Hoare Logic: Part II

COMP2600 — Formal Methods for Software Engineering

Jinbo Huang

Australian National University
Factorial

\{ n \geq 0 \}\]

\texttt{fact := 1;}
\texttt{i := n;}
\texttt{while (i>0) do}
\begin{align*}
& \quad \texttt{fact := fact * i;} \\
& \quad \texttt{i := i-1}
\end{align*}
\{ \texttt{fact = n!} \}
First we need a loop invariant \( P \)

\[
\begin{align*}
\text{fact} & := 1; \\
i & := n; \\
\text{while} \ (i>0) \ \text{do} \\
& \quad \text{fact} := \text{fact} \times i; \\
& \quad i := i-1 \\
{\{ \text{fact} = n! \}}
\end{align*}
\]

After each iteration, \( \text{fact} = n \times (n-1) \times \cdots \times (i+1) \)

\( P \equiv (\text{fact} \times i! = n!) \) seems plausible

However, we want \( (P \land i \leq 0) \Rightarrow (\text{fact} = n!) \)

Take \( P \equiv (\text{fact} \times i! = n! \land i \geq 0) \)
Check \( P \equiv (\text{fact} \ast i! = n! \land i \geq 0) \) is an invariant

That is,

\[
\{ P \land i > 0 \} \text{fact} := \text{fact} \ast i; \ i := i-1 \{P\}
\]

By assignment, we have

\[
\{ \text{fact} \ast (i-1)! = n! \land i - 1 \geq 0 \} \ i := i-1 \ \{ \text{fact} \ast i! = n! \land i \geq 0 \}
\]

Equivalently,

\[
\{ \text{fact} \ast (i-1)! = n! \land i > 0 \} \ i := i-1 \ \{ \text{fact} \ast i! = n! \land i \geq 0 \}
\]

By assignment again, we have

\[
\{ \text{fact} \ast i \ast (i-1)! = n! \land i > 0 \}
\]

\[
\text{fact} := \text{fact} \ast i
\]

\[
\{ \text{fact} \ast (i-1)! = n! \land i > 0 \}
\]

This precondition is precisely \( P \land i > 0 \)
Completing the proof

\[ P \equiv (\text{fact} \times i! = n! \land i \geq 0) \]

Apply the while-rule:

\[
\{P\} \text{ while (i>0) } \ldots \{P \land i \leq 0\}
\]

By postcondition weakening,

\[
\{P\} \text{ while (i>0) } \ldots \{\text{fact} = n!\}
\]

Check that the \textit{initialisation establishes the invariant}:

\[
\{n \geq 0\}
\]

\[
\text{fact := } 1; \ i := n
\]

\[
\{\text{fact} \times i! = n! \land i \geq 0\}\]
Formally

\[ P \equiv (\text{fact} \ast i! = n! \land i \geq 0) \]

1. \[ \{1 \ast n! = n! \land n \geq 0\} \text{fact} := 1 \]
   \[ \{\text{fact} \ast n! = n! \land n \geq 0\} \] (Assignment)

2. \[ \{n \geq 0\} \text{fact} := 1 \{\text{fact} \ast n! = n! \land n \geq 0\} \] (1, Equivalence)

3. \[ \{\text{fact} \ast n! = n! \land n \geq 0\} i := n \{P\} \] (Assignment)

4. \[ \{\text{fact} \ast i \ast (i - 1)! = n! \land i > 0\} \text{fact} := \text{fact} \ast i \]
   \[ \{\text{fact} \ast (i - 1)! = n! \land i > 0\} \] (Assignment)

5. \[ \{P \land i > 0\} \text{fact} := \text{fact} \ast i \]
   \[ \{\text{fact} \ast (i - 1)! = n! \land i > 0\} \] (4, Equivalence)

6. \[ \{\text{fact} \ast (i - 1)! = n! \land i - 1 \geq 0\} i := i-1 \{P\} \] (Assignment)
7. \{fact \ast (i - 1)! = n! \land i > 0\} i := i-1 \{P\} \quad (6, \text{Equivalence})

8. \{P \land i > 0\} fact := fact \ast i; i := i-1 \{P\} \quad (5, 7, \text{Sequencing})

9. \{P\} \textbf{while} (i>0) \ldots \{P \land i \leq 0\} \quad (8, \text{While})

10. \{P\} \textbf{while} (i>0) \ldots \{fact = n!\} \quad (9, \text{Postcondition Weakening})

11. \{n \geq 0\} \textit{Program} \{fact = n!\} \quad (2, 3, 10, \text{Sequencing})
Completeness of Hoare Logic

All true Hoare triples can be proved (with expressive assertion language)

However, we may not be able to verify a proof!

Consider precondition strengthening rule:

\[ P_s \Rightarrow P_w \quad \{P_w\} \quad S \quad \{Q\} \]

\[ \{P_s\} \quad S \quad \{Q\} \]

Not all arithmetic truths \((P_s \Rightarrow P_w)\) can be proved (Gödel’s incompleteness theorem)

Hence, completeness is relative to access to arithmetic truths
Weakest Precondition

An assertion/condition maps program states to \( \{true, false\} \)

- \((x > y)(\sigma) = true\) iff \(x > y\) in state \(\sigma\)

Define assertion \(wp\) for all programs \(S\) and postconditions \(Q\)

- \(wp(S, Q)(\sigma) = true\) iff for all states \(\sigma'\),
  (executing \(S\) in \(\sigma\) results in \(\sigma'\)) \(\Rightarrow (Q(\sigma') = true)\)

With quantification, assertion language can express all \(wp(S, Q)\)

- Quantification required, e.g., to say “\(x\) is a multiple of \(y\)”:\( \exists i. x = i \ast y \)
- Formal proof uses Gödel’s \(\beta\) function to encode sequence of numbers (representing states) of arbitrary length with small number of variables
Properties of $wp$

Definition: $wp(S, Q)(\sigma) = true$ iff for all states $\sigma'$, (executing $S$ in $\sigma$ results in $\sigma'$) $\Rightarrow (Q(\sigma') = true)$

$\models \{ wp(S, Q) \} S \{ Q \}$

- $wp$ is indeed a valid precondition

If $\models \{ P \} S \{ Q \}$ then $P \Rightarrow wp(S, Q)$

- $wp$ is the weakest possible precondition

("$\models$" denotes "true" or "valid")
Proving Completeness of Hoare Logic

Need only show \( \vdash \{ wp(S, Q) \} S \{ Q \} \) for all \( S, Q \)

- If \( \vdash \{ P \} S \{ Q \} \) then \( P \Rightarrow wp(S, Q) \)
- So \( \vdash \{ P \} S \{ Q \} \) by precondition strengthening

Proof by structural induction

- Assignment
- Sequencing
- Conditional
- While

(“\( \vdash \)” denotes “provable”)
Proof of Completeness: Assignment

Need to show

\[ \vdash \{ \text{wp}(x:=e, \, Q) \} \text{ } x:=e \{ Q \} \]

But

\[ \text{wp}(x:=e, \, Q) = Q(e/x) \]

Hence result follows by assignment rule
**Proof of Completeness:** $S_1; S_2$

Need to show

\[ \vdash \{ wp(S_1; S_2, Q) \} S_1; S_2 \{ Q \} \]

By induction,

\[ \vdash \{ wp(S_2, Q) \} S_2 \{ Q \} \]

\[ \vdash \{ wp(S_1, wp(S_2, Q)) \} S_1 \{ wp(S_2, Q) \} \]

By sequencing rule,

\[ \vdash \{ wp(S_1, wp(S_2, Q)) \} S_1; S_2 \{ Q \} \]

Finally, $wp(S_1; S_2, Q) \Rightarrow wp(S_1, wp(S_2, Q))$, or

$wp(S_1; S_2, Q)(\sigma) \Rightarrow wp(S_1, wp(S_2, Q))(\sigma)$
**Proof of Completeness:** if $b$ then $S_1$ else $S_2$

Need to show

$$
\vdash \{wp(if \ b \ then \ S_1 \ else \ S_2, Q)\} \ if \ b \ then \ S_1 \ else \ S_2 \ \{Q\}$$

By induction,

$$
\vdash \{wp(S_1, Q)\} \ S_1 \ \{Q\}, \{wp(S_2, Q)\} \ S_2 \ \{Q\}
$$

Define $P \equiv (b \land wp(S_1, Q)) \lor (\neg b \land wp(S_2, Q))$

By precondition equivalence and conditional rule,

$$
\vdash \{P \land b\} \ S_1 \ \{Q\}, \{P \land \neg b\} \ S_2 \ \{Q\}$$

$$
\vdash \{P\} \ if \ b \ then \ S_1 \ else \ S_2 \ \{Q\}
$$

Finally, by case analysis on $b$,

$$
wp(if \ b \ then \ S_1 \ else \ S_2, Q) \Rightarrow P
$$
Proof of Completeness: while $b$ do $S$

Let $P = \text{wp}(\text{while } b \text{ do } S, Q)$

Lemma 1 $\neg b \land P \Rightarrow Q$

- $P$ guarantees that $Q$ holds in resulting state
- If $\neg b$ then while terminates immediately without modifying state
- Hence $\neg b \Rightarrow (P \Rightarrow Q)$
Proof of Completeness: while $b$ do $S$

Let $P = \text{wp}(\text{while } b \text{ do } S, Q)$

**Lemma 2**  
$b \land P \Rightarrow \text{wp}(S, P)$, or equivalently $\models \{b \land P\} S \{P\}$

- Let $\sigma$ be initial state where $b \land P$ holds, $\sigma'$ the state after $S$ terminates (trivial if $S$ doesn’t terminate). Wish to show $P(\sigma') = true$

- If while doesn’t terminate from $\sigma'$, $P(\sigma') = true$ by definition of $\text{wp}$

- Let $\sigma''$ be state after while terminates
  - Consider the run of while from $\sigma$: $Q(\sigma'') = true$
  - Consider the run of while from $\sigma'$: $P(\sigma') = true$ by definition of $\text{wp}$
Proof of Completeness: while $b$ do $S$

Let $P = \text{wp}(\text{while } b \text{ do } S, Q)$

By induction,
\[ \vdash \{\text{wp}(S, P)\} S \{P\} \]

By Lemma 2 ($b \land P \Rightarrow \text{wp}(S, P)$) and precondition strengthening,
\[ \vdash \{b \land P\} S \{P\} \]

By while-rule, Lemma 1 ($\neg b \land P \Rightarrow Q$), and postcondition weakening,
\[ \vdash \{P\} \text{while } b \text{ do } S \{Q\} \]
Hoare logic expresses *partial correctness*

\[ \{ P \} \ S \ \{ Q \} \]

means

if \( P \) holds initially and \( S \) terminates
then \( Q \) holds afterwards

Does *not* guarantee that \( S \) terminates

All Hoare triples *are true* where \( S \) does not terminate

Hence no Hoare triple can express that \( S \) terminates

Can express that \( S \) does not terminate though:

- \( \{ true \} \ S \ \{ false \} \)

- Provable if *while* condition is an invariant
An Extension to Handle Total Correctness

\[ [P] \ S \ [Q] \] means

if \( P \) holds initially
then \( S \) terminates and \( Q \) holds afterwards

Total correctness = partial correctness + termination

To assert termination alone:

\[ [P] \ S \ [true] \]
Rules for Total Correctness

For deriving total correctness assertions

\[ [P] \; S \; [Q] \]

all old rules continue to work, except one

- Assignment
- Precondition strengthening, postcondition weakening
- Sequencing
- Conditional
- While
Rules for Total Correctness

\[ [Q(e)] \ x := e \ [Q(x)] \]  

(Assignment)

Assumes evaluation of expression \( e \) always terminates

Fine for our toy language

- Require expression to always return a number

In general, depends on the language

- Can expression involve calls to functions, which may not terminate?
  - Even absent \texttt{while}-loops, recursion could cause nontermination
- What if there’s an error, e.g., division by zero?
Rules for Total Correctness

\[ P_s \Rightarrow P_w \quad [P_w] \quad S \quad [Q] \]

\[ [P_s] \quad S \quad [Q] \]

(Precondition Strengthening)

\[ [P] \quad S \quad [Q_s] \quad Q_s \Rightarrow Q_w \]

\[ [P] \quad S \quad [Q_w] \]

(Postcondition Weakening)

\[ [P] \quad S_1 \quad [Q] \quad [Q] \quad S_2 \quad [R] \]

\[ [P] \quad S_1; \quad S_2 \quad [R] \]

(Sequencing)

\[ [P \land b] \quad S_1 \quad [Q] \quad [P \land \neg b] \quad S_2 \quad [Q] \]

\[ [P] \quad \text{if} \ b \ \text{then} \ S_1 \ \text{else} \ S_2 \quad [Q] \]

(Conditional)
Termination of while

\[ y > 0 \]

while \( y \leq r \) do

\[
\begin{align*}
    r &:= r - y; \\
    q &:= q + 1
\end{align*}
\]

\[ true \]
Termination of \texttt{while}

\[ y > 0 \]

\texttt{while} (\texttt{y} <= \texttt{r}) do
\begin{align*}
    \texttt{r} & := \texttt{r} - \texttt{y}; \\
    \texttt{q} & := \texttt{q} + 1
\end{align*}

\[ \text{true} \]

Observations:

\begin{itemize}
    \item \texttt{q} := \texttt{q} + 1 \text{ irrelevant}
    \item \texttt{y} doesn't change, stays > 0
    \item \texttt{r} strictly decreases on each iteration
    \item Hence \texttt{y} <= \texttt{r} will eventually be false
\end{itemize}
Termination of while: General Condition

\[ y > 0 \]

while (y <= r) do
  r := r - y;
  q := q + 1

\[ true \]

Find a variant, an expression \( E \) such that

- \( E \) stays \( \geq 0 \) at beginning of each iteration
- \( E \) strictly decreases on each iteration

What’s a good \( E \) for this example?
**Total Correctness of while**

To show

\[
[P] \textbf{while } b \textbf{ do } S \ [P \land \lnot b]
\]

*in addition* to showing partial correctness, find \( E \) such that

- \( E \) stays \( \geq 0 \) at beginning of each iteration:
  \[P \land b \Rightarrow E \geq 0\]

- \( E \) strictly decreases on each iteration
  \[
  [P \land b \land (E = n)] S [P \land (E < n)]
  \]

\((n \text{ is auxiliary variable } \textit{not appearing elsewhere}, \text{ used to “remember” initial value of } E)\)
While-Rule for Total Correctness

\[ P \land b \Rightarrow E \geq 0 \quad [P \land b \land (E = n)] S [P \land (E < n)] \]
\[ [P] \textbf{while} \ b \textbf{ do } S \ [P \land \neg b] \]

(While)

where \( n \) is an auxiliary variable \textit{not appearing elsewhere}
Using the Rule

\[ P \land b \Rightarrow E \geq 0 \quad [P \land b \land (E = n)] \ S \ [P \land (E < n)] \]

\[ [P] \ while \ b \ do \ S \ [P \land \neg b] \]

Prove \([y > 0] \ while \ (y \leq r) \ do \ \{r := r-y; \ q := q+1\} \ [true]\]

1. \((y > 0) \land (y \leq r) \Rightarrow (r \geq 0)\) (Math)

2. \([ (y > 0) \land (r < n)] \ q := q+1 \ [(y > 0) \land (r < n)]\) (Assignment)

3. \([ (y > 0) \land (r - y < n)] \ r := r-y \ [(y > 0) \land (r < n)]\) (Assignment)

4. \((y > 0) \land (y \leq r) \land (r = n) \Rightarrow (y > 0) \land (r - y < n)\) (Math)

5. \([ (y > 0) \land (y \leq r) \land (r = n)] \ r := r-y \ [(y > 0) \land (r < n)]\) (3, 4, Precondition Strengthening)

6. \([ (y > 0) \land (y \leq r) \land (r = n)] \ r := r-y; \ q := q+1 \ [(y > 0) \land (r < n)]\)
7. $[y > 0]$ while ... $[(y > 0) \land (y > r)]$  

(1, 6, While)

8. $[y > 0]$ while ... $[true]$  

(7, Postcondition Weakening)
Using the Rule

\[
P \land b \Rightarrow E \geq 0 \quad [P \land b \land (E = n)] S [P \land (E < n)] \\
[\{P\} \text{ while } b \text{ do } S [P \land \neg b]]
\]

\([n \geq 0]\)

\[
\begin{align*}
\text{fact} & := 1; \\
i & := n; \\
\text{while } (i > 0) & \text{ do} \\
\quad \text{fact} & := \text{fact} \ast i; \\
\quad i & := i - 1 \\
\end{align*}
\]

\([\text{fact} = n!]\)

Same invariant \(P \equiv (\text{fact} \ast i! = n! \land i \geq 0)\)

Variant \(E \equiv i\)

(\text{use } m \text{ instead of } n \text{ in while-rule})
While-Rule for Total Correctness: Soundness

\[ P \land b \Rightarrow E \geq 0 \quad [P \land b \land (E = n)] \quad S \quad [P \land (E < n)] \]
\[
\begin{array}{c}
\frac{}{[P] \text{ while } b \text{ do } S \ [P \land \neg b]}
\end{array}
\]

Premises imply \textit{partial correctness} of the \textit{while}:\

\[
[P \land b \land (E = n)] \quad S \quad [P \land (E < n)]
\]
\[
\Rightarrow [P \land b \land (E = n)] \quad S \quad [P] \quad \text{(postcondition weakening)}
\]
\[
\Rightarrow [P \land b] \quad S \quad [P] \quad \text{(} n \text{ doesn’t appear elsewhere)}
\]
\[
\Rightarrow \{P \land b\} \quad S \quad \{P\} \quad \text{(total correctness implies partial correctness)}
\]
\[
\Rightarrow \{P\} \text{ while } b \text{ do } S \quad \{P \land \neg b\} \quad \text{(soundness of original while-rule)}
\]

(All triples in the above chain of reasoning are semantic assertions)
While-Rule for Total Correctness: Soundness

\[ P \land b \Rightarrow E \geq 0 \quad [P \land b \land (E = n)] \ S \ [P \land (E < n)] \]

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

Premises imply termination of the while: Show \textbf{while} \ b \ \textbf{do} \ S \ terminates from any state \( \sigma \) satisfying \( P \), by induction on the value of \( E \) in \( \sigma \)

- If \( \neg b(\sigma) \), done. Assume \( b(\sigma) \); have \( E(\sigma) \geq 0 \) by left premise

- Base case \( (E(\sigma) = 0) \):
  - Behavior of \( S \) (or the while) does not depend on value of \( n \)
  - May as well assume \( n = 0 \) in \( \sigma \)
  - Right premise implies \( P \land (E < 0) \) after one iteration
  - This, together with left premise, implies \( \neg b \), hence termination of \textbf{while}
While-Rule for Total Correctness: Soundness

\[ \begin{align*}
  P \land b & \Rightarrow E \geq 0 & & \left[ P \land b \land (E = n) \right] S \left[ P \land (E < n) \right] \\
  & \quad \quad \quad \quad \left[ P \right] \textbf{while} \ b \textbf{ do } S \left[ P \land \neg b \right]
\end{align*} \]

(While)

Premises imply \textit{termination} of the \textbf{while}: Show \textbf{while} \ b \textbf{ do } S terminates from any state \( \sigma \) satisfying \( P \), by induction on the value of \( E \) in \( \sigma \)

- Inductive step:
  - Hypothesis: \textbf{while} \ldots \text{ terminates from any state } \sigma' \text{ satisfying } P, \text{ if } E(\sigma') \leq k
  - Assume \( E(\sigma) = k + 1 \); may as well assume \( n = k + 1 \) in \( \sigma \)
  - Right premise implies \( P \land (E \leq k) \) after one iteration
  - Hence \textbf{while} \ldots \text{ terminates from that state, by induction hypothesis}
Total Correctness = Partial Correctness + Termination

Do our rules agree with this equation?

\[ \vdash [P] S [Q] \Rightarrow \vdash \{P\} S \{Q\} \quad \text{and} \quad \vdash [P] S [true] \quad ? \]

\[ \vdash [P] S [Q] \Rightarrow \models [P] S [Q] \quad \text{(Soundness)} \]
\[ \Rightarrow \models \{P\} S \{Q\} \quad \text{(Semantics of Hoare Triples)} \]
\[ \Rightarrow \vdash \{P\} S \{Q\} \quad \text{(Completeness of Rules for Partial Correctness)} \]

\[ \vdash [P] S [Q] \Rightarrow \vdash [P] S [true] \quad \text{(Postcondition Weakening)} \]
Total Correctness = Partial Correctness + Termination

Do our rules agree with this equation?

\[ \vdash \{ P \} S \{ Q \} \quad \text{and} \quad \vdash [P] S [true] \Rightarrow \vdash [P] S [Q] \]

(You won’t be asked to prove/disprove this)
While-Rule for Total Correctness: Completeness

\[ P \land b \Rightarrow E \geq 0 \quad [P \land b \land (E = n)] \quad S \quad [P \land (E < n)] \]

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

(While)

Do we have a complete Hoare logic for total correctness?

If \([P]\ \textbf{while}\ldots\ [Q]\), can we always find a suitable \(E\) to prove it?

- Values of variables in the \texttt{while} can change with no apparent pattern
- Can we always map them to a decreasing number with a simple expression \(E\) (using only +, −, *)?
While-Rule for Total Correctness: Generalized Version

\[
P(z + 1) \Rightarrow b \quad P(0) \Rightarrow \neg b \quad [P(z + 1)] S [P(z)]
\]

\[
[\exists z. P(z)] \textbf{while } b \textbf{ do } S \ [P(0)]
\]

where auxiliary variable $z$ ranges over natural numbers (i.e., $z \geq 0$)

Think of $z$ as “remaining number of iterations before termination”
Using Generalized While-Rule

\[
\begin{align*}
P(z + 1) & \Rightarrow b \\
P(0) & \Rightarrow \neg b \\
\end{align*}
\]

\[\begin{align*}
[P(z + 1)] S [P(z)] \\
\text{where auxiliary variable } z \text{ ranges over natural numbers (i.e., } z \geq 0) \\
\end{align*}\]

Prove \([true] \text{ while } (i > 0) \text{ do } i := i-1 [true]\]

\[
P(z) \equiv (i = z) \lor (i < 0 \land z = 0). \text{ Assuming } z \geq 0, \text{ we have}
\]

- \(P(z + 1) = (i = z + 1) \lor (i < 0 \land z + 1 = 0) = (i = z + 1); \text{ this } \Rightarrow b\)
- \(P(0) = (i = 0) \lor (i < 0 \land 0 = 0) = (i \leq 0); \text{ this is } \neg b\)
- \([(i = z + 1) \lor (i < 1 \land z = 0)] i := i-1 [P(z)] \quad \text{(Assignment, Equiv.)} \]
- \([P(z + 1)] i := i-1 [P(z)] \quad \text{(Precondition Strengthening)}\)
Using Generalized While-Rule

\[ P(z + 1) \Rightarrow b \quad P(0) \Rightarrow \neg b \quad [P(z + 1)] S [P(z)] \]

\[ \exists z. P(z) \quad \textbf{while} \quad b \quad \textbf{do} \quad S \quad [P(0)] \]

where auxiliary variable \( z \) ranges over natural numbers (i.e., \( z \geq 0 \))

Prove \([true]\) \textbf{while} (i > 0) \textbf{do} i := i-1 \[true\]

\( P(z) \equiv (i = z) \lor (i < 0 \land z = 0) \)

Now apply generalized while-rule:

\[ \exists z. (i = z) \lor (i < 0 \land z = 0) = [true] \]

\textbf{while} (i > 0) \textbf{do} i := i-1

\[ [(i = 0) \lor (i < 0 \land 0 = 0)] = [i \leq 0] \]

Final step by postcondition weakening
Generalized While-Rule: Completeness

\[ P(z + 1) \Rightarrow b \quad P(0) \Rightarrow \neg b \quad [P(z + 1)] \quad S \quad [P(z)] \]

\[ [\exists z. \quad P(z)] \quad \textbf{while} \quad b \quad \textbf{do} \quad S \quad [P(0)] \]

where \( z \) ranges over natural numbers (i.e., \( z \geq 0 \))

Don’t we have the same issue—finding a decreasing expression?

• No, because we no longer need to \textit{express a function} that computes \( z \)

Given arbitrary numbers \( x_0, x_1, \ldots, x_k \) (representing sequence of states), there is a function \( \beta(m, n, i) \) such that \( \beta(m, n, i) = x_i \) for all \( 0 \leq i \leq k \)

• Predicate \( \beta(m, n, z) = x \) \textit{implicitly} turns arbitrary sequence of states \( (x) \) into a decreasing number \( z \)

Resulting Hoare logic is complete