Eine denotationelle Semantik für Featherweight Java

Thomas Studer
Universität Bern

Informatik Kolloquium
Universität Tübingen
Oktober 2000

Übersicht

1. Featherweight Java
2. Das Objekt Modell
3. Beispiele
4. Explizite Mathematik
5. Die Interpretation von FJ
6. Ausblick

Featherweight Java

Von Igarashi, Pierce und Wadler.

Syntax.

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

Featherweight Java

Class Table CT: Abbildung von Klassennamen `C` auf Klassendeklaration `CL`.
Programm: Paar $(CT, \mathtt{e})$ bestehend aus einer Class Table und einem Ausdruck. Die Class Table erfüllt folgende Bedingungen:

1. $CT(\mathtt{C}) = \mathtt{class\ C} \ldots$ für jedes `C` im Domain von CT, d.h. der Klassenname `C` wird auf die Deklaration der Klasse `C` abgebildet,

2. `Object` ist kein Element des Domains von CT,

3. Jede Klasse `C` (ausser `Object`) die irgendwo in CT auftritt gehört zum Domain von CT,

4. Die Subtyp-Relation welche von CT induziert wird enthält keine Zyklen, d.h. $<:$ ist anti-symmetrisch.

Featherweight Java

Subtyping. $<:$

$$\mathtt{C} <: \mathtt{C}$$

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

$$\frac{CT(\mathtt{C}) = \mathtt{class\ C\ extends\ D}\ \{\ldots\}}{\mathtt{C} <: \mathtt{D}}$$

Computation. $\longrightarrow$

$$\frac{\mathit{fields}(\mathtt{C}) = \overline{\mathtt{C}}\ \overline{\mathtt{f}}}{\mathtt{new\ C}(\overline{\mathtt{e}}).\mathtt{f_i} \longrightarrow \mathtt{e_i}}$$

$$\frac{\mathit{mbody}(\mathtt{m}, \mathtt{C}) = (\overline{\mathtt{x}}, \mathtt{e_0})}{\mathtt{new\ C}(\overline{\mathtt{e}}).\mathtt{m}(\overline{\mathtt{d}}) \longrightarrow \mathtt{e_o}[\overline{\mathtt{d}}/\overline{\mathtt{x}}, \mathtt{new\ C}(\overline{\mathtt{e}})/\mathtt{this}]}$$

$$\frac{\mathtt{C} <: \mathtt{D}}{(\mathtt{D})\mathtt{new\ C}(\overline{\mathtt{e}}) \longrightarrow \mathtt{new\ C}(\overline{\mathtt{e}})}$$

Ein Ausdruck e ist in *Normalform*, falls es keinen Ausdruck d gibt mit $\mathtt{e} \longrightarrow \mathtt{d}$.

Featherweight Java

Expression typing.

$$\Gamma \vdash \mathtt{x} \in \Gamma(\mathtt{x})$$

$$\frac{\Gamma \vdash \mathtt{e}_0 \in \mathtt{C}_0 \qquad \mathit{fields}(\mathtt{C}_0) = \overline{\mathtt{C}}\ \overline{\mathtt{f}}}{\Gamma \vdash \mathtt{e}_0.\mathtt{f}_\mathtt{i} \in \mathtt{C}_\mathtt{i}}$$

$$\frac{\begin{array}{c}\Gamma \vdash \mathtt{e}_\mathtt{o} \in \mathtt{C}_0 \\ \mathit{mtype}(\mathtt{m}, \mathtt{C}_0) = \overline{\mathtt{D}} \to \mathtt{C} \\ \Gamma \vdash \overline{\mathtt{e}} \in \overline{\mathtt{C}} \qquad \overline{\mathtt{C}} <: \overline{\mathtt{D}}\end{array}}{\Gamma \vdash \mathtt{e}_0.\mathtt{m}(\overline{\mathtt{e}}) \in \mathtt{C}}$$

$$\frac{\begin{array}{c}\mathit{fields}(\mathtt{C}) = \overline{\mathtt{D}}\ \overline{\mathtt{f}} \\ \Gamma \vdash \overline{\mathtt{e}} \in \overline{\mathtt{C}} \qquad \overline{\mathtt{C}} <: \overline{\mathtt{D}}\end{array}}{\Gamma \vdash \mathtt{new}\ \mathtt{C}(\overline{\mathtt{e}}) \in \mathtt{C}}$$

$$\frac{\Gamma \vdash \mathtt{e}_0 \in \mathtt{D} \qquad \mathtt{D} <: \mathtt{C}}{\Gamma \vdash (\mathtt{C})\mathtt{e}_0 \in \mathtt{C}}$$

Featherweight Java

Expression typing (2).

$$\frac{\Gamma \vdash \mathsf{e}_0 \in \mathtt{D} \qquad \mathtt{C} <: \mathtt{D} \qquad \mathtt{C} \neq \mathtt{D}}{\Gamma \vdash (\mathtt{C})\mathsf{e}_0 \in \mathtt{C}}$$

$$\frac{\Gamma \vdash \mathsf{e}_0 \in \mathtt{D} \qquad \mathtt{C} \not<: \mathtt{D} \qquad \mathtt{C} \not<: \mathtt{D}}{\text{stupid warning}}$$
$$\Gamma \vdash (\mathtt{C})\mathsf{e}_0 \in \mathtt{C}$$

Featherweight Java

Method typing.

$$\frac{\overline{\mathtt{x}} : \overline{\mathtt{C}}, \mathtt{this} : \mathtt{C} \vdash \mathtt{e}_0 \in \mathtt{E}_0 \qquad \mathtt{E}_0 <: \mathtt{C}_0 \qquad CT(\mathtt{C}) = \mathtt{class\ C\ extends\ D}\ \{\ldots\} \qquad \text{if } mtype(\mathtt{m},\mathtt{D}) = \overline{\mathtt{D}} \to \mathtt{D}_0, \text{ then } \overline{\mathtt{C}} = \overline{\mathtt{D}} \text{ and } \mathtt{C}_0 = \mathtt{D}_0}{\mathtt{C}_0\ \mathtt{m}\ (\overline{\mathtt{C}}\ \overline{\mathtt{x}})\ \{\mathtt{return\ e}_0;\}\ \text{OK in } \mathtt{C}}$$

Class typing.

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

Featherweight Java

Hilfsfunktionen für die Berechnungs- und Typisierungsregeln.
Field lookup.

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

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

Method type lookup.

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

$$\frac{CT(\texttt{C}) = \texttt{class C extends D } \{\overline{\texttt{C}}\ \overline{\texttt{f}};\ \texttt{K}\ \overline{\texttt{M}}\} \qquad \texttt{m} \text{ is not defined in } \overline{\texttt{M}}}{\mathit{mtype}(\texttt{m}, \texttt{C}) = \mathit{mtype}(\texttt{m}, \texttt{D})}$$

Featherweight Java

Method body lookup.

$$\frac{CT(\mathtt{C}) = \mathtt{class\ C\ extends\ D}\ \{\overline{\mathtt{C}}\ \overline{\mathtt{f}};\ \mathtt{K}\ \overline{\mathtt{M}}\} \quad \mathtt{B\ m}\ (\overline{\mathtt{B}}\ \overline{\mathtt{x}})\ \{\mathtt{return\ e;}\}\ \text{belongs to}\ \overline{\mathtt{M}}}{mbody(\mathtt{m}, \mathtt{C}) = (\overline{\mathtt{x}}, \mathtt{e})}$$

$$\frac{CT(\mathtt{C}) = \mathtt{class\ C\ extends\ D}\ \{\overline{\mathtt{C}}\ \overline{\mathtt{f}};\ \mathtt{K}\ \overline{\mathtt{M}}\} \quad \mathtt{m}\ \text{is not defined in}\ \overline{\mathtt{M}}}{mbody(\mathtt{m}, \mathtt{C}) = mbody(\mathtt{m}, \mathtt{D})}$$

Ein Ausdruck e heisst *wohl-getypt*, falls $\Gamma \vdash \mathtt{e} \in \mathtt{C}$ hergeleitet werden kann für ein Γ und eine Klasse C.

Featherweight Java

Theorem 1. *Sei* $\mathtt{e}$ *ein wohl-getypter Ausdruck.*

1. *Falls* $\mathtt{e}$ *von der Form* $\mathtt{new}\ \mathtt{C}_0(\overline{\mathtt{e}}).\mathtt{f}$ *ist oder einen solchen Teilausdruck enthält, dann gilt* $\mathit{fields}(\mathtt{C}_0) = \overline{\mathtt{D}}\ \overline{\mathtt{f}}$ *und* $\mathtt{f} \in \overline{\mathtt{f}}$.

2. *Falls* $\mathtt{e}$ *von der Form* $\mathtt{new}\ \mathtt{C}_0(\overline{\mathtt{e}}).\mathtt{m}(\overline{\mathtt{d}})$ *ist oder einen solchen Teilausdruck enthält, dann* $\mathit{mbody}(\mathtt{m}, \mathtt{C}_0) = (\overline{\mathtt{x}}, \mathtt{e}_0)$ *und* $\sharp(\overline{\mathtt{x}}) = \sharp(\overline{\mathtt{d}})$.

Das Objekt Modell

Das *overloading* basierte Objekt Modell stammt von Castagna, Ghelli, Longo.

- n Funktionen $f_i : S_i \to T_i \quad (1 \leq i \leq n)$ werden zu einer überladenen Funktion f des Typs $\{S_1 \to T_1, \ldots, S_n \to T_n\}$ kombiniert

- f_i heisst *Ast* von f

- Der Typ des Arguments x wählt einen Ast f_i. Das Resultat von $f(x)$ ist dann $f_i(x)$

- Early-binding vs. late-binding

- Subtyping

Das Objekt Modell

- Zustand eines Objektes ist von seinen Methoden getrennt.
- Nur die Felder sind zu einer Einheit zusammengefasst.
- Methoden sind nicht gekapselt
- Methoden sind als Äste von globalen überladenen Funktionen implementiert.
- Wenn eine Message einem Objekt geschickt wird, dann bestimmt diese Message eine Funktion, welche dann auf das empfangende Objekt angewendet wird.

Das Objekt Modell

- Messages identifizieren nicht gewöhnliche Funktionen.
- Wenn dieselbe Message zu Objekten aus verschiedenen Klassen geschickt wird, dann können verschiedene Methoden bezeichnet werden, d.h. unterschiedlicher Code wird ausgeführt.
- Messages identifizieren überladene Funktionen: Je nach Argument-Typ (Klasse des empfangenden Objektes) wird eine andere Methode gewählt.
- Diese Auswahl basiert auf dem dynamischen Typ des Objekts: Wir müssen mit late-binding arbeiten.

Das Objekt Modell

- Ein Objekt wird durch einen Record modelliert der aus allen Feldern des Objektes besteht.
- Ebenfalls wird der run-time Typ des Objektes im interpretierenden Record kodiert.
- Alle Methoden mit demselben Namen werden zu einer überladenen Funktion zusammengefasst. Diese nimmt das emfangende Objekt als zusätzliches Argument.
- Also kann die Typ-Information, welche in der Interpretation des empfangenden Objektes enthalten ist, zur Auflösung dieser überladenen Anwendung verwendet werden.
- Methoden können durch verschränkte Rekursion definiert werden, also muss deren Interpretation durch eine rekursive Definition gegeben werden, welche alle Methoden auf einmal behandelt.

Beispiel: Endlosschleifen

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

  C m() {
    return this.m();
  }
}

class C extends Object {
    int x;
    C y;
    C (int a,C b) {
        super();
        this.x = a;
        this.y = b;
    }
}
```

`new A().m()` ist eine Endlosschleife.
Sei e der Ausdruck `new C(5,new A().m()).x`.
Java: nicht-terminierend.
FJ: nicht-terminierend oder 5.
EM: $\neg[\![e]\!]\downarrow$

Beispiel: Illegaler Down-Cast oder korrekte Terminierung ?

```
class main extends Object{
  public static void main (String arg[]) {
      System.out.println
        (new C(5,(C)(new Object())).x);
  }
}
```

In Java wirft diese `main` Methode eine `java.lang.ClassCastException`.

In FJ reduziert

```
new C(5,(C)(new Object())).x
```

zu 5.
Der Term `(C)(new Object())` verursacht die Exception in Java.

Beispiel: Illegaler Down-Cast oder korrekte Terminierung ?

Der Wert ex bezeichnet den Typ des Resultates einer Berechnung welche eine Exception wirft. Es gilt

$$\begin{aligned}[\![\texttt{(C)(new Object())}]\!] &= \mathsf{cast}\,\texttt{C}^*\,[\![\texttt{new Object()}]\!] \\ &= \mathsf{cast}\,\texttt{C}^*\,(\texttt{Object}^*, 0) \\ &= (\mathsf{ex}, 0),\end{aligned}$$

da $\mathsf{sub}(\texttt{Object}^*, \texttt{C}^*) \neq 1$, d.h. `Object` ist keine Subklasse von `C`.

Unser Modell kann nicht korrekt sein bezüglich der Reduktionsrelation von FJ. Wir werden eine neue Reduktionsrelation $\longrightarrow'$ einfuehren, welche eine call-by-value Strategie implementiert und welche illegale Down-Casts respektiert. Bezüglich dieser neuen Relation können wir Korrektheit beweisen.

Beispiel: Typisierung

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

  C m() {
    return this.m();
  }
}
```

- `new A().m()` $\in$ `C` ist herleitbar, obwohl `new A().m()` eine Endlosschleife ist.

- e $\in$ C muss interpretiert werden als

$$[\![\mathsf{e}]\!]\downarrow \rightarrow [\![\mathsf{e}]\!] \in [\![\mathsf{C}]\!].$$

Beispiel: kleinste Fixpunkte

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

  C m() {
    return this.m();
  }
}

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

  D m() {
    return this.m();
  }
}
```

Beispiel: kleinste Fixpunkte

- In beiden Klassen `A` und `B` ist der Methodenkörper für `m` derselbe.
- Also sollten `(new A()).m()` und `(new B()).m()` gleich interpretiert werden, d.h.

$$[\![\texttt{new A().m()}]\!] \simeq [\![\texttt{new B().m()}]\!].$$

- `C` und `D` können beliebige Klassen sein, insbesondere disjunkte.
- Also können wir $\neg[\![\texttt{(new B()).m()}]\!]{\downarrow}$ beweisen.
- Wir müssen einen *kleinsten* Fixpunkt-Operator verwenden, um die Berechnungen zu modellieren.

Beispiel: freie Variablen

```
class A extends Object{
  A () { super(); }
  C m() {
    return this.m();
  }
}

class B extends A{
  B () { super(); }
  C m() {
    return new C();
  }
}
```

- Sei `x` eine freie Variable mit (statischem) Typ `A`.
- Man weiss nicht welche Methode beim Aufruf von `x.m()` ausgeführt wird.
- Es gibt Terme in Normalform die undefiniert sind.
- Nur die Interpretation von Termen in *geschlossener* Normalform ist immer definiert.

Explizite Mathematik

- Feferman, 1975
- Formale Grundlage für Bishop's konstruktive Mathematik
- Wichtig in der Beweistheorie, speziell für die Analyse von Teilsystemen der zweitstufigen Arithmetik und der Mengenlehre.
- Axiomatischer Rahmen um Programme zu repräsentieren und um Eigenschaften von Programmen zu formulieren und zu beweisen

Explizite Mathematik

- Zweisortige Sprache: Individuenvariablen $a, b, c, \ldots$, Typvariablen $S, T, U, \ldots$
- Konstanten: $\mathsf{k}, \mathsf{s}, \mathsf{p}, \mathsf{p}_0, \mathsf{p}_1, 0, \mathsf{s}_\mathsf{N}, \mathsf{p}_\mathsf{N}, \mathsf{d}_\mathsf{N}, \mathsf{c}_t$ (für jeden geschlossenen Term t)
- Generatoren: $\mathsf{nat}, \mathsf{c}_e$ (für jede natürliche Zahl e), j, dc
- Terme sind abgeschlossen unter (partieller) Applikation $\cdot$
- Formeln: $s{\downarrow}, \mathsf{N}(s), s = t, s \in U, U = V$ und $\Re(s, U)$. Abgeschlossen unter den üblichen Junktoren sowie Quantifikation in beiden Sorten
- Elementare Formel: ohne $\Re$ Prädikat und ohne Quantifikation über Typen

Explizite Mathematik

I. Applikative Axiome

(1) $\mathsf{k}ab = a$,

(2) $\mathsf{s}ab{\downarrow} \wedge \mathsf{s}abc \simeq ac(bc)$,

(3) $\mathsf{p}_0(a,b) = a \wedge \mathsf{p}_1(a,b) = b$,

(4) $0 \in \mathsf{N} \wedge (\forall x \in \mathsf{N})(\mathsf{s}_\mathsf{N}x \in \mathsf{N})$,

(5) $(\forall x \in \mathsf{N})(\mathsf{s}_\mathsf{N}x \neq 0 \wedge \mathsf{p}_\mathsf{N}(\mathsf{s}_\mathsf{N}x) = x)$,

(6) $(\forall x \in \mathsf{N})(x \neq 0 \rightarrow \mathsf{p}_\mathsf{N}x \in \mathsf{N} \wedge \mathsf{s}_\mathsf{N}(\mathsf{p}_\mathsf{N}x) = x)$,

(7) $a \in \mathsf{N} \wedge b \in \mathsf{N} \wedge a = b \rightarrow \mathsf{d}_\mathsf{N}xyab = x$,

(8) $a \in \mathsf{N} \wedge b \in \mathsf{N} \wedge a \neq b \rightarrow \mathsf{d}_\mathsf{N}xyab = y$.

Explizite Mathematik

Lemma 2. *1. Für jeden Term t und alle Variablen x gibt es einen Term $(\lambda x.t)$ mit denselben Variablen wie t, ohne x, so dass* PTN *beweist:*

$$(\lambda x.t)\downarrow \wedge\ (\lambda x.t)x \simeq t,$$

$$s\downarrow \rightarrow (\lambda x.t)s \simeq t[s/x].$$

2. Es gibt einen geschlossenen Term rec*, so dass* PTN *beweist:*

$$\mathsf{rec}f\downarrow \wedge\ \forall x(\mathsf{rec}fx \simeq f(\mathsf{rec}f)x).$$

Explizite Mathematik

II. Ontologische Axiome

(9) $\exists x \Re(x, U)$,

(10) $\Re(a, U) \wedge \Re(a, V) \rightarrow U = V$,

(11) $\forall x(x \in U \leftrightarrow x \in V) \rightarrow U = V$.

Explizite Mathematik

III. Typ Existenz Axiome:

Elementare Komprehension: Sei $F(x, \vec{y}, \vec{Z})$ eine elementare Formel mit höchstens den angezeigten freien Variablen und mit Gödelnummer e.

(12) $\Re(\vec{b}) \to \Re(\mathsf{c}_e(\vec{a}, \vec{b}))$,

(13) $\Re(\vec{b}, \vec{T}) \to \forall x(x \mathrel{\dot\in} \mathsf{c}_e(\vec{a}, \vec{b}) \leftrightarrow F(x, \vec{a}, \vec{T}))$.

Explizite Mathematik

Join

(14) $\Re(a) \wedge (\forall x \mathrel{\dot\in} a)\Re(fx) \rightarrow \Re(\mathsf{j}(a,f)) \wedge \Sigma(a,f,\mathsf{j}(a,f))$.

wobei $\Sigma(a,f,b)$ bedeutet, dass b ein Name ist für die disjunkte Vereinigung von f über a, d.h.

$$\Sigma(a,f,b) := \forall x(x \mathrel{\dot\in} b \leftrightarrow \exists y \exists z(x = (y,z) \wedge y \mathrel{\dot\in} a \wedge z \mathrel{\dot\in} fy)).$$

Explizite Mathematik

Dependent choice.

(dc.1) $\Re(a) \wedge f \in (\Re \to \Re) \to \mathsf{dc}(a, f) \in (\Re \to \Re)$,

(dc.2) $\Re(a) \wedge f \in (\Re \to \Re) \to$
$\mathsf{dc}(a, f)0 \simeq a \wedge (\forall n \in \mathsf{N})\mathsf{dc}(a, f)(n + 1) \simeq f(\mathsf{dc}(a, f)n)$.

VI. Typinduktion

(T-$\mathsf{I_N}$) $\forall X(0 \in X \wedge (\forall x \in \mathsf{N})(x \in X \to \mathsf{s_N}x \in X) \to (\forall x \in \mathsf{N})x \in X)$.

Explizite Mathematik

Eine *Fixpunkt Spezifikation* ist ein System von Formeln der Form

$$\begin{aligned} (X)_1 &= Y_{11} \times \cdots \times Y_{1m_1} \\ &\vdots \\ (X)_n &= Y_{n1} \times \cdots \times Y_{nm_n} \end{aligned}$$

wobei Y_{ij} eine Typvariable (ungleich X) ist oder von der Form

$$\{x \in X \mid \mathsf{p}_0 x = k_1\} \cup \cdots \cup \{x \in X \mid \mathsf{p}_0 x = k_l\}$$

für $k_i \leq n$.

Explizite Mathematik

Es gibt einen geschlossenen Term t, so dass

1. $\Re(\vec{a}) \wedge \Re(b) \rightarrow \Re(t(\vec{a}, b))$,

2. $\Re(\vec{a}, \vec{Y}) \wedge \Re(b, X) \rightarrow \forall x(x \mathrel{\dot\in} t(\vec{a}, b) \leftrightarrow (x)_1 \in Y_{11} \times \cdots \times Y_{1m_1} \vee \ldots \vee (x)_n \in Y_{n1} \times \cdots \times Y_{nm_n})$.

Theorem 3. *Es gibt einen geschlossenen Term* fp*, so dass*

$$\Re(\vec{a}) \rightarrow \forall x(x \mathrel{\dot\in} \mathsf{fp}(\vec{a}) \leftrightarrow x \mathrel{\dot\in} t(\vec{a}, \mathsf{fp}(\vec{a}))).$$

Explizite Mathematik

Computability.

(Comp.1) $\forall x(\forall n \in \mathsf{N})(\mathsf{c}_t(x,n) = 0 \vee \mathsf{c}_t(x,n) = 1)$,

(Comp.2) $\forall x(tx{\downarrow} \leftrightarrow (\exists n \in \mathsf{N})\mathsf{c}_t(x,n) = 0)$.

Alles ist eine natürliche Zahl: $\forall x \mathsf{N}(x)$.

Explizite Mathematik

Definition 4. *Seien $A_1, \ldots, A_n, B_1, \ldots, B_n$ Typen und sei $\mathcal{T}$ der Typ $(A_1 \curvearrowright B_1) \cap \cdots \cap (A_n \curvearrowright B_n)$. Wir definieren:*

$$
\begin{aligned}
f \sqsubseteq_{B_i} g &:= f \in B_i \to f = g, \\
f \sqsubseteq_{\mathcal{T}} g &:= f \in \mathcal{T} \to \\
&\qquad (g \in \mathcal{T} \wedge \bigwedge_{1 \le i \le n} (\forall x \in A_i) fx \sqsubseteq_{B_i} gx), \\
f \cong_{B_i} g &:= f \sqsubseteq_{B_i} g \wedge g \sqsubseteq_{B_i} f, \\
f \cong_{\mathcal{T}} g &:= f \sqsubseteq_{\mathcal{T}} g \wedge g \sqsubseteq_{\mathcal{T}} f.
\end{aligned}
$$

Definition 5. *Sei $\mathcal{T}$ wie in Definition 4. Eine Funktion $f \in (\mathcal{T} \to \mathcal{T})$ heisst $\mathcal{T}$* monoton, falls

$$(\forall g \in \mathcal{T})(\forall h \in \mathcal{T})(g \sqsubseteq_{\mathcal{T}} h \to fg \sqsubseteq_{\mathcal{T}} fh).$$

Explizite Mathematik

Theorem 6. *Es gibt einen geschlossenen Term* I*, so dass für alle Typen* $\mathcal{T}$ *wie in Definition 4 und alle monotonen Funktionale* $g \in (\mathcal{T} \to \mathcal{T})$ *gilt:*

1. $\mathsf{I}g \in \mathcal{T}$,

2. $\mathsf{I}g \cong_{\mathcal{T}} g(\mathsf{I}g)$,

3. $gf \cong_{\mathcal{T}} f \to \mathsf{I}g \sqsubseteq_{\mathcal{T}} f$.

Explizite Mathematik

- Berechenbarkeitsaxiome sind durch Kleene's T motiviert. Also werden diese Axiome durch eine rekursionstheoretische Interpretation der applikativen Axiome erfüllt.

- Probst entwickelte eine rekursionstheoretische Semantik für *dependent choice*.

- PTN ist eine prädikative Theorie, welche beweistheoretisch ein wenig stärker ist als Peano Arithmetik aber schwächer als z.B. ML_1.

- Unser System ist viel schwächer als die Theorien, welche üblicherweise verwendet werden um objekt-orientierte Programmierung zu modellieren. Bruce, Cardelli und Pierce z.B. verwenden Erweiterungen von System F um verschiede Objekt-Modelle zu vergleichen.

Interpretation von FJ

Sei $*$ eine injektive Abbildung von allen Namen für Klassen, Basic Types, Felder und Methoden die in der Class Table auftreten in die natürlichen Zahlen.

Ein Objekt der Klasse C mit den Feldern $f_1, \ldots, f_m$ wird durch einen Sequenz

$$(\mathtt{C}^*, (s_1, \ldots, s_n))$$

interpretiert, wobei der Inhalt des Feldes f_i an der f_i^*-ten Stelle der Sequenz modelliert wird.

Wir können geschlossene Terme proj, new und cast definieren.

Interpretation von FJ

$$
\begin{aligned}
[\![\mathtt{x}]\!]_{\mathsf{mc}} &:= x \\
[\![\mathtt{e.f}]\!]_{\mathsf{mc}} &:= \mathsf{proj}\,\mathtt{f}^*\,[\![\mathtt{e}]\!]_{\mathsf{mc}} \\
[\![\mathtt{e.m}(\overline{\mathtt{f}})]\!]_{\mathsf{mc}} &:= \mathsf{mc}(\mathtt{m}^*, [\![\mathtt{e}]\!]_{\mathsf{mc}}, [\![\overline{\mathtt{f}}]\!]_{\mathsf{mc}}) \\
[\![\mathtt{new\ C}(\overline{\mathtt{e}})]\!]_{\mathsf{mc}} &:= \mathsf{new}\,\mathtt{C}^*([\![\overline{\mathtt{e}}]\!]_{\mathsf{mc}}) \\
[\![(\mathtt{C})\mathtt{e}]\!]_{\mathsf{mc}} &:= \mathsf{cast}\,\mathtt{C}^*\,[\![\mathtt{e}]\!]_{\mathsf{mc}}
\end{aligned}
$$

Interpretation von FJ

Die Methode $\mathtt{m}$ sei genau in den Klassen $\mathtt{C_1}, \ldots, \mathtt{C_n}$ definiert und $mbody(\mathtt{m}, \mathtt{C_i}) = (\overline{\mathtt{x}}_\mathtt{i}, \mathtt{e_i})$ für alle $i \leq n$. Sei $\overline{\mathtt{x}}_i = \mathtt{x}_1, \ldots, \mathtt{x}_z$, dann definieren wir $g^{\mathsf{mc}}_{\mathsf{e}_i}$, so dass für $\vec{b} = b_1, \ldots, b_z$ gilt:

1. Falls $a{\downarrow}$ und $\vec{b}{\downarrow}$ und falls es j gibt, so dass $\mathsf{p}_0 b_j = \mathsf{ex}$, dann folgt $g^{\mathsf{mc}}_{\mathsf{e}_i}(a, \vec{b}) = b_j$ wobei j die kleinste Zahl mit $\mathsf{p}_0 b_j = \mathsf{ex}$.

2. Sonst gilt $g^{\mathsf{mc}}_{\mathsf{e}_i}(a, \vec{b}) \simeq (\lambda this.\lambda[\![\overline{\mathtt{x}}_i]\!]_{\mathsf{mc}}.[\![\mathsf{e}_i]\!]_{\mathsf{mc}})a\vec{b}.$

$g^{\mathsf{mc}}_{\mathsf{e}_i}$ ist abhängig von mc. Wähle r so dass für jede Methode $\mathtt{m}$ in CT gilt:

$$\mathsf{r}\,\mathsf{mc}(\mathtt{m}^*, x, \vec{y}) \simeq \mathsf{over}_{\mathtt{C}_1^*,\ldots,\mathtt{C}_\mathtt{n}^*}(g^{\mathsf{mc}}_{\mathsf{e}_1}, \ldots, g^{\mathsf{mc}}_{\mathsf{e}_n})(x, \vec{y}).$$

Wir definieren mc als kleinsten Fixpunkt von r, d.h. $\mathsf{mc} := \mathsf{l}\,\mathsf{r}$.

Interpretation von FJ

Die Zahlen 0 und 1 sollen "false" und "true" repräsentieren. Dann ist die Interpretation $[\![\texttt{boolean}]\!]$ des Basic Type `boolean` gegeben durch

$$\{(\texttt{boolean}^*, b) \mid b = 0 \vee b = 1\}.$$

Weiter gilt

$$[\![\texttt{false}]\!] = (\texttt{boolean}^*, 0)$$

$$[\![\texttt{true}]\!] = (\texttt{boolean}^*, 1).$$

Interpretation von FJ

Klassen werden als Fixpunkttypen modelliert. Falls CT eine Klasse $\mathtt{C}$ enthält mit $\mathtt{C}^* = i$, dann enthält die Spezifikation $(X)_i = Y_{i1} \times \cdots \times Y_{im_\mathtt{C}}$, wobei Y_{ij} ist wie folgt:

1. Falls es Basic Type $\mathtt{D}$ und Feld $\mathtt{f}$ gibt, so dass $\mathtt{D}\ \mathtt{f}$ in $\mathit{fields}(\mathtt{C})$ auftritt, dann ist $Y_{i\mathtt{f}^*}$ gleich der Interpretation von $\mathtt{D}$.

2. Falls es Klasse $\mathtt{D}$ und Feld $\mathtt{f}$ gibt, so dass $\mathtt{D}\ \mathtt{f}$ in $\mathit{fields}(\mathtt{C})$ auftritt und falls $\mathtt{E}_1, \ldots, \mathtt{E}_n$ ist die Liste aller Klassen $\mathtt{E}_j$ für die $\mathtt{E}_j <: \mathtt{D}$ herleitbar ist, dann

$$Y_{i\mathtt{f}^*} = \{(\mathtt{E}_1^*, x) \mid x \in (X)_{\mathtt{E}_1^*}\} \cup \cdots \cup \{(\mathtt{E}_n^*, x) \mid x \in (X)_{\mathtt{E}_n^*}\} \cup \{(\mathsf{ex}, 0)\}.$$

3. Falls es kein Feld $\mathtt{f}$ in $\mathit{fields}(\mathtt{C})$ gibt, so dass $\mathtt{f}^* = j$, dann ist Y_{ij} gleich V.

Interpretation von FJ

Sei $\mathtt{E}_1, \ldots, \mathtt{E}_n$ ist die Liste aller Klassen $\mathtt{E}_j$ für die $\mathtt{E}_j <: \mathtt{C}$ herleitbar ist. Setze

$$[\![\mathtt{C}]\!] = \{(\mathtt{E}_1^*, x) \mid x \in (\mathsf{FP})_{\mathtt{E}_1^*}\} \cup \cdots \cup \{(\mathtt{E}_n^*, x) \mid x \in (\mathsf{FP})_{\mathtt{E}_n^*}\} \cup \{(\mathsf{ex}, 0)\}.$$

Theorem 7. *Für alle Klassen* $\mathtt{C}$ *und* $\mathtt{D}$ *mit* $\mathtt{C} <: \mathtt{D}$ *gilt* $[\![\mathtt{C}]\!] \subset [\![\mathtt{D}]\!]$.

Theorem 8. *Für alle Klassen* $\mathtt{C}$ *und* $\mathtt{D}$ *mit* $\mathtt{C} <: \mathtt{D}$ *gilt*

$$x \in [\![\mathtt{C}]\!] \wedge \mathsf{p}_0 x \neq \mathsf{ex} \rightarrow (\mathtt{D}^*, \mathsf{p}_1 x) \in [\![\mathtt{D}]\!].$$

Interpretation von FJ

Definition 9. $[\![\overline{\mathtt{e}}]\!]_{\mathsf{mc}} \in [\![\overline{\mathtt{D}}]\!]$ *bedeutet*

$$([\![\mathtt{e}_1]\!]_{\mathsf{mc}}, \ldots, [\![\mathtt{e}_n]\!]_{\mathsf{mc}}) \in [\![\mathtt{D}_1]\!] \times \cdots \times [\![\mathtt{D}_n]\!].$$

Für $\Gamma = \mathtt{x}_1 : \mathtt{D}_1, \ldots, \mathtt{x}_n : \mathtt{D}_n$ *setzen wir*

$$[\![\Gamma]\!]_{\mathsf{mc}} := [\![\mathtt{x}_1]\!]_{\mathsf{mc}} \in [\![\mathtt{D}_1]\!] \wedge \cdots \wedge [\![\mathtt{x}_n]\!]_{\mathsf{mc}} \in [\![\mathtt{D}_n]\!].$$

Definition 10. *Der Typ* $\mathcal{T}$ *ist der Schnitt von*

$$(\{\mathtt{m}^*\} \times [\![\mathtt{C}]\!] \times [\![\overline{\mathtt{D}}]\!]) \curvearrowright [\![\mathtt{B}]\!]$$

für alle Methoden $\mathtt{m}$ *und alle Klassen* $\mathtt{C}$ *in* CT *mit* $mtype(\mathtt{m}, \mathtt{C}) = \overline{\mathtt{D}} \to \mathtt{B}$.

Interpretation von FJ

Lemma 11. *Sei* $h \in \mathcal{T}$. *Falls* $\Gamma \vdash \mathsf{e} \in \mathtt{C}$ *in FJ herleitbar ist, dann gilt*

$$[\![\Gamma]\!]_h \wedge [\![\mathsf{e}]\!]_h\!\downarrow \rightarrow [\![\mathsf{e}]\!]_h \in [\![\mathtt{C}]\!].$$

Lemma 12. *Seien* $g, h \in \mathcal{T}$, *so dass* $g \sqsubseteq_{\mathcal{T}} h$. *Falls* $\Gamma \vdash \mathsf{e} \in \mathtt{C}$ *in FJ herleitbar ist, dann gilt*

$$[\![\Gamma]\!]_g \wedge [\![\mathsf{e}]\!]_g\!\downarrow \rightarrow [\![\mathsf{e}]\!]_g = [\![\mathsf{e}]\!]_h.$$

Lemma 13. *Es gilt* $\mathsf{r} \in (\mathcal{T} \rightarrow \mathcal{T})$.

Lemma 14. *Es gilt* $\mathsf{mc} \in \mathcal{T}$.

Theorem 15. *Falls* $\Gamma \vdash \mathsf{e} \in \mathtt{C}$ *in FJ herleitbar ist, dann gilt*

$$[\![\Gamma]\!] \wedge [\![\mathsf{e}]\!]\!\downarrow \rightarrow [\![\mathsf{e}]\!] \in [\![\mathtt{C}]\!].$$

Interpretation von FJ

Definition 16. *Seien* $\mathtt{a}$ *und* $\mathtt{b}$ *zwei FJ Ausdrücke. Wir definieren* $\longrightarrow'$ *durch Induktion über* $\mathtt{a}$*:* $\mathtt{a} \longrightarrow' \mathtt{b}$ *genau dann wenn* $\mathtt{a} \longrightarrow \mathtt{b}$*, wobei alle Teilausdrücke von* $\mathtt{a}$ *in geschlossener Normalform bezüglich* $\longrightarrow'$ *und* $\mathtt{a}$ *enthählt keine Teilausdrücke der Form* $(\mathtt{D})\mathtt{new}\ \mathtt{C}(\overline{\mathtt{e}})$ *mit* $\mathtt{C} \not<: \mathtt{D}$*.*

Lemma 17. *Sei* $\mathtt{e}$ *ein wohl-getypter Ausdruck in geschlossener Normalform bezüglich* $\longrightarrow'$*. Dann gilt* $[\![\mathtt{e}]\!]{\downarrow}$*. Falls* $\mathtt{e}$ *nicht die Form* $(\mathtt{D})\mathtt{new}\ \mathtt{C}(\overline{\mathtt{e}})$ *hat mit* $\mathtt{C} \not<: \mathtt{D}$ *und auch keine solchen Teilausdrücke enthält, dann gilt* $\mathsf{p}_0[\![\mathtt{e}]\!] \neq \mathsf{ex}$*.*

Lemma 18. *Für alle Ausdrücke* $\mathtt{e}, \overline{\mathtt{d}}$ *und Variablen* $\overline{\mathtt{x}}$ *gilt* $[\![\mathtt{e}]\!][[\![\overline{\mathtt{d}}]\!]/[\![\overline{\mathtt{x}}]\!]] \simeq [\![\mathtt{e}[\overline{\mathtt{d}}/\overline{\mathtt{x}}]]\!]$*.*

Theorem 19. *Seien* $\mathtt{g}, \mathtt{h}$ *zwei Ausdrücke so dass* $\mathtt{g}$ *wohl-getypt ist und* $\mathtt{g} \longrightarrow' \mathtt{h}$ *herleitbar ist, dann gilt* $[\![\mathtt{g}]\!] \simeq [\![\mathtt{h}]\!]$*.*

Zusammenfassung

- Rekursionstheoretische denotationelle Semantik für Featherweight Java
- Prädikatives Objekt Modell
- Stützt Feferman's These
- Beweisbar nicht-terminierende Programme
- Interpretation durch partielle Funktionen
- Korrektheit bewiesen bezüglich Subtypen, Typisierung und Reduktionen

Ausblick

- Dynamische Definition von neuen Klassen
- Kombination von funktionalen und objekt-orientierten Konzepten
- Mixins
- Rekursionstheoretische Semantik von nebenläufigen und verteilten Berechnungen

Literatur

L. Cardelli. A semantics of multiple inheritance. Information and Computation, 76(23):138-164, 1988.

G. Castagna. Object-Oriented Programming: A Unified Foundation. Birkhäuser, 1997.

G. Castagna, G. Ghelli, and G. Longo. A calculus for overloaded functions with subtyping. Information and Computation, 117(1):115-135, 1995.

G. Ghelli. A static type system for late binding overloading. In Proc. of the Sixth International ACM Conference on Object-Oriented Programming Systems and Applications, pages 129-145. Addison-Wesley, 1991.

A. Igarashi and B. Pierce. On inner classes. In Informal Proc. of the Seventh Internatianal Workshop on Foundations of Object-Oriented Languages (FOOL), 2000.

A. Igarashi, B. Pierce, and P. Wadler. Featherweight Java: A minimal core calculus for Java and GJ. In Object Oriented Programming: Systems, Languages and Applications (OOPSLA 99), volume 34 of ACM SIGPLAN Notices, pages 132-146, 1999.

Literatur (2)

S. Feferman. A language and axioms for explicit mathematics. In Algebra and Logic, volume 450 of LNM, pages 87-139. Springer, 1975.

S. Feferman. Constructive theories of functions and classes. In Logic Colloquium 78, pages 159-224. North Holland, 1979.

G. Jäger. Induction in the elementary theory of types and names. In Computer Science Logic 87, volume 329 of LNCS, pages 118-128. Springer, 1988.

R. Kahle and T. Studer. Formalizing non-termination of recursive programs. Journal of Logic and Algebraic Programming. 49:1–14, 2001.

T. Studer. A semantics for $\lambda^{\{\}}_{\mathsf{str}}$: a calculus with overloading and late-binding. Journal of Logic and Computation. 11:527–544, 2001.

T. Studer. Constructive foundations for Featherweight Java. In Proc. of the International Seminar on Proof Theory in Computer Science, pages 202–238. Springer, 2001.

T. Studer. Explicit mathematics: power types and overloading. Annals of Pure and Applied Logic. 134:284–302, 2005.