8. Program Verification  8.3 Formalization and soundness of the Hoare logic

Section 8.3

Formalization and soundness of the Hoare logic

Formalization in Isabelle/HOL

Issues to solve:

1. How are assertions represented and how is syntactical substitution handled, e.g., the assignment axiom?
2. How to formalize axioms, rules, and derivations?
3. How to express (and prove) soundness?

Related Isabelle/HOL theory:

» HoareIMP.thy

Approaches

1. Deep embedding:
   - model assertions by a datatype, say assndeep (similar to IMP)
   - model Hoare triples as triples of type assndeep × com × assndeep
   - define validity for assndeep × com × assndeep

2. Shallow embedding:
   - express assertions by Isabelle/HOL formulas such that they have type:
     type_synonym assn = state ⇒ bool
   - model Hoare triples as triples of type assn × com × assn
   - define validity for assn × com × assn

Type and validity:

- Type of Hoare triples is:
  assn × com × assn
  where assn = state ⇒ bool and state = var ⇒ int

- Validity is defined as a ternary predicate (with mixfix syntax):
  definition hoare_valid :: assn ⇒ com ⇒ assn ⇒ bool
  ("|= {{P}} c {{Q}}") 50
  where
  |= (P) c (Q) ≡ ∀ s t. s -c→ t → P s → Q t
Deep vs. shallow embedding

Discussion
- Advantages of deep embedding:
  - Faithfully reflects logic as syntactical calculus
  - Assignment axiom can be realized by substitution
  - Simplifies to prove meta-logical properties (e.g., soundness and completeness)
- Advantages of shallow embedding:
  - Less work
  - Base logic already available
  - Full support of Isabelle/HOL for assertions

Soundness

Definition (Soundness/Korrektheit)
A logical calculus/proof system is sound (German: korrekt) if all derivable formulas are valid.

Proof technique:
Use induction over the (height of) the derivation tree:
- Show that all instances of the axiom schemas are valid
- Assuming that the instances of the premisses in a rule application are valid, show that the instance of the conclusion is valid.

Formalizing Hoare axioms and rules

We demonstrate 2 approaches (see HoareIMP.thy):
1. Hoare axioms and rules as lemmas stating their soundness
2. An inductive definition of what it means to derive a Hoare triple

Remarks:
- The assignment axiom is realized by function update instead of substitution.
- Transformation of boolean expressions is done by the semantic function beval.
- In both approaches, rule application is managed by Isabelle/HOL.
- The second approach is the preferred technique; the first approach is shown for discussion.
Introduction

Using the Hoare logic in its classical form is tedious: ~→ Hoare logic in a form supporting weakest precondition reasoning

Overview

- Hoare logic in wp-form (see HoareIMPwp.thy)
- Automated wp-technique in Isabelle/HOL for IMP (see HoareIMPwp.thy)
- Extension to IMP by arrays (see HoareIMParry.thy)
- Application in a case study (see HoareIMParray.thy)

Discussion

- If preconditions are considered as sets of states, the weakest precondition is unique (Why?).
- For more complex programming languages and Hoare logics, the assertion language might be not sufficiently expressive to specify the weakest precondition.
- WP-transformation provides a proof strategy.
- WP-transformation reduces program verification to reasoning on assertions:

\[
\{ P \} C \{ Q \} \iff (P \rightarrow wp(C, Q))
\]

WP-transformation for IMP

\[
wp(SKIP, Q) = Q
\]
\[
wp(x:=E, Q) = Q[E/x]
\]
\[
wp(C1;C2, Q) = wp(C1, wp(C2, Q))
\]
\[
wp(IF \ b \ THEN \ C1 \ ELSE \ C2 \ END, Q) =
(T(b) \rightarrow wp(C1, Q)) \land (\neg T(b) \rightarrow wp(C2, Q))
\]

Weakest preconditions for while statements can – in general – not be computed.
WP reasoning for loops

To handle programs with while statements, we assume that they are annotated with appropriate invariants $P$:

\[
\text{WHILE } b \text{ INV } P \text{ DO } C \text{ END}
\]

Let $Q$ be the computed postcondition of the while statement, then:

- Use $P$ as precondition for the while statement
- Proof the following two implications:
  1. $P$ is an invariant: $T(b) \land P \rightarrow \text{wp}(C, P)$
  2. $P$ is sufficiently strong: $\neg T(b) \land P \rightarrow Q$

Hoare logic in “WP-form”

Axioms and rules are \textit{WP-form} if they allow to compute the precondition from the postcondition.

Skip and assignment axiom are already in WP-form; the new rules are:

\[
\begin{align*}
\{ Q \} C_2 \{ R \} & \quad \{ P \} C_1 \{ Q \} \\
\{ P \} C_1; C_2 \{ R \} & \\
\end{align*}
\]

\[
\begin{align*}
\{ R \} C \{ P \} & \quad T(b) \land P \rightarrow R \\
\neg T(b) \land P & \rightarrow Q \\
\{ P \} \text{ WHILE } b \text{ INV } P \text{ DO } C \text{ END } \{ Q \} \\
\end{align*}
\]

A strengthening rule similar to the consequence rule is needed to prove that the precondition of a program implies the computed precondition.

Discussion

Remarks:

- In Isabelle/HOL, the described proof strategy can be implemented as a method (see HoareIMPwp.thy)
- Note: Here, we explicitly use schema variables in proof goals.
- Implementing the corresponding WP-transformer directly would be difficult because of the shallow embedding of the assertions.

Example:

» HoareIMPwp.thy, see lecture