Programming Language Semantics: Chapter 02 - Hoare Logic

Tarek Boutefara – t_boutefara@univ-jijel.dz – Version 0.1, 12-10-2024

Table of Contents

1. Introduction
2. Hoare logic principales
   2.1. Basic principale
   2.2. Hoare triplet
   2.3. Partial correction
3. Axioms and rules
   3.1. Assignment rule
   3.2. Sequential (composition) rule
   3.3. Implication rules
   3.4. Conditional rules
   3.5. Iteration rule
4. First example
5. Conclusion

1. Introduction

Proving the properties of a program requires the ability to represent and reason about these properties. Hoare logic can be used to describe the properties of a state and its evolution during instruction execution.

Hoare Logic is a formal system for proving the correctness of computer programs. It was developed by British computer scientist Tony Hoare in the late 1960s. The core idea is to use logical assertions to describe the state of a program before and after the execution of a statement.

2. Hoare logic principales

2.1. Basic principale

As with Formal Systems and Constructivist Logic, a proof in Hoare Logic consists of a sequence of instructions on an initial state that verifies certain properties to obtain a final state that verifies other properties.

2.2. Hoare triplet

Precondition

```
This is a proposition concerning the state of the memory, which is
assumed to be verified before the execution of a code.
```

Post-condition

```
This is a statement about the state of the memory supposed to be
checked after the execution of a code.
```

Specification

```
A program is specified by a pre-condition and a post-condition
determining the cases in which the program will be executed and its result.
```

Triplet de Hoare

A Hoare triplet, denoted {P} C {Q}, is given by a precondition, a code fragment and a postcondition.

If P is true in the initial state and C terminates, then Q is true in the final state.

2.3. Partial correction

A formula {P} C {Q} is said to be valid, i.e. a program C is partially correct with respect to a precondition P and a postcondition Q when: If for any initial state verifying P and if execution terminates, then the final state verifies Q.

3. Axioms and rules

To define a complete logic, we need a programming language with a few exceptions. So, the following axioms and rules are for generic principles present in the majority of programming languages.

3.1. Assignment rule

The most fundamental rule of Hoare Logic is the assignment rule:

$\frac{}{\{P\}x := E\{Q\}}$ such as P = Q[E/x]

Per example:

- $(x + 1 > 0)\{x := x + 1\}(x > 0)$

This can be interpreted as follow: if $x + 1 > 0$ is true before the assignment $x := x + 1$, then the condition $x > 0$ is true after the assignment.

ie. : For the condition $x > 0$ to be true after the assignment $x := x + 1$, it is necessary that the condition $x + 1 > 0$ is verified before the assignment.

3.2. Sequential (composition) rule

$\frac{\{E\}P1\{F\},\ \{F\}P2\{S\}}{\{E\}\text{P1, P2}\{S\}}$] (SEQ)

3.3. Implication rules

1. $\frac{E \Rightarrow F \text{ and } \{F\}P\{S\}}{\{E\}P\{S\}}$ (IMP1)
2. $\frac{\{E\}P\{F\} \text{ and } F \Rightarrow S}{\{E\}P\{S\}}$ (IMP2)

3.4. Conditional rules

1. $\frac{\{E \text{ and } B\}P1\{S\}, \{E \text{ and } \neg B\}P2\{S\}}{\{E\}\text{IF B : P1 ELSE P2 ENDIF}\{S\}}$ (COND1)
2. $\frac{\{E \text{ and } B\}P1\{S\}, E \text{ and } \neg B \Rightarrow S}{\{E\}\text{IF B : P1 ENDIF}\{S\}}$ (COND2)

3.5. Iteration rule

$$\frac{\{E \text{ and } B\}P\{E\}}{\{E\}\text{WHILE B : P ENDWHILE}\{E \text{ and } \neg B\}} \text{ (ITE)}$$

4. First example

Consider the following program:

```
x := 5;
y := x + 2;
```

We want to prove that this program correctly computes the value 7 for y.

1. **Specify the preconditions and postconditions:**
   - Precondition: True (no specific requirements before the program starts)
   - Postcondition: y = 7
2. **Apply the rules of inference:**
   - Assignment 1:
     - Precondition: True
     - Postcondition: x = 5
     - Rule: x := 5 {x = 5}
   - Assignment 2:
     - Precondition: x = 5
     - Postcondition: y = 7
     - Rule: y := x + 2 {y = x + 2}
3. **Combine the assertions:**
   - We can combine the postcondition of the first assignment with the precondition of the second assignment: x = 5. This means that the value of x is indeed 5 before the second assignment.
4. **Prove the final postcondition:** *Since y = x + 2 and x = 5, we can substitute x with 5 in the equation: y = 5 + 2. This simplifies to y = 7.

5. Conclusion

This chapter gives a very brief introduction to Hoare logic. This logic is designed to enable programs to be proven. We will then see that this logic forms the basis of the semantics of programming languages, known as Axiomatic Semantics.

Version 0.1
Last updated 2024-10-13 23:38:19 +0100