Java$_{\ell ight}$ is Type-Safe — Definitely

David von Oheimb[1] and Tobias Nipkow
Institut für Informatik
TU München

[1]Research supported by DFG SPP *Deduktion*.

Aims

Observation:

Checking programming language designs is error-prone.

General thesis:

Theorem proving is mature enough to provide help.

Specific instance:

BALI**: Formalizing and analysing Java in Isabelle.**

First step:

Java$_{\ell ight} \subseteq$ **Java is type-safe.**

Related work

"Well-typed expressions do not go wrong."

- Robin Milner. *A Theory of Type Polymorphism in Programming*. 1978.
- . . . , Kim Bruce et al., . . .
- Sophia Drossopoulou and Susan Eisenbach. *Java is Type Safe — Probably*. ECOOP 1997.
- Donald Syme. *Proving Java Type Soundness*. Tech Report, Cambridge Computer Lab, June 1997.
- Zhenyu Qian. *A Formal Specification of Java Virtual Machine Instructions*. Unpublished, Dept. of Comp. Sci., University of Bremen, 1997.

Overview

1. Java$_{\ell ight}$

2. Isabelle/HOL

3. Formalization

   (a) Abstract syntax
   (b) Type system
   (c) Well-formedness
   (d) Operational semantics

4. Proof of type-safety

5. Experience

6. Conclusions

Java$_{\ell ight}$ $\subseteq$ Java ™

Features:

- classes and interfaces with
  fields and methods,
- subinterface, subclass and implementation,
- inheritance, overriding and hiding,
- some primitive types, arrays,
- method calls with
  static overloading and dynamic binding,
- exception throwing and handling.

Missing yet:

- modifiers
- packages
- multi-threading

Isabelle/HOL: *A theory for abstract tables*

datatype $(\alpha)\ option\ \ =\ \ \mathsf{None} \mid \mathsf{Some}\ \alpha$
types $(\alpha,\beta)\ table\ \ =\ \ \alpha \Rightarrow (\beta)\ option$

consts $\mathsf{empty}\ :: (\alpha,\beta)\ table$
$\_[\_\mapsto\_] :: (\alpha,\beta)\ table \Rightarrow \alpha \Rightarrow \beta \Rightarrow (\alpha,\beta)\ table$
$\mathsf{table\_of} :: (\alpha\times\beta)\ list \Rightarrow (\alpha,\beta)\ table$

defs $\mathsf{empty} \stackrel{\mathsf{def}}{=} \lambda k.\ \mathsf{None}$
$t[x\mapsto y] \stackrel{\mathsf{def}}{=} \lambda k.\ \mathsf{if}\ k = x\ \mathsf{then\ Some}\ y\ \mathsf{else}\ t\ k$

primrec $\mathsf{table\_of}\ [] \quad = \quad \mathsf{empty}$
$\mathsf{table\_of}\ ((k,x)\#t) = (\mathsf{table\_of}\ t)[k\mapsto x]$

Abstract Syntax

Declarations:

$$
\begin{aligned}
prog &= (idecl)list \times (cdecl)list \\
cdecl &= tname \times class \\
class &= (tname)option \times (tname)option \\
&\quad \times (fdecl)list \times (mdecl)list
\end{aligned}
$$

Terms:

$$
\begin{aligned}
expr = {} & \mathsf{Lit}\ val \\
& \mid \mathtt{new}\ tname \\
& \mid (ty)expr \\
& \mid expr\{ref\_ty\}\mathtt{.}ename \\
& \mid expr\{ref\_ty\}\mathtt{.}ename\mathtt{:=}expr \\
& \mid expr\mathtt{.}mname\{ty\}(expr) \\
& \mid \ldots
\end{aligned}
$$

$$
\begin{aligned}
stmt = {} & stmt;\ stmt \\
& \mid \mathtt{while}(expr)\ stmt \\
& \mid \mathtt{try}\ stmt\ \mathtt{catch}(tname\ ename)\ stmt \\
& \mid \ldots
\end{aligned}
$$

Type System

Types:

$$
\begin{aligned}
ty &= \textsf{PrimT}\ \mathit{prim\_ty} \\
&\ \mid\ \textsf{RefT}\ \ \mathit{ref\_ty}
\end{aligned}
$$

$$
\begin{aligned}
\mathit{ref\_ty} = &\ \textsf{NullT} \\
&\mid \textsf{IfaceT}\ \mathit{tname} \\
&\mid \textsf{ClassT}\ \mathit{tname} \\
&\mid \textsf{ArrayT}\ ty
\end{aligned}
$$

Type Relations: $prog \vdash tname \prec_{\mathsf{c}} tname$, $prog \vdash ty \preceq ty$, . . .

$$
\frac{\Gamma \vdash C \prec^1_{\mathsf{c}} E}{\Gamma \vdash C \prec_{\mathsf{c}} E}, \quad \frac{\Gamma \vdash C \prec_{\mathsf{c}} D;\ \ \Gamma \vdash D \prec_{\mathsf{c}} E}{\Gamma \vdash C \prec_{\mathsf{c}} E}
$$

$$
\frac{\textsf{is\_type}\ \Gamma\ \ T}{\Gamma \vdash T \preceq T}, \quad \frac{\Gamma \vdash C \prec_{\mathsf{c}} D}{\Gamma \vdash \textsf{Class}\ \ C\ \preceq\ \textsf{Class}\ \ D}, \quad \ldots
$$

where Class tn $==$ RefT (ClassT tn))

Rule format:

$env = prog \times (ename,\ ty)\ table$

$env \vdash stmt :: \Diamond$

$env \vdash expr :: ty$

Typing Rules:

$$\frac{E \vdash e :: \mathsf{PrimT}\ \texttt{boolean};\quad E \vdash c :: \Diamond}{E \vdash \texttt{while}(e)\ c :: \Diamond}$$

$$\frac{\begin{array}{c} E = (\Gamma,\Lambda);\quad E \vdash e :: \mathsf{RefT}\ R;\quad E \vdash p :: T; \\ \mathsf{max\_spec}\ \Gamma\ R\ (mn, T) = \{((md, pn, rT), T')\} \end{array}}{E \vdash e.mn\{T'\}(p) :: rT}$$

Well-Formedness

$\mathsf{wf\_mdecl}\ \Gamma\ C\ (sig, mh, lvars, blk, res) \stackrel{\mathsf{def}}{=}$
$\quad \mathsf{wf\_mhead}\ \Gamma\ (sig, mh) \wedge \mathsf{unique}\ lvars \wedge$
$\quad \forall (vn, T) \in \mathsf{set}\ lvars.\ \mathsf{is\_type}\ \Gamma\ T \wedge \ldots$

Operational Semantics

State & Object Model:

$$state = (xcpt)option \times (heap \times locals)$$

$$\begin{aligned} heap &= (loc,\ obj)table \\ obj &= \mathsf{Obj}\ tname\ fields \\ &\mid\ \mathsf{Arr}\ ty\ components \\ fields &= (ename \times ref\_ty,\ val)\ table \end{aligned}$$

Rule format:

$$prog \vdash state\ {-stmt}\quad\rightarrow\ state$$
$$prog \vdash state\ {-expr} \triangleright val \rightarrow\ state$$

$$\frac{}{\Gamma \vdash (\mathsf{Some}\ xc,\sigma)\ -c\rightarrow\ (\mathsf{Some}\ xc,\sigma)}$$

$$\frac{\begin{array}{c}\Gamma \vdash (\mathsf{None},\sigma_0)\ -e \triangleright a' \rightarrow\ (x_1,\sigma_1); \\ fields{=}\mathsf{snd}(\mathsf{the\_Obj}(\mathsf{heap}\ \sigma_1\ (\mathsf{the\_Addr}\ a'))); \\ v = \mathsf{the}\ (fields\ (fn, T))\end{array}}{\Gamma \vdash (\mathsf{None},\sigma_0)\ -e\{T\}.fn \triangleright v \rightarrow\ (\mathsf{np}\ a'\ x_1,\sigma_1)}$$

Proofs on BALI

Lemmas: Well-foundedness, . . .

$$\text{wf\_prog}\ \Gamma \Longrightarrow \text{wf}\ ((\Gamma \vdash \_ \prec_{\mathsf{i}} \_)^{-1}) \wedge \text{wf}\ ((\Gamma \vdash \_ \prec_{\mathsf{c}} \_)^{-1})$$

Notions: a value conforming to its type

$$\Gamma,h \vdash v{::}\preceq T \stackrel{\text{def}}{=} \exists T'.\ \text{typeof}\ (\ldots\ h)\ v = \text{Some}\ T' \wedge \Gamma \vdash T' \preceq T$$

Type Soundness Theorem:
here for statements

$$E = (\Gamma,\Lambda)\ \wedge\ \text{wf\_prog}\ \Gamma\ \wedge\ E \vdash s{::}\Diamond\ \wedge\ (x\ ,\sigma\ ){::}\preceq E\ \wedge\ \Gamma \vdash (x,\sigma)\ {-s\rightarrow}\ (x',\sigma')\ \Longrightarrow\ (x',\sigma'){::}\preceq E$$

Corollary:
"method not understood" errors
cannot occur at run-time.

Experience

Faithfulness

- Isabelle/HOL is sufficiently expressive.
- Validation is an issue.

Succinctness

- Restrictions and simplicity are vital.
- More proof automation would help.

Maintainability

- Structuring is essential.
- Automatic proof adaption would help.

Statistics

Space:

- Specifications: 1200 lines, incl. comments
- Proofs : 2600 tactic calls

Time:

- Initial version: 12 weeks
- First revision : 3 weeks

Conclusions

Results

- Java$_{\ell ight}$ is definitely type-safe.
- Some restrictions of Java can be relaxed.
- Foundation for further work is laid.

Caveats

- Bytecode level not considered.
- Java$_{\ell ight}$ is not yet Java.
- Not all languages and
  language properties are equal.

Moral

Machine-checked designs ...

- are meanwhile more than feasible
- still require dedication
- are invaluable for change management.