COMPP2111 Week 5

Term 1, 2019

Hoare Logic II

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

$\mathcal{L}$: A simple imperative programming language

Consider the vocabulary of basic arithmetic:

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

- An **(arithmetic) expression** is a term over this vocabulary.
- A **boolean expression** is a predicate formula over this vocabulary.

$\mathcal{L}$: A simple imperative programming language

Consider the vocabulary of basic arithmetic:

- Constant symbols: $0, 1, 2, \ldots$
- Function symbols: $+, *, \ldots$
- 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 $\mathcal{L}$

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** b **then** P **else** Q **fi**
where b is a boolean expression.

While: **while** b **do** P **od**

Factorial in $\mathcal{L}$

Example

$$
\begin{array}{l}
f := 1; \\
k := 0; \\
\textbf{while } k < n \textbf{ do} \\
\quad k := k + 1; \\
\quad f := f * k \\
\textbf{od}
\end{array}
$$

Summary

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

Hoare triple (Syntax)

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

Intuition:
If φ holds in a state of some computational model
then ψ holds in the state reached after a successful execution of P.

Summary

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

Hoare Logic

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

Assignment

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

Intuition:

If x has property φ *after* executing the assignment; then e must have property φ *before* executing the assignment

Assignment

$$\frac{}{\{\varphi(e)\}\; x := e\; \{\varphi(x)\}} \quad \text{(ass)}$$

Intuition:
If x has property φ *after* executing the assignment; then e must have property φ *before* executing the assignment

Sequence

$$\frac{\{\varphi\}\ P\ \{\psi\} \qquad \{\psi\}\ Q\ \{\rho\}}{\{\varphi\}\ P; Q\ \{\rho\}}\ (\text{seq})$$

Intuition:
If the postcondition of P matches the precondition of Q we can sequentially combine the two program fragments

Conditional

$$\frac{\{\varphi \wedge g\}\ P\ \{\psi\} \qquad \{\varphi \wedge \neg g\}\ Q\ \{\psi\}}{\{\varphi\}\ \textbf{if}\ g\ \textbf{then}\ P\ \textbf{else}\ Q\ \textbf{fi}\ \{\psi\}} \quad \text{(if)}$$

Intuition:

- When a conditional is executed, either P or Q will be executed.
- If ψ is a postcondition of the conditional, then it must be a postcondition of *both* branches
- Likewise, f φ 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 (strengthen the preconditions).

While

$$\frac{\{\varphi \wedge g\}\ P\ \{\varphi\}}{\{\varphi\}\ \textbf{while}\ g\ \textbf{do}\ P\ \textbf{od}\ \{\varphi \wedge \neg g\}} \quad \text{(loop)}$$

Intuition:

- φ is a **loop-invariant**. It must be both a pre- and postcondition of P so that sequences of Ps can be run together.
- If the while loop terminates, g cannot hold.

Precondition strengthening and Postcondition weakening

$$\frac{\varphi' \to \varphi \qquad \{\varphi\}\, P\, \{\psi\} \qquad \psi \to \psi'}{\{\varphi'\}\, P\, \{\psi'\}} \quad \text{(cons)}$$

Intuition:

- $\varphi' \to \varphi$: φ' is **stronger** than φ
  - Stronger conditions impose more restrictions

  $\Rightarrow$ States which satisfy φ' are a subset of states which satisfy φ

  $\Rightarrow$ States reached after executing P are a subset

  $\Rightarrow$ The postcondition will hold in the smaller set of terminal states
- $\psi \to \psi'$: ψ' is **weaker** than ψ
  - Weaker conditions impose fewer restrictions

  $\Rightarrow$ States which satisfy ψ are a subset of states which satisfy ψ'

  $\Rightarrow$ States reached after executing P are a subset of those which satisfy ψ'

Example

Example

$$\{\textsc{True}\}$$

$f := 1;$
$k := 0;$
while $\neg(k = n)$ **do**
  $k := k + 1;$
  $f := f * k$
od

$$\{f = n!\}$$

Example

Example

$$
\begin{array}{l}
\{\textsc{True}\} \\
f := 1; \\
k := 0; \\
\textbf{while } \neg(k = n) \textbf{ do} \\
\quad k := k + 1; \\
\quad f := f * k \\
\textbf{od} \\
\{f = n!\}
\end{array}
$$

Example (full proof)

Example

1.	$\{1 = 0!\}\ f := 1\ \{f = 0!\}$	(ass)
2.	$\{f = 0!\}\ k := 0\ \{f = k!\}$	(ass)
3.	$\{1 = 0!\}\ f := 1; k := 0\ \{f = k!\}$	(seq) : 1, 2
4.	$\{f(k+1) = (k+1)!\}\ k := k+1\ \{fk = k!\}$	(ass)
5.	$\{fk = k!\}\ f := f * k\ \{f = k!\}$	(ass)
6.	$\{f(k+1) = (k+1)!\}\ \textsc{Loop}\ \{f = k!\}$	(seq) : 4, 5
7.	$(f = k!) \wedge \neg(k = n)\ \rightarrow\ f(k+1) = (k+1)!$	math
8.	$\{(f = k!) \wedge \neg(k = n)\}\ \textsc{Loop}\ \{f = k!\}$	(cons): 6,7
9.	$\{f = k!\}$ **while** ... **od** $\{(f = k!) \wedge (k = n)\}$	(loop): 8
10.	$\{1 = 0!\}\ \textsc{Factorial}\ \{(f = k!) \wedge (k = n)\}$	(seq)
11.	$\textsc{True} \rightarrow (1 = 0!)$	math
12.	$((f = k!) \wedge (k = n)) \rightarrow f = n!$	math
13.	$\{\textsc{True}\}\ \textsc{Factorial}\ \{f = n!\}$	(cons): 10,11,12

Example (full proof)

Example

1.	$\{1 = 0!\}\ f := 1\ \{f = 0!\}$	(ass)
2.	$\{f = 0!\}\ k := 0\ \{f = k!\}$	(ass)
3.	$\{1 = 0!\}\ f := 1; k := 0\ \{f = k!\}$	(seq) : 1, 2
4.	$\{f(k+1) = (k+1)!\}\ k := k+1\ \{fk = k!\}$	(ass)
5.	$\{fk = k!\}\ f := f * k\ \{f = k!\}$	(ass)
6.	$\{f(k+1) = (k+1)!\}\ \textsc{Loop}\ \{f = k!\}$	(seq) : 4, 5
7.	$(f = k!) \wedge \neg(k = n) \ \rightarrow\ f(k+1) = (k+1)!$	math
8.	$\{(f = k!) \wedge \neg(k = n)\}\ \textsc{Loop}\ \{f = k!\}$	(cons): 6,7
9.	$\{f = k!\}$ **while** ... **od** $\{(f = k!) \wedge (k = n)\}$	(loop): 8
10.	$\{1 = 0!\}\ \textsc{Factorial}\ \{(f = k!) \wedge (k = n)\}$	(seq)
11.	$\textsc{True} \rightarrow (1 = 0!)$	math
12.	$((f = k!) \wedge (k = n)) \rightarrow f = n!$	math
13.	$\{\textsc{True}\}\ \textsc{Factorial}\ \{f = n!\}$	(cons): 10,11,12

Example (full proof)

Example

1.	$\{1 = 0!\}\ f := 1\ \{f = 0!\}$	(ass)
2.	$\{f = 0!\}\ k := 0\ \{f = k!\}$	(ass)
3.	$\{1 = 0!\}\ f := 1; k := 0\ \{f = k!\}$	(seq) : 1, 2
4.	$\{f(k+1) = (k+1)!\}\ k := k+1\ \{fk = k!\}$	(ass)
5.	$\{fk = k!\}\ f := f * k\ \{f = k!\}$	(ass)
6.	$\{f(k+1) = (k+1)!\}\ \textsc{Loop}\ \{f = k!\}$	(seq) : 4, 5
7.	$(f = k!) \wedge \neg(k = n) \ \rightarrow\ f(k+1) = (k+1)!$	math
8.	$\{(f = k!) \wedge \neg(k = n)\}\ \textsc{Loop}\ \{f = k!\}$	(cons): 6,7
9.	$\{f = k!\}$ **while** ... **od** $\{(f = k!) \wedge (k = n)\}$	(loop): 8
10.	$\{1 = 0!\}\ \textsc{Factorial}\ \{(f = k!) \wedge (k = n)\}$	(seq)
11.	$\textsc{True} \rightarrow (1 = 0!)$	math
12.	$((f = k!) \wedge (k = n)) \rightarrow f = n!$	math
13.	$\{\textsc{True}\}\ \textsc{Factorial}\ \{f = n!\}$	(cons): 10,11,12

Example (full proof)

Example

1.	$\{1 = 0!\}\ f := 1\ \{f = 0!\}$	(ass)
2.	$\{f = 0!\}\ k := 0\ \{f = k!\}$	(ass)
3.	$\{1 = 0!\}\ f := 1; k := 0\ \{f = k!\}$	(seq) : 1, 2
4.	$\{f(k+1) = (k+1)!\}\ k := k+1\ \{fk = k!\}$	(ass)
5.	$\{fk = k!\}\ f := f * k\ \{f = k!\}$	(ass)
6.	$\{f(k+1) = (k+1)!\}\ \textsc{Loop}\ \{f = k!\}$	(seq) : 4, 5
7.	$(f = k!) \wedge \neg(k = n)\ \rightarrow\ f(k+1) = (k+1)!$	math
8.	$\{(f = k!) \wedge \neg(k = n)\}\ \textsc{Loop}\ \{f = k!\}$	(cons): 6,7
9.	$\{f = k!\}$ **while** ... **od** $\{(f = k!) \wedge (k = n)\}$	(loop): 8
10.	$\{1 = 0!\}\ \textsc{Factorial}\ \{(f = k!) \wedge (k = n)\}$	(seq)
11.	$\textsc{True} \rightarrow (1 = 0!)$	math
12.	$((f = k!) \wedge (k = n)) \rightarrow f = n!$	math
13.	$\{\textsc{True}\}\ \textsc{Factorial}\ \{f = n!\}$	(cons): 10,11,12

Example (full proof)

Example

1.	$\{1 = 0!\}\, f := 1\, \{f = 0!\}$	(ass)
2.	$\{f = 0!\}\, k := 0\, \{f = k!\}$	(ass)
3.	$\{1 = 0!\}\, f := 1; k := 0\, \{f = k!\}$	(seq) : 1, 2
4.	$\{f(k+1) = (k+1)!\}\, k := k+1\, \{fk = k!\}$	(ass)
5.	$\{fk = k!\}\, f := f * k\, \{f = k!\}$	(ass)
6.	$\{f(k+1) = (k+1)!\}\, \textsc{Loop}\, \{f = k!\}$	(seq) : 4, 5
7.	$(f = k!) \wedge \neg(k = n) \rightarrow f(k+1) = (k+1)!$	math
8.	$\{(f = k!) \wedge \neg(k = n)\}\, \textsc{Loop}\, \{f = k!\}$	(cons): 6,7
9.	$\{f = k!\}$ **while** ... **od** $\{(f = k!) \wedge (k = n)\}$	(loop): 8
10.	$\{1 = 0!\}\, \textsc{Factorial}\, \{(f = k!) \wedge (k = n)\}$	(seq)
11.	$\textsc{True} \rightarrow (1 = 0!)$	math
12.	$((f = k!) \wedge (k = n)) \rightarrow f = n!$	math
13.	$\{\textsc{True}\}\, \textsc{Factorial}\, \{f = n!\}$	(cons): 10,11,12

Example (full proof)

Example

1.	$\{1 = 0!\}\, f := 1\, \{f = 0!\}$	(ass)
2.	$\{f = 0!\}\, k := 0\, \{f = k!\}$	(ass)
3.	$\{1 = 0!\}\, f := 1; k := 0\, \{f = k!\}$	(seq) : 1, 2
4.	$\{f(k+1) = (k+1)!\}\, k := k+1\, \{fk = k!\}$	(ass)
5.	$\{fk = k!\}\, f := f * k\, \{f = k!\}$	(ass)
6.	$\{f(k+1) = (k+1)!\}\, \textsc{Loop}\, \{f = k!\}$	(seq) : 4, 5
7.	$(f = k!) \wedge \neg(k = n) \rightarrow f(k+1) = (k+1)!$	math
8.	$\{(f = k!) \wedge \neg(k = n)\}\, \textsc{Loop}\, \{f = k!\}$	(cons): 6,7
9.	$\{f = k!\}$ **while** ... **od** $\{(f = k!) \wedge (k = n)\}$	(loop): 8
10.	$\{1 = 0!\}\, \textsc{Factorial}\, \{(f = k!) \wedge (k = n)\}$	(seq)
11.	$\textsc{True} \rightarrow (1 = 0!)$	math
12.	$((f = k!) \wedge (k = n)) \rightarrow f = n!$	math
13.	$\{\textsc{True}\}\, \textsc{Factorial}\, \{f = n!\}$	(cons): 10,11,12

Example (full proof)

Example

1.	$\{1 = 0!\}\ f := 1\ \{f = 0!\}$	(ass)
2.	$\{f = 0!\}\ k := 0\ \{f = k!\}$	(ass)
3.	$\{1 = 0!\}\ f := 1; k := 0\ \{f = k!\}$	(seq) : 1, 2
4.	$\{f(k+1) = (k+1)!\}\ k := k+1\ \{fk = k!\}$	(ass)
5.	$\{fk = k!\}\ f := f * k\ \{f = k!\}$	(ass)
6.	$\{f(k+1) = (k+1)!\}\ \textsc{Loop}\ \{f = k!\}$	(seq) : 4, 5
7.	$(f = k!) \wedge \neg(k = n)\ \rightarrow\ f(k+1) = (k+1)!$	math
8.	$\{(f = k!) \wedge \neg(k = n)\}\ \textsc{Loop}\ \{f = k!\}$	(cons): 6,7
9.	$\{f = k!\}$ **while** ... **od** $\{(f = k!) \wedge (k = n)\}$	(loop): 8
10.	$\{1 = 0!\}\ \textsc{Factorial}\ \{(f = k!) \wedge (k = n)\}$	(seq)
11.	$\textsc{True} \rightarrow (1 = 0!)$	math
12.	$((f = k!) \wedge (k = n)) \rightarrow f = n!$	math
13.	$\{\textsc{True}\}\ \textsc{Factorial}\ \{f = n!\}$	(cons): 10,11,12

Example (full proof)

Example

1. $\{1 = 0!\}\, f := 1\, \{f = 0!\}$ (ass)
2. $\{f = 0!\}\, k := 0\, \{f = k!\}$ (ass)
3. $\{1 = 0!\}\, f := 1; k := 0\, \{f = k!\}$ (seq) : 1, 2
4. $\{f(k+1) = (k+1)!\}\, k := k+1\, \{fk = k!\}$ (ass)
5. $\{fk = k!\}\, f := f * k\, \{f = k!\}$ (ass)
6. $\{f(k+1) = (k+1)!\}\, \textsc{Loop}\, \{f = k!\}$ (seq) : 4, 5
7. $(f = k!) \wedge \neg(k = n) \ \rightarrow\ f(k+1) = (k+1)!$ math
8. $\{(f = k!) \wedge \neg(k = n)\}\, \textsc{Loop}\, \{f = k!\}$ (cons): 6,7
9. $\{f = k!\}$ **while** ... **od** $\{(f = k!) \wedge (k = n)\}$ (loop): 8
10. $\{1 = 0!\}\, \textsc{Factorial}\, \{(f = k!) \wedge (k = n)\}$ (seq)
11. $\textsc{True} \rightarrow (1 = 0!)$ math
12. $((f = k!) \wedge (k = n)) \rightarrow f = n!$ math
13. $\{\textsc{True}\}\, \textsc{Factorial}\, \{f = n!\}$ (cons): 10,11,12

Example (full proof)

Example

1.	$\{1 = 0!\}\ f := 1\ \{f = 0!\}$	(ass)
2.	$\{f = 0!\}\ k := 0\ \{f = k!\}$	(ass)
3.	$\{1 = 0!\}\ f := 1; k := 0\ \{f = k!\}$	(seq) : 1, 2
4.	$\{f(k+1) = (k+1)!\}\ k := k+1\ \{fk = k!\}$	(ass)
5.	$\{fk = k!\}\ f := f * k\ \{f = k!\}$	(ass)
6.	$\{f(k+1) = (k+1)!\}\ \textsc{Loop}\ \{f = k!\}$	(seq) : 4, 5
7.	$(f = k!) \wedge \neg(k = n)\ \rightarrow\ f(k+1) = (k+1)!$	math
8.	$\{(f = k!) \wedge \neg(k = n)\}\ \textsc{Loop}\ \{f = k!\}$	(cons): 6,7
9.	$\{f = k!\}$ **while** $\ldots$ **od** $\{(f = k!) \wedge (k = n)\}$	(loop): 8
10.	$\{1 = 0!\}\ \textsc{Factorial}\ \{(f = k!) \wedge (k = n)\}$	(seq)
11.	$\textsc{True} \rightarrow (1 = 0!)$	math
12.	$((f = k!) \wedge (k = n)) \rightarrow f = n!$	math
13.	$\{\textsc{True}\}\ \textsc{Factorial}\ \{f = n!\}$	(cons): 10,11,12

Example (full proof)

Example

1.	$\{1 = 0!\}\ f := 1\ \{f = 0!\}$	(ass)
2.	$\{f = 0!\}\ k := 0\ \{f = k!\}$	(ass)
3.	$\{1 = 0!\}\ f := 1; k := 0\ \{f = k!\}$	(seq) : 1, 2
4.	$\{f(k+1) = (k+1)!\}\ k := k+1\ \{fk = k!\}$	(ass)
5.	$\{fk = k!\}\ f := f * k\ \{f = k!\}$	(ass)
6.	$\{f(k+1) = (k+1)!\}\ \textsc{Loop}\ \{f = k!\}$	(seq) : 4, 5
7.	$(f = k!) \wedge \neg(k = n)\ \rightarrow\ f(k+1) = (k+1)!$	math
8.	$\{(f = k!) \wedge \neg(k = n)\}\ \textsc{Loop}\ \{f = k!\}$	(cons): 6,7
9.	$\{f = k!\}$ **while** ... **od** $\{(f = k!) \wedge (k = n)\}$	(loop): 8
10.	$\{1 = 0!\}\ \textsc{Factorial}\ \{(f = k!) \wedge (k = n)\}$	(seq)
11.	$\textsc{True} \rightarrow (1 = 0!)$	math
12.	$((f = k!) \wedge (k = n)) \rightarrow f = n!$	math
13.	$\{\textsc{True}\}\ \textsc{Factorial}\ \{f = n!\}$	(cons): 10,11,12

Example (full proof)

Example

1.	$\{1 = 0!\}\ f := 1\ \{f = 0!\}$	(ass)
2.	$\{f = 0!\}\ k := 0\ \{f = k!\}$	(ass)
3.	$\{1 = 0!\}\ f := 1; k := 0\ \{f = k!\}$	(seq) : 1, 2
4.	$\{f(k+1) = (k+1)!\}\ k := k + 1\ \{fk = k!\}$	(ass)
5.	$\{fk = k!\}\ f := f * k\ \{f = k!\}$	(ass)
6.	$\{f(k+1) = (k+1)!\}\ \textsc{Loop}\ \{f = k!\}$	(seq) : 4, 5
7.	$(f = k!) \wedge \neg(k = n) \ \rightarrow\ f(k+1) = (k+1)!$	math
8.	$\{(f = k!) \wedge \neg(k = n)\}\ \textsc{Loop}\ \{f = k!\}$	(cons): 6,7
9.	$\{f = k!\}$ **while** ... **od** $\{(f = k!) \wedge (k = n)\}$	(loop): 8
10.	$\{1 = 0!\}\ \textsc{Factorial}\ \{(f = k!) \wedge (k = n)\}$	(seq)
11.	$\textsc{True} \rightarrow (1 = 0!)$	math
12.	$((f = k!) \wedge (k = n)) \rightarrow f = n!$	math
13.	$\{\textsc{True}\}\ \textsc{Factorial}\ \{f = n!\}$	(cons): 10,11,12

Example (full proof)

Example

1.	$\{1 = 0!\}\ f := 1\ \{f = 0!\}$	(ass)
2.	$\{f = 0!\}\ k := 0\ \{f = k!\}$	(ass)
3.	$\{1 = 0!\}\ f := 1; k := 0\ \{f = k!\}$	(seq) : 1, 2
4.	$\{f(k+1) = (k+1)!\}\ k := k+1\ \{fk = k!\}$	(ass)
5.	$\{fk = k!\}\ f := f * k\ \{f = k!\}$	(ass)
6.	$\{f(k+1) = (k+1)!\}\ \textsc{Loop}\ \{f = k!\}$	(seq) : 4, 5
7.	$(f = k!) \wedge \neg(k = n)\ \rightarrow\ f(k+1) = (k+1)!$	math
8.	$\{(f = k!) \wedge \neg(k = n)\}\ \textsc{Loop}\ \{f = k!\}$	(cons): 6,7
9.	$\{f = k!\}$ **while** ... **od** $\{(f = k!) \wedge (k = n)\}$	(loop): 8
10.	$\{1 = 0!\}\ \textsc{Factorial}\ \{(f = k!) \wedge (k = n)\}$	(seq)
11.	$\textsc{True} \rightarrow (1 = 0!)$	math
12.	$((f = k!) \wedge (k = n)) \rightarrow f = n!$	math
13.	$\{\textsc{True}\}\ \textsc{Factorial}\ \{f = n!\}$	(cons): 10,11,12

Example (proof outline)

Example

	$\{\textsc{True}\}$
	$\{1 = 0!\}$
$f := 1;$	$\{f = 0!\}$
$k := 0;$	$\{f = k!\}$
while $\neg(k = n)$ **do**	$\{(f = k!) \wedge \neg(k = n)\}$
	$\{f(k+1) = (k+1)!\}$
$\quad k := k + 1;$	$\{fk = k!\}$
$\quad f := f * k$	$\{f = k!\}$
od	$\{(f = k!) \wedge (k = n)\}$
	$\{f = n!\}$

Example (proof outline)

Example

	$\{\textsc{True}\}$
	$\{1 = 0!\}$
$f := 1;$	$\{f = 0!\}$
$k := 0;$	$\{f = k!\}$
while $\neg(k = n)$ **do**	$\{(f = k!) \wedge \neg(k = n)\}$
	$\{f(k+1) = (k+1)!\}$
$\quad k := k + 1;$	$\{fk = k!\}$
$\quad f := f * k$	$\{f = k!\}$
od	$\{(f = k!) \wedge (k = n)\}$
	$\{f = n!\}$

Example (proof outline)

Example

	$\{\textsc{True}\}$
	$\{1 = 0!\}$
$f := 1;$	$\{f = 0!\}$
$k := 0;$	$\{f = k!\}$
while $\neg(k = n)$ **do**	$\{(f = k!) \wedge \neg(k = n)\}$
	$\{f(k+1) = (k+1)!\}$
$\quad k := k + 1;$	$\{fk = k!\}$
$\quad f := f * k$	$\{f = k!\}$
od	$\{(f = k!) \wedge (k = n)\}$
	$\{f = n!\}$

Example (proof outline)

Example

	$\{\textsc{True}\}$
	$\{1 = 0!\}$
$f := 1;$	$\{f = 0!\}$
$k := 0;$	$\{f = k!\}$
while $\neg(k = n)$ **do**	$\{(f = k!) \wedge \neg(k = n)\}$
	$\{f(k+1) = (k+1)!\}$
$\quad k := k + 1;$	$\{fk = k!\}$
$\quad f := f * k$	$\{f = k!\}$
od	$\{(f = k!) \wedge (k = n)\}$
	$\{f = n!\}$

Example (proof outline)

Example

$$\{\textsc{True}\}$$

$$\{1 = 0!\}$$

$f := 1;$ $\qquad \{f = 0!\}$

$k := 0;$ $\qquad \{f = k!\}$

while $\neg(k = n)$ **do** $\qquad \{(f = k!) \wedge \neg(k = n)\}$

$$\{f(k+1) = (k+1)!\}$$

$\quad k := k + 1;$ $\qquad \{fk = k!\}$

$\quad f := f * k$ $\qquad \{f = k!\}$

od $\qquad \{(f = k!) \wedge (k = n)\}$

$$\{f = n!\}$$

Example (proof outline)

Example

$$
\begin{array}{ll}
 & \{\textsc{True}\} \\
 & \{1 = 0!\} \\
f := 1; & \{f = 0!\} \\
k := 0; & \{f = k!\} \\
\textbf{while } \neg(k = n) \textbf{ do} & \{(f = k!) \wedge \neg(k = n)\} \\
 & \{f(k+1) = (k+1)!\} \\
\quad k := k + 1; & \{fk = k!\} \\
\quad f := f * k & \{f = k!\} \\
\textbf{od} & \{(f = k!) \wedge (k = n)\} \\
 & \{f = n!\}
\end{array}
$$

Example (proof outline)

Example

$$
\begin{array}{ll}
 & \{\textsc{True}\} \\
 & \{1 = 0!\} \\
f := 1; & \{f = 0!\} \\
k := 0; & \{f = k!\} \\
\textbf{while } \neg(k = n) \textbf{ do} & \{(f = k!) \wedge \neg(k = n)\} \\
 & \{f(k+1) = (k+1)!\} \\
\quad k := k + 1; & \{fk = k!\} \\
\quad f := f * k & \{f = k!\} \\
\textbf{od} & \{(f = k!) \wedge (k = n)\} \\
 & \{f = n!\}
\end{array}
$$

Example (proof outline)

Example

	$\{\text{TRUE}\}$
	$\{1 = 0!\}$
$f := 1;$	$\{f = 0!\}$
$k := 0;$	$\{f = k!\}$
while $\neg(k = n)$ **do**	$\{(f = k!) \wedge \neg(k = n)\}$
	$\{f(k+1) = (k+1)!\}$
$\quad k := k + 1;$	$\{fk = k!\}$
$\quad f := f * k$	$\{f = k!\}$
od	$\{(f = k!) \wedge (k = n)\}$
	$\{f = n!\}$

Example (proof outline)

Example

$$
\begin{array}{ll}
 & \{\textsc{True}\} \\
 & \{1 = 0!\} \\
f := 1; & \{f = 0!\} \\
k := 0; & \{f = k!\} \\
\textbf{while } \neg(k = n) \textbf{ do} & \{(f = k!) \wedge \neg(k = n)\} \\
 & \{f(k+1) = (k+1)!\} \\
\quad k := k + 1; & \{fk = k!\} \\
\quad f := f * k & \{f = k!\} \\
\textbf{od} & \{(f = k!) \wedge (k = n)\} \\
 & \{f = n!\}
\end{array}
$$

Example (proof outline)

Example

$$
\begin{array}{lr}
 & \{\textsc{True}\} \\
 & \{1 = 0!\} \\
f := 1; & \{f = 0!\} \\
k := 0; & \{f = k!\} \\
\textbf{while } \neg(k = n) \textbf{ do} & \{(f = k!) \wedge \neg(k = n)\} \\
 & \{f(k+1) = (k+1)!\} \\
\quad k := k + 1; & \{fk = k!\} \\
\quad f := f * k & \{f = k!\} \\
\textbf{od} & \{(f = k!) \wedge (k = n)\} \\
 & \{f = n!\}
\end{array}
$$

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:

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

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: $\models \{\varphi\}\,P\,\{\psi\}$?

If φ holds in a state of some computational model
then ψ holds in the state reached after a successful execution of P.

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 φ holds in a state of some computational model
then ψ holds in the state reached after a successful execution of P.

Informal semantics: Programs

What is a program?

A partial function mapping system states to system states

Informal semantics: Programs

What is a program?

A partial function mapping system states to system states

Informal semantics: Programs

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
- Abstract: from a mathematical perspective

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

  $\Rightarrow$ States are **logical interpretations** (Model + Environment)
  - There is only one model of interest: standard interpretations of arithmetical symbols

  $\Rightarrow$ States are fully determined by **environments**

  $\Rightarrow$ 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

  $\Rightarrow$ States are **logical interpretations** (Model + Environment)

  - There is only one model of interest: standard interpretations of arithmetical symbols

  $\Rightarrow$ States are fully determined by **environments**

  $\Rightarrow$ 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
  - $\Rightarrow$ States are **logical interpretations** (Model + Environment)
  - There is only one model of interest: standard interpretations of arithmetical symbols
  - $\Rightarrow$ States are fully determined by **environments**
  - $\Rightarrow$ 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
  - $\Rightarrow$ States are **logical interpretations** (Model + Environment)
  - There is only one model of interest: standard interpretations of arithmetical symbols
  - $\Rightarrow$ States are fully determined by **environments**
  - $\Rightarrow$ States are functions that map variables to values

Informal semantics: States and Programs

State space (ENV)

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

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

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

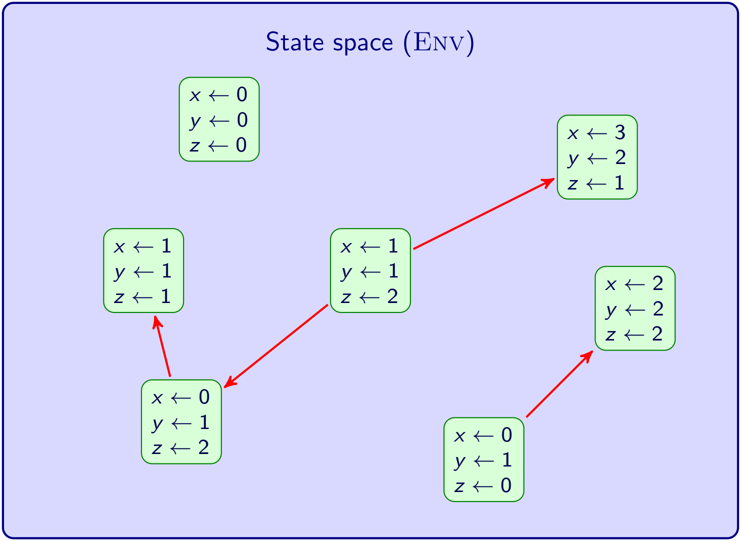
$x \leftarrow 1$
$y \leftarrow 1$
$z \leftarrow 2$

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

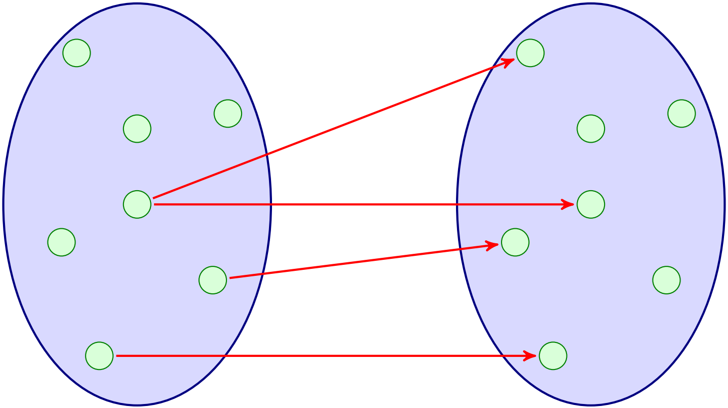
$x \leftarrow 0$
$y \leftarrow 1$
$z \leftarrow 2$

$x \leftarrow 0$
$y \leftarrow 1$
$z \leftarrow 0$

Informal semantics: States and Programs



Informal semantics: States and Programs



Semantics for $\mathcal{L}$

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

NB

An environment, η, 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 $\textsc{Env}$ in the following manner...

Semantics for $\mathcal{L}$

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

NB

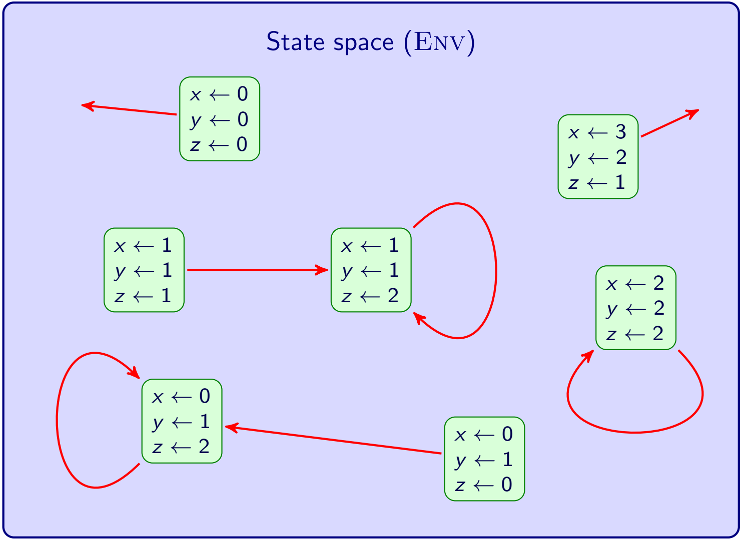
An environment, η, assigns a numeric value $[\![e]\!]^\eta$ to all expressions e, and a boolean value $[\![b]\!]^\eta$ to all boolean expressions b.

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

Assignment

$$(\eta, \eta') \in [\![x := e]\!] \quad \text{if, and only if} \quad \eta' = \eta[x \mapsto [\![e]\!]^{\eta}]$$

Assignment: $[\![z := 2]\!]$



Sequencing

$$[\![P; Q]\!] = [\![P]\!]; [\![Q]\!]$$

where, on the RHS, ; is relational composition.

Conditional, first attempt

$$[\![\textbf{if } b \textbf{ then } P \textbf{ else } Q \textbf{ fi}]\!] = \begin{cases} [\![P]\!] & \text{if } [\![b]\!]^{\eta} = \texttt{true} \\ [\![Q]\!] & \text{otherwise.} \end{cases}$$

Detour: Predicates as programs

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

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

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

$$[\![b]\!] = \{(\eta, \eta) \ : \ \eta \in \langle b \rangle\}$$

Intuitively, b corresponds to the program

$$\textbf{if } b \textbf{ then skip else } \bot \textbf{ fi}$$

Detour: Predicates as programs

A boolean expression b defines a subset (or unary relation) of $\textrm{E}\textsc{nv}$:

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

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

$$[\![b]\!] = \{(\eta, \eta) \ : \ \eta \in \langle b \rangle\}$$

Intuitively, b corresponds to the program

if b **then skip else** $\perp$ **fi**

Conditional, better attempt

$$[\![\textbf{if } b \textbf{ then } P \textbf{ else } Q \textbf{ fi}]\!] = [\![b; P]\!] \cup [\![\neg b; Q]\!]$$

While

while b **do** P **od**

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

How to do "0 or more" executions of $(b; P)$?

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 $(b; P)$?

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

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

where

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

NB

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

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

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

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

where

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

NB

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

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

While

$$[\![\textbf{while } b \textbf{ do } P \textbf{ od}]\!] = [\![b; P]\!]^*; [\![\neg b]\!]$$

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

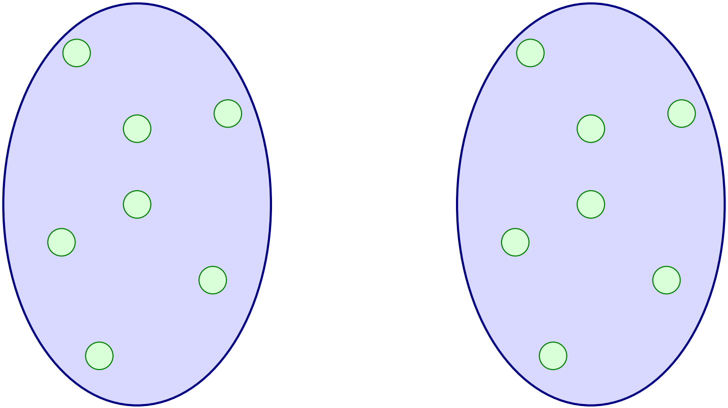
Validity

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

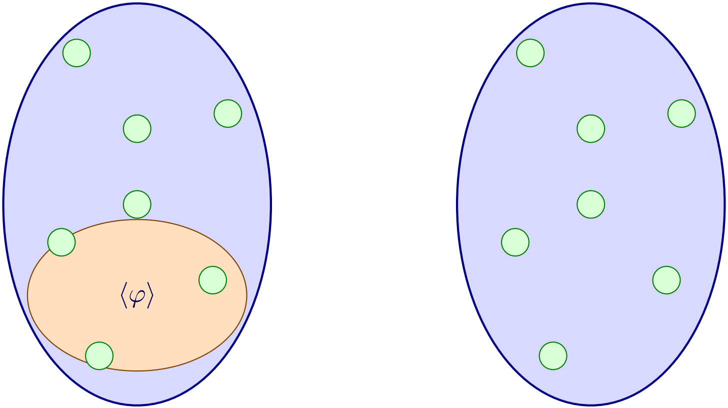
$$[\![P]\!](\langle\varphi\rangle) \subseteq \langle\psi\rangle.$$

That is, the relational image under $[\![P]\!]$ of the set of states where φ holds is contained in the set of states where ψ holds.

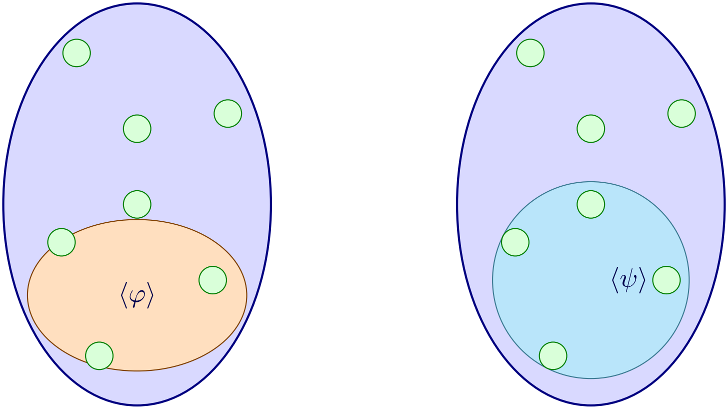
Validity



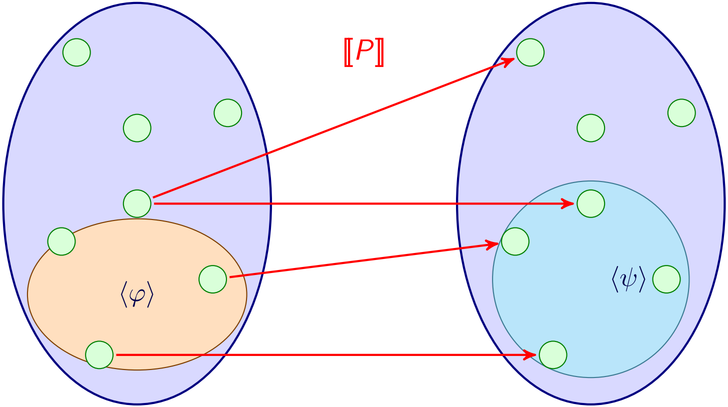
Validity



Validity



Validity



Validity