Hoare Logic

Deepak D’Souza, K. V. Raghavan

Department of Computer Science and Automation
Indian Institute of Science, Bangalore.

April 2012
Outline

- Hoare triples as assertions of partial correctness.
- Hoare logic rules.
- Weakest Precondition calculus.
A way of asserting properties of programs.

Hoare triple: \( \{A\} P \{B\} \) asserts that “If program \( P \) is started in a state satisfying condition \( A \), if it terminates, it will terminate in a state satisfying condition \( B \).”

A proof system for proving such assertions.

A way of reasoning about such assertions using the notion of “Weakest Preconditions” (due to Dijkstra).
A simple programming language

- skip
- $x := e$ (assignment)
- if $b$ then $S$ else $T$ (if-then-else)
- while $b$ do $S$ (while)
- $S$ ; $T$ (sequencing)
Example program

```plaintext
x := n;
a := 1;
while (x ≥ 1) {
    a := a * x;
    x := x - 1
}
```
View program $P$ as a partial map $[P] : Stores \rightarrow Stores$.

$x \mapsto 2, \ y \mapsto 10, \ z \mapsto 3$

$y = y + 1$;
$z = x + y$

$x \mapsto 2, \ y \mapsto 11, \ z \mapsto 12$
Predicates on States

All States

States satisfying Predicate A
Eg. $x \geq 0 \land x < y$
Assertion of “Partial Correctness” \{A\}P\{B\}

\{A\}P\{B\} asserts that “If program \(P\) is started in a state satisfying condition \(A\), either it will not terminate, or it will terminate in a state satisfying condition \(B\).”

\[
\{10 \leq y\}
\]

\[
y = y + 1; \\
z = x + y
\]

\[
\{x < z\}
\]
Give “weakest” preconditions

1. \(\{?\} x := x + 2 \{x \geq 5\}\)
2. \(\{?\} \text{if (} y < 0 \text{) then } x := x + 1 \text{ else } x := y \{x > 0\}\)
3. \(\{?\} \text{while (} x \leq 5 \text{) do } x := x + 1 \{x = 6\}\)
Proof rules of Hoare Logic

Skip:

{A} skip {A}

Assignment

{A[e/x]} x := e {A}
Proof rules of Hoare Logic

If-then-else:

\[
\{P \land b\} S \{Q\}, \quad \{P \land \neg b\} T \{Q\}
\]

\[
\{P\} \text{ if } b \text{ then } S \text{ else } T \{Q\}
\]

While (here \(P\) is called a loop invariant)

\[
\{P \land b\} S \{P\}
\]

\[
\{P\} \text{ while } b \text{ do } S \{P \land \neg b\}
\]

Sequencing:

\[
\{P\} S \{Q\}, \quad \{Q\} T \{R\}
\]

\[
\{P\} S; T \{R\}
\]

Weakening:

\[
P \quad \Rightarrow \quad Q, \quad \{Q\} S \{R\}, \quad R \quad \Rightarrow \quad T
\]

\[
\{P\} S \{T\}
\]
Some examples to work on

1. \( \{ x \geq 3 \} \ x := x + 2 \ \{ x \geq 5 \} \)

2. \( \{(y < 0 \land x > -1) \lor (y > 0)\} \) if \( y < 0 \) then \( x := x + 1 \) else \( x := y \) \( \{ x > 0 \} \)

3. \( \{ x \leq 6 \} \ \text{while} \ (x \leq 5) \ \text{do} \ x := x + 1 \ \{ x = 6 \} \)
Exercise

Prove using Hoare logic \( \{x \geq 1 \land x = n \land a = 1\} \ P \ \{a = n!\} \), where \( P \) is:

```plaintext
while (x \geq 1) {
    a := a \times x;
    x := x - 1
}
```
Relative completeness of Hoare rules

Does \( \{A\} P \{B\} \) mean there exists a proof tree for the same using the rules mentioned above?

Yes, provided the underlying logic is complete.

- That is, whenever \( A \Rightarrow B \) there ought to exist a proof for the same using the rules of the underlying logic.
- For example, (plain) first-order logic, and presburger arithmetic (first-order logic, plus natural numbers with addition) are complete. Peano arithmetic (which includes multiplication) is not complete.
**Weakest Precondition** $WP(P, B)$

$WP(P, B)$ is “a predicate that describes the exact set of states $s$ such that when program $P$ is started in $s$, if it terminates it will terminate in a state satisfying condition $B$.”

\[
\{ -1 < y \} \\

y = y + 1; \\
z = x + y; \\
\{x < z\}
Using weakest pre-conditions for verification

- Note that \( \{A\} P \{B\} \iff A \implies WP(P, B) \).
- Therefore, if we have an algorithm for \( WP \) we can verify Hoare triples automatically.
- Tools such as Spec\# verify Hoare triples, using the above approach.
Illustration

To check:

\{ y > 10 \} \\
y = y + 1; \\
z = x + y; \\
\{ x < z \}

Check verification condition:

\[(y > 10) \implies (y > -1).\]
Rules for Computing Weakest Precondition

For assignment statement \( x = e \):

\[
\{ B[e/x] \} \\

x = e; \\

\{ B \} \]
Rules for Computing Weakest Precondition

For assignment statement $x = e$:

$$\{B[e/x]\}$$

$$x = e;$$

$$\{B\}$$

$$\{(x + y) > 0 \land y = 0\}$$

$$z = x + y;$$

$$\{z > 0 \land y = 0\}$$
Rules for Computing Weakest Precondition

If-the-else statement if c then $S_1$ else $S_2$:

\[
\{(c \land WP(S_1, B)) \lor \\
(\neg c \land WP(S_2, B))\}\]

if (c)
    $S_1$;
else
    $S_2$;

\{B\}
Rules for Computing Weakest Precondition

If-the-else statement \( \text{if } c \text{ then } S_1 \text{ else } S_2 \):

\[
\begin{align*}
&\{(c \land \text{WP}(S_1, B)) \lor \\
&\text{(} \neg c \land \text{WP}(S_2, B)\text{)}\} \\
&\text{if (c)} \\
&\quad \text{S1;} \\
&\text{else} \\
&\quad \text{S2;} \\
&\{B\}
\end{align*}
\]

\[
\begin{align*}
&\{((x < y) \land (y > w)) \lor \\
&\text{(} ((x \geq y) \land (x > w))\text{)}\} \\
&\text{if (x < y)} \\
&\quad z = y; \\
&\text{else} \\
&\quad z = x; \\
&\{z > w\}
\end{align*}
\]
WP rule for sequencing

\[ WP(S; T, B) = WP(S, WP(T, B)). \]
Weakest Precondition for while statements

- Let $W = \text{"while b do S"}$. 
- In general it is not possible to compute the precise $WP(W, B)$. 
- It is possible to compute an under-approximating condition $WP'(W, B)$ such that $WP'(W, B) \implies WP(W, B)$. 
  - Unroll the loop $k$ times, for some chosen value $k \geq 0$, and let $W'$ be the thus unrolled loop. 
  - For e.g., for $k = 0$ 
    - $W' = \text{skip}$ 
    - for $k = 2$, 
      - $W' = \text{"if (b) \{ S; if (b) S \}"}$. 
  - Now, $WP'(W, B) \equiv WP(W', B \land (\neg b))$. 
  - Higher value of $k$ gives a better $WP'(W, B)$. 
- Using this, one can verify a hoare triple $\{A\} P \{B\}$ conservatively. 
  - That is, the above triple is true if $A \implies WP'(W, B)$ (the converse is not necessarily true).
Another approach: under-approximating weakest pre-conditions given loop invariants

while loops

- $i$ is said to be a correct loop invariant in $W = \text{"while } b \text{ invariant } i \text{ do } S\text{"}$ iff $(i \land b) \implies WP(S, i)$.
- $WP'(W, B) \equiv (B \land \neg b) \lor (((i \land \neg b) \implies B) \land i)$. 
Illustration

Consider the example loop $W$ below

```plaintext
while (i < n) invariant i
    i++;
```

- Let $B = \text{“i == n”}$.  
  - $i \equiv \text{“i < n”}$, is not a correct loop invariant. 
  - $i \equiv \text{“i <= n”}$ is correct, and is sufficient to imply the post-condition $B$. In this case $WP'(W, B) = WP(W, B) = \text{“i <= n”}$. 
  - $i \equiv \text{“i <= n+1”}$ is a correct (but weak) loop invariant, and is not sufficient to imply the post-condition. In this case $WP'(W, B)$ is false.

- Let $B = \text{“n == 10”}$.  
  - $i \equiv \text{“n == 10”}$ is a correct loop invariant, and is necessary to imply the post-condition $B$. 

Conclusion

- Hoare logic can be extended to reason about programs with arrays, pointers [Separation Logic], function calls, etc.
- Finds application in recent program analysis techniques like finding “path conditions” in automated directed testing, and null-deference analysis.