

Mechanization and Overhaul of Feature Featherweight Java with Coq

Pedro da C. Abreu Jr.

Universidade de Brasília

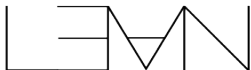
08 de Agosto, 2017

- Mecanizar significa provar teoremas com a assistência do computador.

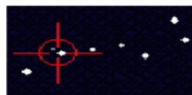
- Mecanizar significa provar teoremas com a assistência do computador.
 - Automaticamente;

- Mecanizar significa provar teoremas com a assistência do computador.
 - Automaticamente;
 - Iterativamente.

Provedores Iterativos de Teoremas



PRL Project



Por que mecanizar?


- Aplicações de segurança crítica: Bugs são inaceitáveis
 - Controladores de Avião
 - Equipamentos Médicos
 - Carros

Por que mecanizar?

- Aplicações de segurança crítica: Bugs são inaceitáveis
 - Controladores de Avião
 - Equipamentos Médicos
 - Carros
- A medida que o software e o hardware crescem dificulta encontrar bugs através de testes.

Por que mecanizar?

- Aplicações de segurança critica: Bugs são inaceitáveis
 - Controladores de Avião
 - Equipamentos Médicos
 - Carros
- A medida que o software e o hardware crescem dificulta encontrar bugs através de testes.
- Bugs encontrados tardiamente são caros. ¹

¹Fonte: The Economic Impacts of Inadequate Infrastructure for Software Testing 

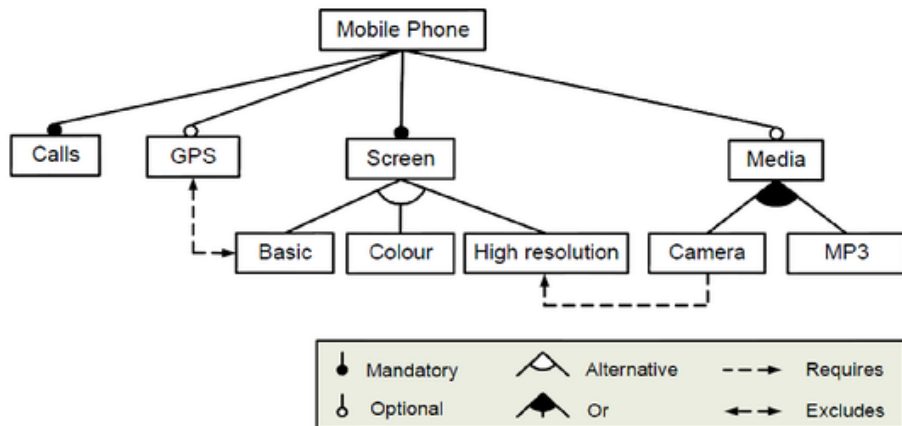
O que é feature?

Feature

'A feature is a unit of functionality of a software system that satisfies a requirement, represents a design decision, and provides a potential configuration option.' - S. Apel & K. Kastner

Feature Oriented Programming

Software Product Line



Mecanizar Feature Featherweight Java

$CD ::=$ *class declarations:*
 `class C extends D { \bar{C} \bar{f} ; K \bar{M} }`

$K ::=$ *constructor declarations:*
 `C(\bar{C} \bar{f}){super(\bar{f}); this. \bar{f} = \bar{f} ;}`

$M ::=$ *method declarations:*
 `C m (\bar{C} \bar{x}) {return e;}`

$e ::= \text{expressions:}$

x

$e.f$

$e.m(\bar{e})$

$\text{new } C(\bar{e})$

$(C)e$

$v ::= \text{values:}$

$\text{new } C(\bar{e})$

Featherweight Java

Relação de Subtipo

$$\frac{}{C <: C} \qquad \frac{C <: D \quad D <: E}{C <: E} \qquad \frac{\text{class } C \text{ extends } D \{ \dots \}}{C <: D}$$

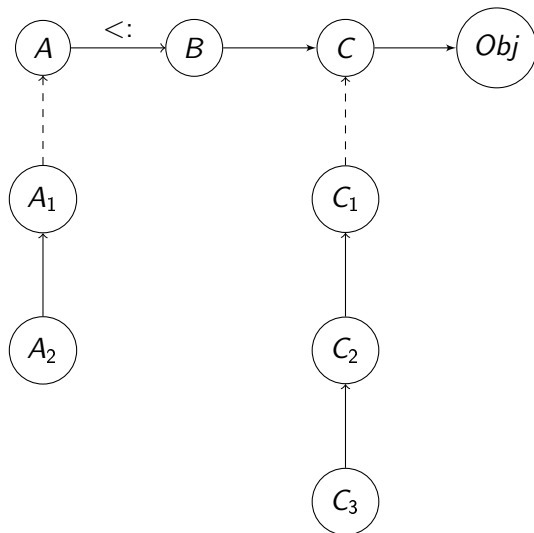
Featherweight Java

Relação de Subtipo



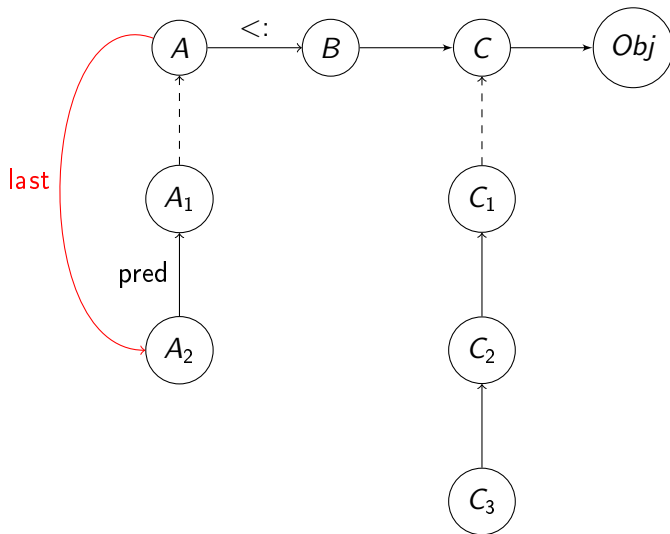
Feature Featherweight Java

Refinement Chain



Feature Featherweight Java

Refinement Chain



$R ::=$ *refinement names:*

$C@feat$

$CR ::=$ *class refinements:*

$refines\ class\ R\ \{\bar{C}\ \bar{f};\ KD\ \bar{M}\ \bar{MR}\}$

$KD ::=$ *constructor refinements:*

$refines\ C(\bar{E}\ \bar{h},\ \bar{C}\ \bar{f})\{original(\bar{f});\ this.\bar{f}=\bar{f};\}$

$MR ::=$ *method refinements:*

$refines\ C\ m\ (\bar{R}\ \bar{x})\ \{return\ e;\}$

fields Object = •

$$\frac{\text{class } C \text{ extends } D \{ \bar{C} \ \bar{f}; K \ \bar{M} \} \quad \neg \text{last } C}{\text{fields } C = \text{fields } D, \ \bar{C} \ \bar{f}}$$
$$\frac{\text{class } C \text{ extends } D \{ \bar{C} \ \bar{f}; K \ \bar{M} \}}{\text{fields } C = \text{fields } D, \ \bar{C} \ \bar{f}, \ \text{fields}_R (\text{last } C)}$$

fields Object = •

$$\frac{\text{class } C \text{ extends } D \{ \bar{C} \ \bar{f}; K \ \bar{M} \}}{\text{fields } C = \text{fields } D, \ \bar{C} \ \bar{f}} \quad \neg \text{last } C$$
$$\frac{\text{class } C \text{ extends } D \{ \bar{C} \ \bar{f}; K \ \bar{M} \}}{\text{fields } C = \text{fields } D, \ \bar{C} \ \bar{f}, \text{fields}_R (\text{last } C)}$$

fields Object = •

$$\frac{\text{class } C \text{ extends } D \{ \bar{C} \bar{f}; K \bar{M} \} \quad \neg \text{last } C}{\text{fields } C = \text{fields } D, \bar{C} \bar{f}}$$
$$\frac{\text{class } C \text{ extends } D \{ \bar{C} \bar{f}; K \bar{M} \}}{\text{fields } C = \text{fields } D, \bar{C} \bar{f}, \text{fields}_R (\text{last } C)}$$

fields Object = •

$$\frac{\text{class } C \text{ extends } D \{ \bar{C} \ \bar{f}; K \ \bar{M} \} \quad \neg \text{last } C}{\text{fields } C = \text{fields } D, \ \bar{C} \ \bar{f}}$$
$$\frac{\text{class } C \text{ extends } D \{ \bar{C} \ \bar{f}; K \ \bar{M} \}}{\text{fields } C = \text{fields } D, \ \bar{C} \ \bar{f}, \text{fields}_R (\text{last } C)}$$

Field Lookup (Refinement)

$$\frac{\text{refines } R \ \{\bar{C} \ \bar{f}; \ KR \ \bar{M} \ \bar{M}R\} \quad \neg \text{pred } R}{\text{fields}_R R = \bar{C} \ \bar{f}}$$

$$\frac{\text{refines } R \ \{\bar{C} \ \bar{f}; \ KR \ \bar{M} \ \bar{M}R\}}{\text{fields}_R R = \text{fields}_R (\text{pred } R), \bar{C} \ \bar{f}}$$

Field Lookup (Refinement)

$$\frac{\text{refines } R \ \{\bar{C} \ \bar{f}; \ KR \ \bar{M} \ \bar{M}R\} \quad \neg \text{pred } R}{\text{fields}_R R = \bar{C} \ \bar{f}}$$

$$\frac{\text{refines } R \ \{\bar{C} \ \bar{f}; \ KR \ \bar{M} \ \bar{M}R\}}{\text{fields}_R R = \text{fields}_R (\text{pred } R), \bar{C} \ \bar{f}}$$

Field Lookup (Refinement)

$$\frac{\text{refines } R \ \{\bar{C} \ \bar{f}; \ KR \ \bar{M} \ \bar{M}R\} \quad \neg \text{pred } R}{\text{fields}_R R = \bar{C} \ \bar{f}}$$

$$\frac{\text{refines } R \ \{\bar{C} \ \bar{f}; \ KR \ \bar{M} \ \bar{M}R\}}{\text{fields}_R R = \text{fields}_R (\text{pred } R), \bar{C} \ \bar{f}}$$

Method Type Lookup

$$\frac{\text{class } C \text{ extends } D \{ \bar{C} \ \bar{f}; K \ \bar{M} \} \quad B \ m \ (\bar{B} \ \bar{x}) \ \{\text{return } e;\} \in \bar{M} \quad \neg mtype_R(m, last\ C)}{mtype(m, C) = \bar{B} \rightarrow B}$$
$$\frac{\text{class } C \text{ extends } D \{ \bar{C} \ \bar{f}; K \ \bar{M} \} \quad m \notin \bar{M} \quad \neg mtype_R(m, last\ C)}{mtype(m, C) = mtype(m, D)}$$
$$\frac{\text{class } C \text{ extends } D \{ \bar{C} \ \bar{f}; K \ \bar{M} \}}{mtype(m, C) = mtype_R(m, last\ C)}$$

Method Type Lookup

$$\text{class } C \text{ extends } D \{ \bar{C} \ \bar{f}; K \ \bar{M} \} \quad B \ m \ (\bar{B} \ \bar{x}) \ \{\text{return } e;\} \in \bar{M}$$
$$\neg \text{mtype}_R(m, \text{last } C)$$

$$\text{mtype}(m, C) = \bar{B} \rightarrow B$$
$$\text{class } C \text{ extends } D \{ \bar{C} \ \bar{f}; K \ \bar{M} \} \quad m \notin \bar{M}$$
$$\neg \text{mtype}_R(m, \text{last } C)$$

$$\text{mtype}(m, C) = \text{mtype}(m, D)$$
$$\text{class } C \text{ extends } D \{ \bar{C} \ \bar{f}; K \ \bar{M} \}$$

$$\text{mtype}(m, C) = \text{mtype}_R(m, \text{last } C)$$

Method Type Lookup

$$\frac{\text{class } C \text{ extends } D \{ \bar{C} \ \bar{f}; K \ \bar{M} \} \quad B \ m \ (\bar{B} \ \bar{x}) \ \{\text{return } e;\} \in \bar{M} \quad \neg mtype_R(m, last\ C)}{mtype(m, C) = \bar{B} \rightarrow B}$$
$$\frac{\text{class } C \text{ extends } D \{ \bar{C} \ \bar{f}; K \ \bar{M} \} \quad m \notin \bar{M} \quad \neg mtype_R(m, last\ C)}{mtype(m, C) = mtype(m, D)}$$
$$\frac{\text{class } C \text{ extends } D \{ \bar{C} \ \bar{f}; K \ \bar{M} \}}{mtype(m, C) = mtype_R(m, last\ C)}$$

Method Type Lookup

$$\text{class } C \text{ extends } D \{ \bar{C} \ \bar{f}; K \ \bar{M} \} \quad B \ m \ (\bar{B} \ \bar{x}) \ \{ \text{return } e; \} \in \bar{M}$$
$$\neg \text{mtype}_R(m, \text{last } C)$$

$$\text{mtype}(m, C) = \bar{B} \rightarrow B$$
$$\text{class } C \text{ extends } D \{ \bar{C} \ \bar{f}; K \ \bar{M} \} \quad m \notin \bar{M}$$
$$\neg \text{mtype}_R(m, \text{last } C)$$

$$\text{mtype}(m, C) = \text{mtype}(m, D)$$
$$\text{class } C \text{ extends } D \{ \bar{C} \ \bar{f}; K \ \bar{M} \}$$

$$\text{mtype}(m, C) = \text{mtype}_R(m, \text{last } C)$$

Method Type Lookup

$$\text{class } C \text{ extends } D \{ \bar{C} \ \bar{f}; K \ \bar{M} \} \quad B \ m \ (\bar{B} \ \bar{x}) \ \{\text{return } e; \} \in \bar{M} \\ \neg \text{mtype}_R(m, \text{last } C)$$

$$\text{mtype}(m, C) = \bar{B} \rightarrow B$$
$$\text{class } C \text{ extends } D \{ \bar{C} \ \bar{f}; K \ \bar{M} \} \quad m \notin \bar{M} \\ \neg \text{mtype}_R(m, \text{last } C)$$

$$\text{mtype}(m, C) = \text{mtype}(m, D)$$
$$\text{class } C \text{ extends } D \{ \bar{C} \ \bar{f}; K \ \bar{M} \} \\ \text{mtype}(m, C) = \text{mtype}_R(m, \text{last } C)$$

Method Type Lookup

$$\frac{\text{class } C \text{ extends } D \{ \bar{C} \ \bar{f}; K \ \bar{M} \} \quad \textcolor{red}{B} \ m \ (\textcolor{red}{\bar{B}} \ \bar{x}) \ \{ \text{return } e; \} \in \bar{M} \quad \neg mtype_R(m, \text{last } C)}{mtype(m, C) = \textcolor{red}{\bar{B}} \rightarrow \textcolor{red}{B}}$$

$$\frac{\text{class } C \text{ extends } D \{ \bar{C} \ \bar{f}; K \ \bar{M} \} \quad m \notin \bar{M} \quad \neg mtype_R(m, \text{last } C)}{mtype(m, C) = mtype(m, D)}$$

$$\frac{\text{class } C \text{ extends } D \{ \bar{C} \ \bar{f}; K \ \bar{M} \}}{mtype(m, C) = mtype_R(m, \text{last } C)}$$

Method Type Lookup

$$\text{class } C \text{ extends } D \{ \bar{C} \ \bar{f}; K \ \bar{M} \} \quad B \ m \ (\bar{B} \ \bar{x}) \ \{ \text{return } e; \} \in \bar{M} \\ \neg mtype_R(m, \text{last } C)$$

$$mtype(m, C) = \bar{B} \rightarrow B$$
$$\text{class } C \text{ extends } D \{ \bar{C} \ \bar{f}; K \ \bar{M} \} \quad m \notin \bar{M} \\ \neg mtype_R(m, \text{last } C)$$

$$mtype(m, C) = mtype(m, D)$$
$$\text{class } C \text{ extends } D \{ \bar{C} \ \bar{f}; K \ \bar{M} \} \\ mtype(m, C) = mtype_R(m, \text{last } C)$$

Method Type Lookup

$$\frac{\text{class } C \text{ extends } D \{ \bar{C} \ \bar{f}; K \ \bar{M} \} \quad B \ m \ (\bar{B} \ \bar{x}) \ \{\text{return } e;\} \in \bar{M} \quad \neg mtype_R(m, last\ C)}{mtype(m, C) = \bar{B} \rightarrow B}$$
$$\frac{\text{class } C \text{ extends } D \{ \bar{C} \ \bar{f}; K \ \bar{M} \} \quad m \notin \bar{M} \quad \neg mtype_R(m, last\ C)}{mtype(m, C) = mtype(m, D)}$$
$$\frac{\text{class } C \text{ extends } D \{ \bar{C} \ \bar{f}; K \ \bar{M} \}}{mtype(m, C) = mtype_R(m, last\ C)}$$

Method Type Lookup (Refinement)

$$\frac{\text{refines class } R \{ \bar{C} \ \bar{f}; \ KR \ \bar{M} \ \overline{MR} \} \quad B \ m \ (\bar{B} \ \bar{x}) \ \{\text{return } e;\} \in \bar{M}}{mtype_R(m, R) = \bar{B} \rightarrow B}$$

$$\frac{\begin{array}{l} \text{refines class } R \{ \bar{C} \ \bar{f}; \ KR \ \bar{M} \ \overline{MR} \} \quad m \notin \bar{M} \\ \text{refines } B \ m \ (\bar{B} \ \bar{x}) \ \{\text{return } e;\} \in \overline{MR} \end{array}}{mtype_R(m, R) = \bar{B} \rightarrow B}$$

$$\frac{\begin{array}{l} \text{refines class } R \{ \bar{C} \ \bar{f}; \ KR \ \bar{M} \ \overline{MR} \} \\ m \notin \bar{M} \quad m \notin \overline{MR} \end{array}}{mtype_R(m, R) = mtype_R(m, pred \ P)}$$

Method Type Lookup (Refinement)

$$\frac{\text{refines class } R \{ \bar{C} \ \bar{f}; \ KR \ \bar{M} \ \overline{MR} \} \quad B \ m \ (\bar{B} \ \bar{x}) \ \{\text{return } e;\} \in \bar{M}}{mtype_R(m, R) = \bar{B} \rightarrow B}$$

$$\frac{\begin{array}{l} \text{refines class } R \{ \bar{C} \ \bar{f}; \ KR \ \bar{M} \ \overline{MR} \} \quad m \notin \bar{M} \\ \text{refines } B \ m \ (\bar{B} \ \bar{x}) \ \{\text{return } e;\} \in \overline{MR} \end{array}}{mtype_R(m, R) = \bar{B} \rightarrow B}$$

$$\frac{\begin{array}{l} \text{refines class } R \{ \bar{C} \ \bar{f}; \ KR \ \bar{M} \ \overline{MR} \} \\ m \notin \bar{M} \quad m \notin \overline{MR} \end{array}}{mtype_R(m, R) = mtype_R(m, pred \ P)}$$

Method Type Lookup (Refinement)

$$\frac{\text{refines class } R \{ \bar{C} \ \bar{f}; \ KR \ \bar{M} \ \overline{MR} \} \quad \textcolor{red}{B} \ m \ (\textcolor{red}{\bar{B}} \ \bar{x}) \ \{\text{return } e;\} \in \bar{M}}{mtype_R(m, R) = \textcolor{red}{\bar{B}} \rightarrow \textcolor{red}{B}}$$

$$\frac{\begin{array}{l} \text{refines class } R \{ \bar{C} \ \bar{f}; \ KR \ \bar{M} \ \overline{MR} \} \quad m \notin \bar{M} \\ \text{refines } B \ m \ (\bar{B} \ \bar{x}) \ \{\text{return } e;\} \in \overline{MR} \end{array}}{mtype_R(m, R) = \bar{B} \rightarrow B}$$

$$\frac{\begin{array}{l} \text{refines class } R \{ \bar{C} \ \bar{f}; \ KR \ \bar{M} \ \overline{MR} \} \\ m \notin \bar{M} \quad m \notin \overline{MR} \end{array}}{mtype_R(m, R) = mtype_R(m, pred \ P)}$$

Method Type Lookup (Refinement)

$$\frac{\text{refines class } R \{ \bar{C} \ \bar{f}; \ KR \ \bar{M} \ \overline{MR} \} \quad B \ m \ (\bar{B} \ \bar{x}) \ \{\text{return } e;\} \in \bar{M}}{mtype_R(m, R) = \bar{B} \rightarrow B}$$

$$\frac{\text{refines class } R \{ \bar{C} \ \bar{f}; \ KR \ \bar{M} \ \overline{MR} \} \quad m \notin \bar{M} \quad \text{refines } B \ m \ (\bar{B} \ \bar{x}) \ \{\text{return } e;\} \in \overline{MR}}{mtype_R(m, R) = \bar{B} \rightarrow B}$$

$$\frac{\text{refines class } R \{ \bar{C} \ \bar{f}; \ KR \ \bar{M} \ \overline{MR} \} \quad m \notin \bar{M} \quad m \notin \overline{MR}}{mtype_R(m, R) = mtype_R(m, pred \ P)}$$

Method Type Lookup (Refinement)

$$\frac{\text{refines class } R \{ \bar{C} \ \bar{f}; \ KR \ \bar{M} \ \overline{MR} \} \quad B \ m \ (\bar{B} \ \bar{x}) \ \{\text{return } e;\} \in \bar{M}}{mtype_R(m, R) = \bar{B} \rightarrow B}$$

$$\frac{\begin{array}{l} \text{refines class } R \{ \bar{C} \ \bar{f}; \ KR \ \bar{M} \ \overline{MR} \} \quad m \notin \bar{M} \\ \text{refines } B \ m \ (\bar{B} \ \bar{x}) \ \{\text{return } e;\} \in \overline{MR} \end{array}}{mtype_R(m, R) = \bar{B} \rightarrow B}$$

$$\frac{\begin{array}{l} \text{refines class } R \{ \bar{C} \ \bar{f}; \ KR \ \bar{M} \ \overline{MR} \} \\ m \notin \bar{M} \quad m \notin \overline{MR} \end{array}}{mtype_R(m, R) = mtype_R(m, pred \ P)}$$

Method Type Lookup (Refinement)

$$\frac{\text{refines class } R \{ \bar{C} \ \bar{f}; \ KR \ \bar{M} \ \overline{MR} \} \quad B \ m \ (\bar{B} \ \bar{x}) \ \{\text{return } e;\} \in \bar{M}}{mtype_R(m, R) = \bar{B} \rightarrow B}$$

$$\frac{\begin{array}{l} \text{refines class } R \{ \bar{C} \ \bar{f}; \ KR \ \bar{M} \ \overline{MR} \} \quad m \notin \bar{M} \\ \text{refines } B \ m \ (\bar{B} \ \bar{x}) \ \{\text{return } e;\} \in \overline{MR} \end{array}}{mtype_R(m, R) = \bar{B} \rightarrow B}$$

$$\frac{\begin{array}{l} \text{refines class } R \{ \bar{C} \ \bar{f}; \ KR \ \bar{M} \ \overline{MR} \} \\ m \notin \bar{M} \quad m \notin \overline{MR} \end{array}}{mtype_R(m, R) = mtype_R(m, \text{pred } P)}$$

Method Body Lookup

$$\frac{\text{class } C \text{ extends } D \{ \bar{C} \ \bar{f}; \ K \ \bar{M} \} \quad B \ m \ (\bar{B} \ \bar{x}) \ \{ \text{return } e; \} \in \bar{M} \quad \neg mbody_R(m, last\ C)}{mbody(m, C) = \bar{x}.e}$$
$$\frac{\text{class } C \text{ extends } D \{ \bar{C} \ \bar{f}; \ K \ \bar{M} \} \quad m \notin \bar{M} \quad \neg mbody_R(m, last\ C)}{mbody(m, C) = mbody(m, D)}$$
$$\frac{\text{class } C \text{ extends } D \{ \bar{C} \ \bar{f}; \ K \ \bar{M} \}}{mbody(m, C) = mbody_R(m, last\ C)}$$

Method Body Lookup (Refinement)

$$\frac{\text{refines class } R \{ \bar{C} \ \bar{f}; \ KR \ \bar{M} \ \overline{MR} \} \quad B \ m \ (\bar{B} \ \bar{x}) \ \{\text{return } e;\} \in \bar{M}}{mbody_R(m, R) = \bar{x}.e}$$

$$\frac{\begin{array}{l} \text{refines class } R \{ \bar{C} \ \bar{f}; \ KR \ \bar{M} \ \overline{MR} \} \quad m \notin \bar{M} \\ \text{refines } B \ m \ (\bar{B} \ \bar{x}) \ \{\text{return } e;\} \in \overline{MR} \end{array}}{mbody_R(m, R) = \bar{x}.e}$$

$$\frac{\begin{array}{l} \text{refines class } R \{ \bar{C} \ \bar{f}; \ KR \ \bar{M} \ \overline{MR} \} \\ m \notin \bar{M} \quad m \notin \overline{MR} \end{array}}{mbody_R(m, R) = mbody_R(m, \text{last } P)}$$

$$\Gamma \vdash e : C$$

$$\begin{array}{c} \Gamma \vdash e : C \\ \Downarrow \\ a : A, b : B, \dots, x : X \vdash e : C \end{array}$$

Regras de Tipagem

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

$$\frac{\Gamma \vdash e_0 : C_0 \quad \text{fields}(C_0) = \overline{C} \ \overline{f}}{\Gamma \vdash e_0.f_i : C_i} \quad (\text{T-Field})$$

$$\frac{\Gamma \vdash e_0 : C_0 \quad \text{mtypes}(m, C_0) = \overline{D} \rightarrow C \quad \Gamma \vdash \overline{e} : \overline{C} \quad \overline{C} <: \overline{D}}{\Gamma \vdash e_0.m(\overline{e}) : C} \quad (\text{T-Invk})$$

$$\frac{\text{fields}(C) = \overline{D} \ \overline{f} \quad \Gamma \vdash \overline{e} : \overline{C} \quad \overline{C} <: \overline{D}}{\Gamma \vdash \text{new } C(\overline{e}) : C} \quad (\text{T-New})$$

$$\frac{\Gamma \vdash e_0 : D \quad D <: C}{\Gamma \vdash (C) e_0 : C} \quad (\text{T-UCast})$$

$$\frac{\Gamma \vdash e_0 : D \quad C <: D \quad C \neq D}{\Gamma \vdash (C) e_0 : C} \quad (\text{T-DCast})$$

$$\frac{\Gamma \vdash e_0 : D \quad C \not<: D \quad D \not<: C \quad \text{stupid warning}}{\Gamma \vdash (C) e_0 : C} \quad (\text{T-SCast})$$

$$e \rightarrow e'$$

$$\frac{fields(C) = \bar{C}\bar{f}}{(new\ C(\bar{e})).f_i \rightarrow e_i} \quad (\text{R-Field})$$

$$\frac{mbody(m, C) = \bar{x}.e_0}{(new\ C(\bar{e})).m(\bar{d}) \rightarrow [\bar{d}/\bar{x}, new\ C(\bar{e})/this]e_0} \quad (\text{R-Invk})$$

$$\frac{C <: D}{(D)(new\ C(\bar{e})) \rightarrow new\ C(\bar{e})} \quad (\text{R-Cast})$$

Regras de Congruência

$$\frac{e_0 \rightarrow e'_0}{e_0.f \rightarrow e'_0.f} \quad (\text{RC-Field})$$

$$\frac{e_0 \rightarrow e'_0}{e_0.m(\bar{e}) \rightarrow e'_0.m(\bar{e})} \quad (\text{RC-Invk-Recv})$$

$$\frac{e_i \rightarrow e'_i}{e_0.m(\dots, e_i, \dots) \rightarrow e'_0.m(\dots, e'_i, \dots)} \quad (\text{RC-Invk-Arg})$$

$$\frac{e_i \rightarrow e'_i}{\text{new } C(\dots, e_i, \dots) \rightarrow \text{new } C(\dots, e'_i, \dots)} \quad (\text{RC-New-Arg})$$

$$\frac{e_0 \rightarrow e'_0}{(C)e_0 \rightarrow (C)e'_0} \quad (\text{RC-Cast})$$

- Um programa bem tipado não está preso

- Um programa bem tipado não está preso
 - Ou é um valor;

- Um programa bem tipado não está preso
 - Ou é um valor;
 - Ou dá um passo.

- Se um termo bem tipado dá um passo, então o resultado também é bem tipado.

Type Safety (ou Soundness)

- Um termo bem tipado nunca fica preso durante a computação.

Safety = Progress + Preservation

- "*Well-typed program cannot go wrong*" – Robin Milner

Safety = Progress + Preservation

- "*Well-typed program cannot go wrong*" – Robin Milner
- Formulado por Wright e Felleisen em 1994 como definição padrão de type safety para linguagens formuladas por *operational semantics*

Theorem (Preservation)

Se $\Gamma \vdash e : C$ e $e \rightarrow e'$, então $\Gamma \vdash e' : C'$ para algum $C' \leq C$.

Theorem (Progress)

*Se $\vdash e : C$ e não existe e' tal que $e \rightarrow e'$
Então ou e é um valor, ou contém um downcast como subtermo.*

- Tudo isto foi devidamente implementado em Coq.^{2 3}

²<https://github.com/hephaestus-pl/coqffj>

³<https://github.com/hephaestus-pl/coqfj>



Harper, Robert.

Practical foundations for programming languages.
Cambridge University Press, 2012.



Igarashi, Atsushi, Benjamin C. Pierce, and Philip Wadler.

Featherweight Java: a minimal core calculus for Java and GJ.
ACM Transactions on Programming Languages and Systems
(TOPLAS) 2001.



Apel, Sven, Christian Kästner, and Christian Lengauer.

Feature Featherweight Java: A calculus for feature-oriented
programming and stepwise refinement
Proceedings of the 7th international conference on Generative
programming and component engineering. ACM, 2008.

Perguntas?

Class Name

$$\frac{R = C@feat}{class_name\ R = C}$$

Refinements of a class

$$\frac{filter\ (\lambda R \cdot class_name\ R == C)\ RT = \bar{R}}{refinements_of\ C = \bar{R}}$$

Predecessor

$$\frac{refinements_of\ (class_name\ R) = \bar{R} \quad index\ R\ \bar{R} = n \quad get\ (n - 1)\ \bar{R} = P}{pred\ R = P}$$

Last

$$\frac{refinements_of\ C = \bar{R} \quad tail\ \bar{R} = R}{last\ C = R}$$

$$\frac{mtype(m, D) = \bar{D} \rightarrow D \text{ implies } \bar{C} = \bar{D} \text{ and } C_0 = D}{\text{override } m \ D \ \bar{C} \ C_0}$$

$$\frac{\text{class } C \text{ extends } D \ \{\bar{C} \ \bar{f}; \ K \ \bar{M}\} \quad C_0 \ m \ (\bar{C} \ \bar{x}) \ \{\text{return } e;\} \in \bar{M} \quad \neg \text{pred } R \quad R = C@feat}{\text{override}_R \ m \ R \ \bar{C} \ C_0}$$

$$\frac{\text{refines class } P \ \{\bar{C} \ \bar{f}; \ KR \ \bar{M} \ \overline{MR}\} \quad C_0 \ m \ (\bar{C} \ \bar{x}) \ \{\text{return } e;\} \in \bar{M} \quad \text{pred } R = P}{\text{override}_R \ m \ R \ \bar{C} \ C_0}$$

$$\frac{\text{refines class } P \ \{\bar{C} \ \bar{f}; \ KR \ \bar{M} \ \overline{MR}\} \quad m \notin \bar{M} \quad \text{pred } R = P \quad \text{override}_R \ m \ P \ \bar{C} \ C_0}{\text{override}_R \ m \ R \ \bar{C} \ C_0}$$

$$\frac{\text{pred } R = S \quad \neg \text{mtype}_R (m, S)}{\text{introduce } m \ R}$$

$$\frac{\neg \text{pred } R \quad R = C@feat \quad \text{class } C \text{ extends } D \ \{\bar{C} \ \bar{f}; \ K \ \bar{M}\} \quad m \notin \bar{M}}{\text{introduce } m \ R}$$

$$\frac{\bar{x} : \bar{C}, \text{this} : C \vdash t_0 : E_0 \quad E_0 <: C_0 \quad \text{CT}(C) = \text{class } C \text{ extends } D \{ \dots \} \quad \text{override}(m, D, \bar{C} \rightarrow C)}{C_0 \text{ m } (\bar{C} \bar{x}) \{ \text{return } t_0; \} \text{ OK in } C}$$

$$\frac{\bar{x} : \bar{C}, \text{this} : C \vdash t_0 : E_0 \quad E_0 <: C_0 \quad R = C @ \text{feat} \quad \text{CT}(C) = \text{class } C \text{ extends } D \{ \dots \} \quad \text{RT}(R) = \text{refines } R \{ \dots \bar{M} \dots \} \quad \text{override}(m, D, \bar{C} \rightarrow C) \quad \text{introduce m R} \quad m \in \bar{M}}{C_0 \text{ m } (\bar{C} \bar{x}) \{ \text{return } t_0; \} \text{ OK in } R}$$

$$\frac{\bar{x} : \bar{C}, \text{this} : C \vdash t_0 : E_0 \quad E_0 <: C_0 \quad R = C @ \text{feat} \quad \text{RT}(R) = \text{refines } R \{ \dots \bar{M}, \bar{MR} \dots \} \quad m \notin \bar{M} \quad m \in \bar{MR} \quad \text{override}_R(m, R, \bar{C} \rightarrow C) \quad \text{introduce m R}}{\text{refines } C_0 \text{ m } (\bar{C} \bar{x}) \{ \text{return } t_0; \} \text{ OK in } R}$$

Appendix

Class and Refinement Typing

$$\frac{\begin{array}{c} K = C(\bar{D} \bar{g}, \bar{C} \bar{f}) \{ \text{super}(\bar{g}); \text{this}.\bar{f} = \bar{f} \} \quad \text{fields}(D) = \bar{D} \bar{g} \\ \bar{M} \text{ OK in } C \end{array}}{\text{class } C \text{ extends } D \{ \bar{C} \bar{f}; K \bar{M} \} \text{ OK}}$$
$$\frac{\bar{M} \text{ OK in } R \quad \bar{MR} \text{ OK in } R}{\text{refines class } R \{ \bar{C} \bar{f}; KR \bar{M} \bar{MR} \} \text{ OK}}$$

Lemma (Typed method has body)

*If $mtype(m, C) = B \rightarrow \overline{B}$
then $\exists \overline{x} \exists e$ such that $mbody(m, C) = \overline{x}.e$*

Lemma (Typed method has body - Refinement)

*If $mtype_R(m, R) = B \rightarrow Bs$
then $\exists \bar{x} \exists e$ such that $mbody_R(m, R) = \bar{x}.e$*

Lemma (Body method has type - Refinement)

*If $mbody_R(m, R) = \bar{x}.e$
then $\exists \bar{B} \exists B$ such that $mtype_R(m, R) = B \rightarrow Bs$*

Lemma (Subtype respects method types)

*If class C extends $D \{ \bar{C} \ \bar{f}; \ K \ \bar{M} \}$
then $mtype(m, C) = mtype(m, D)$*

Lemma (Refinement respects method types)

*If class C extends D $\{\bar{C} \ \bar{f}; K \ \bar{M}\}$
then $\forall feat, mtype_R(m, C@feat) = mtype(m, D)$*

Lemma (A1.4 - Method Body is Typable)

If $mtype(m, \mathcal{C}) = \overline{D} \rightarrow D$

and $mbody(m, \mathcal{C}) = \overline{x}.e$

then $\exists C <: D, \exists C_0 <: D_0, this : D, \overline{x} : \overline{D} \vdash e : C_0$

Lemma (Method Body is Typable - Refinement)

*If $mtype_R(m, C@feat) = \overline{D} \rightarrow D$
and $mbody_R(m, C@feat) = \overline{x}.e$
then $\exists C <: D, \exists C_0 <: D_0, \text{ this} : D, \overline{x} : \overline{D} \vdash e : C_0$*

$$E ::= \square \mid E.f_i \mid E.m(\bar{e}) \mid e.m(\bar{e}_l, E, \bar{e}_r) \mid (C) E \mid \text{new } C(\bar{e}_l, E, \bar{e}_r)$$

Table: Evaluation Context

Theorem (Progress – Using Evaluation Context)

Suppose e is closed, well-typed normal form.

Then either (1) e is a value, or (2) for some evaluation context E , we can express e as $e = E[(C)(\text{new } D(\bar{e}))]$, with $D \not\vdash C$.