt{new}\ \mathtt{D}(\overline{\mathtt{d}})$, *where* $\overline{\mathtt{d}}$ *are expansions of the erasures of* $\overline{\mathtt{e}}$, *at the same position (modulo synthetic casts) as the erasure of* e'.

Similarly, if $\Delta,\Gamma \vdash e : T$ *and* $|e|_{\Delta,\Gamma} \longrightarrow^*_{FJ} e'$, *where* e' *has a stuck subexpression* $(\mathtt{C})\ \mathtt{new}\ \mathtt{D}(\overline{\mathtt{d}})$, *then there exists an FGJ expression* d *such that* $e \longrightarrow^*_{FGJ} d$ *and* $|\Gamma|_\Delta \vdash |d|_{\Delta,\Gamma} \stackrel{exp}{\Longrightarrow} e'$ *and* d *has a stuck subexpression* $(\mathtt{C} < \overline{\mathtt{S}} >)\ \mathtt{new}\ \mathtt{D} < \overline{\mathtt{T}} > (\overline{\mathtt{d}})$, *where* $\overline{\mathtt{e}}$ *are expansions of the erasures of* $\overline{\mathtt{d}}$, *at the same position (modulo synthetic casts) as* e'.

The end

That's all for today. Questions?

Homework

You have to implement:

- typechecker for FJ
- evaluator for FJ
- type erasure (translation from FGJ to FJ)
- ...glue it all together, provide some examples

Any functional language (Haskell, OCaml, Scala, F#, ...)