Hoare triples as assertions of partial correctness. Programs as state transformers Hoare logic rules Weakest Preconditions

Hoare Logic

Deepak D’Souza
K. V. Raghavan

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

November 18, 2015
Outline

1. Hoare triples as assertions of partial correctness.
2. Programs as state transformers
3. Hoare logic rules
4. Weakest Preconditions
Hoare Logic

- A way of asserting properties of programs.
- Hoare triple: \( \{A\} P \{B\} \) asserts that “Whenever program \( P \) is started in a state satisfying condition \( A \), if it terminates, it will terminate in a state satisfying condition \( B \).”
- Example assertion: \( \{n \geq 1\} P \{a = n!\} \), where \( P \) is the program:
  
  ```plaintext
  x := n;
  a := 1;
  while (x \geq 1) {
    a := a * x;
    x := x - 1
  }
  ```
- 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
}
```
Programs as State Transformers

View program \( P \) as a partial map \([P] : State \rightarrow State\). (Assume that \( State = Var \rightarrow \mathbb{Z} \).)

\[ P \langle x \mapsto 2, \ y \mapsto 10, \ z \mapsto 3 \rangle \]

\[
\begin{align*}
y & := y + 1; \\
z & := x + y
\end{align*}
\]

\[ \langle x \mapsto 2, \ y \mapsto 11, \ z \mapsto 13 \rangle \]
Predicates on States

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

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

\[
\begin{align*}
&\{10 \leq y\} \\
y := y + 1; \\
z := x + y \\
&\{x < z\}
\end{align*}
\]
Mathematical meaning of a Hoare triple

- View program $P$ as a relation

$$[P] \subseteq \text{State} \times \text{State}.$$  

so that $(s, t) \in [P]$ iff it is possible to start $P$ in the state $s$ and terminate in state $t$.

- As usual here elements of $\text{State}$ are maps from variables to integers.

- $[P]$ is possibly non-deterministic, in case we also want to model non-deterministic assignment etc.

- Then the Hoare triple $\{A\} P \{B\}$ is true iff for all states $s$ and $t$: whenever $s \models A$ and $(s, t) \in [P]$, then $t \models B$.

- In other words $\text{Post}_{[P]}([A]) \subseteq [B]$. 
Give “weakest” preconditions

1. \{?\} \ x := x + 2 \ \{x \geq 5\}
Give “weakest” preconditions

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

2. if (y < 0) then x := x + 1 else x := y
   \{ x > 0 \}
Give “weakest” preconditions

1. \( \{ x \geq 3 \} \ x := x + 2 \ \{ x \geq 5 \} \)
   \( \{ (y < 0 \land x > -1) \lor (y > 0) \} \)
2. if \( (y < 0) \) then \( x := x + 1 \) else \( x := y \)
   \( \{ x > 0 \} \)
3. \( \{ ? \} \) while \( (x \leq 5) \) do \( x := x + 1 \) \( \{ x = 6 \} \)
Give “weakest” preconditions

1. \{ \ x \geq 3 \} \ x := x + 2 \ \{ \ x \geq 5 \}
   \{ (y < 0 \land x > -1) \lor (y > 0) \}

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

3. \{ x \leq 6 \} while (x \leq 5) do x := x+1 \ \{ x = 6 \}
Proof rules of Hoare Logic

Axiom of Valid formulas:

\[ \frac{\text{provided } \models A}{A} \]
provided \( \models A \) (i.e. \( A \) is a valid logical formula, eg. \( x > 10 \implies x > 0 \)).

Skip:

\[ \{ A \} \text{skip} \{ A \} \]

Assignment

\[ \{ A[e/x] \} \text{x := e} \{ A \} \]
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 \text{ is called a } \textit{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:

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

\{P\} \; S \; \{T\}
Some examples to work on

Use the rules of Hoare logic to prove the following assertions:

1. \( \{ x \geq 3 \} \quad x := x + 2 \quad \{ x \geq 5 \} \)
2. \( \{(y < 0) \land (x > -1)\} \quad \text{if } (y < 0) \text{ then } x := x + 1 \text{ else } x := y \quad \{ x > 0 \} \)
3. \( \{ x \leq 0 \} \quad x = x + 1; \text{ while } (x \leq 5) \text{ do } x := x + 1 \quad \{ x \leq 7 \} \)
Hoare triples as assertions of partial correctness. Programs as state transformers. Hoare logic rules. Weakest Preconditions.

Illustration

Note: Need to guess loop invariant:
\( x \leq 6 \)

\[ \begin{align*}
\{ x \leq 6 \} & \quad \text{while} \quad \ldots \quad \{ x \leq 6 \land x > 5 \} \\
\{ x \leq 7 \} & \quad \text{Sequencing}
\end{align*} \]
Exercise

Prove using Hoare logic:

\[ \{ n \geq 1 \} \; P \; \{ a = n! \}, \]

where \( P \) is the program:

\[
\begin{align*}
\text{x} & \;:=\; \text{n}; \\
\text{a} & \;:=\; 1; \\
\text{while} \; (x \geq 1) \{} \\
\text{a} & \;:=\; \text{a} \times \text{x}; \\
\text{x} & \;:=\; \text{x} - 1 \\
\text{\}} \\
\end{align*}
\]

Assume that factorial is defined as follows:

\[
n! = \begin{cases} 
n \times (n - 1) \times \cdots \times 1 & \text{if } n \geq 1 \\
1 & \text{if } n = 0 \\
-1 & \text{if } n < 0 \end{cases}
\]
Exercise

Prove using Hoare logic:

\[ \{ n \geq 1 \} \ P \ \{ a = n! \}, \]

where \( P \) is the program:

S1: \( x := n; \)
S2: \( a := 1; \)
S3: while (\( x \geq 1 \)) {
    S4: \( a := a \times x; \)
    S5: \( x := x - 1 \)
}

Assume that factorial is defined as follows:

\[
  n! = \begin{cases} 
    n \times (n - 1) \times \cdots \times 1 & \text{if } n \geq 1 \\
    1 & \text{if } n = 0 \\
    -1 & \text{if } n < 0 
  \end{cases}
\]
Solution

Need a loop invariant $P$ satisfying:

1. $\{ n \geq 1 \} \ S1; \ S2 \ \{ P \}$
2. $\{ P \land (x \geq 1) \} \ S4; \ S5 \ \{ P \}$
3. $(P \land \neg(x \geq 1)) \implies (a = n!)$
Solution

Need a loop invariant $P$ satisfying:

1. $\{ n \geq 1 \} \ S1; \ S2 \ \{P\}$
2. $\{ P \land (x \geq 1) \} \ S4; \ S5 \ \{P\}$
3. $(P \land \neg (x \geq 1)) \implies (a = n!)$

A potential $P$: $(x \geq 0) \land (a \times x! = n!)$.  

Soundness and Completeness of Hoare logic

- Hoare logic is sound (i.e. if we can prove “{A} P {B}” in the logic, then {A} P {B} is true.)
- Conversely, is it “complete”? That is, if {A} P {B} is true for a program P and pre/post-conditions A and B, does there exists a proof tree for {A} P {B} using the rules of Hoare logic?
- Yes, provided the underlying logic L can express all “weakest preconditions” (for all programs and post-conditions expressed in L).
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 \( B \)."

\[
\begin{align*}
\text{All States} & \quad WP(P, B) \\
\end{align*}
\]

\[
\begin{align*}
A & \quad WP(P, B) \\
\end{align*}
\]

\[
\begin{align*}
P & \quad A = \Rightarrow WP(P, B) \\
\end{align*}
\]

\[
\begin{align*}
\{ -1 < y \} \\
\end{align*}
\]

\[
\begin{align*}
y & := y + 1; \\
z & := x + y; \\
\{ x < z \} \\
\end{align*}
\]
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 $B$.”

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

\[
y := y + 1; \\
z := x + y; \\
\{ x < z \}
\]

Checking $\{A\} P \{B\}$

First compute $WP(P, B)$. Then check if $A \implies WP(P, B)$.
Generating Verification Conditions

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 $x = e$:

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

$$x = e;$$

$$\{B\}$$
For assignment statement $x = e$:

$$\begin{align*}
\{ B[e/x] \} & \quad \{ (x + y) > 0 \land y = 0 \} \\
x = e; & \quad z = x + y; \\
\{ B \} & \quad \{ z > 0 \land y = 0 \}
\end{align*}$$
If-then-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)
  S1;
else
  S2;

$$\{ B \}$$
If-then-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
    $S_2$;

$\{B\}$

$\{((x < y) \land (y > w)) \lor ((x \geq y) \land (x > w))\}$

if (x < y)
    $z = y$;
else
    $z = x$;

$\{z > w\}$
WP rule for sequencing

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

We can “approximate” \( WP(\text{while } b \text{ do } c) \).

- Let \( w \) be the loop mentioned above.
- \( WP_i \) is the set of states just before the loop from which control enters the loop body at most \( i \) times and then breaks out, and the state when control leaves the loop satisfies the post-condition \( A \).
- \( WP_i \) defined inductively as follows:

\[
\begin{align*}
WP_0 &= \neg b \land A \\
WP_{i+1} &= (\neg b \land A) \lor (b \land WP(c, WP_i))
\end{align*}
\]

- Then \( WP(w, A) \) can be shown to be “or” of all the sets \( WP_0, WP_1, \ldots \).
Weakest Precondition for while statements

Another way to approximate this (this time, a shrinking over-approximation):

- $WP_i(w, A) =$ the set of states from which the body $c$ of the loop is either entered more than $i$ times or we exit the loop in a state satisfying $A$.

- $WP_i$ defined inductively as follows:

$$
WP_0 = b \lor A \\
WP_{i+1} = (\neg b \land A) \lor (b \land WP(c, WP_i))
$$

- Then $WP(w, A)$ can be shown to be the “limit” or least upper bound of the chain $WP_0(w, A), WP_1(w, A), \ldots$ in a suitably defined lattice (here the join operation is “And” or intersection).
Consider the program $w$ below:

\begin{verbatim}
while (x \geq 10) do
  x := x - 1
\end{verbatim}

- What is the weakest precondition of $w$ with respect to the postcondition ($x \leq 0$)?
- Compute $WP_0(w, (x \leq 0))$, $WP_1(w, (x \leq 0))$, ...
Illustration of $WP_i$ through example

Consider the program $w$ below:

\begin{verbatim}
while (x ≥ 10) do
    x := x - 1
\end{verbatim}

- What is the weakest precondition of $w$ with respect to the postcondition $(x ≤ 0)$?
- Compute $WP_0(w, (x ≤ 0)), WP_1(w, (x ≤ 0)), \ldots$
Relative completeness of Hoare logic

- Hoare logic is complete provided the underlying logic $L$ can express the WP for any program $P$ and post-condition $B$.
- Proved by Cook in 1974.
- Proof uses WP predicates and proceeds by induction on the structure of the program $P$. 
Conclusion

• Hoare logic can be extended to reason about programs with arrays, pointers [Separation Logic], function calls, etc.

• Elements of Hoare logic and Weakest Preconditions find application in many program analysis techniques.