Imperative Featherweight Java

dichiarazione di variabili
assegnamenti
sequenze di istruzioni

G. Bierman, M.J. Parkinson, A.M. Pitts
MJ: An imperative core calculus for Java and Java with effects
Technical Report 563, University of Cambridge - Computer Laboratory, 2003

A. Ahern, N. Yoshida
Formalising Java RMI with explicit code mobility OOPSLA 2005.

IFJ: sintassi

Class Table con le definizioni di classi analoga a FJ

Classi	CL	::=	class C extends $D\,\{\tilde{A}\,f;\ K\ \tilde{M}\}$
Metodi	M	::=	$C\ m\ (\tilde{D}\,x)\{\text{return}\ e\}$
Costruttori	K	::=	$C(\tilde{A}\,g, \tilde{B}\,f)\{\text{super}(\tilde{g});\ \text{this}.\tilde{f} = \tilde{f}\}$

IFJ: sintassi

La sintassi sottolineata occorre solo runtime

Espressioni	e	::=	$v \mid x \mid \text{this} \mid e.f \mid \text{new}\ C(\tilde{e}) \mid e.m(\tilde{e})$
		$\mid$	e ; e | C x = e ; e | x = e | e.f = e | NPE
Valori	v	::=	null | o

Dove o è un **object identifier**: un riferimento ad un oggetto

$$C\ x = \text{new}\ C(\tilde{v})$$

dichiara una variabile x inizializata con il valore o, che è un riferimento all'oggetto costruito *in memoria*.

IFJ: sintassi

La sintassi sottolineata occorre solo runtime

Espressioni	e	::=	$v \mid x \mid \text{this} \mid e.f \mid \text{new}\ C(\tilde{e}) \mid e.m(\tilde{e})$
		$\mid$	e ; e | C x = e ; e | x = e | e.f = e | NPE
Valori	v	::=	null | o
Store	σ	::=	$\varnothing \mid \sigma \cdot [x \mapsto v] \mid \sigma \cdot [o \mapsto (C, \tilde{f}{:}v)]$
Configurazioni	F	::=	$\langle \sigma\ ,\ e \rangle$

Una configurazione rappresenta **uno stato della macchina virtuale**

IFJ: esempio

```
class D extends Object {
    Object f;
    D(Object f) { super(); this.f=f;}
    Object m() { return this; }
}
```

```
class C extends D {
    C(Object f) { super(f); }
    Object m() {return this;}
}
```

```
Object z=new Object();
C x=new C(z);
C y=new D(x);
x.m();  compila? è stuck? che metodo esegue?
x=y;
x.m()
y.f=new Object();  quanto vale x.f?
z=null; x.f=z; y.f.m();
```

$\langle \emptyset \, , \, e \rangle$
configurazione
iniziale

IFJ: Semantica Operazionale $\langle \sigma \, , \, e \rangle \longrightarrow \langle \sigma' \, , \, e' \rangle$

$$\frac{o \notin Dom(\sigma) \quad fields(C) = \tilde{A}\, \tilde{f}}{\langle \sigma \, , \, \text{new } C(\tilde{v}) \rangle \longrightarrow \langle \sigma \cdot [o \mapsto (C, \tilde{f}{:}\tilde{v})] \, , \, o \rangle} \; (NewObj)$$

$$\frac{\sigma(o) = (C, \tilde{f}{:}\tilde{v}) \quad f_i \in \tilde{f}}{\langle \sigma \, , \, o.f_i \rangle \longrightarrow \langle \sigma \, , \, v_i \rangle} \; (Field)$$

$$\frac{\sigma(o) = (C, \tilde{f}{:}\tilde{v}) \quad f \in \tilde{f}}{\langle \sigma \, , \, o.f = v' \rangle \longrightarrow \langle \sigma[o.f \mapsto v'] \, , \, v' \rangle} \; (Assegn \; Campi)$$

IFJ: Semantica Operazionale $\langle \sigma \, , \, e \rangle \longrightarrow \langle \sigma' \, , \, e' \rangle$

$$\frac{}{\langle \sigma \, , \, v; e \rangle \longrightarrow \langle \sigma \, , \, e \rangle} \; (Seq)$$

$$\frac{}{\langle \sigma \, , \, x \rangle \longrightarrow \langle \sigma \, , \, \sigma(x) \rangle} \; (Var)$$

$$\frac{}{\langle \sigma \, , \, x = v \rangle \longrightarrow \langle \sigma[x \mapsto v] \, , \, v \rangle} \; (Assegn)$$

IFJ: Semantica Operazionale

$$\frac{x \notin Dom(\sigma)}{\langle \sigma \, , \, C \; x = v; e \rangle \longrightarrow \langle \sigma \cdot [x \mapsto v] \, , \, e \rangle} \; (Dichiar \; Var)$$

$$\frac{\sigma(o) = (C, \ldots) \quad mbody(m, C) = (\tilde{x}, e) \quad \tilde{x} \notin Dom(\sigma)}{\langle \sigma \, , \, o.m(\tilde{v}) \rangle \longrightarrow \langle \sigma \cdot [\tilde{x} \mapsto \tilde{v}] \, , \, e\{\text{this} := o\} \rangle} \; (Method \; Call)$$

$$\frac{x \notin \text{fv}(e)}{\langle \sigma \cdot [x \mapsto v] \, , \, e \rangle \longrightarrow \langle \sigma \, , \, e \rangle}$$

Garbage Collection

IFJ: Garbage Collection?

$$\frac{x \notin \mathrm{fv}(e)}{\langle \sigma \cdot [x \mapsto v]\,,\ e\rangle \longrightarrow \langle \sigma\,,\ e\rangle} \qquad \frac{o \notin \mathrm{fref}(e)???}{\langle \sigma \cdot [o \mapsto (C, \tilde{f{:}v})]\,,\ e\rangle \longrightarrow \langle \sigma\,,\ e\rangle}$$

$[x \mapsto o] \in \sigma,\ e = E[..x..]$

$[x \mapsto o_1] \cdot [o_1 \mapsto (D,\ f{:}o)] \in \sigma,\ e = E[..x..]$ oppure $e = E[..o_1..]$.

$$\frac{o \notin \mathrm{fref}(e) \quad o \notin \mathrm{Codom}(\sigma)}{\langle \sigma \cdot [o \mapsto (C, \tilde{f{:}v})]\,,\ e\rangle \longrightarrow \langle \sigma\,,\ e\rangle}$$

IFJ: Semantica Operazionale

$$\frac{\langle \sigma\,,\ e\rangle \longrightarrow \langle \sigma'\,,\ e'\rangle}{\langle \sigma\,,\ E[\,e\,]\rangle \longrightarrow \langle \sigma'\,,\ E[\,e'\,]\rangle}\ (\mathit{Cong})$$

$$E ::= \ [\,] \mid E.f \mid E; e \mid x = E \mid E.f = e \mid o.f = E \mid \mathrm{new}\ C(\tilde{v}, E, \tilde{e}) \mid E.m(\tilde{e}) \mid o.m(\tilde{v}, E, \tilde{e}) \mid C\ x = E$$

$$\frac{\langle \sigma\,,\ e\rangle \longrightarrow \langle \sigma'\,,\ e'\rangle}{\langle \sigma, \mathrm{new}\ C(v, e)\rangle \longrightarrow \langle \sigma', \mathrm{new}\ C(v, e')\rangle} \qquad \frac{\langle \sigma\,,\ e\rangle \longrightarrow \langle \sigma'\,,\ e'\rangle}{\langle \sigma, C\ x{=}o.m(e)\rangle \longrightarrow \langle \sigma', C\ x{=}o.m(e')\rangle}$$

$\mathrm{new}\ C(v, e) = E[e]$ with $E = \mathrm{new}\ C(v, [])$

$C\ x = o.m(e) = E_1[E_2[e]]$ with $E_1 = C\ x = E_2[]\ \ E_2 = o.m([])$

IFJ: Semantica Operazionale

$$\frac{}{\langle \sigma\,,\ \mathrm{null}.f\rangle \longrightarrow \langle \sigma\,,\ \mathrm{NPE}\rangle} \qquad \frac{}{\langle \sigma\,,\ \mathrm{null}.f = v\rangle \longrightarrow \langle \sigma\,,\ \mathrm{NPE}\rangle}$$

$$\frac{}{\langle \sigma\,,\ \mathrm{null}.m(\tilde{v})\rangle \longrightarrow \langle \sigma\,,\ \mathrm{NPE}\rangle} \qquad \frac{}{\langle \sigma\,,\ E[\,\mathrm{NPE}\,]\rangle \longrightarrow \langle \sigma\,,\ \mathrm{NPE}\rangle}$$

e.g. $\langle \sigma\,,\ o.m(v_1, \mathrm{NPE}, e)\rangle \longrightarrow \langle \sigma\,,\ \mathrm{NPE}\rangle$

$$\langle \sigma\,,\ e.f\rangle \longrightarrow^* \begin{cases} \langle \sigma'\,,\ o.f\rangle & \longrightarrow\ v & \text{se } f \text{ è uno dei campi} \\ \langle \sigma'\,,\ o.f\rangle & \mathrm{stuck} & \text{se } f \text{ non è uno dei campi} \\ \langle \sigma'\,,\ \mathrm{null}.f\rangle & \longrightarrow & \mathrm{NPE} \end{cases}$$

IFJ: esempio

```
class D extends Object {
    Object f;
    D(Object f) { super(); this.f=f;}
    Object m() { return this; }
}
```

```
class C extends D {
    C(Object f) { super(f); }
    Object m() {return this;}
}
```

```
Object z=new Object();
C x=new C(z);
C y=new D(x);
x.m();  compila? è stuck? che metodo esegue?
x=y;
x.m()
y.f=new Object();  quanto vale x.f?
z=null; x.f=z; y.f.m();
```

IFJ: esempio

$\langle \emptyset\,,\ \ \text{Object } z = \text{new Object}();\ \ C\ x = \text{new } C(z); e\rangle$

$\longrightarrow \langle [o_1 \mapsto \text{Object}]\,,\ \ \text{Object } z = o_1;\ \ C\ x = \text{new } C(z); e\rangle$

$\longrightarrow \langle [o_1 \mapsto \text{Object}][z \mapsto o_1]\,,\ \ C\ x = \text{new } C(z); e\rangle$

$\longrightarrow \langle [o_1 \mapsto \text{Object}][z \mapsto o_1]\,,\ \ C\ x = \text{new } C(o_1); e\rangle$

$\longrightarrow \langle [o_1 \mapsto \text{Object}][z \mapsto o_1][o_2 \mapsto (C, f{:}o_1)]\,,\ \ C\ x = o_2; e\rangle$

$\longrightarrow \langle [o_1 \mapsto \text{Object}][z \mapsto o_1][o_2 \mapsto (C, f{:}o_1)][x \mapsto o_2]\,,\ \ e\rangle$

IFJ: type system

Anche in IFJ i tipi sono i nomi delle classi

$$\textit{Contesti}\quad \Gamma \quad ::= \quad \varnothing \mid \Gamma, x : C \mid \Gamma, o : C$$

Il subtyping è identico a FJ.

Il type system utilizza i seguenti giudizi:

- $D\ m\ (\tilde{C}\,x)\ \{\text{return } e;\}\ OK\ in\ C$
  $\text{class } C \text{ extends } D\,\{\tilde{C}\,f;\ K\ \tilde{M}\}\ OK$ con stesse regole del caso FJ
- $\Gamma \vdash e : C$ per indicare che il termine e ha tipo C in Γ.
- $\Gamma \vdash \langle \sigma\,,\ e\rangle$ per indicare che la configurazione è ben tipata in Γ.
- $\Gamma \vdash \sigma$ per indicare che lo store è ben tipato in Γ.

IFJ: type system

$$\frac{\Gamma \vdash e : C \qquad \Gamma \vdash \sigma}{\Gamma \vdash \langle \sigma\,,\ e\rangle}\ (\textit{Config})$$

$$\frac{}{\Gamma \vdash \varnothing} \qquad \frac{\Gamma \vdash \sigma \quad x \notin \textit{Dom}(\sigma) \quad \Gamma \vdash x : C \quad \Gamma \vdash v : D \quad D <: C}{\Gamma \vdash \sigma \cdot [x \mapsto v]}$$

$$\frac{\Gamma \vdash \sigma \quad o \notin \textit{Dom}(\sigma) \quad \Gamma \vdash o : C \quad \textit{fields}(C) = \tilde{A}\,\tilde{f} \quad \Gamma \vdash \tilde{v} : \tilde{B} \quad \tilde{B} <: \tilde{A}}{\Gamma \vdash \sigma \cdot [o \mapsto (C, \tilde{f{:}v})]}$$

(IFJ: type system)

$$\frac{}{\Gamma \vdash \text{null} : C} \qquad \frac{o : C \in \Gamma}{\Gamma \vdash o : C} \qquad \frac{x : C \in \Gamma}{\Gamma \vdash x : C}$$

$$\frac{\Gamma \vdash e_1 : C \quad \Gamma \vdash e_2 : D}{\Gamma \vdash e_1; e_2 : D}\ (\textit{Seq}) \qquad \frac{\Gamma \vdash e : C \quad C <: D \quad \Gamma \vdash x : D}{\Gamma \vdash x = e : C}\ (\textit{Assegn})$$

$$\frac{\Gamma \vdash e : C \quad C <: D \quad \Gamma, x : D \vdash e' : A}{\Gamma \vdash D\ x = e; e' : A}\ (\textit{Dichiar})$$

IFJ: type system

$$\frac{\Gamma \vdash e.f : D \quad C <: D \quad \Gamma \vdash e' : C}{\Gamma \vdash e.f = e' : C} \ (\textit{Field Assign})$$

$$\frac{\Gamma \vdash e : C \quad \textit{field}(C) = \tilde{D}\, f}{\Gamma \vdash e.f_i : D_i} \ (\textit{Field})$$

$$\frac{\textit{fileds}(C) = \tilde{D}\, f \quad \Gamma \vdash \tilde{e} : \tilde{C} \quad \tilde{C} <: \tilde{D}}{\Gamma \vdash \text{new } C(\tilde{e}) : C} \ (\textit{New})$$

$$\frac{\Gamma \vdash e : C \quad \textit{mtype}(m, C) = \tilde{A} \to B \quad \Gamma \vdash \tilde{e} : \tilde{D} \quad \tilde{D} <: \tilde{A}}{\Gamma \vdash e.m(\tilde{e}) : B} \ (\textit{Method})$$

IFJ: type system

$$\frac{}{\Gamma \not\vdash \text{NPE} : T}$$

Nel linguaggio funzionale c'era $\Gamma \vdash \text{throw } v : T$ perché c'era anche try... catch ..

qui invece NPE è un errore non gestibile
nessun termine $E[\text{NPE}]$ sarà mai ben tipato.

IFJ: Proprietà del sistema di tipi

Subject Reduction

- Se $\Gamma \vdash F$ e $F \longrightarrow F'$ con $F' \neq E[\text{NPE}]$, allora $\Gamma \vdash F'$
- Se $\Gamma \vdash e : C$ e $\Gamma \vdash \sigma$ ed inoltre $\langle \sigma , e \rangle \longrightarrow \langle \sigma' , e' \rangle$ con $e' \neq E[\text{NPE}]$, allora $\Gamma \vdash e' : C'$ con $C' <: D$ e $\Gamma \vdash \sigma'$

Progressione

Se $\varnothing \vdash F$, allora $\exists F'.\ F \longrightarrow F'$ oppure $F = \langle \sigma , v \rangle$.

Safety

Se $\varnothing \vdash F$, allora $F \longrightarrow^* \langle \sigma , v \rangle$ per qualche v oppure $F \longrightarrow^* \text{NPE}$.

Tipiamo l'espressione $C\ x = e_1\ ;\ e_2\ ;\ x = o$.

$$\frac{\dfrac{\Gamma \vdash e_1 : A \quad A <: C \quad \dfrac{\ldots x \notin \textit{Dom}(\Gamma) \ldots}{\Gamma, x : C \vdash e_2 : D'}}{\Gamma \vdash C\ x = e_1; e_2 : D'} \quad \dfrac{\Gamma \vdash o : D \quad \dfrac{x \in \textit{Dom}(\Gamma)}{\Gamma \vdash x : B} \quad D <: B}{\Gamma \vdash x = o : D}\ (\textit{Seq})}{\Gamma \vdash (C\ x = e_1; e_2)\ ; x = o : D}$$

$$\frac{\Gamma \vdash e_1 : A \quad A <: C \qquad \dfrac{\dfrac{\ldots}{\Gamma, x : C \vdash e_2 : D'} \qquad \dfrac{\Gamma, x : C \vdash o : D \quad \dfrac{x \in \textit{Dom}(\Gamma, x : C)}{\Gamma, x : C \vdash x : C} \quad D <: C}{\Gamma, x : C \vdash x = o : D}\ (\textit{Seq})}{\Gamma, x : C \vdash e_2\ ;\ x = o : D}\ (\textit{Dich})}{\Gamma \vdash C\ x = e_1\ ;\ (e_2\ ;\ x = o) : D}$$

Che cosa deve essere inteso come un errore?

Quali errori prevenire **staticamente** e
su quali posticipare il controllo a **runtime**??

È una scelta di design!

Nei linguaggi OO un **message-not-understood** è un errore ?

- Nei linguaggi OO con ***typing statico*** la compilazione assicura che non ci siano **mai** errori runtime MNU
- Nei linguaggi con ***duck typing*** possono comparire runtime MNU

Nei linguaggi OO **fare un cast ad un tipo non maggiore** è un errore ?
Nei linguaggi OO **dereferenziare un riferimento null** è un errore ?

Che cosa deve essere inteso come un errore?

Quali errori prevenire staticamente e
su quali posticipare il controllo a runtime??

È una scelta di design!

- in **Java** si prevengono staticamente i MNU e tutte le user-defined exceptions. Si posticipa il controllo per le `RuntimeExceptions`, es. NPE, CCE ..
- in **Scala** nessuna eccezione deve essere per forza gestita.

  Il linguaggio offre primitive/astrazioni adatte a programmare in modo uniforme anche i casi speciali (e.g. `Option[T]`)

  Quindi le *"eccezione"* rappresentano solo *"errori runtime"*, che dunque terminano l'esecuzione.