COMP2111 Week 8/9
Term 1, 2024
Hoare Logic
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, …
- Function symbols: +, ∗, …
- Predicate symbols: <, ≤, ≥, |, …
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: \( +, \ast, \ldots \)
- 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
Example

\[
i := 0;
m := 1;
\text{while } i < N \text{ do}
\quad i := i + 1;
\quad m := m \times i
\text{od}
\]
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
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)\}
<table>
<thead>
<tr>
<th>Hoare triple: Examples</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example</strong></td>
<td></td>
</tr>
<tr>
<td>{(x = 0)} x := 1 {(x = 1)}</td>
<td></td>
</tr>
<tr>
<td>{(x = 499)} x := x + 1 {(x = 500)}</td>
<td></td>
</tr>
</tbody>
</table>
Hoare triple: Examples

Example

\[(x = 0) \implies x := 1 \implies (x = 1)\]

\[(x = 499) \implies x := x + 1 \implies (x = 500)\]

\[(x > 0) \implies y := 0 - x \implies (y < 0) \land (x \neq y)\]
Example

\{N \geq 0\}
i := 0;
m := 1;
while i < N do
  i := i + 1;
  m := m \times i
od
\{m = N!\}
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 \( \varphi \) after executing the assignment; then \( e \) must have property \( \varphi \) before executing the assignment
Assignment: Example

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

\[
\begin{align*}
&\{ (y = 0) \} x := y \{ (x = 0) \} \\
&\{ (y = y) \} x := y \{ (x = y) \} \\
&\{ (1 < 2) \} x := 1 \{ (x < 2) \} \\
&\{ (y = 3) \} x := y \{ (x > 2) \} \\
\end{align*}
\]

Problem!
Sequence

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

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

Example

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

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

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

(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

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

(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:

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

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

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

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)\} x := y \{(x > 2)\}  \hspace{1cm} \textit{Problem!}
Example

\{(y = 3)\} x := y \{(x > 2)\}  \hspace{1cm} \text{Problem!}

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

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

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:

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

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

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

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

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

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

\[
\{ N \geq 0 \}
\]
\[
i := 0;
\]
\[
m := 1;
\]
\[
\{ m = i! \land N \geq 0 \}
\]
while \( i \neq N \) do

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

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

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

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

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

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

\[
\begin{align*}
\{\varphi \land g\} & \quad P \{\psi\} \quad \{\varphi \land \neg g\} & \quad Q \{\psi\} \\
\{\varphi\} & \quad \text{if } g \text{ then } P \text{ else } Q \text{ fi } \{\psi\} & \\
\{\varphi[x := e]\} & \quad x := e \{\varphi\} & \\
\{\varphi \land g\} & \quad P \{\varphi\} \\
\{\varphi\} & \quad \text{while } g \text{ do } P \od \{\varphi \land \neg g\} & \\
\{\varphi\} & \quad P \{\alpha\} \quad \{\alpha\} & \quad Q \{\psi\} & \\
\{\varphi\} & \quad P ; Q \{\psi\} & \\
\varphi' & \Rightarrow \varphi \quad \{\varphi\} & \quad P \{\psi\} \quad \psi \Rightarrow \psi' & \\
\{\varphi'\} & \quad P \{\psi'\}
\end{align*}
\]
Let's verify the Factorial program using our Hoare rules:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Example

\[
\begin{align*}
m &:= 1; \\
n &:= 1; \\
i &:= 1; \\
&\text{while } i < N \text{ do} \\
&t := m; \\
&m := n; \\
&n := m + t; \\
&i := i + 1 \\
&\text{od}
\end{align*}
\]
<table>
<thead>
<tr>
<th>Example</th>
</tr>
</thead>
</table>
| \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 \( L \) 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
  - 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 (Env)

- $x \leftarrow 0$
- $y \leftarrow 0$
- $z \leftarrow 0$

- $x \leftarrow 3$
- $y \leftarrow 2$
- $z \leftarrow 1$

- $x \leftarrow 1$
- $y \leftarrow 1$
- $z \leftarrow 1$

- $x \leftarrow 2$
- $y \leftarrow 2$
- $z \leftarrow 2$

- $x \leftarrow 0$
- $y \leftarrow 1$
- $z \leftarrow 2$
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 \((\text{ENV})\)

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

- \( x \leftarrow 1 \)
  \( y \leftarrow 1 \)
  \( z \leftarrow 1 \)

- \( x \leftarrow 0 \)
  \( y \leftarrow 1 \)
  \( z \leftarrow 2 \)

- \( x \leftarrow 0 \)
  \( y \leftarrow 1 \)
  \( z \leftarrow 0 \)

- \( x \leftarrow 3 \)
  \( y \leftarrow 2 \)
  \( z \leftarrow 1 \)

- \( x \leftarrow 2 \)
  \( y \leftarrow 2 \)
  \( z \leftarrow 2 \)
Sequencing

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

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

\[
\begin{align*}
\text{[if } b \text{ then } P \text{ else } Q \text{ fi]} &= \left\{ \begin{array}{ll}
[P] & \text{if } [b]^\eta = \text{true} \\
[Q] & \text{otherwise.}
\end{array} \right.
\end{align*}
\]
Detour: Predicates as programs

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

$$\langle b \rangle = \{ \eta : \llbracket b \rrbracket^\eta = \text{true} \}$$

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

$$\llbracket b \rrbracket = \{ (\eta, \eta') : \eta \in \langle b \rangle \}$$
Detour: Predicates as programs

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

$$\langle b \rangle = \{ \eta : \llbracket b \rrbracket^\eta = \text{true} \}$$

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

$$\llbracket b \rrbracket = \{ (\eta, \eta) : \eta \in \langle b \rangle \}$$

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

While

while \( b \) do \( P \) od

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

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

- 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
- $R^{n+1} = R^n \cdot R$

**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 $f(X) = \Delta \cup X ; R$
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 $\models \{ \varphi \} P \{ \psi \}$ if

$$\llbracket P \rrbracket(\langle \varphi \rangle) \subseteq \langle \psi \rangle.$$  

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

\[ [(P)] \]
Validity

\[ \langle \varphi \rangle \] \[ [P] \] \[ [P] \langle \varphi \rangle \] \[ \langle \psi \rangle \]
Hoare Logic is **sound** with respect to the semantics given. That is,

**Theorem**

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

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

- Set theory revisited
- Soundness of Hoare Logic
- Completeness of Hoare Logic
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) $R(S(A)) = (S; R)(A)$
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):

\[
y \in R(A) \iff \exists x \in A \text{ such that } (x, y) \in R
\]\[
\iff \exists x \in B \text{ such that } (x, y) \in R
\]\[
\iff y \in R(B)
\]
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):

\[ z \in R(S(A)) \iff \exists y \in S(A) \text{ s.t. } (y, z) \in R \]
\[ \iff \exists x \in A, y \in S(A) \text{ s.t. } (x, y) \in S \text{ and } (y, z) \in R \]
\[ \iff \exists x \in A \text{ s.t. } (x, z) \in (S; R) \]
\[ \iff z \in (S; R)(A) \]
Corollary

If $R(A) \subseteq A$ then $R^*(A) \subseteq A$
Some results on relational images

Corollary

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

Proof:
Corollary

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

Proof:

$$R(A) \subseteq A \implies R^{i+1}(A) = R^i(R(A)) \subseteq R^i(A)$$

$$\implies R^{i+1}(A) \subseteq R(A) \subseteq A$$

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**

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

Proof:
By induction on the structure of the proof.
Soundness of Hoare Logic

Theorem

If \( \vdash \{ \varphi \} P \{ \psi \} \) then \( \models \{ \varphi \} P \{ \psi \} \)

Proof:
Soundness of Hoare Logic

Theorem

If \( \vdash \{ \phi \} P \{ \psi \} \) then \( \models \{ \phi \} P \{ \psi \} \)

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

\[
\begin{array}{c}
\{\varphi[e/x]\} x := e \{\varphi\} \\
\end{array}
\]  

(ass)
Base case: Assignment rule

\[ \{ \varphi[e/x] \} x := e \{ \varphi \} \]  \hspace{1cm} \text{(ass)}

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

\[ \llbracket x := e \rrbracket (\llangle \varphi[e/x] \rrangle) \subseteq \llangle \varphi \rrangle. \]
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,

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

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

\[
[\textcolor{black}{x := e}] (\langle \varphi[e/x] \rangle) \subseteq \langle \varphi \rangle.
\]

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

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

\[
[x := e](\sector{\varphi[e/x]}) \subseteq \sector{\varphi}.
\]

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

So if \( \eta \in \sector{\varphi[e/x]} \) then \( \eta' \in \sector{\varphi} \)

Recall: \( (\eta, \eta'') \in [x := e] \) if and only if \( \eta'' = \eta[x \mapsto \sector{e}^\eta] \),
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: \(\{\varphi[e/x]\}^\eta = \{\varphi\}^{\eta'}\) 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

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

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

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 \(\eta'' = \eta[x \mapsto \langle e \rangle^\eta]\),

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

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

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

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

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

Recall: 
\[
\left[\left[ P; Q \right]\right] = \left[\left[ P \right]\right]; \left[\left[ Q \right]\right]
\]

So:
\[
\left[\left[ P; Q \right]\right](\langle \varphi \rangle) = \left[\left[ Q \right]\right](\left[\left[ P \right]\right](\langle \varphi \rangle))
\]

(see Lemma 1(c))

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

So:
\[
\left[\left[ Q \right]\right](\left[\left[ P \right]\right](\langle \varphi \rangle)) \subseteq \left[\left[ Q \right]\right](\langle \psi \rangle) \subseteq \langle \rho \rangle
\]

(see Lemma 1(a))
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.
Inductive case 1: Sequence rule

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

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

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

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

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

(\text{seq})

\[
\{\varphi\} P; Q \{\rho\}
\]

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

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

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

\[
\begin{array}{c}
\{\varphi\} P \{\psi\} \\
\{\psi\} Q \{\rho\} \\
\hline
\{\varphi\} P; Q \{\rho\}
\end{array}
\] (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) $[\varphi](X) = \langle \varphi \rangle \cap X$

(b) $R(\langle \varphi \land \psi \rangle) = ([\varphi]; R)(\langle \psi \rangle)$
Two more useful results

Lemma

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

(a) $[\varphi](X) = \langle \varphi \rangle \cap X$

(b) $R(\langle \varphi \land \psi \rangle) = ([\varphi]; R)(\langle \psi \rangle)$

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):

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

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

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

**Lemma**

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

(a) $[[\varphi]](X) = \langle \varphi \rangle \cap X$

(b) $R(\langle \varphi \land \psi \rangle) = ([[\varphi]; R](\langle \psi \rangle))$

Proof (b):

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

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

$$= ([[\varphi]; R](\langle \psi \rangle)) \quad (\text{see Lemma 1(b)})$$
Inductive case 2: Conditional rule

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

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

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

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

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

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

(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

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

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

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

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

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

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

(if)

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

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

\[
\begin{array}{c}
\{ \varphi \land g \} \ P \ \{ \varphi \} \\
\{ \varphi \} \text{ while } g \text{ do } P \text{ od } \{ \varphi \land \neg g \}
\end{array}
\] (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.
Inductive case 3: While rule

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

(loop)

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

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

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

\([g; P]([\varphi]) = [P]([g \land \varphi]) \quad \text{(see Lemma 2(b))}\)
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 \(\{\varphi \land g\} P \{\varphi\}\) is valid. Need to show that \(\{\varphi\} \text{ while } g \text{ do } P \text{ od } \{\varphi \land \neg g\}\) is valid.

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

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

\[[g; P]^*; [\neg g] \subseteq [\neg g](\angle \varphi \rangle) \quad \text{(IH)}\]
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\} while \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)
Inductive case 3: While rule

\[
\begin{align*}
\{\varphi \land g\} & \textcolor{red}{P} \{\varphi\} \\
\{\varphi\} \text{ while } g \text{ do } P \text{ od } \{\varphi \land \neg g\}
\end{align*}
\]  \hspace{1cm} \text{(loop)}

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

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

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

So \(\left[g; P\right]^\ast(\langle \varphi \rangle) \subseteq \langle \varphi \rangle\) \hspace{1cm} \text{(see Corollary)}

So \(\left[g; P\right]^\ast; \left[\neg g\right](\langle \varphi \rangle) = \left[\neg g\right](\left[g; P\right]^\ast(\langle \varphi \rangle))\) \hspace{1cm} \text{(see Lemma 1(c))}
Inductive case 3: While rule

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

(loop)

Assume \(\{\varphi \land g\} \frac{}{P} \{\varphi\}\) 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]\

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

(see Lemma 2(b))

\[\subseteq \langle \varphi \rangle\]

(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

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

(loop)

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

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

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

(IH)

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

So \([g; P]^*; [\neg g](\langle \varphi \rangle) = [\neg g](\langle g; P]^*(\langle \varphi \rangle)) \subseteq [\neg g](\langle \varphi \rangle) \quad \text{(see Lemma 1(c))}

(see Lemma 1(a))

\[
= \langle \neg g \land \varphi \rangle \quad \text{(see Lemma 2(a))}
\]
Inductive case 4: Consequence rule

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

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

(cons)
Inductive case 4: Consequence rule

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

(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.
Inductive case 4: Consequence rule

\[
\begin{array}{ccc}
\varphi' & \rightarrow & \varphi \\
\{\varphi\} & P & \{\psi\} \\
\{\varphi'\} & P & \{\psi'\} \\
\end{array}
\]

(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 \(\langle \varphi' \rangle \subseteq \langle \varphi \rangle\)
Inductive case 4: Consequence rule

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

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 \(\langle \varphi' \rangle \subseteq \langle \varphi \rangle\)

\[
\llbracket P \rrbracket(\langle \varphi' \rangle) \subseteq \llbracket P \rrbracket(\langle \varphi \rangle) \quad \text{(see Lemma 1(a))}
\]
Inductive case 4: Consequence rule

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

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 \(\langle \varphi' \rangle \subseteq \langle \varphi \rangle\)

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

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

\(\subseteq \langle \psi' \rangle\)
Soundness of Hoare Logic

Theorem

If $\vdash \{ \varphi \} P \{ \psi \}$ 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.
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 $\llbracket \cdot \rrbracket$
- Definition of composition and transitive closure