Sir Tony Hoare

- Pioneer in formal verification
- Invented: Quicksort,
- the null reference (called it his “billion dollar mistake”)
- CSP (formal specification language), and
- Hoare Logic
Summary

- \( \mathcal{L} \): A simple imperative programming language
- Hoare triples (SYNTAX)
- Hoare logic (PROOF)
- Semantics for Hoare logic
Summary

- \( \mathcal{L} \): A simple imperative programming language
- Hoare triples (SYNTAX)
- Hoare logic (PROOF)
- Semantics for Hoare logic
Imperative Programming

imperō

**Definition**

*Imperative programming* is where programs are described as a series of *statements* or commands to manipulate mutable *state* or cause externally observable *effects*.

*States* may take the form of a *mapping* from variable names to their values, or even a model of a CPU state with a memory model (for example, in an *assembly language*).
Consider the vocabulary of basic arithmetic:

- **Constant symbols:** 0, 1, 2, \ldots
- **Function symbols:** +, *, \ldots
- **Predicate symbols:** <, \leq, \geq, |, \ldots
Consider the vocabulary of basic arithmetic:

- Constant symbols: 0, 1, 2, …
- Function symbols: +, *, …
- Predicate symbols: <, ≤, ≥, |, …

An (arithmetic) expression is a term over this vocabulary.
Consider the vocabulary of basic arithmetic:

- **Constant symbols**: 0, 1, 2, \ldots
- **Function symbols**: +, \times, \ldots
- **Predicate symbols**: <, \leq, \geq, |, \ldots

An **(arithmetic) expression** is a term over this vocabulary.

A **boolean expression** is a predicate formula over this vocabulary.
The language $\mathcal{L}$ is a simple imperative programming language made up of four statements:

**Assignment:** $x := e$

where $x$ is a variable and $e$ is an arithmetic expression.
The language $\mathcal{L}$ is a simple imperative programming language made up of four statements:

**Assignment:** $x := e$

where $x$ is a variable and $e$ is an arithmetic expression.

**Sequencing:** $P; Q$
The language $\mathcal{L}$ is a simple imperative programming language made up of four statements:

**Assignment:** $x := e$

where $x$ is a variable and $e$ is an arithmetic expression.

**Sequencing:** $P; Q$

**Conditional:** if $g$ then $P$ else $Q$ fi

where $g$ is a boolean expression.
The language $\mathcal{L}$ is a simple imperative programming language made up of four statements:

**Assignment:** $x := e$

where $x$ is a variable and $e$ is an arithmetic expression.

**Sequencing:** $P;Q$

**Conditional:** if $g$ then $P$ else $Q$ fi

where $g$ is a boolean expression.

**While:** while $g$ do $P$ od
Factorial in $\mathcal{L}$

Example

\[
i := 0; \\
m := 1; \\
\text{while } i < N \text{ do} \\
    i := i + 1; \\
    m := m \times i \\
\text{od}
\]
Summary

- $\mathcal{L}$: A simple imperative programming language
- Hoare triples (SYNTAX)
- Hoare logic (PROOF)
- Semantics for Hoare logic
Summary

- $\mathcal{L}$: A simple imperative programming language
- Hoare triples (SYNTAX)
- Hoare logic (PROOF)
- Semantics for Hoare logic
We’ll define a *Hoare Logic* for $\mathcal{L}$ to allow us to prove properties of our program. We write a *Hoare triple* judgement as:

$$\{ \varphi \} \ P \ \{ \psi \}$$

Where $\varphi$ and $\psi$ are logical formulae about states, called *assertions*, and $P \in \mathcal{L}$. This triple states that if the program $P$ terminates successfully from a starting state satisfying the *precondition* $\varphi$, then the final state will satisfy the *postcondition* $\psi$. 
Example

\[(\{x = 0\}) \ x := 1 \ {\{x = 1\}}\]
Hoare triple: Examples

Example

\{(x = 0)\} \ x := 1 \ \{(x = 1)\}

\{(x = 499)\} \ x := x + 1 \ \{(x = 500)\}
Hoare triple: Examples

Example

\[
\begin{align*}
\{(x = 0)\} & \quad x := 1 \quad \{(x = 1)\} \\
\{(x = 499)\} & \quad x := x + 1 \quad \{(x = 500)\} \\
\{(x > 0)\} & \quad y := 0 - x \quad \{(y < 0) \land (x \neq y)\}
\end{align*}
\]
Example

\{ N \geq 0 \}
\begin{align*}
i &:= 0; \\
m &:= 1; \\
\text{while } i < N \text{ do} \\
&\quad i := i + 1; \\
&\quad m := m \times i \\
\text{od} \\
\{ m = N! \} 
\end{align*}
Summary

- $\mathcal{L}$: A simple imperative programming language
- Hoare triples (SYNTAX)
- Hoare logic (PROOF)
- Semantics for Hoare logic
Motivation

Question

We know what we want informally; how do we establish when a triple is valid?
Motivation

Question

We know what we want informally; how do we establish when a triple is valid?

- Develop a semantics, OR

Hoare logic consists of one axiom and four inference rules for deriving Hoare triples.
Motivation

Question

We know what we want informally; how do we establish when a triple is valid?

- Develop a semantics, OR
- Derive the triple in a syntactic manner (i.e. Hoare proof)

**Hoare logic** consists of one axiom and four inference rules for deriving Hoare triples.
Assignment

\[
\{\varphi[e/x]\} x := e \{\varphi\} \quad \text{(assign)}
\]

Intuition:
If \( x \) has property \( \varphi \) \textit{after} executing the assignment; then \( e \) must have property \( \varphi \) \textit{before} executing the assignment.
Example

\{(y = 0)\} x := y \{ (x = 0)\}
Example

\{(y = 0)\} x := y \{(x = 0)\}

\{ \} x := y \{(x = y)\}
Assignment: Example

Example

\[ \{(y = 0)\} x := y \{(x = 0)\} \]

\[ \{(y = y)\} x := y \{(x = y)\} \]
Assignment: Example

Example

\{(y = 0)\} x ::= y \{(x = 0)\}

\{(y = y)\} x ::= y \{(x = y)\}

\{\} x ::= 1 \{(x < 2)\}
Assignment: Example

Example

\[ \{(y = 0)\} \ x := y \ {(x = 0)} \]

\[ \{(y = y)\} \ x := y \ {(x = y)} \]

\[ \{(1 < 2)\} \ x := 1 \ {(x < 2)} \]

\[ \{(y = 3)\} \ x := y \ {(x > 2)} \]
Assignment: Example

Example

\{(y = 0)\} \ x := y \ { (x = 0) \}

\{(y = y)\} \ x := y \ { (x = y) \}

\{(1 < 2)\} \ x := 1 \ { (x < 2) \}

\{(y = 3)\} \ x := y \ { (x > 2) \}  \ Problem!
Sequence

\[
\begin{array}{c}
\{\varphi\} P \{\psi\} \quad \{\psi\} Q \{\rho\} \\
\{\varphi\} P; Q \{\rho\}
\end{array}
\]

Intuition:
If the postcondition of \( P \) matches the precondition of \( Q \) we can sequentially combine the two program fragments
Sequence: Example

Example

\[
\begin{array}{c}
\{ \} x := 0 \{ \} \quad \{ \} y := 0 \{ (x = y) \} \\
\{ \} x := 0; y := 0 \{ (x = y) \}
\end{array}
\]
Sequence: Example

Example

\[
\begin{align*}
\{ \} x & := 0 \{ (x = 0) \} \\
\{ \} y & := 0 \{ (x = y) \}
\end{align*}
\]

\( (seq) \)
Sequence: Example

Example

\[
\begin{align*}
\{(0 = 0)\} & \ x := 0 \ \{(x = 0)\} & \ (x = 0) & \ y := 0 \ \{(x = y)\} \\
(0 = 0) & \ x := 0; \ y := 0 \ \{(x = y)\} & \ (\text{seq})
\end{align*}
\]
Conditional

\[
\{ \varphi \land g \} \ P \ \{ \psi \} \quad \{ \varphi \land \neg g \} \ Q \ {\psi} \\
\quad \{ \varphi \} \ \text{if} \ g \ \text{then} \ P \ \text{else} \ Q \ \text{fi} \ \{ \psi \} \quad \text{(if)}
\]

Intuition:

- When a conditional is executed, either \( P \) or \( Q \) will be executed.
- For the postcondition \( \psi \) to be established, \textit{either} branch must terminate in a state satisfying \( \psi \).
While

\[
\{ \varphi \land g \} \quad P \quad \{ \varphi \} \\
\{ \varphi \} \text{ while } g \text{ do } P \text{ od } \{ \varphi \land \neg g \}
\]  

(loop)

Intuition:

- \( \varphi \) is a **loop invariant**. It must be both a pre- and postcondition of \( P \), so that sequences of \( P \)s can be run together.

- If the while loop terminates, \( g \) cannot hold.
Consequence

There is one more rule, called the rule of consequence, that we need to insert ordinary logical reasoning into our Hoare logic proofs:

\[
\varphi' \rightarrow \varphi \quad \{\varphi\} P \{\psi\} \quad \psi \rightarrow \psi' \quad \frac{}{\{\varphi'\} P \{\psi'\}} \quad \text{(cons)}
\]
Consequence

There is one more rule, called the *rule of consequence*, that we need to insert ordinary logical reasoning into our Hoare logic proofs:

\[
\varphi' \rightarrow \varphi \quad \{ \varphi \} \text{P} \quad \{ \psi \} \quad \psi \rightarrow \psi' \\
\{ \varphi' \} \text{P} \quad \{ \psi' \}
\]

(cons)

**Intuition:**

- Adding assertions to the precondition makes it more likely the postcondition will be reached
- Removing assertions from the postcondition makes it more likely the postcondition will be reached
- If you can reach the postcondition initially, then you can reach it in the more likely scenario
Back to Assignment Example

Example

\{ (y = 3) \} x := y \{ (x > 2) \}  \quad Problem!
Back to Assignment Example

Example

\{(y = 3)\} x := y \{(x > 2)\}   \textit{Problem!}

\{(y > 2)\} x := y \{(x > 2)\} (assign)
Example

\{(y = 3)\} x := y \{(x > 2)\}  \quad Problem!

\{(y = 3)\} x := y \{(x > 2)\} (assign, cons)
\{(y > 2)\} x := y \{(x > 2)\} (assign)
Factorial Example

Let’s verify the Factorial program using our Hoare rules:

\{N \geq 0\}

\quad i := 0;
\quad m := 1;

while \ i < \ N \ do

\quad i := i + 1;
\quad m := m \times i

od

\{m = N!\}

\{\varphi \wedge g\} \quad \{\psi\} \quad \{\varphi \wedge \neg g\} \quad \{\psi\}

\{\varphi\} \ if \ g \ then \ P \ else \ Q \ fi \ \{\psi\}

\{\varphi[x := e]\} \quad \{\varphi\}

\{\varphi \wedge g\} \quad \{\varphi\}

\{\varphi\} \ while \ g \ do \ P \ od \ \{\varphi \wedge \neg g\}

\{\varphi\} \quad \{\alpha\} \quad \{\alpha\} \quad \{\psi\}

\{\varphi\} \ P; \ Q \ \{\psi\}

\varphi' \Rightarrow \varphi \quad \{\varphi\} \ P \ \{\psi\} \quad \psi \Rightarrow \psi'

\{\varphi'\} \ P \ \{\psi'\}
Factorial Example

Let’s verify the Factorial program using our Hoare rules:

\[
\{ N \geq 0 \} \\
i := 0; \\
m := 1; \\
\text{while } i < N \text{ do} \\
i := i + 1; \\
m := m \times i \\
\text{od} \{ m = i! \land N \geq 0 \land i = N \} \\
\{ m = N! \} \\
\]
Factorial Example

Let’s verify the Factorial program using our Hoare rules:

\[
\{N \geq 0\}
\]

\[
i := 0;
\]

\[
m := 1;
\]

\[
\{m = i! \land N \geq 0\}
\]

while \(i < N\) do

\[
i := i + 1;
\]

\[
m := m \times i
\]

od \(\{m = i! \land N \geq 0 \land i = N\}\)

\(\{m = N!\}\)

\[
\{\varphi \land g\} P \{\psi\} \quad \{\varphi \land \neg g\} Q \{\psi\}
\]

\(\{\varphi\} \text{ if } g \text{ then } P \text{ else } Q \text{ fi } \{\psi\}\)

\[
\{\varphi[x := e]\} x := e \{\varphi\}
\]

\[\{\varphi \land g\} P \{\varphi\}\]

\(\{\varphi\} \text{ while } g \text{ do } P \text{ od } \{\varphi \land \neg g\}\)

\[
\{\varphi\} P \{\alpha\} \quad \{\alpha\} Q \{\psi\}
\]

\(\{\varphi\} P; Q \{\psi\}\)

\[\varphi' \Rightarrow \varphi \quad \{\varphi\} P \{\psi\}\]

\(\psi \Rightarrow \psi' \quad \{\varphi'\} P \{\psi'\}\)

\[\text{note: } (i+1)! = i! \times (i+1)\]
Factorial Example

Let’s verify the Factorial program using our Hoare rules:

\[
\{ N \geq 0 \} \\
\quad \quad i := 0; \\
\quad \quad m := 1; \\
\{ m = i! \land N \geq 0 \} \\
\text{while } i < N \text{ do} \\
\quad \quad i := i + 1; \\
\quad \quad m := m \times i \\
\{ m = i! \land N \geq 0 \} \\
\text{od} \{ m = i! \land N \geq 0 \land i = N \} \\
\{ m = N! \} \\
\]

\[
\{ \varphi \land g \} \quad P \quad \{ \psi \} \\
\{ \varphi \land \neg g \} \quad Q \quad \{ \psi \} \\
\{ \varphi \} \quad \text{if } g \text{ then } P \text{ else } Q \text{ fi } \{ \psi \} \\
\]

\[
\{ \varphi[x := e] \} \quad x := e \quad \{ \varphi \} \\
\]

\[
\{ \varphi \land g \} \quad P \quad \{ \varphi \} \\
\{ \varphi \} \quad \text{while } g \text{ do } P \text{ od } \{ \varphi \land \neg g \} \\
\]

\[
\{ \varphi \} \quad P \quad \{ \alpha \} \\
\{ \alpha \} \quad Q \quad \{ \psi \} \\
\{ \varphi \} \quad P ; Q \quad \{ \psi \} \\
\]

\[
\varphi' \Rightarrow \varphi \quad \{ \varphi \} \quad P \quad \{ \psi \} \\
\psi \Rightarrow \psi' \\
\quad \quad \{ \varphi' \} \quad P \quad \{ \psi' \} \\
\]

\[
\text{note: } (i + 1)! = i! \times (i + 1) \\
\]

\[
\]
Factorial Example

Let’s verify the Factorial program using our Hoare rules:

{\( N \geq 0 \)}

\[ i := 0; \]
\[ m := 1;\]

{\( m = i! \land N \geq 0 \)}

while \( i < N \) do \{ \( m = i! \land N \geq 0 \land i < N \) \}

\[ i := i + 1; \]

\[ m := m \times i \]

{\( m = i! \land N \geq 0 \)}

od \{ \( m = i! \land N \geq 0 \land i = N \) \}

{\( m = N! \)}
Factorial Example

Let's verify the Factorial program using our Hoare rules:

\[
\begin{align*}
\{N \geq 0\} & \quad i := 0; \\
\{m = i! \land N \geq 0\} & \quad m := 1; \\
\text{while } i < N \text{ do } \{m = i! \land N \geq 0 \land i < N\} & \\
\quad i := i + 1; \\
\quad \{m \times i = i! \land N \geq 0\} & \\
\quad m := m \times i \\
\text{od } \{m = i! \land N \geq 0 \land i = N\} & \\
\{m = N!\} &
\end{align*}
\]

\[
\begin{align*}
\{\varphi \land g\} & \quad P \{\psi\} \quad \{\varphi \land \neg g\} & \quad Q \{\psi\} \\
\{\varphi\} & \quad \text{if } g \text{ then } P \text{ else } Q \text{ fi } \{\psi\} \\
\{\varphi[x := e]\} & \quad x := e \{\varphi\} \\
\{\varphi \land g\} & \quad P \{\varphi\} \\
\{\varphi\} & \quad \text{while } g \text{ do } P \text{ od } \{\varphi \land \neg g\} \\
\{\varphi\} & \quad P \{\alpha\} \quad \{\alpha\} \quad Q \{\psi\} \\
\{\varphi\} & \quad P; Q \{\psi\} \\
\varphi' \Rightarrow \varphi & \quad \{\varphi\} \quad P \{\psi\} \quad \psi \Rightarrow \psi' \\
\{\varphi'\} & \quad P \{\psi'\}
\end{align*}
\]

\(\text{note: } (i + 1)! = i! \times (i + 1)\)
Factorial Example

Let’s verify the Factorial program using our Hoare rules:

\[
\{N \geq 0\} \\
\text{ } \\
\text{ } \\
\{m = i! \land N \geq 0\} \\
\text{while } i < N \text{ do } \{m = i! \land N \geq 0 \land i < N\} \\
\text{ } \\
\text{ } \\
\{m \times (i + 1) = (i + 1)! \land N \geq 0\} \\
i := i + 1; \\
\{m \times i = i! \land N \geq 0\} \\
m := m \times i \\
\{m = i! \land N \geq 0\} \\
\text{od } \{m = i! \land N \geq 0 \land i = N\} \\
\{m = N!\}
\]
Factorial Example

Let’s verify the Factorial program using our Hoare rules:

\[
\begin{align*}
\{N \geq 0\} & \\
& \quad i := 0; \\
& \quad m := 1; \\
\{m = i! \land N \geq 0\} & \\
\text{while } i < N \text{ do } \{m = i! \land N \geq 0 \land i < N\} & \\
& \quad \{m \times (i + 1) = (i + 1)! \land N \geq 0\} \\
& \quad i := i + 1; \\
& \quad \{m \times i = i! \land N \geq 0\} \\
& \quad m := m \times i \\
\text{od } \{m = i! \land N \geq 0 \land i = N\} \\
\{m = N!\}
\end{align*}
\]

note: \((i + 1)! = i! \times (i + 1)\)
Factorial Example

Let’s verify the Factorial program using our Hoare rules:

{ \( N \geq 0 \) }

\[
i := 0;
\]
\[
m := 1; \{ m = i! \land N \geq 0 \}
\]

\[
\text{while } i < N \text{ do } \{ m = i! \land N \geq 0 \land i < N \}
\]
\[
\{ m \times (i+1) = (i+1)! \land N \geq 0 \}
\]
\[
i := i + 1;
\]
\[
\{ m \times i = i! \land N \geq 0 \}
\]
\[
m := m \times i
\]
\[
\{ m = i! \land N \geq 0 \}
\]
\[\text{od } \{ m = i! \land N \geq 0 \land i = N \}
\]
\[
\{ m = N! \}
\]

Note: \((i + 1)! = i! \times (i + 1)\)
Factorial Example

Let’s verify the Factorial program using our Hoare rules:

\[
\begin{align*}
\{N \geq 0\} & \quad i := 0; \quad \{1 = i! \land N \geq 0\} \quad m := 1; \quad \{m = i! \land N \geq 0\} \\
\{m = i! \land N \geq 0\} & \quad \text{while } i < N \text{ do } \{m = i! \land N \geq 0 \land i < N\} \\
& \quad \{m \times (i + 1) = (i + 1)! \land N \geq 0\} \\
& \quad i := i + 1; \\
& \quad \{m \times i = i! \land N \geq 0\} \\
& \quad m := m \times i \\
& \quad \{m = i! \land N \geq 0\} \\
\od & \quad \{m = i! \land N \geq 0 \land i = N\} \\
\{m = N!\} \\ \\
\end{align*}
\]

note: \((i + 1)! = i! \times (i + 1)\)
Factorial Example

Let’s verify the Factorial program using our Hoare rules:

\[
\{N \geq 0\} \\
\quad i := 0; \{1 = i! \land N \geq 0\} \\
\{1 = i! \land N \geq 0\} \ m := 1; \{m = i! \land N \geq 0\} \\
\{m = i! \land N \geq 0\} \text{ while } i < N \text{ do } \{m = i! \land N \geq 0 \land i < N\} \\
\quad \{m \times (i + 1) = (i + 1)! \land N \geq 0\} \\
\quad i := i + 1; \\
\quad \{m \times i = i! \land N \geq 0\} \\
\quad m := m \times i \\
\quad \{m = i! \land N \geq 0\} \\
\text{ od } \{m = i! \land N \geq 0 \land i = N\} \\
\{m = N!\}
\]

\[
\begin{align*}
\{\varphi \land g\} & \text{ } P \{\psi\} & \{\varphi \land \neg g\} & \text{ } Q \{\psi\} \\
\{\varphi\} & \text{ if } g \text{ then } P \text{ else } Q \text{ fi } \{\psi\} \\
\{\varphi[x := e]\} & \times := e \{\varphi\} \\
\{\varphi \land g\} & \text{ } P \{\varphi\} \\
\{\varphi\} & \text{ while } g \text{ do } P \text{ od } \{\varphi \land \neg g\} \\
\{\varphi\} & \text{ } P \{\alpha\} & \{\alpha\} & \text{ } Q \{\psi\} \\
\{\varphi\} & \text{ } P; Q \{\psi\} \\
\varphi' & \Rightarrow \varphi & \{\varphi\} & \text{ } P \{\psi\} & \psi \Rightarrow \psi' \\
\{\varphi'\} & \text{ } P \{\psi'\}
\end{align*}
\]

note: \((i + 1)! = i! \times (i + 1)\)
Factorial Example

Let’s verify the Factorial program using our Hoare rules:

\[
\begin{align*}
\{ N \geq 0 \} \\
\{ 1 = 0! \land N \geq 0 \} i := 0; \{ 1 = i! \land N \geq 0 \} \\
\{ 1 = i! \land N \geq 0 \} m := 1; \{ m = i! \land N \geq 0 \} \\
\{ m = i! \land N \geq 0 \} \\
\text{while } i < N \text{ do } \{ m = i! \land N \geq 0 \land i < N \} \\
\{ m \times (i + 1) = (i + 1)! \land N \geq 0 \} \\
i := i + 1; \\
\{ m \times i = i! \land N \geq 0 \} \\
m := m \times i; \\
\{ m = i! \land N \geq 0 \} \\
\text{od } \{ m = i! \land N \geq 0 \land i = N \} \\
\{ m = N! \} \\
\end{align*}
\]

\[
\begin{align*}
\{ \varphi \land g \} P \{ \psi \} & \quad \{ \varphi \land \neg g \} Q \{ \psi \} \\
\{ \varphi \} \text{ if } g \text{ then } P \text{ else } Q \text{ fi } \{ \psi \} \\
\{ [x := e] \} x := e \{ \varphi \} \\
\{ \varphi \land g \} P \{ \varphi \} \\
\{ \varphi \} \text{ while } g \text{ do } P \text{ od } \{ \varphi \land \neg g \} \\
\{ \varphi \} P \{ \alpha \} \quad \{ \alpha \} Q \{ \psi \} \\
\{ \varphi \} P; Q \{ \psi \} \\
\end{align*}
\]

\[
\begin{align*}
\varphi' & \Rightarrow \varphi \\
\{ \varphi \} P \{ \psi \} & \quad \psi \Rightarrow \psi' \\
\{ \varphi' \} P \{ \psi' \} \\
\end{align*}
\]

note: \((i + 1)! = i! \times (i + 1)\)
Practice Exercise

Example

\[
m := 1;
n := 1;
i := 1;
\]
\[
\text{while } i < N \text{ do}
\]
\[
\quad t := m;
\quad m := n;
\quad n := m + t;
\quad i := i + 1
\]
\[
\text{od}
\]

What does this program compute?

What is a valid Hoare triple \{ϕ\} P \{ψ\} of this program?

Prove using the inference rules and consequence axiom that this Hoare triple is valid.
Practice Exercise

Example

\[m := 1;\]
\[n := 1;\]
\[i := 1;\]
while \( i < N \) do
\[t := m;\]
\[m := n;\]
\[n := m + t;\]
\[i := i + 1\]
od

- What does this \( \mathcal{L} \) program \( P \) compute?
- What is a valid Hoare triple \( \{ \varphi \} P \{ \psi \} \) of this program?
- Prove using the inference rules and consequence axiom that this Hoare triple is valid.
Summary

- $\mathcal{L}$: A simple imperative programming language
- Hoare triples (SYNTAX)
- Hoare logic (PROOF)
- Semantics for Hoare logic
Semantics

Nope. That’s a topic for another lecture.