A Mechanized Framework for Aspects in Isabelle/HOL

Florian Kammüller and Henry Sudhof

Institut für Softwaretechnik und Theoretische Informatik

WMM, 4 October 2007
Motivation and Background

⇒ Verification of object-oriented paradigms
  - Aspect-oriented programming (AOP)
  - Distributed objects (ASP)
  - Isabelle/HOL
    - Isabelle: generic interactive theorem prover
    - Embedding: types, constants, and definitions constitute object logic (theory)
    - Isabelle/HOL: instance for classical HOL
    - Many applications for programming language semantics, e.g. Java,
    - also specification languages: CSP, TLA, Object-Z, …
Overview

1. Aspect-Oriented Programming
2. The Theory of Objects in Isabelle/HOL
3. A Theory of Aspects
4. Discussion
Aspect-Oriented Programming (AOP)

- Idea: Weave Advice into OO-Program
- Advice = code fragments
- Pointcuts: points at which advice is woven in
- Aspect = Advice + Pointcut-definition
- Weave produces combination
- AOP-language: base language (programs and advice)+ Pointcut definition language
AOP-Constructs

• Pointcut selection
  • call: syntactic selection of method calls
e.g., all methods whose name contains “set”
  • cflow: selection of control flow points
e.g., from entry to exit of method x

• Advice insertion
  • before, after
  • around: instead of selected command,
  • or around with proceed: before/after original command

⇒ Change of semantics
⇒ Endangers properties of programs
Foundations of AOP

- Formalization of AOP in Isabelle/HOL
- Idea: simple, functional calculus
- Represent pointcuts by *labels*, e.g.

\[ \langle L, \lambda x. e \rangle \downarrow v_1 + l_1(v_2) \xrightarrow{l_1 \in L} v_1 + e[v_2/x] \]

with pointcuts \( L \), advice \( \lambda x. e \), and weaving operator \( \downarrow \)

- Based on object calculus (Theory of Objects \( \varsigma \))
- Advanced features: type preserving compilation
Theory of Objects: $\varsigma$-calculus

- Terms in the $\varsigma$-calculus: “labelled lists” of methods/fields
  - Objects: $[l_1 = \varsigma(x_0)b_0, \ldots, l_n = \varsigma(x_n)b_n]$ where $x_j$ “self”-parameter
  - Method call/ field selection: $a.l_j$ where $j \in 1..n$
  - Update of method/field: $a.l_j := \varsigma(x)b$ where $j \in 1..n$
- Semantics: reduction relation $\rightarrow_\beta$
- Substitution of formal parameter with a it”self”

$$a \equiv [l_j = \varsigma(x_j)b_j]^{j \in 1..n}$$

$$a.l_j \rightarrow_\beta b_j[a/x_j] \quad j \in 1..n$$
First step: $\xi$-calculus in Isabelle/HOL

- Formalization of finite maps $L \rightarrow T$
- Simple datatype for (de Bruijn) object terms

```haskell
datatype term =
    Var nat
  | Obj Label \rightarrow term
  | Call term Label
  | Upd term Label term
```

- Definition of substitution on de Bruijn terms $t [s / k]$
- Reduction relation $\rightarrow_\beta$

```
inductive beta 
intros

beta: l \in dom f \Rightarrow Call (Obj f) l \rightarrow_\beta the(f l)[(Obj f)/0]
upd : l \in dom f \Rightarrow Upd (Obj f) l a \rightarrow_\beta Obj (f (l\mapsto a))

sel : s \rightarrow_\beta t \Rightarrow Call s l \rightarrow_\beta Call t l
updL: s \rightarrow_\beta t \Rightarrow Upd s l u \rightarrow_\beta Upd t l u
updR: s \rightarrow_\beta t \Rightarrow Upd u l s \rightarrow_\beta Upd u l t

obj : [s \rightarrow_\beta t; l \in dom f] \Rightarrow Obj(f(l\mapsto s)) \rightarrow_\beta Obj(f(l\mapsto t))
```
Confluence and Type Safety for $\varsigma$-calculus

- Confluence (diamond property)

\[
\begin{array}{c}
M \\
\downarrow \\
N_0 & \rightarrow & N_1 \\
\downarrow \\
\rightarrow & L & \leftarrow \\
\end{array}
\]

- If a term $M$ can be reduced in $n \geq 0$ reduction steps to terms $N_0$ and $N_1$, then there exists $L$ such that $N_0$ and $N_1$ can be reduced to $L$.

- We define simple type system for $\varsigma$-calculus,

\[ E \vdash t : T \]

i.e., *term $t$ has type $T$ in type environment $E$*

- We prove type safety for first-order type system of $\varsigma$

**Theorem (preservation)**

\[ [\; t \rightarrow^* \beta t'; E \vdash t : T \;] \implies E \vdash t' : T \]

**Theorem (progress)**

\[ [\; [] \vdash t : A; \exists c . t = \text{Obj } c \;] \implies \exists b . t \rightarrow \beta b \]
• Extend terms \(t\) by (aspect-)labelled terms, e.g. \(l\langle t \rangle\)

\[
\text{datatype term} = \text{Var} \text{ nat} \\
| \text{Obj} \ \text{label} \rightarrow \text{term} \\
| \text{Call} \ \text{term} \ \text{label} \\
| \text{Upd} \ \text{term} \ \text{label} \ \text{term} \\
| \text{Asp Label term} ("_ \langle \_ \rangle")
\]

• Aspect = \langle pointcut (set of Labels), advice (term function) \rangle

\[
\text{datatype aspect} = \text{Aspect} \ (\text{Label set}) \ \text{term} \ ("\langle \_, \_ \rangle")
\]
Weaving

- Idea of weaving: replace existing labels in program with advice
  
  \[ \text{weave} :: [ \text{term}, \text{aspect} ] \Rightarrow \text{term} ("\downarrow") \]

- For example, central rule now:
  
  \[ l\langle t\rangle \downarrow a = \text{if } l \in \text{pct } a \text{ then } l\langle \text{adv } a \ [t/0]\rangle \text{ else } l\langle t\rangle \]

where \( \text{pct } \langle L, a\rangle = L \) and \( \text{adv } \langle L, a\rangle = a \)
Typing of Aspects

- Problem: AOP **not** type safe in general
- Example: around advice exchanges return value [Kammüller, Vösgen: FOAL06]
- Type system to exclude pathological cases:
  - Extend previous type relation by labels \( L \)
    \[ E, L \vdash t : T \]
    i.e., *term* \( t \) *has type* \( T \) *in type/label environment* \( E, L \)
  - Idea: label types represent “legal” advice
- Define *well-formedness* of program \( t \) wrt set of aspects \( A \) \((wf \ t \ A)\)
- Goal: prove that weaving preserves type relation.

Theorem

\[ [wf \ t \ A; [], L \vdash t : T ] \rightarrow [] , L \vdash \text{Weave } t \ A : T \]
Summary

- The $\varsigma$-calculus as a Basis for AOP (and ASP) in Isabelle/HOL


- Labels representing pointcuts in programs
- Definition of weaving function
- Typing of advice and labels:
  $\Rightarrow$ type safety for aspects in Isabelle/HOL
Discussion

- Nominal Techniques vs HOAS vs de Bruijn
- Code extraction
- Structural vs Nominal Type Systems
- Is $\varsigma$-calculus unrealistic (type preserving compilations)?