Formal Methods for Java

Lecture 21: Proofs in Jahob

Jochen Hoenicke

Software Engineering
Albert-Ludwigs-University Freiburg

Jan 18, 2012

Static Checking vs. Theorem Proving

Goal:

- finds bugs at compile-time,
- proves that there is no violation.

Static Checking:

- e. g. Jahob and ESC/Java
- fully automatic (after annotation)
- can only verify simple properties

Theorem Proving:

- e. g. KeY Prover
- Needs lot of manual interaction
- complete calculus, can verify any property.

The Jahob Proof Language

Goals

- Improve the strength of the provable properties.
- Still fully automatic (after annotation).
- Have intermediate proof steps in annotation.

Paper:

- Karen Zee, Viktor Kuncak, and Martin Rinard. An integrated proof language for imperative programs. In ACM Conf. Programming Language Design and Implementation (PLDI), 2009.

Note command

We already know one command

$$\text{note } \ell : F$$

which abbreviates

$$\text{assert } \ell : F; \text{assume } \ell : F$$

- ℓ is a label (or name) for the formula F
- When F cannot be proven Jahob tells that the check for ℓ failed.
- ℓ can also be used to tell the Jahob which formulas are relevant:

$$\text{assert } G \text{ from } \ell$$

This rule is correct, i. e., $wp(\text{note } F, H) \rightarrow H$:

$$\begin{aligned} wp(\text{note } F, H) &\leftrightarrow F \wedge (F \rightarrow H) \\ &\leftrightarrow F \wedge H \\ &\rightarrow H \end{aligned}$$

Proving implications

To prove an implication $F \to G$, the syntax is

$$\begin{array}{l} \text{assuming } F \\ \vdots \\ \text{note } G \end{array}$$

This is an abbreviation for

$$\begin{array}{ll} (& \text{assume } F \\ & \vdots \\ & \text{assert } G \\ & \text{assume false} \\ \square & \\ & \text{assume} F \to G \\ ) & \end{array}$$

- $\vdots$ stands for arbitrary proof statements

Correctness of assuming statement

The implication rule is correct, provided the proof statements used in between are correct.

$$
\begin{aligned}
& wp((\text{assume } F; p; \text{assert } G; \text{assume false} \,\Box\, \text{assume } F \rightarrow G, H) \\
& \quad \equiv (F \rightarrow wp(p, G)) \wedge ((F \rightarrow G) \rightarrow H) \\
& \quad \rightarrow [\text{assuming that proof statments } p \text{ are correct}] \\
& \qquad (F \rightarrow G) \wedge ((F \rightarrow G) \rightarrow H) \\
& \quad \rightarrow H
\end{aligned}
$$

Case Splits

One can split cases, e. g.

$$\text{cases } x \geq 0, x < 0 \text{ for } abs(x) \geq 0$$

$$\text{cases } F_1, \ldots, F_n \text{ for } G$$

is an abbreviation for

$$\text{assert } F_1 \vee \cdots \vee F_n;$$
$$\text{assert } F_1 \rightarrow G; \ldots$$
$$\text{assert } F_n \rightarrow G;$$
$$\text{assume } G$$

- Proof that $F_1, \ldots, F_n$ are all possible cases.
- Proof for each case G separately.
- Assume G holds.

Proving Universal Quantifiers

To prove a universal quantified formula the syntax is

pickAny x
$\vdots$
note F

This is an abbreviation for

$$
\begin{array}{ll}
(& \text{havoc } x \\
 & \vdots \\
 & \text{assert } F[x] \\
 & \text{assume false} \\
\Box & \\
 & \text{assume } \forall x. F[x] \\
) &
\end{array}
$$

Removing Universal Quantifiers

The inverse operation removes universal quantifiers:

instantiate $\forall x.F[x]$ with t

This is an abbreviation for

assert $\forall x.F[x]$
assume $F[t]$

Proving Existential Quantifiers

To prove an existential quantified formula the syntax is

$$\text{witness } t \text{ for } \exists x. F[x]$$

This is an abbreviation for

$$\begin{array}{l}\text{assert } F[t]\\ \text{assume } \exists x. F[x]\end{array}$$

Removing Existential Quantifiers

The syntax is

pickWitness x for $F[x]$
$\vdots$ where x does not occur in G
note G

This is an abbreviation for

(assert $\exists x.F[x]$
havoc x
assume $F[x]$
$\vdots$
assert G
assume false
$\square$
assume G
)