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

![Sir Tony Hoare](image)
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**

**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: \(+, \times, \ldots\)
- 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**: +, *, \ldots
- **Predicate symbols**: <, ≤, ≥, |, \ldots

- **An (arithmetic) expression** is a term over this vocabulary.
- **A boolean expression** is a predicate formula over this vocabulary.
The language $\mathcal{L}$

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

\[
\begin{align*}
i & := 0; \\
m & := 1; \\
\text{while } i < N \text{ do} \\
  & \quad i := i + 1; \\
  & \quad m := m \ast i \\
\text{od}
\end{align*}
\]
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 are going to define what’s called 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 $P$ is a program. This triple states that if the program $P$ terminates and it successfully evaluates from a starting state satisfying the **precondition** $\varphi$, then the result state will satisfy the **postcondition** $\psi$. 
Hoare triple: Examples

Example

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

\{(x > 0)\} \ y := 0 - x \ {y < 0 \land x \neq y\}
Example

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

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

Example

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

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

\{(x > 0)\} y := 0 - x \{ (y < 0) \land (x \neq y) \}
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 \\
& \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

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

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

Example

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

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)\}

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

Example

\[
\begin{align*}
&\{(y = 0)\} x := y \ (x = 0) \\
&\{(y = y)\} x := y \ (x = y) \\
&\{(1 < 2)\} x := 1 \ (x < 2) \\
&\{(y = 3)\} x := y \ (x > 2)
\end{align*}
\]
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)\}  \hspace{1cm} \textit{Problem!}
Sequence

\[
\{\varphi\} \enspace P \{\psi\} \quad \{\psi\} \enspace Q \{\rho\} \\
\{\varphi\} \enspace P; \enspace Q \{\rho\} \tag{seq}
\]

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

Example

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

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

Example

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

Example

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

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

(if)

Intuition:

- When a conditional is executed, either \( P \) or \( Q \) will be executed.
- If \( \psi \) is a postcondition of the conditional, then it must be a postcondition of both branches.
- Likewise, if \( \varphi \) is a precondition of the conditional, then it must be a precondition of both branches.
- Which branch gets executed depends on \( g \), so we can assume \( g \) to be a precondition of \( P \) and \( \neg g \) to be a precondition of \( Q \).
While

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

(loop)

Intuition:

- \( \varphi \) is a \textbf{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:

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

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

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

\[
\{\varphi'\} \; P \; \{\psi'\}
\]

(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.
Example

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

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

Problem!

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

Example

\{(y = 3)\} \times \leftarrow y \{(x > 2)\}

Problem!

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

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

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

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

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

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

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

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

\[
\{\varphi'\} \quad P \quad \{\psi'\}
\]
Factorial Example

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

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

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

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

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

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

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

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

\[
\begin{align*}
{\varphi} & \land g \quad P \quad {\psi} \quad {\varphi} \land \lnot g \quad Q \quad {\psi} \\
{\varphi} & \land g \quad P \quad {\psi} \\
{\varphi} & \land \lnot g \quad Q \quad {\psi} \\
{\varphi} & \land g \quad P \quad {\psi} \\
{\varphi} & \land \lnot g \quad Q \quad {\psi} \\
{\varphi} & \Rightarrow \varphi \quad {\varphi} \quad P \quad {\psi} \\
{\varphi} & \Rightarrow \varphi \quad \psi \Rightarrow \psi' \quad {\varphi}' \quad P \quad {\psi}'
\end{align*}
\]
Factorial Example

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

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

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

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

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

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

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

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

\[
\begin{align*}
\{ \varphi \land g \} & \quad P \{ \psi \} \quad \{ \varphi \land \neg g \} \quad Q \{ \psi \} \\
\{ \varphi \} & \quad \text{if } g \text{ then } P \text{ else } Q \text{ fi } \{ \psi \}
\end{align*}
\]

\[
\begin{align*}
\{ \varphi[x := e]\} & \quad x := e \{ \varphi \}
\end{align*}
\]

\[
\begin{align*}
\{ \varphi \land g \} & \quad P \{ \varphi \} \\
\{ \varphi \} & \quad \text{while } g \text{ do } P \text{ od } \{ \varphi \land \neg g \}
\end{align*}
\]

\[
\begin{align*}
\{ \varphi \} & \quad P \{ \alpha \} \quad \{ \alpha \} \quad Q \{ \psi \} \\
\{ \varphi \} & \quad P; Q \{ \psi \}
\end{align*}
\]

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

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

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

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

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

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

\{N \geq 0\}

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

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

while \(i < N\) 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\}

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

\{m = N!\}

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

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

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

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

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

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

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

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

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

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

Example

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

What does this program compute?

What is a valid Hoare triple \{\phi\} P \{\psi\} of this program?

Prove using the inference rules and consequence axiom that this Hoare triple is valid.
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 \( L \) 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
Recall

If $R$ and $S$ are binary relations, then the relational composition of $R$ and $S$, $R; S$ is the relation:

$$R; S := \{(a, c) : \exists b \text{ such that } (a, b) \in R \text{ and } (b, c) \in S\}$$

If $R \subseteq A \times B$ is a relation, and $X \subseteq A$, then the image of $X$ under $R$, $R(X)$ is the subset of $B$ defined as:

$$R(X) := \{b \in B : \exists a \text{ in } X \text{ such that } (a, b) \in R\}.$$
Informal semantics

Hoare logic gives a proof of $\{\varphi\} P \{\psi\}$, that is: $\vdash \{\varphi\} P \{\psi\}$ (axiomatic semantics)

How do we determine when $\{\varphi\} P \{\psi\}$ is valid, that is: $\models \{\varphi\} P \{\psi\}$?
Informal semantics

Hoare logic gives a proof of $\{\varphi\} P \{\psi\}$, that is: $\vdash \{\varphi\} P \{\psi\}$ (axiomatic semantics)

How do we determine when $\{\varphi\} P \{\psi\}$ is valid, that is: $\models \{\varphi\} P \{\psi\}$?

If $\varphi$ holds in a state of some computational model then $\psi$ holds in the state reached after a successful execution of $P$. 
Informal semantics: Programs

What is a program?
Informal semantics: Programs

What is a program?

A function mapping system states to system states
What is a program?

A partial function mapping system states to system states
Informal semantics: Programs

What is a program?

A relation between system states
Informal semantics: States

What is a state of a computational model?

Two approaches:

Concrete: from a physical perspective
States are memory configurations, register contents, etc.
Store of variables and the values associated with them

Abstract: from a mathematical perspective
The pre-/postcondition predicates hold in a state ⇒ States are logical interpretations (Model + Environment)
There is only one model of interest: standard interpretations of arithmetical symbols ⇒ States are fully determined by environments ⇒ States are functions that map variables to values
Informal semantics: States

What is a state of a computational model?

Two approaches:

- Concrete: from a physical perspective
  - States are memory configurations, register contents, etc.
  - Store of variables and the values associated with them

- Abstract: from a mathematical perspective
  - Pre-/postcondition predicates hold in a state
  - States are logical interpretations (Model + Environment)
  - There is only one model of interest: standard interpretations of arithmetical symbols
  - States are fully determined by environments
  - States are functions that map variables to values
Informal semantics: States

What is a state of a computational model?

Two approaches:

- Concrete: from a physical perspective
  - States are memory configurations, register contents, etc.
  - Store of variables and the values associated with them
- Abstract: from a mathematical perspective
  - The pre-/postcondition predicates hold in a state
  - States are logical interpretations (Model + Environment)
  - There is only one model of interest: standard interpretations of arithmetical symbols
  - States are fully determined by environments
  - States are functions that map variables to values
Informal semantics: States

What is a state of a computational model?

Two approaches:

- **Concrete**: from a physical perspective
  - States are memory configurations, register contents, etc.
  - Store of variables and the values associated with them

- **Abstract**: from a mathematical perspective
  - The pre-/postcondition predicates *hold* in a state
  - States are **logical interpretations** (Model + Environment)
Informal semantics: States

What is a state of a computational model?

Two approaches:

- **Concrete:** from a physical perspective
  - States are memory configurations, register contents, etc.
  - Store of variables and the values associated with them

- **Abstract:** from a mathematical perspective
  - The pre-/postcondition predicates *hold* in a state
  - States are **logical interpretations** *(Model + Environment)*
  - There is only one model of interest: standard interpretations of arithmetical symbols
Informal semantics: States

What is a state of a computational model?

Two approaches:

- **Concrete**: from a physical perspective
  - States are memory configurations, register contents, etc.
  - Store of variables and the values associated with them

- **Abstract**: from a mathematical perspective
  - The pre-/postcondition predicates *hold* in a state
  - States are **logical interpretations** (Model + Environment)
    - There is only one model of interest: standard interpretations of arithmetical symbols
  - States are fully determined by **environments**
  - States are functions that map variables to values
Informal semantics: States

State space ($\text{Env}$)

- $x \leftarrow 0$
- $y \leftarrow 0$
- $z \leftarrow 0$
- $x \leftarrow 3$
- $y \leftarrow 2$
- $z \leftarrow 1$
- $x \leftarrow 1$
- $y \leftarrow 1$
- $z \leftarrow 1$
- $x \leftarrow 2$
- $y \leftarrow 2$
- $z \leftarrow 2$
- $x \leftarrow 0$
- $y \leftarrow 1$
- $z \leftarrow 2$
- $x \leftarrow 0$
- $y \leftarrow 1$
- $z \leftarrow 0$
Informal semantics: States and Programs

State space ($\text{ENV}$)

- $x \leftarrow 0$
  - $y \leftarrow 0$
  - $z \leftarrow 0$
- $x \leftarrow 1$
  - $y \leftarrow 1$
  - $z \leftarrow 1$
- $x \leftarrow 0$
  - $y \leftarrow 1$
  - $z \leftarrow 2$
- $x \leftarrow 1$
  - $y \leftarrow 1$
  - $z \leftarrow 2$
- $x \leftarrow 0$
  - $y \leftarrow 1$
  - $z \leftarrow 2$
- $x \leftarrow 2$
  - $y \leftarrow 1$
  - $z \leftarrow 1$
- $x \leftarrow 3$
  - $y \leftarrow 2$
  - $z \leftarrow 1$
Informal semantics: States and Programs
An **environment** or **state** is a function from variables to numeric values. We denote by $\text{Env}$ the set of all environments.

**NB**

An environment, $\eta$, assigns a numeric value $[e]^{\eta}$ to all expressions $e$, and a boolean value $[b]^{\eta}$ to all boolean expressions $b$. 
Semantics for $\mathcal{L}$

An **environment** or **state** is a function from variables to numeric values. We denote by $\text{Env}$ the set of all environments.

**NB**

An environment, $\eta$, assigns a numeric value $[e]^\eta$ to all expressions $e$, and a boolean value $[b]^\eta$ to all boolean expressions $b$.

Given a program $P$ of $\mathcal{L}$, we define $[P]$ to be a **binary relation** on $\text{Env}$ in the following manner...
Assignment

\[(\eta, \eta') \in [x := e] \text{ if, and only if } \eta' = \eta[x \mapsto [e]^{\eta}]\]
Assignment: \([z := 2]\)

State space (\(E_{\text{NV}}\))

- \(x \leftarrow 0\)
- \(y \leftarrow 0\)
- \(z \leftarrow 0\)
Sequencing

\[[P; Q] = [P]; [Q]\]

where, on the RHS, ; is relational composition.
Conditional, first attempt

\[
[\text{if } b \text{ then } P \text{ else } Q \text{ fi}] = \begin{cases} 
[P] & \text{if } [b]^{\eta} = \text{true} \\
[Q] & \text{otherwise.}
\end{cases}
\]
Detour: Predicates as programs

A boolean expression $b$ defines a subset (or unary relation) of $\text{Env}$:

$$\langle b \rangle = \{ \eta : [b]^{\eta} = \text{true} \}$$

This can be extended to a binary relation (i.e. a program):

$$[b] = \{ (\eta, \eta) : \eta \in \langle b \rangle \}$$
A boolean expression \( b \) defines a subset (or unary relation) of \( \text{Env} \):

\[
\langle b \rangle = \{ \eta : \lbrack b \rbrack^\eta = \text{true} \}
\]

This can be extended to a binary relation (i.e. a program):

\[
\lbrack b \rbrack = \{ (\eta, \eta) : \eta \in \langle b \rangle \}
\]

Intuitively, \( b \) corresponds to the program

\[
\text{if } b \text{ then skip else } \bot \text{ fi}
\]
Conditional, better attempt

\[ [\text{if } b \text{ then } P \text{ else } Q \text{ fi}] = [b; P] \cup [\neg b; Q] \]
While

while $b$ do $P$ od

- Do 0 or more executions of $P$ while $b$ holds
- Terminate when $b$ does not hold
While

\[ \text{while } b \text{ do } P \text{ od} \]

- Do 0 or more executions of \((b; P)\)
- Terminate with an execution of \(\neg b\)
While

while $b$ do $P$ od

- Do 0 or more executions of $(b; P)$
- Terminate with an execution of $\neg b$

How to do “0 or more” executions of $(b; P)$?
Transitive closure

Given a binary relation $R \subseteq E \times E$, the transitive closure of $R$, $R^*$ is defined to be the limit of the sequence

$$R^0 \cup R^1 \cup R^2 \ldots$$

where

- $R^0 = \Delta$, the diagonal relation
- $R^{n+1} = R^n \cdot R$

**NB**

- $R^*$ is the smallest transitive relation which contains $R$
- Related to the Kleene star operation seen in languages: $\Sigma^*$
Transitive closure

Given a binary relation $R \subseteq E \times E$, the transitive closure of $R$, $R^*$ is defined to be the limit of the sequence

$$R^0 \cup R^1 \cup R^2 \ldots$$

where

- $R^0 = \Delta$, the diagonal relation
- $R^{n+1} = R^n \cup R$

**NB**

- $R^*$ is the smallest transitive relation which contains $R$
- Related to the Kleene star operation seen in languages: $\Sigma^*$

Technically, $R^*$ is the least-fixed point of $f(X) = X \cup X; R$
While

\[ [\text{while } b \text{ do } P \text{ od}] = [b; P]^* ; [\neg b] \]

- Do 0 or more executions of \((b; P)\)
- Conclude with an execution of \(\neg b\)
Validity

A Hoare triple is valid, written $\vdash \{ \varphi \} P \{ \psi \}$ if

$$[P](\langle \varphi \rangle) \subseteq \langle \psi \rangle.$$

That is, the relational image under $[P]$ of the set of states where $\varphi$ holds is contained in the set of states where $\psi$ holds.
Validity
Validity

\[ \langle \varphi \rangle \]
Validity

⟨ϕ⟩

⟨ψ⟩
Validity

\[ \langle \varphi \rangle \rightarrow [P] \rightarrow \langle \psi \rangle \]

\[ [P] \langle \varphi \rangle \]

\[ [P]([\langle \varphi \rangle]) \]
Soundness of Hoare Logic

Hoare Logic is **sound** with respect to the semantics given. That is,

**Theorem**

\[
\text{If } \vdash \{ \varphi \} P \{ \psi \} \text{ then } \models \{ \varphi \} P \{ \psi \}
\]
Summary

- Set theory revisited
- Soundness of Hoare Logic
- Completeness of Hoare Logic
Summary

- Set theory revisited
- Soundness of Hoare Logic
- Completeness of Hoare Logic
Lemma

For any binary relations $R, S \subseteq X \times Y$ and subsets $A, B \subseteq X$:

(a) If $A \subseteq B$ then $R(A) \subseteq R(B)$

(b) $R(A) \cup S(A) = (R \cup S)(A)$

(c) $R(S(A)) = (S; R)(A)$
Some results on relational images

Lemma

For any binary relations $R, S \subseteq X \times Y$ and subsets $A, B \subseteq X$:

(a) If $A \subseteq B$ then $R(A) \subseteq R(B)$

(b) $R(A) \cup S(A) = (R \cup S)(A)$

(c) $R(S(A)) = (S; R)(A)$

Proof (a):
Some results on relational images

Lemma

For any binary relations $R, S \subseteq X \times Y$ and subsets $A, B \subseteq X$:

(a) If $A \subseteq B$ then $R(A) \subseteq R(B)$

(b) $R(A) \cup S(A) = (R \cup S)(A)$

(c) $R(S(A)) = (S; R)(A)$

Proof (a):

$y \in R(A) \iff \exists x \in A \text{ such that } (x, y) \in R$

$\Rightarrow \exists x \in B \text{ such that } (x, y) \in R$

$\iff y \in R(B)$
Some results on relational images

Lemma

For any binary relations $R, S \subseteq X \times Y$ and subsets $A, B \subseteq X$:

(a) If $A \subseteq B$ then $R(A) \subseteq R(B)$

(b) $R(A) \cup S(A) = (R \cup S)(A)$

(c) $R(S(A)) = (S; R)(A)$

Proof (b):
Some results on relational images

Lemma

For any binary relations $R, S \subseteq X \times Y$ and subsets $A, B \subseteq X$:

(a) If $A \subseteq B$ then $R(A) \subseteq R(B)$

(b) $R(A) \cup S(A) = (R \cup S)(A)$

(c) $R(S(A)) = (S; R)(A)$

Proof (b):

$y \in R(A) \cup S(A) \iff y \in R(A) \text{ or } y \in S(A)$

$\iff \exists x \in A \text{ s.t. } (x, y) \in R \text{ or } \exists x \in A \text{ s.t. } (x, y) \in S$

$\iff \exists x \in A \text{ s.t. } (x, y) \in R \text{ or } (x, y) \in S$

$\iff \exists x \in A \text{ s.t. } (x, y) \in (R \cup S)$

$\iff y \in (R \cup S)(A)$
Some results on relational images

**Lemma**

For any binary relations $R, S \subseteq X \times Y$ and subsets $A, B \subseteq X$:

(a) If $A \subseteq B$ then $R(A) \subseteq R(B)$

(b) $R(A) \cup S(A) = (R \cup S)(A)$

(c) $R(S(A)) = (S; R)(A)$

Proof (c):
Some results on relational images

Lemma

For any binary relations $R, S \subseteq X \times Y$ and subsets $A, B \subseteq X$:

(a) If $A \subseteq B$ then $R(A) \subseteq R(B)$
(b) $R(A) \cup S(A) = (R \cup S)(A)$
(c) $R(S(A)) = (S; R)(A)$

Proof (c):

$z \in R(S(A)) \iff \exists y \in S(A) \text{ s.t. } (y, z) \in R$
$\iff \exists x \in A, y \in S(A) \text{ s.t. } (x, y) \in S \text{ and } (y, z) \in R$
$\iff \exists x \in A \text{ s.t. } (x, z) \in (S; R)$
$\iff z \in (S; R)(A)$
Some results on relational images

**Corollary**

If $R(A) \subseteq A$ then $R^*(A) \subseteq A$
Corollary

If \( R(A) \subseteq A \) then \( R^*(A) \subseteq A \)

Proof:
Corollary

If \( R(A) \subseteq A \) then \( R^*(A) \subseteq A \)

Proof:

\[
R(A) \subseteq A \quad \Rightarrow \quad R^{i+1}(A) = R^i(R(A)) \subseteq R^i(A) \\
\Rightarrow \quad R^{i+1}(A) \subseteq R(A) \subseteq A \\
\text{So } R^*(A) = \left( \bigcup_{i=0}^{\infty} R^i \right)(A) \\
= \bigcup_{i=0}^{\infty} R^i(A) \\
\subseteq A
\]
Summary

- Set theory revisited
- Soundness of Hoare Logic
- Completeness of Hoare Logic
Soundness of Hoare Logic

**Theorem**

\[ \text{If } \vdash \{ \varphi \} P \{ \psi \} \text{ then } \models \{ \varphi \} P \{ \psi \} \]
Soundness of Hoare Logic

**Theorem**

If $\vdash \{ \varphi \} P \{ \psi \}$ then $\models \{ \varphi \} P \{ \psi \}$

**Proof:**
Soundness of Hoare Logic

Theorem

\[ \text{If } \vdash \{ \varphi \} P \{ \psi \} \text{ then } \models \{ \varphi \} P \{ \psi \} \]

Proof:
By induction on the structure of the proof.
Base case: Assignment rule

\[
\{\varphi[e/x]\} x := e \{\varphi\} \quad \text{(ass)}
\]
Base case: Assignment rule

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

Need to show \(\{\varphi[e/x]\} x := e \{\varphi\}\) is always valid. That is,

\[
[x := e](\langle \varphi[e/x] \rangle) \subseteq \langle \varphi \rangle.
\]
Base case: Assignment rule

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

Need to show \(\{\varphi[e/x]\} x := e \{\varphi\}\) is always valid. That is,

\[
[x := e](\langle \varphi[e/x] \rangle) \subseteq \langle \varphi \rangle.
\]

Observation: \(\llbracket \varphi[e/x] \rrbracket^\eta = \llbracket \varphi \rrbracket^\eta'\) where \(\eta' = \eta[x \mapsto \llbracket e \rrbracket^\eta]\)
**Base case: Assignment rule**

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

Need to show \(\{\varphi[e/x]\} \ x := e \{\varphi\}\) is always valid. That is,

\[
[x := e](\langle \varphi[e/x] \rangle) \subseteq \langle \varphi \rangle.
\]

Observation: \(\llbracket \varphi[e/x] \rrbracket^\eta = \llbracket \varphi \rrbracket^\eta'\) where \(\eta' = \eta[x \mapsto \llbracket e \rrbracket^\eta]\)

So if \(\eta \in \langle \varphi[e/x] \rangle\) then \(\eta' \in \langle \varphi \rangle\)
Base case: Assignment rule

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

Need to show \( \{ \varphi[e/x] \} x := e \{ \varphi \} \) is always valid. That is,

\[
[x := e](\langle \varphi[e/x] \rangle) \subseteq \langle \varphi \rangle.
\]

Observation: \( [\varphi[e/x]]^\eta = [\varphi]^\eta' \) where \( \eta' = \eta[x \mapsto [e]^\eta] \)

So if \( \eta \in \langle \varphi[e/x] \rangle \) then \( \eta' \in \langle \varphi \rangle \)

Recall: \( (\eta, \eta'') \in [x := e] \) if and only if \( \eta'' = \eta[x \mapsto [e]^\eta] \),
Base case: Assignment rule

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

(ass)

Need to show \(\{\varphi[e/x]\} x := e \{\varphi\}\) is always valid. That is,

\[
[x := e] (\langle \varphi[e/x] \rangle) \subseteq \langle \varphi \rangle.
\]

Observation: \(\{\varphi[e/x]\}^\eta = \{\varphi\}^{\eta'}\) where \(\eta' = \eta[x \mapsto [e]^\eta]\)

So if \(\eta \in \langle \varphi[e/x] \rangle\) then \(\eta' \in \langle \varphi \rangle\)

Recall: \((\eta, \eta'') \in [x := e]\) if and only if \(\eta'' = \eta[x \mapsto [e]^\eta]\),

So \([x := e](\eta) \in \langle \varphi \rangle\) for all \(\eta \in \langle \varphi[e/x] \rangle\)
Base case: Assignment rule

\[
\{\phi[e/x]\} x := e \{\phi\} \quad \text{(ass)}
\]

Need to show \(\{\phi[e/x]\} x := e \{\phi\}\) is always valid. That is,

\[
[x := e](\langle \phi[e/x] \rangle) \subseteq \langle \phi \rangle.
\]

Observation: \([\phi[e/x]]^\eta = [\phi]^\eta'\) where \(\eta' = \eta[x \mapsto [e]^\eta]\)

So if \(\eta \in \langle \phi[e/x] \rangle\) then \(\eta' \in \langle \phi \rangle\)

Recall: \((\eta, \eta'') \in [x := e]\) if and only if \(\eta'' = \eta[x \mapsto [e]^\eta]\),

So \([x := e](\eta) \in \langle \phi \rangle\) for all \(\eta \in \langle \phi[e/x] \rangle\)

So \([x := e](\langle \phi[e/x] \rangle) \subseteq \langle \phi \rangle\)
Inductive case 1: Sequence rule

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

\[
\{\varphi\} \quad P; \quad Q \quad \{\rho\}
\] (seq)

Assume \( \{\varphi\} P \{\psi\} \) and \( \{\psi\} Q \{\rho\} \) are valid. Need to show that \( \{\varphi\} P; \quad Q \quad \{\rho\} \) is valid.

Recall: \( [\{P\}; \{Q\}] = [\{P\}] ; [\{Q\}] \)

So: \( [\{P\}; \{Q\}] (\langle \varphi \rangle) = [\{Q\}] ([\{P\}] (\langle \varphi \rangle)) \) (see Lemma 1(c))

By IH: \( [\{P\}] (\langle \varphi \rangle) \subseteq \langle \psi \rangle \) and \( [\{Q\}] (\langle \psi \rangle) \subseteq \langle \rho \rangle \)

So: \( [\{Q\}] ([\{P\}] (\langle \varphi \rangle)) \subseteq [\{Q\}] (\langle \psi \rangle) \subseteq \langle \rho \rangle \) (see Lemma 1(a))
Inductive case 1: Sequence rule

\[
\begin{array}{c}
\{\varphi\} P \{\psi\} \\
\{\psi\} Q \{\rho\}
\end{array}
\quad\frac{}{\{\varphi\} P; Q \{\rho\}} \quad \text{(seq)}
\]

Assume \(\{\varphi\} P \{\psi\}\) and \(\{\psi\} Q \{\rho\}\) are valid. Need to show that \(\{\varphi\} P; Q \{\rho\}\) is valid.
Inductive case 1: Sequence rule

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

\[ \{ \varphi \} P; Q \{ \rho \} \] (seq)

Assume \( \{ \varphi \} P \{ \psi \} \) and \( \{ \psi \} Q \{ \rho \} \) are valid. Need to show that \( \{ \varphi \} P; Q \{ \rho \} \) is valid.

Recall: \([P; Q] = [P]; [Q]\)
Inductive case 1: Sequence rule

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

(seq)

Assume \( \{ \varphi \} \ P \ {\psi} \) and \( \{ {\psi} \} \ Q \ {\rho} \) are valid. Need to show that \( \{ \varphi \} \ P; Q \ {\rho} \) is valid.

Recall: \( [P; Q] = [P]; [Q] \)

So: \( [P; Q](\langle \varphi \rangle) = [Q]( [P](\langle \varphi \rangle)) \) (see Lemma 1(c))
Inductive case 1: Sequence rule

\[
\begin{array}{c}
\{\varphi\} P \{\psi\} \quad \{\psi\} Q \{\rho\} \\
\hline
\{\varphi\} P; Q \{\rho\}
\end{array}
\]  
(\text{seq})

Assume \(\{\varphi\} P \{\psi\}\) and \(\{\psi\} Q \{\rho\}\) are valid. Need to show that \(\{\varphi\} P; Q \{\rho\}\) is valid.

Recall: \([P; Q] = [P]; [Q]\)

So: \([P; Q](\langle \varphi \rangle) = [Q][P](\langle \varphi \rangle)\)  
(see Lemma 1(c))

By IH: \([P](\langle \varphi \rangle) \subseteq \langle \psi \rangle\) and \([Q](\langle \psi \rangle) \subseteq \langle \rho \rangle\)
Inductive case 1: Sequence rule

\[
\begin{array}{cc}
\{\varphi\} & P & \{\psi\} \\
\{\psi\} & Q & \{\rho\} \\
\hline
\{\varphi\} & P; Q & \{\rho\}
\end{array}
\]

(seq)

Assume \(\{\varphi\} P \{\psi\}\) and \(\{\psi\} Q \{\rho\}\) are valid. Need to show that \(\{\varphi\} P; Q \{\rho\}\) is valid.

Recall: \([P; Q] = [P]; [Q]\)

So: \([P; Q](\langle \varphi \rangle) = [Q]([P](\langle \varphi \rangle))\) (see Lemma 1(c))

By IH: \([P](\langle \varphi \rangle) \subseteq \langle \psi \rangle\) and \([Q](\langle \psi \rangle) \subseteq \langle \rho \rangle\)

So: \([Q]([P](\langle \varphi \rangle)) \subseteq [Q](\langle \psi \rangle) \subseteq \langle \rho \rangle\) (see Lemma 1(a))
Two more useful results

**Lemma**

*For* $R \subseteq \text{Env} \times \text{Env}$, predicates $\varphi$ and $\psi$, and $X \subseteq \text{Env}$:

| (a) | $\llbracket \varphi \rrbracket (X) = \langle \varphi \rangle \cap X$ |
| (b) | $R(\langle \varphi \land \psi \rangle) = (\llbracket \varphi \rrbracket; R)(\langle \psi \rangle)$ |
Two more useful results

Lemma

For $R \subseteq \text{Env} \times \text{Env}$, predicates $\varphi$ and $\psi$, and $X \subseteq \text{Env}$:

(a) $\llbracket \varphi \rrbracket(X) = \langle \varphi \rangle \cap X$

(b) $R(\langle \varphi \land \psi \rangle) = (\llbracket \varphi \rrbracket; R)(\langle \psi \rangle)$

Proof (a):
Two more useful results

Lemma

For $R \subseteq \text{Env} \times \text{Env}$, predicates $\varphi$ and $\psi$, and $X \subseteq \text{Env}$:

(a) $\llbracket \varphi \rrbracket (X) = \langle \varphi \rangle \cap X$

(b) $R(\langle \varphi \land \psi \rangle) = (\llbracket \varphi \rrbracket ; R)(\langle \psi \rangle)$

Proof (a):

$\forall \eta' \in \llbracket \varphi \rrbracket (X) \iff \exists \eta \in X \text{ s.t. } (\eta, \eta') \in \llbracket \varphi \rrbracket$

$\iff \exists \eta \in X \text{ s.t. } \eta = \eta' \text{ and } \eta \in \langle \varphi \rangle$

$\iff \eta' \in X \cap \langle \varphi \rangle$
Two more useful results

Lemma

For $R \subseteq \text{Env} \times \text{Env}$, predicates $\varphi$ and $\psi$, and $X \subseteq \text{Env}$:

(a) $\llbracket \varphi \rrbracket(X) = \langle \varphi \rangle \cap X$

(b) $R(\langle \varphi \land \psi \rangle) = (\llbracket \varphi \rrbracket; R)(\langle \psi \rangle)$

Proof (b):

\[
\langle \varphi \land \psi \rangle = \langle \varphi \rangle \cap \langle \psi \rangle = \llbracket \varphi \rrbracket(\langle \psi \rangle)
\]

So $R(\langle \varphi \land \psi \rangle) = R(\llbracket \varphi \rrbracket(\langle \psi \rangle))$

$= (\llbracket \varphi \rrbracket; R)(\langle \psi \rangle)$ (see Lemma 1(b))
Inductive case 2: Conditional rule

\[
\begin{array}{c}
\{\varphi \land g\} P \{\psi\} \\
\{\varphi \land \neg g\} Q \{\psi\}
\end{array}
\]

(if)

\[
\{\varphi\} \text{ if } g \text{ then } P \text{ else } Q \text{ fi } \{\psi\}
\]
Inductive case 2: Conditional rule

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

Assume \(\{\varphi \land g\} P \{\psi\}\) and \(\{\varphi \land \neg g\} Q \{\psi\}\) are valid. Need to show that \(\{\varphi\} \text{ if } g \text{ then } P \text{ else } Q \text{ fi } \{\psi\}\) is valid.
Inductive case 2: Conditional rule

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

(if)

Assume \( \{ \varphi \land g \} \ P \ \{ \psi \} \) and \( \{ \varphi \land \neg g \} \ Q \ \{ \psi \} \) are valid. Need to show that \( \{ \varphi \} \ \text{if } g \ \text{then } P \ \text{else } Q \ \text{fi } \{ \psi \} \) is valid.

Recall: \([\text{if } g \ \text{then } P \ \text{else } Q \ \text{fi}] = [g; P] \cup [\neg g; Q]\)
Inductive case 2: Conditional rule

\[
\begin{align*}
\{ \varphi \land g \} & \quad P \{ \psi \} \quad \{ \varphi \land \neg g \} & \quad Q \{ \psi \} \\
\{ \varphi \} & \text{ if } g \text{ then } P \text{ else } Q \text{ fi } \{ \psi \}
\end{align*}
\]

(if)

Assume \( \{ \varphi \land g \} \quad P \{ \psi \} \) and \( \{ \varphi \land \neg g \} \quad Q \{ \psi \} \) are valid. Need to show that \( \{ \varphi \} \text{ if } g \text{ then } P \text{ else } Q \text{ fi } \{ \psi \} \) is valid.

Recall: \( \llbracket \text{if } g \text{ then } P \text{ else } Q \text{ fi} \rrbracket = [g; P] \cup [\neg g; Q] \)

\( \llbracket \text{if } g \text{ then } P \text{ else } Q \text{ fi} \rrbracket(\langle \varphi \rangle) \)
Inductive case 2: Conditional rule

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

Assume \(\{\varphi \land g\} P \{\psi\}\) and \(\{\varphi \land \neg g\} Q \{\psi\}\) are valid. Need to show that \(\{\varphi\} \text{ if } g \text{ then } P \text{ else } Q \text{ fi } \{\psi\}\) is valid.

Recall: \([\text{if } g \text{ then } P \text{ else } Q \text{ fi}] = [g; P] \cup [\neg g; Q]\)

\[
[\text{if } g \text{ then } P \text{ else } Q \text{ fi}](\langle\varphi\rangle)
= [g; P](\langle\varphi\rangle) \cup [\neg g; Q](\langle\varphi\rangle) \quad (\text{see Lemma 1(b)})
\]
Inductive case 2: Conditional rule

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

(if)

Assume \( \{ \varphi \land g \} \quad P \{ \psi \} \) and \( \{ \varphi \land \neg g \} \quad Q \{ \psi \} \) are valid. Need to show that \( \{ \varphi \} \quad \text{if } g \text{ then } P \text{ else } Q \text{ fi } \{ \psi \} \) is valid.

Recall: \( [\text{if } g \text{ then } P \text{ else } Q \text{ fi}] = [g; P] \cup [\neg g; Q] \)

\[
[\text{if } g \text{ then } P \text{ else } Q \text{ fi}](\langle \varphi \rangle)
\]

\[= [g; P](\langle \varphi \rangle) \cup [\neg g; Q](\langle \varphi \rangle) \quad \text{(see Lemma 1(b))}
\]

\[= [P](\langle g \land \varphi \rangle) \cup [Q](\langle \neg g \land \varphi \rangle) \quad \text{(see Lemma 2(b))}
\]
Inductive case 2: Conditional rule

\[
\begin{align*}
\{ \varphi \land g \} & \quad P \quad \{ \psi \} \\
\{ \varphi \land \neg g \} & \quad Q \quad \{ \psi \}
\end{align*}
\]

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

Assume \( \{ \varphi \land g \} \ P \ {\psi} \) and \( \{ \varphi \land \neg g \} \ Q \ {\psi} \) are valid. Need to show that \( \{ \varphi \} \text{ if } g \text{ then } P \text{ else } Q \text{ fi } \{ \psi \} \) is valid.

Recall: \[[\text{if } g \text{ then } P \text{ else } Q \text{ fi}] = [g; P] \cup [\neg g; Q]\]

\[[\text{if } g \text{ then } P \text{ else } Q \text{ fi}] (\langle \varphi \rangle) \]

\[= [g; P](\langle \varphi \rangle) \cup [\neg g; Q](\langle \varphi \rangle) \quad (\text{see Lemma 1(b)})\]

\[= [P](\langle g \land \varphi \rangle) \cup [Q](\langle \neg g \land \varphi \rangle) \quad (\text{see Lemma 2(b)})\]

\[\subseteq \langle \psi \rangle \quad (\text{by IH})\]
Inductive case 3: While rule

\[
\begin{align*}
{\varphi \land g} & \quad P \quad {\varphi} \\
\{\varphi\} & \quad \text{while } g \text{ do } P \text{ od } \{\varphi \land \neg g\}
\end{align*}
\] (loop)
Inductive case 3: While rule

\[
\begin{align*}
\{\varphi \land g\} & \ P \ \{\varphi\} \\
\{\varphi\} \text{ while } g \text{ do } P \text{ od } \{\varphi \land \neg g\} & \quad \text{(loop)}
\end{align*}
\]

Assume \( \{\varphi \land g\} \ P \ \{\varphi\}\) is valid. Need to show that \( \{\varphi\} \text{ while } g \text{ do } P \text{ od } \{\varphi \land \neg g\}\) is valid.
Inductive case 3: While rule

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

(loop)

Assume \( \{ \varphi \land g \} \quad P \quad \{ \varphi \} \) is valid. Need to show that \( \{ \varphi \} \text{ while } g \text{ do } P \text{ od } \{ \varphi \land \neg g \} \) is valid.

Recall: \( [\text{while } g \text{ do } P \text{ od}] = [g; P]^*; [\neg g] \)
Inductive case 3: While rule

\[
\frac{\{\varphi \land g\} \{\varphi\}}{\{\varphi\} \text{ while } g \text{ do } P \text{ od } \{\varphi \land \neg g\}} \quad (\text{loop})
\]

Assume \(\{\varphi \land g\} \{\varphi\}\) is valid. Need to show that \(\{\varphi\} \text{ while } g \text{ do } P \text{ od } \{\varphi \land \neg g\}\) is valid.

Recall: 
\[
[\text{while } g \text{ do } P \text{ od}] = [g; P]^*; [\neg g]
\]

\[
[g; P](\langle \varphi \rangle) = [P](\langle g \land \varphi \rangle) \quad \text{(see Lemma 2(b))}
\]
Inductive case 3: While rule

\[
\begin{align*}
\{ \varphi \land g \} & \quad P \quad \{ \varphi \} \\
\{ \varphi \} \text{ while } g \text{ do } \quad P \quad \text{ od } \quad \{ \varphi \land \neg g \} \\
\end{align*}
\]

(\text{loop})

Assume \{ \varphi \land g \} \quad P \quad \{ \varphi \} \quad \text{is valid. Need to show that} \\
\{ \varphi \} \text{ while } g \text{ do } \quad P \quad \text{ od } \quad \{ \varphi \land \neg g \} \quad \text{is valid.}

Recall: \([\text{while } g \text{ do } P \text{ od}] = [g; P]^*; [\neg g]\\n\]

\[
[g; P](\langle \varphi \rangle) = [P](\langle g \land \varphi \rangle) \subseteq \langle \varphi \rangle \\
\quad \text{(see Lemma 2(b))} \\
\quad \text{(IH)}
\]


Inductive case 3: While rule

\[
\begin{array}{c}
\{ \varphi \land g \} \vdash P \{ \varphi \} \\
\{ \varphi \} \text{ while } g \text{ do } P \text{ od } \{ \varphi \land \neg g \}
\end{array}
\]

(loop)

Assume \( \{ \varphi \land g \} \vdash P \{ \varphi \} \) is valid. Need to show that \( \{ \varphi \} \text{ while } g \text{ do } P \text{ od } \{ \varphi \land \neg g \} \) is valid.

Recall: \([\text{while } g \text{ do } P \text{ od}] = [g; P]^*; [\neg g]\)

\([g; P](\langle \varphi \rangle) = [P](\langle g \land \varphi \rangle) \subseteq \langle \varphi \rangle \) (see Lemma 2(b))

\(\subseteq \langle \varphi \rangle \) (IH)

So \([g; P]^*(\langle \varphi \rangle) \subseteq \langle \varphi \rangle \) (see Corollary)
Inductive case 3: While rule

\[
\begin{align*}
\{ \varphi \land g \} & P \{ \varphi \} \\
\{ \varphi \} \text{ while } g \text{ do } P \text{ od } \{ \varphi \land \neg g \}
\end{align*}
\]  

(loop)

Assume \( \{ \varphi \land g \} \ P \ {\varphi} \) is valid. Need to show that \( \{ \varphi \} \text{ while } g \text{ do } P \text{ od } \{ \varphi \land \neg g \} \) is valid.

Recall: \([\text{while } g \text{ do } P \text{ od}] = [g; P]^*; [\neg g]\)

\[
[g; P](\langle \varphi \rangle) = [P](\langle g \land \varphi \rangle) \subseteq \langle \varphi \rangle
\]

(see Lemma 2(b))

(IH)

So \([g; P]^*(\langle \varphi \rangle) \subseteq \langle \varphi \rangle \) (see Corollary)

So \([g; P]^*; [\neg g](\langle \varphi \rangle) = [\neg g]( [g; P]^*(\langle \varphi \rangle)) \) (see Lemma 1(c))
Inductive case 3: While rule

\[
\begin{array}{c}
\{\varphi \land g\} \mathcal{P} \{\varphi\} \\
\{\varphi\} \text{ while } g \text{ do } \mathcal{P} \text{ od } \{\varphi \land \neg g\}
\end{array}
\]

(loop)

Assume \(\{\varphi \land g\} \mathcal{P} \{\varphi\}\) is valid. Need to show that
\(\{\varphi\} \text{ while } g \text{ do } \mathcal{P} \text{ od } \{\varphi \land \neg g\}\) is valid.

Recall: \([\text{while } g \text{ do } \mathcal{P} \text{ od}] = [g; P]^*; [\neg g]\)

\[ [g; P](\langle \varphi \rangle) = [P](\langle g \land \varphi \rangle) \]
\[ \subseteq \langle \varphi \rangle \]
\[ \text{(IH)} \]

So \([g; P]^*(\langle \varphi \rangle) \subseteq \langle \varphi \rangle\)

(see Corollary)

So \([g; P]^*; [\neg g](\langle \varphi \rangle) = [\neg g][g; P]^*(\langle \varphi \rangle)\)

(see Lemma 1(c))

\[ \subseteq [\neg g](\langle \varphi \rangle) \]

(see Lemma 1(a))
Inductive case 3: While rule

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

Assume \( \{ \varphi \land g \} \text{ } P \{ \varphi \} \) is valid. Need to show that \( \{ \varphi \} \text{ while } g \text{ do } P \text{ od } \{ \varphi \land \neg g \} \) is valid.

Recall: \([\text{while } g \text{ do } P \text{ od}] = [g; P]^*; [\neg g] \]

\[ [g; P](\langle \varphi \rangle) = [P](\langle g \land \varphi \rangle) \]
\[
\subseteq \langle \varphi \rangle \quad \text{(IH)}
\]

So \([g; P]^*(\langle \varphi \rangle) \subseteq \langle \varphi \rangle \) (see Corollary)

So \([g; P]^*; [\neg g](\langle \varphi \rangle) = [\neg g][g; P]^*(\langle \varphi \rangle) \) (see Lemma 1(c))
\[
\subseteq [\neg g](\langle \varphi \rangle) \quad \text{(see Lemma 1(a))}
\]
\[ = \langle \neg g \land \varphi \rangle \quad \text{(see Lemma 2(a))} \]
Inductive case 4: Consequence rule

$$\varphi' \rightarrow \varphi \quad \{\varphi\} P \{\psi\} \quad \psi \rightarrow \psi'$$

(cons)

Assume $$\{\varphi\} P \{\psi\}$$ is valid and $$\varphi' \rightarrow \varphi$$ and $$\psi \rightarrow \psi'$$. Need to show that $$\{\varphi'\} P \{\psi'\}$$ is valid.
Inductive case 4: Consequence rule

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

(cons)

Assume \( \{\varphi\} P \{\psi\} \) is valid and \( \varphi' \rightarrow \varphi \) and \( \psi \rightarrow \psi' \). Need to show that \( \{\varphi'\} P \{\psi'\} \) is valid.
Inductive case 4: Consequence rule

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

(cons)

Assume \(\{\varphi\} \quad P \quad \{\psi\}\) is valid and \(\varphi' \rightarrow \varphi\) and \(\psi \rightarrow \psi'\). Need to show that \(\{\varphi'\} \quad P \quad \{\psi'\}\) is valid.

Observe: If \(\varphi' \rightarrow \varphi\) then \(\langle \varphi' \rangle \subseteq \langle \varphi \rangle\)
Inductive case 4: Consequence rule

\[
\begin{array}{c}
\varphi' \rightarrow \varphi \\
\{\varphi\} P \{\psi\} \\
\{\varphi'\} P \{\psi'\}
\end{array}
\]  
(cons)

Assume \{\varphi\} P \{\psi\} is valid and \varphi' \rightarrow \varphi and \psi \rightarrow \psi'. Need to show that \{\varphi'\} P \{\psi'\} is valid.

Observe: If \varphi' \rightarrow \varphi then \langle \varphi' \rangle \subseteq \langle \varphi \rangle

\[[P](\langle \varphi' \rangle) \subseteq [P](\langle \varphi \rangle) \] (see Lemma 1(a))
Inductive case 4: Consequence rule

\[
\begin{array}{c}
\varphi' \rightarrow \varphi \\
\{ \varphi \} \quad P \quad \{ \psi \} \\
\varphi' \quad P \quad \{ \psi' \} \\
\end{array}
\]

(cons)

Assume \( \{ \varphi \} \ P \ \{ \psi \} \) is valid and \( \varphi' \rightarrow \varphi \) and \( \psi \rightarrow \psi' \). Need to show that \( \{ \varphi' \} \ P \ \{ \psi' \} \) is valid.

Observe: If \( \varphi' \rightarrow \varphi \) then \( \langle \varphi' \rangle \subseteq \langle \varphi \rangle \)

\[
\llbracket P \rrbracket(\langle \varphi' \rangle) \subseteq \llbracket P \rrbracket(\langle \varphi \rangle) \quad \text{(see Lemma 1(a))}
\]

\[
\subseteq \langle \psi \rangle \quad \text{(IH)}
\]

\[
\subseteq \langle \psi' \rangle
\]
Soundness of Hoare Logic

Theorem

\[ \text{If } \vdash \{ \varphi \} P \{ \psi \} \text{ then } \models \{ \varphi \} P \{ \psi \} \]
Summary

- Set theory revisited
- Soundness of Hoare Logic
- Completeness of Hoare Logic
Incompleteness

Theorem (Gödel’s Incompleteness Theorem)

There is no proof system that can prove every valid first-order sentence about arithmetic over the natural numbers.
Incompleteness

**Theorem (Gödel’s Incompleteness Theorem)**

*There is no proof system that can prove every valid first-order sentence about arithmetic over the natural numbers.*

⇒ There are true statements that do not have a proof.
Incompleteness

Theorem (Gödel’s Incompleteness Theorem)

There is no proof system that can prove every valid first-order sentence about arithmetic over the natural numbers.

⇒ There are true statements that do not have a proof.

⇒ Because of (cons) there are valid triples that result from valid, but unprovable, consequences.
Incompleteness

**Theorem (Gödel’s Incompleteness Theorem)**

There is no proof system that can prove every valid first-order sentence about arithmetic over the natural numbers.

⇒ There are true statements that do not have a proof.

⇒ Because of (cons) there are valid triples that result from valid, but unprovable, consequences.

⇒ Hoare Logic is not complete.
Relative completeness of Hoare Logic

**Theorem (Relative completeness of Hoare Logic)**

*With an oracle that decides the validity of predicates,*

\[
\text{if } \models \{ \varphi \} P \{ \psi \} \text{ then } \vdash \{ \varphi \} P \{ \psi \}.
\]
Need to know for this course

- Write programs in $\mathcal{L}$.
- Give proofs using the Hoare logic rules (full and outline)
- Definition of $[\cdot]$  
- Definition of composition and transitive closure