1 Featherweight Java progress

Consider the Featherweight Java language without casts. Prove the following theorem:

Progress: If t is a well typed term, it is either a value or can take a step.

Proceed by induction on typing derivations.

2 Featherweight Java with Field Assignment

The original Featherweight Java (FJ) proposal, as defined by Igarashi, Pierce and Wadler, presented a minimal calculus for Java's type system. In this question, we ask you to extend the calculus with a field assignment operation.

In your solution, we expect you to use a store (similarly to how it was done in STLC with References). Field assignment needs to change the state of the object and return the object itself as a result. No new types or sequencing should be added. You need to ensure that progress and preservation still hold, but there is no need to prove it.

1. Adapt the syntax (terms and/or values) of Featherweight Java.
2. Define any new evaluations rule(s) and if your calculus requires changes in the original `E-ProjNew`, `E-InvkNew`, `E-CastNew`, `E-Field` evaluation rules, please write them again.
3. Define any new typing rule(s) and if your calculus requires changes in the original `T-Invk`, `T-New` typing rules, please write them again.

3 Featherweight Java big-step evaluation

Featherweight Java has been formally defined by Igarashi, Pierce and Wadler using small-step semantics. During the lectures and exercises we have been also working on the formalization that uses big-step semantics.

Your task is to:

1. Briefly state the difference between big-step and small-step semantics in general.

2. Give one reason why big-step might be more appropriate than small-step semantics.

3. Define a big-step evaluation relation for Featherweight Java. You are allowed to use the helper functions that have been defined in the original formalization, i.e. $fields(C) = \bar{C}\bar{f}$ and $mbody(m, C) = (\bar{x}, s)$.

4. State the **preservation** theorem for the big-step evaluation relation that you have defined without proving it.

Featherweight Java (TAPL pages 254 - 259):

CL	::=		**class declarations :**
		`class C extends D {C̄ f̄; K M̄}`	
K	::=		**constructor declarations :**
		`C(C̄ f̄) {super(`$\bar{g}$`); this.f̄=f̄;}`	
M	::=		**method declarations :**
		`C m (C̄ x̄) {return  t;}`	
t	::=		`terms:`
	|	x	**variable**
	|	t.`f`	**field access**
	|	t.`m`$(\bar{t})$	**method invocation**
	|	`new` $C(\bar{t})$	**object creation**
	|	$(C)t$	**cast**
v	::=		`values:`
	|	`new` $C(\bar{v})$	**object creation**

Typing rules:

$$\text{(T-VAR)}\ \frac{x : C \in \Gamma}{\Gamma \vdash x : C}$$

$$\text{(T-FIELD)}\ \frac{\Gamma \vdash t_0 : C_0 \qquad \mathit{fields}(C_0) = \bar{C}\bar{f}}{\Gamma \vdash t_0.\mathtt{f}_i : C_i}$$

$$\text{(T-INVK)}\ \frac{\Gamma \vdash t_0 : C_0 \qquad \mathit{mtype}(m, C_0) = \bar{D} \to C \qquad \Gamma \vdash \bar{t} : \bar{C} \quad \bar{C} <: \bar{D}}{\Gamma \vdash t_0.\mathtt{m}(\bar{t}) : C}$$

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

$$\text{(T-UCAST)}\ \frac{\Gamma \vdash t_0 : D \quad D <: C}{\Gamma \vdash (C)t_0 : C}$$

$$\text{(T-DCAST)}\ \frac{\Gamma \vdash t_0 : D \quad C <: D \quad C \neq D}{\Gamma \vdash (C)t_0 : C}$$

$$\text{(T-SCAST)}\ \frac{\Gamma \vdash t_0 : D \quad C \not<: D \quad D \not<: C \qquad \mathit{stupid\ warning}}{\Gamma \vdash (C)t_0 : C}$$

$$\text{(METHOD TYPING)}\ \frac{\bar{x} : \bar{C}, \mathit{this} : C \vdash t_0 : E_0 \quad E_0 <: C_0 \qquad CT(C) = \mathtt{class\ C\ extends\ D\{...\}} \qquad \mathit{override}(m, D, \bar{C} \to C_0)}{C_0\ \mathtt{m}\ (\bar{C}\ \bar{x})\ \{\mathtt{return}\ t_0;\}\ \mathit{OK\ in}\ C}$$

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

Evaluation rules:

$$\text{(E-ProjNew)}\ \frac{\mathit{fields}(C) = \bar{C}\,\bar{f}}{(\texttt{new}\ C(\bar{v})).\texttt{f}_i \longrightarrow v_i}$$

$$\text{(E-Invk-Recv)}\ \frac{t_0 \longrightarrow t_0'}{t_0.\texttt{m}(\bar{t}) \longrightarrow t_0'.\texttt{m}(\bar{t})}$$

$$\text{(E-InvkNew)}\ \frac{\mathit{mbody}(m, C) = (\bar{x}, t_0)}{\begin{array}{c}(\texttt{new}\ C(\bar{v})).\texttt{m}(\bar{u})\\ \longrightarrow [\bar{x} \mapsto \bar{u}, \mathit{this} \mapsto \texttt{new}\ C(\bar{v})]t_0\end{array}}$$

$$\text{(E-Invk-Arg)}\ \frac{t_i \longrightarrow t_i'}{v_0.\texttt{m}(\bar{v}, t_i, \bar{t}) \longrightarrow v_0.\texttt{m}(\bar{v}, t_i', \bar{t})}$$

$$\text{(E-CastNew)}\ \frac{C <: D}{(D)(\texttt{new}\ C(\bar{v})) \longrightarrow \texttt{new}\ C(\bar{v})}$$

$$\text{(E-New-Arg)}\ \frac{t_i \longrightarrow t_i'}{\texttt{new}\ C(\bar{v}, t_i, \bar{t}) \longrightarrow \texttt{new}\ C(\bar{v}, t_i', \bar{t})}$$

$$\text{(E-Field)}\ \frac{t_0 \longrightarrow t_0'}{t_0.\texttt{f} \longrightarrow t_0'.\texttt{f}}$$

$$\text{(E-Cast)}\ \frac{t_0 \longrightarrow t_0'}{(C)t_0 \longrightarrow (C)t_0'}$$

Auxiliary definitions

Field lookup $\mathit{fields}(C) = \bar{C}\ \bar{f}$:

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

$$\frac{\begin{array}{c}CT(C) = \texttt{class C extends D }\{\bar{\texttt{C}}\ \bar{\texttt{f}}\texttt{; K }\bar{\texttt{M}}\}\\ \mathit{fields}(D) = \bar{D}\ \bar{g}\end{array}}{\mathit{fields}(C) = \bar{D}\ \bar{g}, \bar{C}\ \bar{f}}$$

Method type lookup $\mathit{mtype}(m, c) = \bar{C} \to C$:

$$\frac{\begin{array}{c}CT(C) = \texttt{class C extends D }\{\bar{\texttt{C}}\ \bar{\texttt{f}}\texttt{; K }\bar{\texttt{M}}\}\\ \texttt{B m (}\bar{\texttt{B}}\ \bar{\texttt{x}}\texttt{) }\{\texttt{return t;}\} \in \bar{\texttt{M}}\end{array}}{\mathit{mtype}(m, C) = \bar{B} \to B}$$

$$\frac{\begin{array}{c}CT(C) = \texttt{class C extends D }\{\bar{\texttt{C}}\ \bar{\texttt{f}}\texttt{; K }\bar{\texttt{M}}\}\\ \mathbf{m}\ \texttt{is not defined in}\ \bar{M}\end{array}}{\mathit{mtype}(m, C) = \bar{B} \to \mathit{mtype}(m, D)}$$

Method body lookup $\mathit{mbody}(m, C) = (\bar{x}, t)$:

$$\frac{\begin{array}{c}CT(C) = \texttt{class C extends D }\{\bar{\texttt{C}}\ \bar{\texttt{f}}\texttt{; K }\bar{\texttt{M}}\}\\ \texttt{B m (}\bar{\texttt{B}}\ \bar{\texttt{x}}\texttt{) }\{\texttt{return t;}\} \in \bar{\texttt{M}}\end{array}}{\mathit{mbody}(m, C) = \bar{B} \to (\bar{x}, t)}$$

$$\frac{\begin{array}{c}CT(C) = \texttt{class C extends D }\{\bar{\texttt{C}}\ \bar{\texttt{f}}\texttt{; K }\bar{\texttt{M}}\}\\ \texttt{m is not defined in}\ \bar{M}\end{array}}{\mathit{mbody}(m, C) = \bar{B} \to \mathit{mbody}(m, D)}$$

Valid method overloading $\mathit{override}(m, D, \bar{C} \to C_0)$:

$$\frac{\mathit{mtype}(m, D) = \bar{D} \to D_0\ \texttt{implies}\ \bar{C} = \bar{D}\ \texttt{and}\ C_0 = D_0}{\mathit{override}(m, D, \bar{C} \to C_0)}$$