Hoare Triples and Floyd-Hoare Logic

Snorri Agnarsson

snorri@hi.is

Hoare Triples

- Hoare triples are named after C.A.R. Hoare who was prominent in developing the notation along with the associated Floyd-Hoare logic for proving program correctness
  - See https://en.wikipedia.org/wiki/Hoare_logic
- Hoare triples are commonly written as $\{\boldsymbol{P}\}\boldsymbol{S}\{\boldsymbol{Q}\}$ where $\boldsymbol{P}$ and $\boldsymbol{Q}$ are assertions (state descriptions) and $\boldsymbol{S}$ is a command
- A Hoare triple is either true or false
- The meaning of "$\{\boldsymbol{P}\}\boldsymbol{S}\{\boldsymbol{Q}\}$" is:
  "if $\boldsymbol{P}$ is true and then $\boldsymbol{S}$ is executed, then $\boldsymbol{Q}$ will be true"
- A Hoare triple (true): $\{x = 0\}x \coloneqq x + 1\{x = 1\}$
- A Hoare triple (false): $\{x = 0\}x \coloneqq x + 1\{x = 0\}$

Examples of Hoare Triples

- True Hoare triples
  - {x==y}x:=y+1{x==y+1}
  - {x==0}x++{x>0}
  - {x<=0}x++{x<=1}
  - {x>0}x++{x>1}
- False Hoare triples
  - {x==0}x++{x==0}
  - {x==0}x++{x>1}
  - {x<0}x++{x<0}
  - {x<0}x++{x==0}

- This notation for Hoare triples is not convenient in many programming languages and we often use an alternative notation instead of {P}S{Q}, e.g.

```
// P
S;
// Q
```

Essential Floyd-Hoare Logic Rules

Contravariance

$$\frac{P' \Rightarrow P,\ \{P\}S\{Q\},\ Q \Rightarrow Q'}{\{P'\}S\{Q'\}}$$

Premises

Consequence

Sequence

$$\frac{\{P\}S\{Q\},\ \{Q\}T\{R\}}{\{P\}S;T\{R\}}$$

Conditional

$$\frac{\{C \wedge P\}T\{Q\},\ \{\neg C \wedge P\}S\{Q\}}{\{P\}\text{if } C \text{ then } T \text{ else } S\{Q\}}$$

Loop

$$\frac{\{I \wedge C\}S\{I\}}{\{I\}\text{while } C \text{ do } S\{I \wedge \neg C\}}$$

Essential Floyd-Hoare Logic Rules

Contravariance

$$\frac{P' \Rightarrow P,\ \{P\}S\{Q\},\ Q \Rightarrow Q'}{\{P'\}S\{Q'\}}$$

It is allowed to strengthen the precondition and to weaken the postcondition of a piece of code

Sequence

$$\frac{\{P\}S\{Q\},\ \{Q\}T\{R\}}{\{P\}S;T\{R\}}$$

It is allowed to chain together pieces of code if the precondition of the second piece of code is the postcondition of the other

Conditional

$$\frac{\{C \land P\}T\{Q\},\ \{\neg C \land P\}S\{Q\}}{\{P\}\text{if } C \text{ then } T \text{ else } S\{Q\}}$$

The postcondition of an if-statement can be made equal to the postconditions of the then-part and the else-part

Loop

$$\frac{\{I \land C\}S\{I\}}{\{I\}\text{while } C \text{ do } S\{I \land \neg C\}}$$

A while-loop needs a loop invariant that is true before and after each pass through the loop, including after the last pass through the loop