Hoare Logic

Costel Anghel and Mădălina Erașcu

Objectives

- Understand and apply the Hoare rules for a simple programming language [1, Chapter 2].
- Write Dafny program illustrating the principles of Hoare logic.

1 Recalling from the Lecture

- The **state** at a specific point in a program refers to the assignment of values to the variables that are currently within scope at that particular point in the program's execution.
- To show the correctness of a program, one can analyze it **forwards** or **backwards**. **Partial correctness** refers to the property of a program that when it terminates, its postcondition is satisfied given that the precondition holds. Partial correctness does not verify whether a program terminates or not.
- **Total correctness** goes a step further than partial correctness by not only ensuring that the program meets the postcondition, but also makes sure that it terminates. We can say that the total correctness implies both partial correctness and the termination of the program.
- A **Hoare Triple** is a set of logical rules that refers rigorously about the correctness of computer programs.

  $$\{\!\{P\}\!\}S\{\!\{Q\}\!\}$$

  P represents the precondition, S stands for the program statements whose behaviour you're specifying, and Q represents the postcondition.
- The **Hoare logic rules** for a simple programming language are:
  - `skip` command

    $$\{\!\{P\}\!\}\ \texttt{skip}\ \{\!\{P\}\!\}$$

  - `abort` command

    $$\{\!\{true\}\!\}\ \texttt{abort}\ \{\!\{false\}\!\}$$

  - Scalar assignment

    $$\{\!\{Q[e/x]\}\!\}\ x := e\ \{\!\{Q\}\!\}$$

  - Array assignment

    $$\{\!\{Q[a[i \rightarrowtail e]/a]\}\!\}\ a[i] := e\ \{\!\{Q\}\!\}$$

– Command Sequences

$$\frac{\{\!\{P\}\!\}c_1\{\!\{R\}\!\} \qquad \{\!\{R\}\!\}c_2\{\!\{Q\}\!\}}{\{\!\{P\}\!\}c_1; c_2\{\!\{Q\}\!\}}$$

– Conditionals

$$\frac{\{\!\{P \wedge b\}\!\}\ c_1\ \{\!\{Q\}\!\} \qquad \{\!\{P \wedge \neg b\}\!\}\ c_2\ \{\!\{Q\}\!\}}{\{\!\{P\}\!\}\ \textbf{if}\ b\ \textbf{then}\ c_1\ \textbf{else}\ c_2\ \{\!\{Q\}\!\}}$$

$$\frac{\{\!\{P \wedge b\}\!\}\ c\ \{\!\{Q\}\!\} \qquad \{\!\{P \wedge \neg b\}\!\} \Longrightarrow \{\!\{Q\}\!\}}{\{\!\{P\}\!\}\ \textbf{if}\ b\ \textbf{then}\ c\ \{\!\{Q\}\!\}}$$

– Loops (partial correctness)

$$\frac{P \Longrightarrow I \qquad \{\!\{I \wedge b\}\!\}\ c\ \{\!\{I\}\!\} \qquad (I \wedge \neg b) \Longrightarrow Q}{\{\!\{P\}\!\}\ \textbf{while}\ b\ \textbf{do}\ c\ \{\!\{Q\}\!\}}$$

Condition $P \Longrightarrow I$ shows that the loop invariant holds initially. The loop body is $\{I \wedge b\}\ c\ \{I\}$ and it assumes that after executing it, the loop invariant and loop condition hold initially. The condition $(I \wedge \neg b) \Longrightarrow Q$ says that if the loop invariant and the nagation of loop condition hold, it implies the postcondition Q at the end of the loop.

– Loops (total correctness)

$$\frac{P \Longrightarrow I \quad I \Longrightarrow t \geq 0 \quad \{\!\{I \wedge b \wedge t = N\}\!\}\ c\ \{\!\{I \wedge t < N\}\!\}\ (I \wedge \neg b) \Longrightarrow Q}{\{\!\{P\}\!\}\ \textbf{while}\ b\ \textbf{do}\ c \quad \{\!\{Q\}\!\}}$$

Additionally to the rules for partial correctness, for total correctness we need to find a termination function/term t, which has to be positive at the entrance in the loop $(I \Longrightarrow t \geq 0)$ and has to decrease at each loop iteration $\{\!\{I \wedge b \wedge t = N\}\!\}\ c\ \{\!\{I \wedge t < N\}\!\}$.

2 Homework

Remark 1 *For Exercises 1 and 2, you can use Dafny to help you understand your explanations. For example:*

```
method test(x: int, y: int) returns (z: int)
{
    assume(x==y);
    z:=x-y;
    assert(z==0);
}
```

A condition introduced with `assume` *is considered to be true and it is not necessary to be proved.*

1. Explain rigorously why each of these triples holds:
   (a) $\{\!\{x == y\}\!\}z := x - y\{\!\{z == 0\}\!\}$
   (b) $\{\!\{true\}\!\}x := 100\{\!\{x == 100\}\!\}$
   (c) $\{\!\{0 <= x < 100\}\!\}x := x + 1\{\!\{0 <= x <= 100\}\!\}$
2. For each of the following triples, find initial values for x and y that demonstrate that the triple does not hold.

(a) $\{\{true\}\}x := 2 * y\{\{y <= x\}\}$

(b) $\{\{0 <= x\}\}x := x - 1\{\{0 <= x\}\}$

3. For each of the following triples, come up with some predicate to replace the question mark to make it a Hoare triple that holds. Make your conditions as precise as possible.

   (a) $\{\{0 <= x < 100\}\}x := 2 * x\{\{?\}\}$

   (b) $\{\{0 <= x < N\}\}x := x + 1\{\{?\}\}$

4. For each of the following triples, come up with some predicate to replace the question mark to make it a Hoare triple that holds. Make your conditions as precise as possible.

   (a) $\{\{?\}\}x := 400\{\{x == 400\}\}$

   (b) $\{\{?\}\}x := 65\{\{y <= x\}\}$

5. Write the program which computes the sum of first n natural numbers.

   - Write this program of the form $\{\{P\}\}S\{\{Q\}\}$ and rewrite P, S, Q until you reach the atomic statements of the Hoare logic rules presented above.
   - Show the total correctness of the program by "trial and error" in Dafny.
   - Based on the rules of the Hoare calculus, identify which verification conditions are generated and proved by Dafny (in the background).

   Remark 2 *To the best of authors knowledge, there is no way to see pretty printed the verification conditions in Dafny. Pretty printed means something which can be understood by students who attended Computational Logic course. However, if you want to experiment with Dafny and Boogie to check the verification conditions, please check the Stackoverflow post* `https://stackoverflow.com/questions/77827067/visualize-the-verification-conditions-in-dafny/`.

6. Write the program which computes the product of first n natural numbers. Prove its total correctness.

References

[1] K. R. M. Leino. *Program Proofs*. MIT Press, 2023.