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

Motivation

What is program verification?
- Show that a program has specified properties
- Style of specification depends on programming paradigm and language
- 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
8. Program Verification
8.1 Introduction

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

Example: Program

Splitting step in Quicksort:

```c
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:

\[ 0 \leq lwb \land lwb < upb \land upb < arr.length \]

Postcondition:

Splitting array into elements below and above pivot:

\[ lwb \leq result \land result \leq upb \land \]
\[ (\forall i. lwb \leq i \land i < result \rightarrow arr[i] \leq arr[result] ) \land \]
\[ (\forall i. result < i \land i \leq upb \rightarrow arr[i] \geq arr[result] ) \]

Not treated:

Array contains same elements in poststate as in prestate

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
8.2 A Hoare logic for IMP

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:
\[
\{ x = 7 \land y \leq 3 \land p(x) \land z = Z \} \\
\text{IF } x == 7 \land y <= 5 \text{ THEN } z := z + 1 \text{ ELSE } C2 \\
\{ p(x) \land z = Z+1 \}
\]

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
\[
P(s) \equiv_{\text{def}} P[s(v_1)/v_1, \ldots, s(v_n)/v_n]
\]
where \( v_1, \ldots, v_n \) are the program variables in \( P \).
The Hoare triple \( \{ P \} C \{ Q \} \) is valid iff
\[
P(s) \land (s \xrightarrow{C} t) \rightarrow Q(t)
\]
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

{ P } C1 { Q } { Q } C2 { R }
composition rule

{ T ( b ) ∧ P } C { P }
while rule

P → P' { P' } C { Q' } Q' → Q
consequence rule

where
• P[ E / x ] denotes the substitution of x in P by E
• 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 ≡

\[
\begin{align*}
c & := 0; & \quad \text{-- a1} \\
\text{sq} & := 1; & \quad \text{-- a2} \\
\text{WHILE} & \; \text{sq} \leq x \; \text{DO} \\
\text{c} & := \text{c} + 1; & \quad \text{-- a3} \\
\text{sq} & := \text{sq} + (2 \cdot \text{c} + 1) & \quad \text{-- a4} \\
\text{END}
\end{align*}
\]

Prove: \{ 0 \leq x \} C \{ c \cdot c \leq x \land x < (c + 1) \cdot (c + 1) \}