COMP2111 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, …
- Function symbols: +, ∗, …
- Predicate symbols: <, ≤, ≥, |, …

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

A boolean expression is a predicate formula over this vocabulary.
Consider the vocabulary of basic arithmetic:

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

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.

**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{align*}
f &:= 1; \\
k &:= 0; \\
\text{while } k < n \text{ do} \\
    k &:= k + 1; \\
    f &:= f \ast k \\
\text{od}
\end{align*}
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 \( \varphi \) holds in a state of some computational model
then \( \psi \) 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

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

(ass)

Intuition:
If \(x\) has property \(\phi\) after executing the assignment; then \(e\) must have property \(\phi\) before executing the assignment.
Assignment

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

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

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

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

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

(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 $Q$ (strengthen the preconditions).
While

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

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

(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.
Precondition strengthening and Postcondition weakening

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

Intuition:

- \( \varphi' \rightarrow \varphi \): \( \varphi' \) is **stronger** than \( \varphi \)
  - Stronger conditions impose more restrictions
    - States which satisfy \( \varphi' \) are a subset of states which satisfy \( \varphi \)
    - States reached after executing \( P \) are a subset
    - The postcondition will hold in the smaller set of terminal states

- \( \psi \rightarrow \psi' \): \( \psi' \) is **weaker** than \( \psi \)
  - Weaker conditions impose fewer restrictions
    - States which satisfy \( \psi \) are a subset of states which satisfy \( \psi' \)
    - States reached after executing \( P \) are a subset of those which satisfy \( \psi' \)
Example

\[
\begin{align*}
\{ & \text{True} \} \\
& f := 1; \\
& k := 0; \\
\textbf{while} & \neg (k = n) \ \textbf{do} \\
& \quad k := k + 1; \\
& \quad f := f \ast k \\
\textbf{od} \\
\{ & f = n! \}
\end{align*}
\]
Example

\{ \text{TRUE} \}
\begin{align*}
  f &:= 1; \\
  k &:= 0; \\
  \textbf{while} \ &\neg (k = n) \textbf{ do} \\
    &\quad k := k + 1; \\
    &\quad f := f \ast k \\
  \textbf{od} \\
  \{ f = n! \}
\end{align*}
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. $\{fk = k!\} f := f \ast k \{f = k!\}$ (ass)
6. $\{f(k + 1) = (k + 1)!\}$ LOOP $\{f = k!\}$ (seq) : 4, 5
7. $(f = k! \land \neg(k = n)) \rightarrow f(k + 1) = (k + 1)!$ math
8. $\{(f = k!) \land \neg(k = n)\}$ LOOP $\{f = k!\}$ (cons): 6, 7
9. $\{f = k!\}$ while . . . od $\{(f = k! \land (k = n))\}$ (loop): 8
10. $\{1 = 0!\}$ FACTORIAL $\{(f = k!) \land (k = n)\}$ (seq)
11. TRUE $\rightarrow$ $(1 = 0!)$ math
12. $(f = k!) \land (k = n) \rightarrow f = n!$ math
13. $\{\text{TRUE}\}$ FACTORIAL $\{f = n!\}$ (cons): 10, 11, 12
Example (full proof)

1. \( \{1 = 0!\} f := 1 \{f = 0!\} \)
   \((\text{ass})\)

2. \( \{f = 0!\} k := 0 \{f = k!\} \)
   \((\text{ass})\)

3. \( \{1 = 0!\} f := 1; k := 0 \{f = k!\} \)
   \(\text{(seq)} : 1, 2\)

4. \( \{f(k + 1) = (k + 1)!\} k := k + 1 \{fk = k!\} \)
   \((\text{ass})\)

5. \( \{fk = k!\} f := f \times k \{f = k!\} \)
   \((\text{ass})\)

6. \( \{f(k + 1) = (k + 1)!\} \text{LOOP} \{f = k!\} \)
   \(\text{(seq)} : 4, 5\)

7. \( (f = k!) \land \neg(k = n) \rightarrow f(k + 1) = (k + 1)! \)
   \(\text{math}\)

8. \( \{(f = k!) \land \neg(k = n)\} \text{LOOP} \{f = k!\} \)
   \(\text{(cons)}: 6, 7\)

9. \( \{f = k!\} \text{while} \ldots \text{od} \{(f = k!) \land (k = n)\} \)
   \(\text{(loop)}: 8\)

10. \( \{1 = 0!\} \text{FACTORIAL} \{(f = k!) \land (k = n)\} \)
    \(\text{(seq)}\)

11. \( \text{TRUE} \rightarrow (1 = 0!) \)
    \(\text{math}\)

12. \( ((f = k!) \land (k = n)) \rightarrow f = n! \)
    \(\text{math}\)

13. \( \{\text{TRUE}\} \text{FACTORIAL} \{f = n!\} \)
    \(\text{(cons)}: 10, 11, 12\)
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. \( \{ fk = k! \} \ f := f \ast k \{ f = k! \} \) (ass)
6. \( \{ f(k + 1) = (k + 1)! \} \ \text{LOOP} \{ f = k! \} \) (seq) : 4, 5
7. \( (f = k!) \land \neg (k = n) \rightarrow f(k + 1) = (k + 1)! \) math
8. \( \{ (f = k!) \land \neg (k = n) \} \ \text{LOOP} \{ f = k! \} \) (cons): 6, 7
9. \( \{ f = k! \} \ \text{while} \ldots \text{od} \ \{ (f = k!) \land (k = n) \} \) (loop): 8
10. \( \{ 1 = 0! \} \ \text{FACTORSIAL} \ \{ (f = k!) \land (k = n) \} \) (seq)
11. \( \text{TRUE} \rightarrow (1 = 0!) \) math
12. \( ((f = k!) \land (k = n)) \rightarrow f = n! \) math
13. \( \{ \text{TRUE} \} \ \text{FACTORSIAL} \ \{ f = n! \} \) (cons): 10,11,12
Example (full proof)

**Example**

1. \{1 = 0!\} \ f := 1 \ \{f = 0!\} \quad (ass)
2. \{f = 0!\} \ k := 0 \ \{f = k!\} \quad (ass)
3. \{1 = 0!\} \ f := 1; \ k := 0 \ \{f = k!\} \quad (seq) : 1, 2
4. \{f(k + 1) = (k + 1)!\} \ k := k + 1 \ \{fk = k!\} \quad (ass)
5. \{fk = k!\} \ f := f \ast k \ \{f = k!\} \quad (ass)
6. \{f(k + 1) = (k + 1)!\} \ LOOP \ \{f = k!\} \quad (seq) : 4, 5
7. \(f = k!\) \land \neg(k = n) \rightarrow f(k + 1) = (k + 1)! \quad math
8. \{(f = k!) \land \neg(k = n)\} \ LOOP \ \{f = k!\} \quad (cons): 6, 7
9. \{f = k!\} \ while \ldots \ od \ \{(f = k!) \land (k = n)\} \quad (loop): 8
10. \{1 = 0!\} \ FACTORIAL \ \{(f = k!) \land (k = n)\} \quad (seq)
11. True \rightarrow (1 = 0!) \quad math
12. \((f = k!) \land (k = n)\) \rightarrow f = n! \quad math
13. \{True\} \ FACTORIAL \ \{f = n!\} \quad (cons): 10, 11, 12
Example (full proof)

Example

1. \{1 = 0!\} f := 1 \{f = 0!\} \hspace{1cm} (ass)
2. \{f = 0!\} k := 0 \{f = k!\} \hspace{1cm} (ass)
3. \{1 = 0!\} f := 1; k := 0 \{f = k!\} \hspace{1cm} (seq) : 1, 2
4. \{f(k+1) = (k+1)!\} k := k + 1 \{fk = k!\} \hspace{1cm} (ass)
5. \{fk = k!\} f := f \ast k \{f = k!\} \hspace{1cm} (ass)
6. \{f(k+1) = (k+1)!\} LOOP \{f = k!\} \hspace{1cm} (seq) : 4, 5
7. (f = k!) \land \neg(k = n) \rightarrow f(k+1) = (k+1)! \hspace{1cm} math
8. \{(f = k!) \land \neg(k = n)\} LOOP \{f = k!\} \hspace{1cm} (cons): 6, 7
9. \{f = k!\} \textbf{while} \ldots \textbf{od} \{(f = k!) \land (k = n)\} \hspace{1cm} (loop): 8
10. \{1 = 0!\} \textbf{FACTORIAL} \{(f = k!) \land (k = n)\} \hspace{1cm} (seq)
11. \textbf{TRUE} \rightarrow (1 = 0!) \hspace{1cm} math
12. ((f = k!) \land (k = n)) \rightarrow f = n! \hspace{1cm} math
13. \{\textbf{TRUE}\} \textbf{FACTORIAL} \{f = n!\} \hspace{1cm} (cons): 10, 11, 12
Example (full proof)

Example

1. \{1 = 0!\} \ f := 1 \{f = 0!\} \hspace{1cm} (ass)
2. \{f = 0!\} \ k := 0 \{f = k!\} \hspace{1cm} (ass)
3. \{1 = 0!\} \ f := 1; \ k := 0 \{f = k!\} \hspace{1cm} (seq) : 1, 2
4. \{f(k + 1) = (k + 1)!\} \ k := k + 1 \{fk = k!\} \hspace{1cm} (ass)
5. \{fk = k!\} \ f := f \ast k \{f = k!\} \hspace{1cm} (ass)
6. \{f(k + 1) = (k + 1)!\} Loop \{f = k!\} \hspace{1cm} (seq) : 4, 5
7. \((f = k!) \land \neg(k = n)\) \rightarrow \ f(k + 1) = (k + 1)! \hspace{1cm} math
8. \{(f = k!) \land \neg(k = n)\} Loop \{f = k!\} \hspace{1cm} (cons): 6, 7
9. \{f = k!\} \textbf{while} \ldots \textbf{od} \{(f = k!) \land (k = n)\} \hspace{1cm} (loop): 8
10. \{1 = 0!\} \textbf{FACTORIAL} \{(f = k!) \land (k = n)\} \hspace{1cm} (seq)
11. \textbf{TRUE} \rightarrow (1 = 0!) \hspace{1cm} math
12. \((f = k!) \land (k = n)) \rightarrow f = n! \hspace{1cm} math
13. \{\textbf{TRUE}\} \textbf{FACTORIAL} \{f = n!\} \hspace{1cm} (cons): 10, 11, 12
Example (full proof)

1. \{1 = 0!\} \ f := 1 \{f = 0!\} \quad \text{(ass)}
2. \{f = 0!\} \ k := 0 \{f = k!\} \quad \text{(ass)}
3. \{1 = 0!\} \ f := 1; \ k := 0 \{f = k!\} \quad \text{(seq) : 1, 2}
4. \{f(k + 1) = (k + 1)!\} \ k := k + 1 \{fk = k!\} \quad \text{(ass)}
5. \{fk = k!\} \ f := f \ast k \{f = k!\} \quad \text{(ass)}
6. \{f(k + 1) = (k + 1)!\} \text{ LOOP } \{f = k!\} \quad \text{(seq) : 4, 5}
7. \(f = k!\) \land \neg (k = n) \rightarrow f(k + 1) = (k + 1)! \quad \text{math}
8. \{(f = k!) \land \neg (k = n)\} \text{ LOOP } \{f = k!\} \quad \text{(cons): 6,7}
9. \{f = k!\} \text{ while...od } \{(f = k!) \land (k = n)\} \quad \text{(loop): 8}
10. \{1 = 0!\} \text{ FACTORIAL } \{(f = k!) \land (k = n)\} \quad \text{(seq)}
11. \text{ TRUE } \rightarrow (1 = 0!) \quad \text{math}
12. \((f = k!) \land (k = n)) \rightarrow f = n! \quad \text{math}
13. \{\text{ TRUE}\} \text{ FACTORIAL } \{f = n!\} \quad \text{(cons): 10,11,12}
Example (full proof)

Example

1. \{1 = 0!\} \quad f := 1 \quad \{f = 0!\} \quad \text{(ass)}
2. \quad \{f = 0!\} \quad k := 0 \quad \{f = k!\} \quad \text{(ass)}
3. \quad \{1 = 0!\} \quad f := 1; k := 0 \quad \{f = k!\} \quad \text{(seq) : 1, 2}
4. \quad \{f(k + 1) = (k + 1)!\} \quad k := k + 1 \quad \{fk = k!\} \quad \text{(ass)}
5. \quad \{fk = k!\} \quad f := f \ast k \quad \{f = k!\} \quad \text{(ass)}
6. \quad \{f(k + 1) = (k + 1)!\} \quad \text{LOOP} \quad \{f = k!\} \quad \text{(seq) : 4, 5}
7. \quad (f = k!) \land \lnot (k = n) \rightarrow f(k + 1) = (k + 1)! \quad \text{math}
8. \quad \{(f = k!) \land \lnot (k = n)\} \quad \text{LOOP} \quad \{f = k!\} \quad \text{(cons) : 6, 7}
9. \quad \{f = k!\} \quad \text{while...od} \quad \{(f = k!) \land (k = n)\} \quad \text{(loop) : 8}
10. \quad \{1 = 0!\} \quad \text{FACTORIAL} \quad \{(f = k!) \land (k = n)\} \quad \text{(seq)} \quad \text{math}
11. \quad \text{TRUE} \rightarrow (1 = 0!) \quad \text{math}
12. \quad (f = k!) \land (k = n) \rightarrow f = n! \quad \text{math}
13. \quad \{\text{TRUE}\} \quad \text{FACTORIAL} \quad \{f = n!\} \quad \text{(cons) : 10, 11, 12}
Example (full proof)

Example

1. \{1 = 0!\} f := 1 \{f = 0!\} \quad (ass)
2. \{f = 0!\} k := 0 \{f = k!\} \quad (ass)
3. \{1 = 0!\} f := 1; k := 0 \{f = k!\} \quad (seq) : 1, 2
4. \{f(k + 1) = (k + 1)!\} k := k + 1 \{fk = k!\} \quad (ass)
5. \{fk = k!\} f := f \cdot k \{f = k!\} \quad (ass)
6. \{f(k + 1) = (k + 1)!\} \text{LOOP} \{f = k!\} \quad (seq) : 4, 5
7. (f = k!) \land \neg(k = n) \to f(k + 1) = (k + 1)! \quad \text{math}
8. \{(f = k!) \land \neg(k = n)\} \text{LOOP} \{f = k!\} \quad (cons): 6, 7
9. \{f = k!\} \textbf{while} \ldots \textbf{od} \{(f = k!) \land (k = n)\} \quad (loop): 8
10. \{1 = 0!\} \text{FACTORIAL} \{(f = k!) \land (k = n)\} \quad (seq)
11. \text{TRUE} \to (1 = 0!) \quad \text{math}
12. ((f = k!) \land (k = n)) \to f = n! \quad \text{math}
13. \{\text{TRUE}\} \text{FACTORIAL} \{f = n!\} \quad (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. \( \{ fk = k! \} f := f \times k \{ f = k! \} \) (ass)
6. \( \{ f(k+1) = (k+1)! \} \text{LOOP} \{ f = k! \} \) (seq) : 4, 5
7. \( (f = k!) \land \neg(k = n) \rightarrow f(k+1) = (k+1)! \) math
8. \( \{(f = k!) \land \neg(k = n)\} \text{LOOP} \{ f = k! \} \) (cons): 6,7
9. \( \{ f = k! \} \textbf{while} \ldots \textbf{od} \{(f = k!) \land (k = n)\} \) (loop): 8
10. \( \{1 = 0!\} \textbf{FACTORIAL} \{(f = k!) \land (k = n)\} \) (seq)
11. \( \textbf{TRUE} \rightarrow (1 = 0!) \) math
12. \( ((f = k!) \land (k = n)) \rightarrow f = n! \) math
13. \( \{ \textbf{TRUE} \} \textbf{FACTORIAL} \{ f = n! \} \) (cons): 10, 11, 12
Example (full proof)

Example

1. \{1 = 0!\} f := 1 \{f = 0!\} \quad (ass)
2. \{f = 0!\} k := 0 \{f = k!\} \quad (ass)
3. \{1 = 0!\} f := 1; k := 0 \{f = k!\} \quad (seq) : 1, 2
4. \{f(k + 1) = (k + 1)!\} k := k + 1 \{fk = k!\} \quad (ass)
5. \{fk = k!\} f := f \times k \{f = k!\} \quad (ass)
6. \{f(k + 1) = (k + 1)!\} LOOP \{f = k!\} \quad (seq) : 4, 5
7. (f = k!) \land \neg (k = n) \rightarrow f(k + 1) = (k + 1)! \quad math
8. \{(f = k!) \land \neg (k = n)\} LOOP \{f = k!\} \quad (cons): 6, 7
9. \{f = k!\} \textbf{while} \ldots \textbf{od} \{(f = k!) \land (k = n)\} \quad (loop): 8
10. \{1 = 0!\} \textsc{factorial} \{(f = k!) \land (k = n)\} \quad (seq)
11. \textsc{true} \rightarrow (1 = 0!) \quad math
12. ((f = k!) \land (k = n)) \rightarrow f = n! \quad math
13. \{\textsc{true}\} \textsc{factorial} \{f = n!\} \quad (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. \(fk = k!\) \(f := f \ast k\) \(\{f = k!\}\) (ass)
6. \(f(k + 1) = (k + 1)!\) LOOP \(\{f = k!\}\) (seq) : 4, 5
7. \((f = k!) \land \neg(k = n)\) \(\rightarrow\) \(f(k + 1) = (k + 1)!\) math
8. \((f = k!) \land \neg(k = n)\) LOOP \(\{f = k!\}\) (cons): 6,7
9. \(f = k!\) while...od \((f = k!) \land (k = n)\) (loop): 8
10. \{1 = 0!\} FACTORIAL \((f = k!) \land (k = n)\) (seq)
11. \True \rightarrow (1 = 0!\) math
12. \((f = k!) \land (k = n)\) \(\rightarrow\) \(f = n!\) math
13. \{\True\} FACTORIAL \(\{f = n!\}\) (cons): 10,11,12
Example (proof outline)

Example

\[
\begin{align*}
&f := 1; \\
&k := 0; \\
&\textbf{while } \neg(k = n) \textbf{ do} \\
&\quad k := k + 1; \\
&\quad f := f \ast k \\
&\textbf{od}
\end{align*}
\]

\[
\begin{align*}
&\{\text{ TRUE} \} \\
&\{1 = 0! \} \\
&\{f = 0! \} \\
&\{f = k! \} \\
&\{(f = k!) \land \neg(k = n) \} \\
&\{f(k + 1) = (k + 1)! \} \\
&\{fk = k! \} \\
&\{f = k! \} \\
&\{(f = k!) \land (k = n) \} \\
&\{f = n! \} \\
\end{align*}
\]
Example (proof outline)

Example

\[
\begin{align*}
\text{Example} &\quad \{\text{TRUE}\} \\
&\quad \{1 = 0!\} \\
\text{f} &\quad := 1; \quad \{f = 0!\} \\
\text{k} &\quad := 0; \quad \{f = k!\} \\
\text{while} \quad \neg (k = n) \quad \text{do} &\quad \{(f = k!) \land \neg (k = n)\} \\
&\quad (f(k + 1) = (k + 1)! \land f = k!\} \\
\text{k} &\quad := k + 1; \quad \{f(k + 1) = (k + 1)!\} \\
\text{f} &\quad := f \ast k \quad \{fk = k!\} \\
\text{od} &\quad \{(f = k!) \land (k = n)\} \\
&\quad \{f = n!\}
\end{align*}
\]
Example (proof outline)

Example

\[
f := 1;
k := 0;
\text{while } \neg (k = n) \text{ do}
\]

\[
\begin{align*}
k & := k + 1; \\
f & := f \ast k
\end{align*}
\]

\od

\[
\{ \text{TRUE} \}
\]
\[
\{ 1 = 0! \}
\]
\[
\{ f = 0! \}
\]
\[
\{ f = k! \}
\]
\[
\{ f(k + 1) = (k + 1)! \}
\]
\[
\{ fk = k! \}
\]
\[
\{ f = k! \}
\]
\[
\{ (f = k!) \land \neg (k = n) \}
\]
\[
\{ (f = k!) \land (k = n) \}
\]
\[
\{ f = n! \} \]
Example (proof outline)

Example

\[
\begin{align*}
  f &:= 1; \\
  k &:= 0; \\
  \text{while } &\neg (k = n) \text{ do } \\
    k &:= k + 1; \\
    f &:= f \ast k \\
  \text{od}
\end{align*}
\]

\[
\begin{align*}
  \{ \text{TRUE} \} &
  \{ 1 = 0! \} \\
  \{ f = 0! \} &
  \{ f = k! \}
\end{align*}
\]

\[
\begin{align*}
  \{ (f = k!) \land \neg (k = n) \} &
  \{ f(k + 1) = (k + 1)! \}
\end{align*}
\]

\[
\begin{align*}
  \{ fk = k! \} &
  \{ f = k! \}
\end{align*}
\]

\[
\begin{align*}
  \{ (f = k!) \land (k = n) \} &
  \{ f = n! \}
\end{align*}
\]
Example (proof outline)

Example

\[
\begin{align*}
    f & := 1; & \{ \text{TRUE} \} \\
    k & := 0; & \{ 1 = 0! \} \\
    \text{while } \neg (k = n) \textbf{ do} & \{ f = 0! \} \\
        & \{ f = k! \} \\
        & \{(f = k!) \land \neg (k = n)\} \\
        & \{ f(k + 1) = (k + 1)! \} \\
        & \{ fk = k! \} \\
        & \{ f = k! \} \\
        & \{(f = k!) \land (k = n)\} \\
        & \{ f = n! \} \\
    \text{od} & \\
    k & := k + 1; & \\
    f & := f \ast k & \\
\end{align*}
\]
Example (proof outline)

Example

\[
\begin{align*}
\text{True} & \quad \{1 = 0!\} \\
\text{f} & := 1; \quad \{f = 0!\} \\
\text{k} & := 0; \quad \{f = k!\} \\
\text{while } \neg(k = n) \text{ do} & \quad \{(f = k!) \land \neg(k = n)\} \\
\text{k} & := k + 1; \quad \{f_k = k!\} \\
\text{f} & := f \ast k \quad \{f = k!\} \\
\text{od} & \quad \{(f = k!) \land (k = n)\} \\
\text{f} & := f \ast k \quad \{f = n!\}
\end{align*}
\]
Example (proof outline)

Example

<table>
<thead>
<tr>
<th>Statement</th>
<th>Invariants</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f := 1;$</td>
<td>${\text{TRUE}}$</td>
</tr>
<tr>
<td>$k := 0;$</td>
<td>${1 = 0!}$</td>
</tr>
<tr>
<td><strong>while</strong> $\neg(k = n)$ <strong>do</strong></td>
<td>${f = 0!}$</td>
</tr>
<tr>
<td>$k := k + 1;$</td>
<td>${f = k!}$</td>
</tr>
<tr>
<td>$f := f \times k$</td>
<td>${(f = k!) \land \neg(k = n)}$</td>
</tr>
<tr>
<td><strong>od</strong></td>
<td>${f(k + 1) = (k + 1)!}$</td>
</tr>
<tr>
<td></td>
<td>${f(k + 1) = (k + 1)!}$</td>
</tr>
<tr>
<td></td>
<td>${f(k + 1) = (k + 1)!}$</td>
</tr>
<tr>
<td></td>
<td>${fk = k!}$</td>
</tr>
<tr>
<td></td>
<td>${(f = k!) \land (k = n)}$</td>
</tr>
<tr>
<td></td>
<td>${(f = k!) \land (k = n)}$</td>
</tr>
<tr>
<td></td>
<td>${(f = k!) \land (k = n)}$</td>
</tr>
<tr>
<td></td>
<td>${f = n!}$</td>
</tr>
<tr>
<td></td>
<td>${f = n!}$</td>
</tr>
</tbody>
</table>
Example (proof outline)

<table>
<thead>
<tr>
<th>Example</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$f := 1;$</td>
<td>${\text{TRUE}}$</td>
</tr>
<tr>
<td>$k := 0;$</td>
<td>${1 = 0!}$</td>
</tr>
<tr>
<td><strong>while</strong> $\neg(k = n)$ <strong>do</strong></td>
<td>${f = 0!}$</td>
</tr>
<tr>
<td>$k := k + 1;$</td>
<td>${f = k!}$</td>
</tr>
<tr>
<td>$f := f \times k$</td>
<td>${fk = k!}$</td>
</tr>
<tr>
<td><strong>od</strong></td>
<td>${(f = k!) \land (k = n)}$</td>
</tr>
<tr>
<td></td>
<td>${f = n!}$</td>
</tr>
</tbody>
</table>
Example (proof outline)

Example

\[
\begin{align*}
f &:= 1; \quad \{ \text{TRUE} \} \\
k &:= 0; \quad \{ 1 = 0! \} \\
\textbf{while } \neg (k = n) \textbf{ do} & \quad \{ f = 0! \} \\
\quad k &:= k + 1; \quad \{ f = k! \} \\
\quad f &:= f \times k \quad \{ f(k + 1) = (k + 1)! \} \\
\textbf{od} & \quad \{(f = k!) \land \neg (k = n)\} \\
\end{align*}
\]
Example (proof outline)

**Example**

\[
\begin{align*}
f &:= 1; & \{ \text{TRUE} \} \\
k &:= 0; & \{ 1 = 0! \} \\
\textbf{while } \lnot (k = n) \textbf{ do} & \{ f = 0! \} \\
\quad k &:= k + 1; & \{ f = k! \} \\
\quad f &:= f \ast k & \{ f(k + 1) = (k + 1)! \} \\
\textbf{od} & \{ f(k + 1) = (k + 1)! \} \\
\end{align*}
\]
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:
$\models \{\varphi\} P \{\psi\}$?

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

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?

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
  - 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
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)
  - 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)
  - 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 and Programs**

State space ($\text{Env}$)

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

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

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

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

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

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

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

State space ($\text{ENV}$)

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

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

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

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

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

Given a program $P$ of $\mathcal{L}$, we define $[P]$ to be a binary relation on $\text{Env}$ in the following manner...
An **environment** or **state** is a function from variables to numeric values. We denote by \( \mathbb{Env} \) the set of all environments.

**NB**

An environment, \( \eta \), assigns a numeric value \( \llbracket e \rrbracket^\eta \) to all expressions \( e \), and a boolean value \( \llbracket b \rrbracket^\eta \) to all boolean expressions \( b \).

Given a program \( P \) of \( \mathcal{L} \), we define \( \llbracket P \rrbracket \) to be a **binary relation** on \( \mathbb{Env} \) 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 (\(\text{Env}\))

\[
\begin{align*}
\text{x} & \leftarrow 0 \\
\text{y} & \leftarrow 0 \\
\text{z} & \leftarrow 0 \\
\text{x} & \leftarrow 3 \\
\text{y} & \leftarrow 2 \\
\text{z} & \leftarrow 1 \\
\text{x} & \leftarrow 1 \\
\text{y} & \leftarrow 1 \\
\text{z} & \leftarrow 1 \\
\text{x} & \leftarrow 0 \\
\text{y} & \leftarrow 1 \\
\text{z} & \leftarrow 2 \\
\text{x} & \leftarrow 0 \\
\text{y} & \leftarrow 1 \\
\text{z} & \leftarrow 0 \\
\text{x} & \leftarrow 1 \\
\text{y} & \leftarrow 2 \\
\text{z} & \leftarrow 2 \\
\text{x} & \leftarrow 2 \\
\text{y} & \leftarrow 2 \\
\text{z} & \leftarrow 2 \\
\end{align*}
\]
Sequencing

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

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

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

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

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

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

if $b$ then skip else $\perp$ fi
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):

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

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

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

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

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

**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 $f(X) = X \cup X; R$
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
- $R^{n+1} = R^n; R$

**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

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

- Do 0 or more executions of \((b; P)\)
- Conclude with an execution of \(\neg b\)
A Hoare triple is valid, written $\models \{ \varphi \} P \{ \psi \}$ if

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

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

\[ \langle \varphi \rangle \]
Validity

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

\[ \langle \varphi \rangle \quad \text{[}P\text{]}(\langle \varphi \rangle) \quad \langle \psi \rangle \]