Hoare Logic

Jari Stenman

February 10, 2012

Outline

Outline

Hoare Logic

- Hoare - An axiomatic basis for computer programming (1969)
- Describes a deductive system for proving program correctness.
- A set of axioms and inference rules about asserted programs.

While Programs

- Assume that we have an underlying logic L, e.g. Integer Arithmetic

Defined inductively:

- for every variable x and term t, $x := t$ is a program
- if S_1 and S_2 are programs, and e is a boolean expression, the following are also programs
  - S_1 ; S_2
  - **if** e **then** S_1 **else** S_2 **fi**
  - **while** e **do** S_1 **od**

States

- We have a set of variables, typically integers
- A program can be seen as a set of states, and a set of transitions between states

In the case of integers $x_1, \ldots, x_n$, the state space of the program is $\mathbb{Z}^n$.

- A predicate on $x_1, \ldots, x_n$ characterizes a set of states, i.e. a subset of $\mathbb{Z}^n$.

Hoare Triples

The formulas of Hoare Logic are asserted programs

- $\{p\}\, S\, \{q\}$

Here, S is a program, and p, q are *assertions*

- p is called the precondition
- q is called the postcondition

The above formula states that whenever p holds before running S, and S terminates, then q will hold *after* running S.

Hoare Triples

Example

$\{x < 1\}\, x := x + 1;\ x = x + 1\, \{x < 3\}$

- How can we prove this?
- Hoare Logic provides axioms and inference rules for proving asserted programs.

Outline

Assignment Axiom Schema

Assignment

$$\overline{\{p[t/x]\}\, x := t\, \{p\}}$$

- $p[t/x]$ stands for substituting t for free occurences of x in p

Example

$\{y + 5 = 42\}\, x := y + 5\, \{x = 42\}$

Example

What is p if the following is an instance?
$\{p\}\, x := x + 1\, \{x < 10\}$

Composition Rule

Composition

$$\frac{\{p\}\, S_1 \,\{r\} \quad \{r\}\, S_2 \,\{q\}}{\{p\}\, S_1 \,;\, S_2 \,\{q\}}$$

Composition Rule

Example

P: $\{true\}\, x := 2\, ;\, y := x\, \{x > 0 \wedge y = 2\}$

We can infer P if we can infer

- $\{true\}\, x := 2\, \{\varphi\}$ and
- $\{\varphi\}\, y := x\, \{x > 0 \wedge y = 2\}$

for some predicate φ.
By Assignment, we can infer

- $\{x > 0 \wedge x = 2\}\, y := x\, \{x > 0 \wedge y = 2\}$ and
- $\{2 > 0 \wedge 2 = 2\}\, x := 2\, \{x > 0 \wedge x = 2\}$

Since $2 > 0 \wedge 2 = 2 \equiv true$, we have proved the asserted program.

Conditional Rule

Conditional

$$\frac{\{p \land e\}\; S_1 \;\{q\} \quad \{p \land \neg e\}\; S_2 \;\{q\}}{\{p\}\; \textbf{if } e \textbf{ then } S_1 \textbf{ else } S_2 \textbf{ fi} \;\{q\}}$$

Conditional Rule

Conditional

$$\frac{\{p \wedge e\}\, S_1\, \{q\} \quad \{p \wedge \neg e\}\, S_2\, \{q\}}{\{p\}\, \textbf{if } e \textbf{ then } S_1 \textbf{ else } S_2 \textbf{ fi}\, \{q\}}$$

Example

$\{true\}\, \textbf{if } x < 10 \textbf{ then } x := 10 \textbf{ else } x := 0 \textbf{ fi}\, \{x = 10 \vee x = 0\}$

We can infer this if we can infer

- $\{true \wedge x < 10\}\, x := 10\, \{x = 10 \vee x = 0\}$
- $\{true \wedge x \geq 10\}\, x := 0\, \{x = 10 \vee x = 0\}$

Iteration Rule

Iteration

$$\frac{\{p \wedge e\}\, S\, \{p\}}{\{p\}\ \mathbf{while}\ e\ \mathbf{do}\ S\ \mathbf{od}\ \{p \wedge \neg e\}}$$

Iteration Rule

Iteration

$$\frac{\{p \wedge e\}\, S\, \{p\}}{\{p\}\, \textbf{while}\ e\ \textbf{do}\ S\ \textbf{od}\, \{p \wedge \neg e\}}$$

Example

$\{x \leq 10\}\, \textbf{while}\ x < 10\ \textbf{do}\ x := x + 1\ \textbf{od}\, \{x = 10\}$

- A: $\{x + 1 \leq 10\}\, x := x + 1\, \{x \leq 10\}$
- L: $\{x \leq 10 \wedge x + 1 \leq 10\}\, x := x + 1\, \{x \leq 10\}$
- L: $\{x + 1 \leq 10 \wedge x \leq 10\}\, x := x + 1\, \{x \leq 10\}$
- I: $\{x \leq 10\}\, \textbf{while}\ x + 1 \leq 10\ \textbf{do}\ x := x + 1\ \textbf{od}\, \{x \leq 10 \wedge x + 1 \not\leq 10\}$
- L: $\{x \leq 10\}\, \textbf{while}\ x < 10\ \textbf{do}\ x := x + 1\ \textbf{od}\, \{x \leq 10 \wedge x \geq 10\}$
- L: $\{x \leq 10\}\, \textbf{while}\ x < 10\ \textbf{do}\ x := x + 1\ \textbf{od}\, \{x = 10\}$

Rule of Consequence

Consequence

$$\frac{p \Rightarrow p' \quad \{p'\}\, S\, \{q'\} \quad q' \Rightarrow q}{\{p\}\, S\, \{q\}}$$

- We can *strengthen* the precondition, i.e. assume more than we need
- We can *weaken* the postcondition, i.e. conclude less than we are allowed to

Rule of Consequence

Consequence

$$\frac{p \Rightarrow p' \quad \{p'\}\, S\, \{q'\} \quad q' \Rightarrow q}{\{p\}\, S\, \{q\}}$$

Example

$\{true \wedge x < 10\}\, x := 10\, \{x = 10 \vee x = 0\}$

We have

- $\{true\}\, x := 10\, \{x = 10 \vee x = 0\}$ by Assignment
- $true \wedge x < 10 \Rightarrow true$
- $x = 10 \vee x = 0 \Rightarrow x = 10 \vee x = 0$

Outline

Proofs in Hoare Logic

- An asserted program is sequence $\{p_0\}\, S_1\, ;\, S_2\, ;\, \ldots\, ;\, S_n\, \{p_n\}$
- each S_i is either an **if**-statement, a **while**-statement or an assignment.

By Composition, the problem of proving this program correct amounts to finding p_i s.t. $\{p_0\}\, S_1\, \{p_1\}$, $\{p_1\}\, S_2\, \{p_2\}$, ... , $\{p_{n-1}\}\, S_n\, \{p_n\}$

Hoare's paper doesn't include any way of computing these intermediate assertions.

Weakest Preconditions

Dijkstra's paper "Guarded Commands, Nondeterminancy and Formal Derivation of Programs" introduces the notion of weakest precondition.

Definition

The weakest precondition of a predicate q wrt. a program S, denoted by $wp(S, q)$ is the *weakest* predicate characterizing all states from which a run of S is guaranteed to terminate in q.

Weakest Preconditions

Start with postcondition and "push" it backwards

- $\{p\}\, S_1 \,;\, S_2 \,;\, S_3 \,\{q\}$
- $\{p\}\, S_1 \,;\, S_2 \,\{wp(S_3, q)\}$
- $\{p\}\, S_1 \,\{wp(S_2, wp(S_3, q))\}$
- $p \Rightarrow wp(S_1, wp(S_2, wp(S_3, q)))$?

A kind of symbolic execution of statements in the domain of predicates.

Conclusions

Hoare's paper founded a whole school of research

A swarm of extensions, for e.g.

- procedure calls
- arrays
- goto
- pointers

Conclusions

Hoare's paper founded a whole school of research

A number of tools work like follows:

1. Use Hoare-style Logic and WP to generate *verification conditions*
2. Use a general-purpose tool to prove these

Verification conditions do not contain program constucts

Conclusions

Thank You!

Next presentation: Joe Scott on CTL, Friday 17th in P1112