Mechanization and Overhaul of Feature Featherweight Java with Coq

Pedro da C. Abreu Jr.

Universidade de Brasilia

08 de Agosto, 2017

Mecanização

- Mecanizar significa provar teoremas com a assistência do computador.

Mecanização

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

Mecanização

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

Provadores Iterativos de Teoremas

LEAN

PRL Project

Isabelle

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.

Bugs são caros

Custo Relativo para Corrigir Defeitos Encontrados em Diferentes Fases do Desenvolvimento de Software.[1]

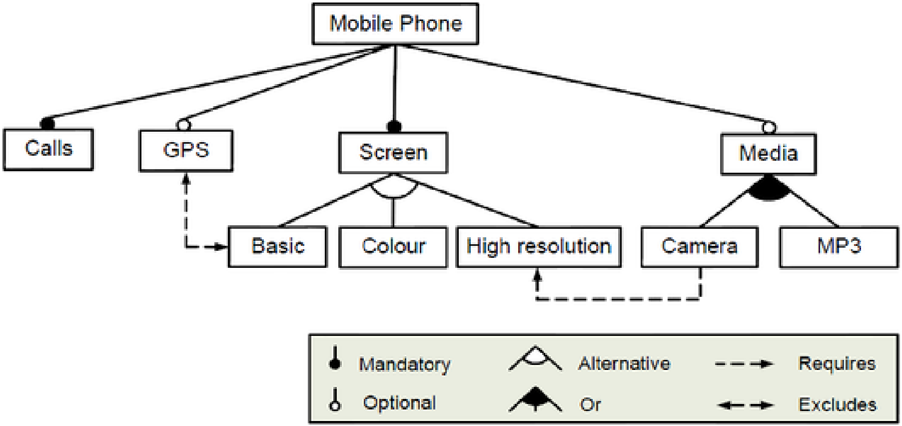
Requirements Gathering and Analysis/ Architectural Design	Coding/Unit Test	Integration and Component/RAISE System Test	Early Customer Feedback/Beta Test Programs	Post-product Release
1X	5X	10X	15X	30X

[1] Fonte: The Economic Impacts of Inadequate Infraestructure 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


Mobile Phone
Calls
GPS
Screen
Media
Basic
Colour
High resolution
Camera
MP3
Mandatory
Optional
Alternative
Or
Requires
Excludes

Objetivo

Mecanizar Feature Featherweight Java

CD ::= *class declarations:*

`class C extends D {C̄ f̄; K M̄}`

K ::= *constructor declarations:*

`C(C̄ f̄){super(f̄); this.f̄=f̄;}`

M ::= *method declarations:*

`C m (C̄ x̄) {return e;}`

e ::= *expressions:*

x

e.f

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

new C($\bar{e}$)

(C)e

v ::= *values:*

new C($\bar{e}$)

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

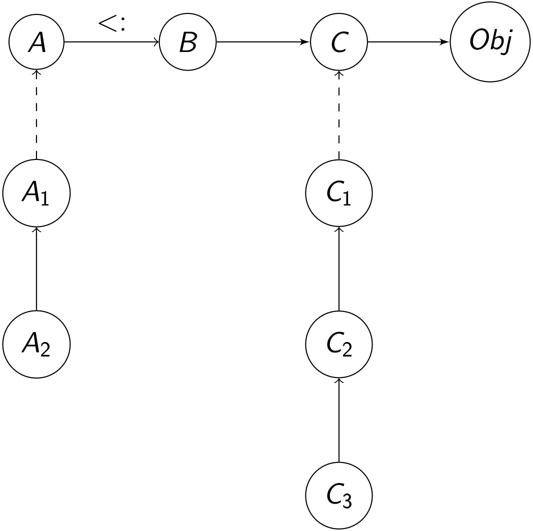
Featherweight Java

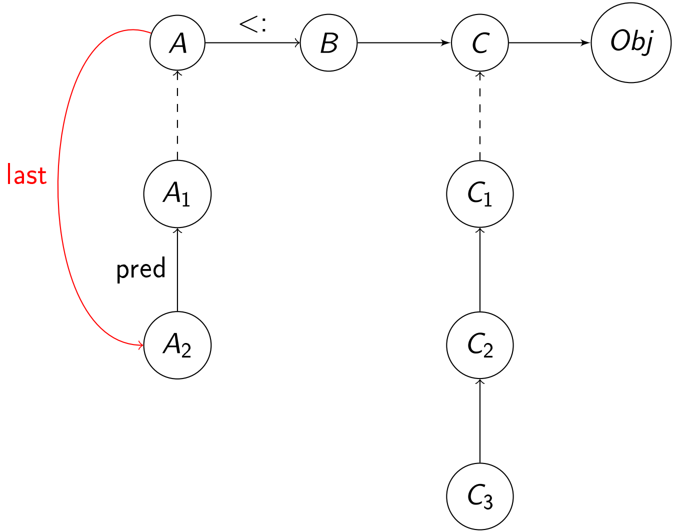
Relação de Subtipo

$$A \xrightarrow{<:} B \longrightarrow C \longrightarrow Obj$$

Feature Featherweight Java

Refinement Chain




A
<:
B
C
Obj
last
A1
C1
pred
A2
C2
C3

$\texttt{R} ::=$ *refinement names:*
  $\texttt{C@feat}$

$\texttt{CR} ::=$ *class refinements:*
  $\texttt{refines class R \{}\bar{\texttt{C}}\ \bar{\texttt{f}}\texttt{; KD}\ \bar{\texttt{M}}\ \bar{\texttt{MR}}\texttt{\}}$

$\texttt{KD} ::=$ *constructor refinements:*
  $\texttt{refines C(}\bar{\texttt{E}}\ \bar{\texttt{h}}\texttt{, }\bar{\texttt{C}}\ \bar{\texttt{f}}\texttt{)\{original(}\bar{\texttt{f}}\texttt{); this.}\bar{\texttt{f}}\texttt{=}\bar{\texttt{f}}\texttt{;\}}$

$\texttt{MR} ::=$ *method refinements:*
  $\texttt{refines C m (}\bar{\texttt{R}}\ \bar{\texttt{x}}\texttt{) \{return e;\}}$

Field Lookup

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

$$\frac{\texttt{class C extends D \{}\bar{\texttt{C}}\ \bar{\texttt{f}}\texttt{; K}\ \bar{\texttt{M}}\texttt{\}} \qquad \neg \mathit{last}\ \texttt{C}}{\mathit{fields}\ \texttt{C} = \mathit{fields}\ \texttt{D},\ \bar{\texttt{C}}\ \bar{\texttt{f}}}$$

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

Field Lookup (Refinement)

$$\frac{\texttt{refines R } \{\bar{\texttt{C}}\ \bar{\texttt{f}};\ \texttt{KR}\ \bar{\texttt{M}}\ \bar{\texttt{M}}\texttt{R}\} \qquad \neg \mathit{pred}\ \texttt{R}}{\mathit{fields}_R\ \texttt{R}\ =\ \bar{\texttt{C}}\ \bar{\texttt{f}}}$$

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

Method Type Lookup

$$\frac{\texttt{class C extends D \{}\bar{\texttt{C}}\ \bar{\texttt{f}}\texttt{; K }\bar{\texttt{M}}\texttt{\}} \qquad \texttt{B m (}\bar{\texttt{B}}\ \bar{\texttt{x}}\texttt{) \{return e;\}} \in \bar{\texttt{M}} \qquad \neg \mathit{mtype}_R\ (\texttt{m}, \mathit{last}\ \texttt{C})}{\mathit{mtype}\ (\texttt{m}, \texttt{C}) = \bar{\texttt{B}} \to \texttt{B}}$$

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

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

Method Type Lookup (Refinement)

$$\frac{\texttt{refines class R } \{\bar{\texttt{C}}\ \bar{\texttt{f}};\ \texttt{KR}\ \bar{\texttt{M}}\ \overline{\texttt{MR}}\} \qquad \texttt{B m } (\bar{\texttt{B}}\ \bar{\texttt{x}})\ \{\texttt{return e;}\} \in \bar{\texttt{M}}}{\mathit{mtype}_R\ (\texttt{m}, \texttt{R}) = \bar{\texttt{B}} \rightarrow \texttt{B}}$$

$$\frac{\begin{array}{c}\texttt{refines class R } \{\bar{\texttt{C}}\ \bar{\texttt{f}};\ \texttt{KR}\ \bar{\texttt{M}}\ \overline{\texttt{MR}}\} \qquad \texttt{m} \notin \bar{\texttt{M}} \\ \texttt{refines B m } (\bar{\texttt{B}}\ \bar{\texttt{x}})\ \{\texttt{return e;}\} \in \overline{\texttt{MR}}\end{array}}{\mathit{mtype}_R\ (\texttt{m}, \texttt{R}) = \bar{\texttt{B}} \rightarrow \texttt{B}}$$

$$\frac{\begin{array}{c}\texttt{refines class R } \{\bar{\texttt{C}}\ \bar{\texttt{f}};\ \texttt{KR}\ \bar{\texttt{M}}\ \overline{\texttt{MR}}\} \\ \texttt{m} \notin \bar{\texttt{M}} \quad \texttt{m} \notin \overline{\texttt{MR}}\end{array}}{\mathit{mtype}_R\ (\texttt{m}, \texttt{R}) = \mathit{mtype}_R\ (\texttt{m}, \mathit{pred}\ \texttt{P})}$$

Method Body Lookup

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

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

$$\frac{\texttt{class C extends D \{}\bar{\texttt{C}}\ \bar{\texttt{f}}\texttt{; K }\bar{\texttt{M}}\texttt{\}}}{\mathit{mbody}\ (\texttt{m}, \texttt{C}) \ = \ \mathit{mbody}_R\ (\texttt{m}, \mathit{last}\ \texttt{C})}$$

Method Body Lookup (Refinement)

$$\frac{\texttt{refines class R } \{\bar{\texttt{C}}\ \bar{\texttt{f}};\ \texttt{KR}\ \bar{\texttt{M}}\ \overline{\texttt{MR}}\} \qquad \texttt{B m } (\bar{\texttt{B}}\ \bar{\texttt{x}})\ \{\texttt{return e;}\} \in \bar{\texttt{M}}}{\mathit{mbody}_R\,(\texttt{m}, \texttt{R}) \;=\; \bar{\texttt{x}}.\texttt{e}}$$

$$\frac{\begin{array}{c}\texttt{refines class R } \{\bar{\texttt{C}}\ \bar{\texttt{f}};\ \texttt{KR}\ \bar{\texttt{M}}\ \overline{\texttt{MR}}\} \qquad \texttt{m} \notin \bar{\texttt{M}} \\ \texttt{refines B m } (\bar{\texttt{B}}\ \bar{\texttt{x}})\ \{\texttt{return e;}\} \in \overline{\texttt{MR}}\end{array}}{\mathit{mbody}_R\,(\texttt{m}, \texttt{R}) \;=\; \bar{\texttt{x}}.\texttt{e}}$$

$$\frac{\begin{array}{c}\texttt{refines class R } \{\bar{\texttt{C}}\ \bar{\texttt{f}};\ \texttt{KR}\ \bar{\texttt{M}}\ \overline{\texttt{MR}}\} \\ \texttt{m} \notin \bar{\texttt{M}} \quad \texttt{m} \notin \overline{\texttt{MR}}\end{array}}{\mathit{mbody}_R\,(\texttt{m}, \texttt{R}) \;=\; \mathit{mbody}_R\,(\texttt{m}, \mathit{last}\ \texttt{P})}$$

Tipagem

$$\Gamma \vdash e : C$$

Tipagem

$$a : A, b : B, ..., x : X \vdash e : C$$

Regras de Tipagem

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

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

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

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

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

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

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

Computação

$$e \rightarrow e'$$

Regras de Computação

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

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

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

Regras de Congruência

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

$$\frac{e_0 \to e_0'}{e_0.m\ (\overline{e}) \to e_0'.m\ (\overline{e})} \quad \text{(RC-Invk-Recv)}$$

$$\frac{e_i \to e_i'}{e_0.m\ (\ldots, e_i, \ldots) \to e_0'.m\ (\ldots, e_i, \ldots)} \quad \text{(RC-Invk-Arg)}$$

$$\frac{e_i \to e_i'}{new\ \ C\ (\ldots, e_i, \ldots) \to new\ \ C\ (\ldots, e_i', \ldots)} \quad \text{(RC-New-Arg)}$$

$$\frac{e_0 \to e_0'}{(C)e_0 \to (C)e_0'} \quad \text{(RC-Cast)}$$

Progress

- Um programa bem tipado não está preso

Progress

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

Progress

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

Preservation

- 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*

Theoremas

Theorem (Preservation)

Se $\Gamma \vdash e : C$ *e* $e \to e'$, *então* $\Gamma \vdash e' : C'$ *para algum* $C' <: C$.

Theorem (Progress)

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

Conclusão

- Tudo isto foi devidamente implementado em Coq.

References I

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.

Fim

Perguntas?

Class Name

$$\frac{\mathtt{R} = \mathtt{C@feat}}{\textit{class\_name}\ \mathtt{R} = \mathtt{C}}$$

Refinements of a class

$$\frac{\textit{filter}\ (\lambda R \cdot \textit{class\_name}\ \mathtt{R} == \mathtt{C})\ \mathsf{RT} = \bar{\mathtt{R}}}{\textit{refinements\_of}\ \mathtt{C} = \bar{\mathtt{R}}}$$

Predecessor

$$\frac{\begin{array}{c}\textit{refinements\_of}\ (\textit{class\_name}\ \mathtt{R}) = \bar{\mathtt{R}} \\ \textit{index}\ \mathtt{R}\ \bar{\mathtt{R}} \ = \ n \qquad \textit{get}\ (n-1)\ \bar{\mathtt{R}} \ = \ \mathtt{P}\end{array}}{\textit{pred}\ \mathtt{R} \ = \mathtt{P}}$$

Last

$$\frac{\textit{refinements\_of}\ \mathtt{C} = \bar{\mathtt{R}} \qquad \textit{tail}\ \bar{\mathtt{R}} \ = \ \mathtt{R}}{\textit{last}\ \mathtt{C} \ = \mathtt{R}}$$

$$\frac{\textit{mtype}\ (\mathtt{m}, \mathtt{D}) \ = \ \bar{\mathtt{D}} \ \rightarrow \ \mathtt{D} \ \textit{implies} \ \bar{\mathtt{C}} \ = \ \bar{\mathtt{D}} \ \textit{and} \ \mathtt{C}_0 \ = \ \mathtt{D}}{\textit{override}\ \mathtt{m}\ \mathtt{D}\ \bar{\mathtt{C}}\ \mathtt{C}_0}$$

$$\frac{\begin{array}{c}\mathtt{class\ C\ extends\ D}\ \{\bar{\mathtt{C}}\ \bar{\mathtt{f}};\ \mathtt{K}\ \bar{\mathtt{M}}\} \qquad \mathtt{C}_0\ \mathtt{m}\ (\bar{\mathtt{C}}\ \bar{\mathtt{x}})\ \{\mathtt{return\ e;}\} \in \bar{\mathtt{M}} \\ \neg\ \textit{pred}\ \mathtt{R} \qquad \mathtt{R}\ =\ \mathtt{C@feat}\end{array}}{\textit{override}_R\ \mathtt{m}\ \mathtt{R}\ \bar{\mathtt{C}}\ \mathtt{C}_0}$$

$$\frac{\begin{array}{c}\mathtt{refines\ class\ P}\ \{\bar{\mathtt{C}}\ \bar{\mathtt{f}};\ \mathtt{KR}\ \bar{\mathtt{M}}\ \overline{\mathtt{MR}}\} \qquad \mathtt{C}_0\ \mathtt{m}\ (\bar{\mathtt{C}}\ \bar{\mathtt{x}})\ \{\mathtt{return\ e;}\} \in \bar{\mathtt{M}} \\ \textit{pred}\ \mathtt{R}\ =\ \mathtt{P}\end{array}}{\textit{override}_R\ \mathtt{m}\ \mathtt{R}\ \bar{\mathtt{C}}\ \mathtt{C}_0}$$

$$\frac{\begin{array}{c}\mathtt{refines\ class\ P}\ \{\bar{\mathtt{C}}\ \bar{\mathtt{f}};\ \mathtt{KR}\ \bar{\mathtt{M}}\ \overline{\mathtt{MR}}\} \qquad \mathtt{m} \notin \bar{\mathtt{M}} \\ \textit{pred}\ \mathtt{R}\ =\ \mathtt{P} \qquad \textit{override}_R\ \mathtt{m}\ \mathtt{P}\ \bar{\mathtt{C}}\ \mathtt{C}_0\end{array}}{\textit{override}_R\ \mathtt{m}\ \mathtt{R}\ \bar{\mathtt{C}}\ \mathtt{C}_0}$$

$$\frac{\textit{pred}\ \mathtt{R} \ = \ \mathtt{S} \qquad \neg\ \textit{mtype}_R\ (\mathtt{m}, \mathtt{S})}{\textit{introduce}\ \mathtt{m}\ \ \mathtt{R}}$$

$$\frac{\begin{array}{c}\neg\ \textit{pred}\ \mathtt{R} \qquad \mathtt{R} \ = \ \mathtt{C@feat} \\ \mathtt{class\ C\ extends\ D\ \{\bar{C}\ \bar{f};\ K\ \bar{M}\}} \qquad \mathtt{m} \notin \bar{\mathtt{M}}\end{array}}{\textit{introduce}\ \mathtt{m}\ \ \mathtt{R}}$$

$$\frac{\begin{array}{c}\overline{\mathtt{x}}:\overline{\mathtt{C}},\mathtt{this}:\mathtt{C}\ \vdash \mathtt{t_0}:\mathtt{E_0} \qquad \mathtt{E_0} <: \mathtt{C_0} \\ \mathtt{CT(C)}\ =\mathtt{class\ C\ extends\ D}\ \{\ldots\} \qquad \mathit{override}(\mathtt{m},\mathtt{D},\overline{\mathtt{C}}\to\mathtt{C})\end{array}}{\mathtt{C_0\ m}\ (\overline{\mathtt{C}}\ \overline{\mathtt{x}})\{\mathtt{return\ t_0};\}\ \mathtt{OK\ in\ C}}$$

$$\frac{\begin{array}{c}\overline{\mathtt{x}}:\overline{\mathtt{C}},\mathtt{this}:\mathtt{C}\ \vdash \mathtt{t_0}:\mathtt{E_0} \qquad \mathtt{E_0} <: \mathtt{C_0} \qquad \mathtt{R}\ =\ \mathtt{C@feat} \\ \mathtt{CT(C)}\ =\mathtt{class\ C\ extends\ D}\ \{\ldots\} \qquad \mathtt{RT(R)}\ =\ \mathtt{refines\ R}\ \{\ldots\ \overline{\mathtt{M}}\ \ldots\} \\ \mathit{override}(\mathtt{m},\mathtt{D},\overline{\mathtt{C}}\to\mathtt{C}) \qquad \mathit{introduce}\ \mathtt{m}\ \mathtt{R} \qquad \mathtt{m}\in\overline{\mathtt{M}}\end{array}}{\mathtt{C_0\ m}\ (\overline{\mathtt{C}}\ \overline{\mathtt{x}})\{\mathtt{return\ t_0};\}\ \mathtt{OK\ in\ R}}$$

$$\frac{\begin{array}{c}\overline{\mathtt{x}}:\overline{\mathtt{C}},\mathtt{this}:\mathtt{C}\ \vdash \mathtt{t_0}:\mathtt{E_0} \qquad \mathtt{E_0} <: \mathtt{C_0} \qquad \mathtt{R}\ =\ \mathtt{C@feat} \\ \mathtt{RT(R)}\ =\ \mathtt{refines\ R}\ \{\ldots\ \overline{\mathtt{M}},\ \overline{\mathtt{MR}}\ \ldots\} \qquad \mathtt{m}\notin\overline{\mathtt{M}} \qquad \mathtt{m}\in\overline{\mathtt{MR}} \\ \mathit{override}_R(\mathtt{m},\mathtt{R},\overline{\mathtt{C}}\to\mathtt{C}) \qquad \mathit{introduce}\ \mathtt{m}\ \mathtt{R}\end{array}}{\mathtt{refines\ C_0\ m}\ (\overline{\mathtt{C}}\ \overline{\mathtt{x}})\{\mathtt{return\ t_0};\}\ \mathtt{OK\ in\ R}}$$

$$\frac{\texttt{K} \;=\; \texttt{C}\;(\bar{\texttt{D}}\;\bar{\texttt{g}},\;\bar{\texttt{C}}\;\bar{\texttt{f}})\{\texttt{super}(\bar{\texttt{g}});\;\texttt{this}.\bar{\texttt{f}} = \bar{\texttt{f}}\} \qquad \mathit{fields}(\texttt{D}) \;=\; \bar{\texttt{D}}\;\bar{\texttt{g}} \qquad \bar{\texttt{M}}\;\texttt{OK in C}}{\texttt{class C extends D}\;\{\bar{\texttt{C}}\;\bar{\texttt{f}};\;\texttt{K}\;\bar{\texttt{M}}\}\;\texttt{OK}}$$

$$\frac{\bar{\texttt{M}}\;\texttt{OK in R} \qquad \overline{\texttt{MR}}\;\texttt{OK in R}}{\texttt{refines class R}\;\{\bar{\texttt{C}}\;\bar{\texttt{f}};\;\texttt{KR}\;\bar{\texttt{M}}\;\overline{\texttt{MR}}\}\;\texttt{OK}}$$

Lemma (Typed method has body)

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

Lemmas

Lemma (Typed method has body - Refinement)

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

Lemma (Body method has type - Refinement)

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

Lemmas

Lemma (Subtype respects method types)

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

Lemmas

Lemma (Refinement respects method types)

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

Lemmas

Lemma (A1.4 - Method Body is Typable)

If mtype $(m, C) = \overline{D} \to D$
and mbody $(\mathtt{m}, \mathtt{C}) = \overline{x}.e$
then $\exists C <: D, \exists C_0 <: D_0,\ this : D, \overline{x} : \overline{D} \vdash e : C_0$

Lemmas

Lemma (Method Body is Typable - Refinement)

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

Evaluation Context

$$E ::= \Box \mid E.f_i \mid E.m(\overline{e}) \mid e.m(\overline{e_l}, E, \overline{e_r}) \mid (C)\ E \mid new\ \ C(\overline{e_l}, E, \overline{e_r})$$

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 $e = E[(C)(newD(\overline{e}))]$, with $D \not<: C$.