Chapter 8

Program Verification

Overview of Chapter

8. Program Verification
8.1 Introduction
8.2 A Hoare logic for IMP
8.3 Formalization and soundness of the Hoare logic
8.4 Program verification with Isabelle/HOL
8.5 Software verification tools

Section 8.1

Introduction

Motivation

What is program verification?

- Show that a program has specified properties
- Style of specification depends on programming paradigm and language

$\leadsto$ We consider imperative programs with properties formulated over pre- and poststates

Why program verification?

- Verification of algorithms and their implementation
- Proof that specific errors cannot happen

Learning objectives

- General concept of program verification
- Hoare logic
- Verification of simple sequential imperative programs
- Relationship of Hoare logic and semantics
- Formalization of Hoare logic and soundness proof
- Other approaches and tools for program verification

Literature

Books on program verification

- Krzysztof R. Apt, Frank S. de Boer, E.-R. Olderog:
  Verification of Sequential and Concurrent Programs (3. Auflage)
- Roland C. Backhouse:
  Program Construction and Verification
- Edsger Dijkstra:
  The Discipline of Programming
- David Gries:
  The Science of Computer Programming

Example: Program

Splitting step in Quicksort:

```
int split( int[] arr,
           int lwb, int upb){
  int  left, pivot, right;
  int  result, tmp;
  left = lwb;
  pivot = arr[upb];
  right = upb-1;
  while( left <= right ) {
    while( arr[left]<pivot ){
      left = left+1;
    }
    while( left <= right &&
       pivot <= arr[right] ){
      right = right-1;
    }
    if( left <= right ) {
      tmp = arr[left];
      arr[left] =
            arr[right];
      arr[right] = tmp;
      left = left+1;
      right = right-1;
    } else { ; }
  }
  tmp = arr[left];
  arr[left] = arr[upb];
  arr[upb] = tmp;
  result = left;
  return result;
}
```


Example: Specification

Precondition:

$0 \leq lwb \wedge lwb < upb \wedge upb < arr.length$

Postcondition:

Splitting array into elements below and above pivot:

$lwb \leq result \wedge result \leq upb \wedge$
$(\forall\, i.\ lwb \leq i \wedge i < result \longrightarrow arr[i] \leq arr[result]) \wedge$
$(\forall\, i.\ result < i \wedge i \leq upb \longrightarrow arr[i] \geq arr[result])$

Not treated:

Array contains same elements in poststate as in prestate

Section 8.2

A Hoare logic for IMP

Syntax of Hoare logic

Hoare triple:

Formulas in Hoare logic are triples of the form `{ P } C { Q }` where

- C is a command/statement of the programming language
- P and Q are first-order formula, so-called *assertions*, such that
  - program variables can appear in P and Q (no quantification over program variables)
  - boolean expressions can be *translated* to equivalent formulas

Example:

Let `C2` be some command:

```
{ x = 7 ∧ y ≤ 3 ∧ p(x) ∧ z = Z }
   IF x == 7 & y <= 5 THEN z := z + 1 ELSE C2
{ p(x) ∧ z = Z+1 }
```


Discussion: Syntax

1. Hoare logics vary w.r.t.:
   - the programming language
   - the relationship between boolean expressions and assertions
   - the treatment of variables:
     - Are program and logical variables be syntactically distinguished?
     - Is quantification over program variables allowed?
   - the datatypes and functions available for writing assertions
2. In addition to rules for reasoning about Hoare triples, Hoare logic needs a base logic to reason about assertions, e.g. FOL. That is, strictly speaking, FOL formulas are part of Hoare logic.

Semantics of Hoare Logic

Definition

Let $s \xrightarrow{C} t$ denote the judgment of the big-step semantics and let

$$P(s) \equiv_{def} P[s(v_1)/v_1, \ldots, s(v_n)/v_n]$$

where $v_1, \ldots, v_n$ are the program variables in P.
The Hoare triple `{ P } C { Q }` is valid iff

$$P(s) \wedge (s \xrightarrow{C} t) \longrightarrow Q(t)$$

is valid.

Discussion:

Often, the semantics of an assertion A is considered to be the set of states satisfying A (assuming no free logical variable).

Rules of Hoare Logic

$\{ P \}$ `SKIP` $\{ P \}$ — skip axiom

$\{ P[E/x] \}$ `x := E` $\{ P \}$ — assignment axiom

$$\frac{\{ P \}\ \texttt{C1}\ \{ Q \} \quad \{ Q \}\ \texttt{C2}\ \{ R \}}{\{ P \}\ \texttt{C1;C2}\ \{ R \}}$$ composition rule

$$\frac{\{ \mathcal{T}(b) \wedge P \}\ \texttt{C1}\ \{ Q \} \quad \{ \neg\mathcal{T}(b) \wedge P \}\ \texttt{C2}\ \{ Q \}}{\{ P \}\ \texttt{IF b THEN C1 ELSE C2 END}\ \{ Q \}}$$ conditional rule

Rules of Hoare Logic (2)

$$\frac{\{ \mathcal{T}(b) \wedge P \}\ \texttt{C}\ \{ P \}}{\{ P \}\ \texttt{WHILE b DO C END}\ \{ \neg\mathcal{T}(b) \wedge P \}}$$ while rule

$$\frac{P \longrightarrow P' \quad \{ P' \}\ \texttt{C}\ \{ Q' \} \quad Q' \longrightarrow Q}{\{ P \}\ \texttt{C}\ \{ Q \}}$$ consequence rule

where

- $P[E/x]$ denotes the substitution of x in P by E
- $\mathcal{T}(b)$ denotes the translation of `b` to an equivalent formula

Remark:
Note: The axioms and rules are schemas.

Applying Hoare logic: an example

Let $C \equiv$

```
c  := 0;                -- a1
sq := 1;                -- a2
WHILE sq <= x DO
  c  := c+1;            -- a3
  sq := sq + (2*c+1)    -- a4
END
```

Prove: $\{ 0 \leq x \}\ C\ \{ c * c \leq x \wedge x < (c+1) * (c+1) \}$

Partial and total correctness

Partial correctness:
If the precondition P holds in a prestate s and the program terminates in a poststate t, then Q holds in t.

Total correctness:
If the precondition P holds in a prestate s, then the program terminates and Q holds in the poststate.

Remark:

- We only considered partial correctness.
- For total correctness, a measure is needed to prove loop termination.