# L11: Hoare Logic

**Software Analysis**

IIT Guwahati · August 2026

# Example

A program is supposed to satisfy some property. Consider one statement:

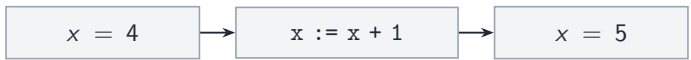
```
x := x + 1;
```


Hoare, *An Axiomatic Basis for Computer Programming*, CACM 12(10), 1969

# Example

A program is supposed to satisfy some property. Consider one statement:

```
x := x + 1;
```

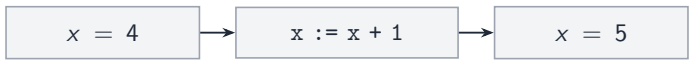



Hoare, *An Axiomatic Basis for Computer Programming*, CACM 12(10), 1969

# Example

A program is supposed to satisfy some property. Consider one statement:

```
x := x + 1;
```




How do we **state** this, and **prove** it, without running the program?

Hoare, *An Axiomatic Basis for Computer Programming*, CACM 12(10), 1969

# Program State

A **state** is the current value of every program variable: $\sigma : \mathit{Var} \to \mathbb{Z}$

$$\sigma = \{\, x \mapsto 4,\ y \mapsto 10 \,\}$$

# Program State

A **state** is the current value of every program variable: $\sigma : \mathit{Var} \to \mathbb{Z}$

$$\sigma = \{\, x \mapsto 4,\ y \mapsto 10 \,\}$$

A command transforms one state into another:


Svendsen, *Hoare Logic and Model Checking*, Cambridge

# Program State

A **state** is the current value of every program variable: $\sigma : \mathit{Var} \to \mathbb{Z}$

$$\sigma = \{\, x \mapsto 4,\ y \mapsto 10 \,\}$$

A command transforms one state into another:


Only x changes; every other variable keeps its value.

Svendsen, *Hoare Logic and Model Checking*, Cambridge

# Assertions

An **assertion** is a logical statement about a state. Examples:

$$x > 0 \qquad x = y + 1 \qquad x \geq 0 \wedge y \geq 0 \qquad x \neq y$$

Svendsen, *Hoare Logic and Model Checking*, Cambridge

# Assertions

An **assertion** is a logical statement about a state. Examples:

$$x > 0 \qquad x = y + 1 \qquad x \geq 0 \wedge y \geq 0 \qquad x \neq y$$

An assertion is true or false *in a given state*. With $\sigma = \{\, x \mapsto 4,\ y \mapsto 10 \,\}$:

$$\sigma \models x < y \qquad\qquad \sigma \not\models x > y$$

Svendsen, *Hoare Logic and Model Checking*, Cambridge

# Assertions

An **assertion** is a logical statement about a state. Examples:

$$x > 0 \qquad x = y + 1 \qquad x \geq 0 \wedge y \geq 0 \qquad x \neq y$$

An assertion is true or false *in a given state*. With $\sigma = \{\, x \mapsto 4,\ y \mapsto 10 \,\}$:

$$\sigma \models x < y \qquad\qquad \sigma \not\models x > y$$

$P \Rightarrow Q$ means: in *every* state, if $P$ holds then $Q$ holds.

Svendsen, *Hoare Logic and Model Checking*, Cambridge

# Pre- and Post-conditions


Svendsen, *Hoare Logic and Model Checking*, Cambridge

# Pre- and Post-conditions


- $P$ is what we **assume** about the initial state

Svendsen, *Hoare Logic and Model Checking*, Cambridge

# Pre- and Post-conditions


- $P$ is what we **assume** about the initial state
- $Q$ is what we **guarantee** about the final state

For `x := x + 1`: $P \equiv x = 4$ and $Q \equiv x = 5$.

Svendsen, *Hoare Logic and Model Checking*, Cambridge

# The Hoare Triple

$$\{ P \}\ C\ \{ Q \}$$

If $P$ holds before executing $C$, and $C$ terminates, then $Q$ holds afterwards.

Hoare, *An Axiomatic Basis for Computer Programming*, CACM 12(10), 1969 · Svendsen, *Hoare Logic and Model Checking*, Cambridge

# The Hoare Triple

$$\{ P \}\ C\ \{ Q \}$$

If $P$ holds before executing $C$, and $C$ terminates, then $Q$ holds afterwards.


Hoare, *An Axiomatic Basis for Computer Programming*, CACM 12(10), 1969 · Svendsen, *Hoare Logic and Model Checking*, Cambridge

# Partial Correctness

The triple $\{P\}\ C\ \{Q\}$ holds if and only if:

- whenever $C$ is executed in an initial state satisfying $P$,
- and that execution terminates,
- then the terminal state satisfies $Q$.

Svendsen, *Hoare Logic and Model Checking*, Cambridge

# Partial Correctness

The triple $\{P\}\ C\ \{Q\}$ holds if and only if:

- whenever $C$ is executed in an initial state satisfying $P$,
- and that execution terminates,
- then the terminal state satisfies $Q$.

- $\{x = 4\}\ x := x + 1\ \{x = 5\}$ **holds**

Svendsen, *Hoare Logic and Model Checking*, Cambridge

# Partial Correctness

The triple $\{P\}\ C\ \{Q\}$ holds if and only if:

- whenever $C$ is executed in an initial state satisfying $P$,
- and that execution terminates,
- then the terminal state satisfies $Q$.

- $\{x = 4\}\ x := x + 1\ \{x = 5\}$ **holds**
- $\{x = 4\}\ x := x + 1\ \{x = 6\}$ **does not hold**

Svendsen, *Hoare Logic and Model Checking*, Cambridge

# Partial Correctness

The triple $\{P\}\ C\ \{Q\}$ holds if and only if:

- whenever $C$ is executed in an initial state satisfying $P$,
- and that execution terminates,
- then the terminal state satisfies $Q$.

- $\{x = 4\}\ x := x + 1\ \{x = 5\}$ **holds**
- $\{x = 4\}\ x := x + 1\ \{x = 6\}$ **does not hold**
- $\{X = 1\}$ `while` $X > 0$ `do` $X := X + 1\ \{X = 0\}$ **holds** — the loop never terminates

Partial correctness says *nothing* about termination.

Svendsen, *Hoare Logic and Model Checking*, Cambridge

# Total Correctness

$$[P]\ C\ [Q]$$

The triple holds if and only if:

- whenever $C$ is executed in an initial state satisfying $P$,
- the execution **must terminate**,
- and the terminal state satisfies $Q$.

Svendsen, *Hoare Logic and Model Checking*, Cambridge

# Total Correctness

$$[P]\ C\ [Q]$$

The triple holds if and only if:

- whenever $C$ is executed in an initial state satisfying $P$,
- the execution **must terminate**,
- and the terminal state satisfies $Q$.

$$\text{total correctness} = \text{partial correctness} + \text{termination}$$

Svendsen, *Hoare Logic and Model Checking*, Cambridge

# How Do We Prove a Triple?

A program is built from a few constructs, so we give one **inference rule** per construct:

`skip` `x := E` $C_1; C_2$ `if` `while`

# How Do We Prove a Triple?

A program is built from a few constructs, so we give one **inference rule** per construct:

`skip` `x := E` $C_1; C_2$ `if` `while`

program → break into constructs → apply rules → proof

# How Do We Prove a Triple?

A program is built from a few constructs, so we give one **inference rule** per construct:

`skip` `x := E` $C_1; C_2$ `if` `while`

program → break into constructs → apply rules → proof

# Inference Rule Notation

$$\frac{\text{premises}}{\text{conclusion}}$$

Svendsen, *Hoare Logic and Model Checking*, Cambridge

# Inference Rule Notation

$$\frac{\text{premises}}{\text{conclusion}}$$

- Above the line: the **premises** — what we must establish first

Svendsen, *Hoare Logic and Model Checking*, Cambridge

# Inference Rule Notation

$$\frac{\text{premises}}{\text{conclusion}}$$

- Above the line: the **premises** — what we must establish first
- Below the line: the **conclusion** — what we may then conclude

Svendsen, *Hoare Logic and Model Checking*, Cambridge

# Inference Rule Notation

$$\frac{\text{premises}}{\text{conclusion}}$$

- Above the line: the **premises** — what we must establish first
- Below the line: the **conclusion** — what we may then conclude
- The horizontal line reads **"therefore"**

Svendsen, *Hoare Logic and Model Checking*, Cambridge

# Inference Rule Notation

$$\frac{\text{premises}}{\text{conclusion}}$$

- Above the line: the **premises** — what we must establish first
- Below the line: the **conclusion** — what we may then conclude
- The horizontal line reads **"therefore"**

$$\frac{A \qquad B}{C}$$

if $A$ and $B$ hold, then we may conclude $C$

Svendsen, *Hoare Logic and Model Checking*, Cambridge

# Inference Rules for Hoare Triples

$$\frac{\text{triples we have already proved}}{\text{triple we may conclude}}$$

Svendsen, *Hoare Logic and Model Checking*, Cambridge

# Inference Rules for Hoare Triples

$$\frac{\text{triples we have already proved}}{\text{triple we may conclude}}$$

$$\frac{\vdash \{P\}\ C_1\ \{R\} \qquad \vdash \{R\}\ C_2\ \{Q\}}{\vdash \{P\}\ C_1; C_2\ \{Q\}}$$

Svendsen, *Hoare Logic and Model Checking*, Cambridge

# Inference Rules for Hoare Triples

$$\frac{\text{triples we have already proved}}{\text{triple we may conclude}}$$

$$\frac{\vdash \{P\}\ C_1\ \{R\} \qquad \vdash \{R\}\ C_2\ \{Q\}}{\vdash \{P\}\ C_1; C_2\ \{Q\}}$$

If $C_1$ takes us from $P$ to $R$, and $C_2$ takes us from $R$ to $Q$, then $C_1; C_2$ takes us from $P$ to $Q$.

Svendsen, *Hoare Logic and Model Checking*, Cambridge

# Inference Rules for Hoare Triples

$$\frac{\text{triples we have already proved}}{\text{triple we may conclude}}$$

$$\frac{\vdash \{P\}\ C_1\ \{R\} \qquad \vdash \{R\}\ C_2\ \{Q\}}{\vdash \{P\}\ C_1; C_2\ \{Q\}}$$

If $C_1$ takes us from $P$ to $R$, and $C_2$ takes us from $R$ to $Q$, then $C_1; C_2$ takes us from $P$ to $Q$.

Svendsen, *Hoare Logic and Model Checking*, Cambridge

# Rule 1: `skip`

`skip` does nothing: it leaves the state unchanged.

$$x = 10,\ y = 20 \xrightarrow{\texttt{skip}} x = 10,\ y = 20$$

# Rule 1: `skip`

`skip` does nothing: it leaves the state unchanged.

$$x = 10,\ y = 20 \xrightarrow{\texttt{skip}} x = 10,\ y = 20$$

$$\frac{}{\vdash \{P\}\ \texttt{skip}\ \{P\}}$$

# Rule 1: `skip`

`skip` does nothing: it leaves the state unchanged.

$$x = 10,\ y = 20 \xrightarrow{\texttt{skip}} x = 10,\ y = 20$$

$$\frac{}{\vdash \{P\}\ \texttt{skip}\ \{P\}}$$

Whatever was true before is still true afterwards — for *every* $P$.

Svendsen, *Hoare Logic and Model Checking*, Cambridge

# Rule 2: Assignment

`x := E` evaluates $E$ in the current state and stores the result in `x`. Nothing else changes.

```
x = 100;   y = 4;
x := y + 1;          // now x = 5, y = 4
```

Svendsen, *Hoare Logic and Model Checking*, Cambridge

# Rule 2: Assignment

`x := E` evaluates $E$ in the current state and stores the result in `x`. Nothing else changes.

```
x = 100;   y = 4;
x := y + 1;          // now x = 5, y = 4
```

Now the question that gives the rule:

$$\{\,?\,\}\; x := x + 2\; \{\,x > 10\,\}$$

Svendsen, *Hoare Logic and Model Checking*, Cambridge

# Rule 2: Assignment

`x := E` evaluates $E$ in the current state and stores the result in `x`. Nothing else changes.

```
x = 100;   y = 4;
x := y + 1;          // now x = 5, y = 4
```

Now the question that gives the rule:

$$\{\,?\,\}\; x := x + 2\; \{\,x > 10\,\}$$

What must hold *before* the assignment, for $x > 10$ to hold *after* it?

Svendsen, *Hoare Logic and Model Checking*, Cambridge

# Assignment: Think Backwards

- We want $x > 10$ to hold *after* `x := x + 2`

Svendsen, *Hoare Logic and Model Checking*, Cambridge

# Assignment: Think Backwards

- We want $x > 10$ to hold *after* `x := x + 2`
- After the assignment, the value of $x$ is the old $x$ plus 2

Svendsen, *Hoare Logic and Model Checking*, Cambridge

# Assignment: Think Backwards

- We want $x > 10$ to hold *after* `x := x + 2`
- After the assignment, the value of $x$ is the old $x$ plus 2
- So we need $x + 2 > 10$ to hold *before*, that is $x > 8$

$$\{\, x > 8 \,\}\ x := x + 2\ \{\, x > 10 \,\}$$

Svendsen, *Hoare Logic and Model Checking*, Cambridge

# Assignment: Think Backwards

- We want $x > 10$ to hold *after* `x := x + 2`
- After the assignment, the value of $x$ is the old $x$ plus 2
- So we need $x + 2 > 10$ to hold *before*, that is $x > 8$

$$\{\, x > 8 \,\}\ x := x + 2\ \{\, x > 10 \,\}$$

The assignment rule is **backward reasoning**: the precondition is computed from the postcondition.

Svendsen, *Hoare Logic and Model Checking*, Cambridge

# The Assignment Rule

$$\frac{}{\vdash \{\, Q[E/x] \,\}\ x := E\ \{\, Q \,\}}$$

- $Q[E/x]$ means: replace every free occurrence of $x$ in $Q$ by $E$

Hoare, *An Axiomatic Basis for Computer Programming*, CACM 12(10), 1969 · Svendsen, *Hoare Logic and Model Checking*, Cambridge

# The Assignment Rule

$$\frac{}{\vdash \{\, Q[E/x] \,\}\ x := E\ \{\, Q \,\}}$$

- $Q[E/x]$ means: replace every free occurrence of $x$ in $Q$ by $E$
- With $Q \equiv x > 10$ and $E \equiv x + 2$: $Q[E/x] \equiv x + 2 > 10 \equiv x > 8$

Hoare, *An Axiomatic Basis for Computer Programming*, CACM 12(10), 1969 · Svendsen, *Hoare Logic and Model Checking*, Cambridge

# The Assignment Rule

$$\frac{}{\vdash \{\, Q[E/x] \,\}\ x := E\ \{\, Q \,\}}$$

- $Q[E/x]$ means: replace every free occurrence of $x$ in $Q$ by $E$
- With $Q \equiv x > 10$ and $E \equiv x + 2$: $Q[E/x] \equiv x + 2 > 10 \equiv x > 8$
- Hence $\{\, x + 2 > 10 \,\}\ x := x + 2\ \{\, x > 10 \,\}$

Hoare, *An Axiomatic Basis for Computer Programming*, CACM 12(10), 1969 · Svendsen, *Hoare Logic and Model Checking*, Cambridge

# Rule 3: Sequence

```
x := x + 1;
y := x + 1;
```

- $\{ x = 4 \}\ x := x + 1\ \{ x = 5 \}$

Svendsen, *Hoare Logic and Model Checking*, Cambridge

# Rule 3: Sequence

```
x := x + 1;
y := x + 1;
```

- $\{ x = 4 \}\ x := x + 1\ \{ x = 5 \}$
- $\{ x = 5 \}\ y := x + 1\ \{ y = 6 \}$

Svendsen, *Hoare Logic and Model Checking*, Cambridge

# Rule 3: Sequence

```
x := x + 1;
y := x + 1;
```

- $\{ x = 4 \}\ x := x + 1\ \{ x = 5 \}$
- $\{ x = 5 \}\ y := x + 1\ \{ y = 6 \}$
- therefore $\{ x = 4 \}\ x := x + 1;\ y := x + 1\ \{ y = 6 \}$

Svendsen, *Hoare Logic and Model Checking*, Cambridge

# Rule 3: Sequence

```
x := x + 1;
y := x + 1;
```

- $\{\, x = 4 \,\}\ x := x + 1\ \{\, x = 5 \,\}$
- $\{\, x = 5 \,\}\ y := x + 1\ \{\, y = 6 \,\}$
- therefore $\{\, x = 4 \,\}\ x := x + 1;\ y := x + 1\ \{\, y = 6 \,\}$

$$\frac{\vdash \{P\}\ C_1\ \{R\} \qquad \vdash \{R\}\ C_2\ \{Q\}}{\vdash \{P\}\ C_1; C_2\ \{Q\}}$$

$R$ is the **intermediate assertion**: the postcondition of $C_1$ and the precondition of $C_2$.

Svendsen, *Hoare Logic and Model Checking*, Cambridge

# Rule 4: Conditional

```
if (x > 0) y := 1; else y := 2;
```

Svendsen, *Hoare Logic and Model Checking*, Cambridge

# Rule 4: Conditional

```
if (x > 0) y := 1; else y := 2;
```

$$\frac{\vdash \{P \wedge B\}\ C_1\ \{Q\} \qquad \vdash \{P \wedge \neg B\}\ C_2\ \{Q\}}{\vdash \{P\}\ \texttt{if}\ B\ \texttt{then}\ C_1\ \texttt{else}\ C_2\ \{Q\}}$$

Svendsen, *Hoare Logic and Model Checking*, Cambridge

# Rule 5: Consequence

$$\frac{P \Rightarrow P' \qquad \vdash \{P'\}\ C\ \{Q'\} \qquad Q' \Rightarrow Q}{\vdash \{P\}\ C\ \{Q\}}$$

Svendsen, *Hoare Logic and Model Checking*, Cambridge

# Rule 5: Consequence

$$\frac{P \Rightarrow P' \qquad \vdash \{P'\}\ C\ \{Q'\} \qquad Q' \Rightarrow Q}{\vdash \{P\}\ C\ \{Q\}}$$

- A precondition may always be **strengthened**

Svendsen, *Hoare Logic and Model Checking*, Cambridge

# Rule 5: Consequence

$$\frac{P \Rightarrow P' \qquad \vdash \{P'\}\ C\ \{Q'\} \qquad Q' \Rightarrow Q}{\vdash \{P\}\ C\ \{Q\}}$$

- A precondition may always be **strengthened**
- A postcondition may always be **weakened**

Example: from $\{x > 8\}\, x := x + 2\, \{x > 10\}$ we obtain $\{x = 20\}\, x := x + 2\, \{x > 0\}$.

Svendsen, *Hoare Logic and Model Checking*, Cambridge

# Loops

```
while (x > 0) x := x - 1;
```

- Unlike `if`, the body may execute $0, 1, 2, 3, \ldots$ times

Svendsen, *Hoare Logic and Model Checking*, Cambridge

# Loops

```
while (x > 0) x := x - 1;
```

- Unlike `if`, the body may execute $0, 1, 2, 3, \ldots$ times
- The number of iterations depends on the input, so we cannot unroll the loop

Svendsen, *Hoare Logic and Model Checking*, Cambridge

# Loops

```
while (x > 0) x := x - 1;
```

- Unlike `if`, the body may execute $0, 1, 2, 3, \ldots$ times
- The number of iterations depends on the input, so we cannot unroll the loop
- We need one property that holds however many iterations have run

**The loop invariant**

Svendsen, *Hoare Logic and Model Checking*, Cambridge

# Loop Invariant Example

```
// pre:  x >= 0
while (x > 0) x := x - 1;
// post:  x == 0
```

$$I \equiv x \geq 0$$

Svendsen, *Hoare Logic and Model Checking*, Cambridge

# Loop Invariant Example

```
// pre:  x >= 0
while (x > 0) x := x - 1;
// post:  x == 0
```

$$I \equiv x \geq 0$$

- Before the loop: $x \geq 0$ is given by the precondition

Svendsen, *Hoare Logic and Model Checking*, Cambridge

# Loop Invariant Example

```
// pre:  x >= 0
while (x > 0) x := x - 1;
// post:  x == 0
```

$$I \equiv x \geq 0$$

- Before the loop: $x \geq 0$ is given by the precondition
- One iteration: from $x \geq 0$ and $x > 0$ we get $x \geq 1$, so $x - 1 \geq 0$ — the property is preserved

Svendsen, *Hoare Logic and Model Checking*, Cambridge

# Loop Invariant Example

```
// pre:  x >= 0
while (x > 0) x := x - 1;
// post:  x == 0
```

$$I \equiv x \geq 0$$

- Before the loop: $x \geq 0$ is given by the precondition
- One iteration: from $x \geq 0$ and $x > 0$ we get $x \geq 1$, so $x - 1 \geq 0$ — the property is preserved
- At exit: $\neg(x > 0)$ gives $x \leq 0$, and with $x \geq 0$ we conclude $x = 0$

Svendsen, *Hoare Logic and Model Checking*, Cambridge

# Loop Invariant Example

```
// pre:  x >= 0
while (x > 0) x := x - 1;
// post:  x == 0
```

$$I \equiv x \geq 0$$

- Before the loop: $x \geq 0$ is given by the precondition
- One iteration: from $x \geq 0$ and $x > 0$ we get $x \geq 1$, so $x - 1 \geq 0$ — the property is preserved
- At exit: $\neg(x > 0)$ gives $x \leq 0$, and with $x \geq 0$ we conclude $x = 0$

Note $x > 0$ is **not** an invariant: it is false on the last visit to the loop test.

Svendsen, *Hoare Logic and Model Checking*, Cambridge

# The While Rule

$$\frac{\vdash \{I \wedge B\}\ C\ \{I\}}{\vdash \{I\}\ \texttt{while}\ B\ \texttt{do}\ C\ \{I \wedge \neg B\}}$$

Hoare, *An Axiomatic Basis for Computer Programming*, CACM 12(10), 1969 · Svendsen, *Hoare Logic and Model Checking*, Cambridge

# The While Rule

$$\frac{\vdash \{I \wedge B\}\ C\ \{I\}}{\vdash \{I\}\ \texttt{while}\ B\ \texttt{do}\ C\ \{I \wedge \neg B\}}$$

- Premise: if $I$ holds and the guard is true, one execution of the body re-establishes $I$

Hoare, *An Axiomatic Basis for Computer Programming*, CACM 12(10), 1969 · Svendsen, *Hoare Logic and Model Checking*, Cambridge

# The While Rule

$$\frac{\vdash \{I \wedge B\}\ C\ \{I\}}{\vdash \{I\}\ \texttt{while}\ B\ \texttt{do}\ C\ \{I \wedge \neg B\}}$$

- Premise: if $I$ holds and the guard is true, one execution of the body re-establishes $I$
- Conclusion: if $I$ holds before the loop, then $I$ holds after it — and the guard is false

Hoare, *An Axiomatic Basis for Computer Programming*, CACM 12(10), 1969 · Svendsen, *Hoare Logic and Model Checking*, Cambridge

# The While Rule

$$\frac{\vdash \{I \wedge B\}\ C\ \{I\}}{\vdash \{I\}\ \texttt{while}\ B\ \texttt{do}\ C\ \{I \wedge \neg B\}}$$

- Premise: if $I$ holds and the guard is true, one execution of the body re-establishes $I$
- Conclusion: if $I$ holds before the loop, then $I$ holds after it — and the guard is false

Hoare, *An Axiomatic Basis for Computer Programming*, CACM 12(10), 1969 · Svendsen, *Hoare Logic and Model Checking*, Cambridge

# Definition: Loop Invariant

**Definition.** $I$ is a **loop invariant** of `while` $B$ `do` $C$ if and only if

$$\models \{I \wedge B\}\ C\ \{I\}$$

Svendsen, *Hoare Logic and Model Checking*, Cambridge · Pfenning & Cervesato, *Contracts*, CMU 15-122

# Definition: Loop Invariant

**Definition.** $I$ is a **loop invariant** of `while` $B$ `do` $C$ if and only if

$$\models \{I \wedge B\}\ C\ \{I\}$$

**Consequence.** If $I$ additionally holds on the first arrival at the loop head, then $I$ holds on **every** arrival at the loop head.

Svendsen, *Hoare Logic and Model Checking*, Cambridge · Pfenning & Cervesato, *Contracts*, CMU 15-122

# Definition: Loop Invariant

**Definition.** $I$ is a **loop invariant** of `while` $B$ `do` $C$ if and only if

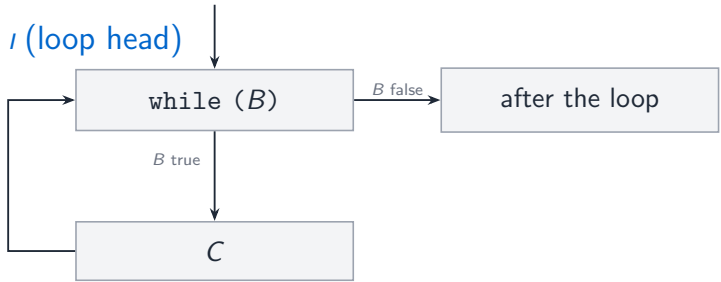
$$\models \{I \wedge B\}\ C\ \{I\}$$

**Consequence.** If $I$ additionally holds on the first arrival at the loop head, then $I$ holds on **every** arrival at the loop head.

**Loop head** = the program point immediately *before* each evaluation of the guard $B$.

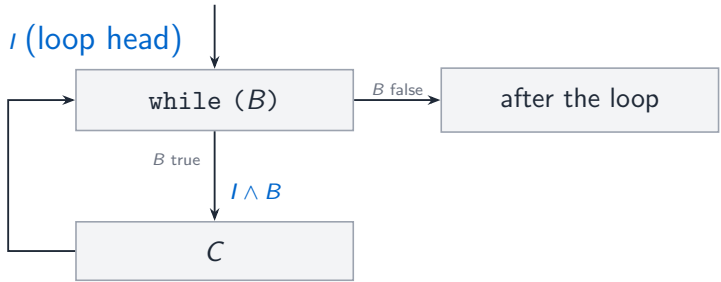
Svendsen, *Hoare Logic and Model Checking*, Cambridge · Pfenning & Cervesato, *Contracts*, CMU 15-122

# Three Points, Three Assertions



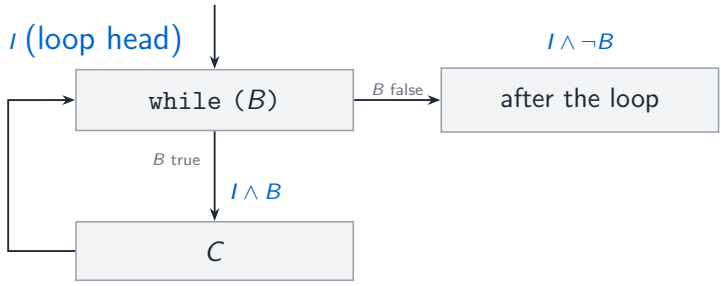

Svendsen, *Hoare Logic and Model Checking*, Cambridge

# Three Points, Three Assertions



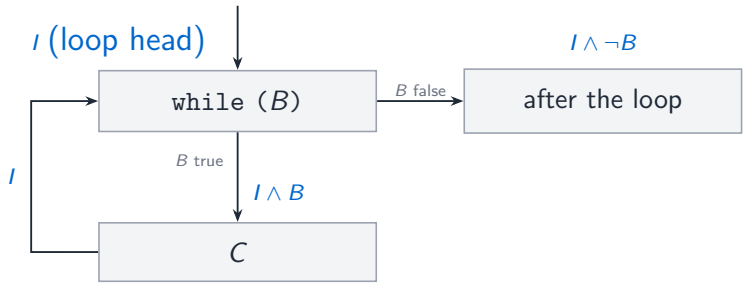

Svendsen, *Hoare Logic and Model Checking*, Cambridge

# Three Points, Three Assertions




Svendsen, *Hoare Logic and Model Checking*, Cambridge

# Three Points, Three Assertions




$I$ may be *false* in the middle of $C$; it must be true again by the end of $C$.

Svendsen, *Hoare Logic and Model Checking*, Cambridge

# Using an Invariant: Three Obligations

$$\frac{P \Rightarrow I \qquad \vdash \{I \wedge B\}\ C\ \{I\} \qquad I \wedge \neg B \Rightarrow Q}{\vdash \{P\}\ \texttt{while}\ B\ \texttt{do}\ C\ \{Q\}}$$

Svendsen, *Hoare Logic and Model Checking*, Cambridge · Pfenning & Cervesato, *Contracts*, CMU 15-122

# Using an Invariant: Three Obligations

$$\frac{P \Rightarrow I \qquad \vdash \{I \wedge B\}\ C\ \{I\} \qquad I \wedge \neg B \Rightarrow Q}{\vdash \{P\}\ \texttt{while}\ B\ \texttt{do}\ C\ \{Q\}}$$

1. **Initialisation**: $I$ holds when the loop head is first reached

Svendsen, *Hoare Logic and Model Checking*, Cambridge · Pfenning & Cervesato, *Contracts*, CMU 15-122

# Using an Invariant: Three Obligations

$$\frac{P \Rightarrow I \qquad \vdash \{I \wedge B\}\ C\ \{I\} \qquad I \wedge \neg B \Rightarrow Q}{\vdash \{P\}\ \texttt{while}\ B\ \texttt{do}\ C\ \{Q\}}$$

1. **Initialisation**: $I$ holds when the loop head is first reached
2. **Preservation**: the body re-establishes $I$

Svendsen, *Hoare Logic and Model Checking*, Cambridge · Pfenning & Cervesato, *Contracts*, CMU 15-122

# Using an Invariant: Three Obligations

$$\frac{P \Rightarrow I \qquad \vdash \{I \wedge B\}\ C\ \{I\} \qquad I \wedge \neg B \Rightarrow Q}{\vdash \{P\}\ \texttt{while}\ B\ \texttt{do}\ C\ \{Q\}}$$

1. **Initialisation**: $I$ holds when the loop head is first reached
2. **Preservation**: the body re-establishes $I$
3. **Exit**: $I$ together with $\neg B$ implies the postcondition

Svendsen, *Hoare Logic and Model Checking*, Cambridge · Pfenning & Cervesato, *Contracts*, CMU 15-122

# `for` Loop Example

```
for (int i = 0; i < 10; i++) { /* body */ }
```

is by definition the same program as

```
int i = 0;
while (i < 10) {            // loop head:  the invariant is asserted here
    /* body */
    i = i + 1;              // the update belongs to the loop body
}
// after the loop
```

Svendsen, *Hoare Logic and Model Checking*, Cambridge

# `for` Loop Example

```
for (int i = 0; i < 10; i++) { /* body */ }
```

is by definition the same program as

```
int i = 0;
while (i < 10) {            // loop head:  the invariant is asserted here
    /* body */
    i = i + 1;              // the update belongs to the loop body
}
// after the loop
```

Guard $B \equiv i < 10$; body $C \equiv \text{body}; i := i + 1$.

Svendsen, *Hoare Logic and Model Checking*, Cambridge

# What Is True at the Loop Head

| visit to the head | value of `i` | guard $i < 10$ | action |
|---|---|---|---|
| 1 | 0 | true | run body, then `i++` |
| 2 | 1 | true | run body, then `i++` |
| ⋮ | ⋮ | ⋮ | ⋮ |
| 10 | 9 | true | run body, then `i++` |
| 11 | **10** | **false** | exit the loop |

# What Is True at the Loop Head

| visit to the head | value of `i` | guard $i < 10$ | action |
|---|---|---|---|
| 1 | 0 | true | run body, then `i++` |
| 2 | 1 | true | run body, then `i++` |
| ⋮ | ⋮ | ⋮ | ⋮ |
| 10 | 9 | true | run body, then `i++` |
| 11 | **10** | **false** | exit the loop |

- At the head, `i` takes every value in $\{0, 1, \ldots, 10\}$ — **eleven** visits, not ten

# What Is True at the Loop Head

| visit to the head | value of `i` | guard $i < 10$ | action |
|---|---|---|---|
| 1 | 0 | true | run body, then `i++` |
| 2 | 1 | true | run body, then `i++` |
| ⋮ | ⋮ | ⋮ | ⋮ |
| 10 | 9 | true | run body, then `i++` |
| 11 | **10** | **false** | exit the loop |

- At the head, `i` takes every value in $\{0, 1, \ldots, 10\}$ — **eleven** visits, not ten
- So any invariant must be true when $i = 10$

# The Loop Invariant

$$I \equiv 0 \leq i \leq 10$$

- **Initialisation**: after $i := 0$ we have $0 \leq 0 \leq 10$ **holds**

Svendsen, *Hoare Logic and Model Checking*, Cambridge

# The Loop Invariant

$$I \equiv 0 \leq i \leq 10$$

- **Initialisation**: after $i := 0$ we have $0 \leq 0 \leq 10$ **holds**
- **Preservation**: $I[i+1/i] \equiv 0 \leq i+1 \leq 10 \equiv -1 \leq i \leq 9$.
  Since $I \wedge B \equiv (0 \leq i \leq 10) \wedge (i < 10)$, we have $I \wedge B \Rightarrow I[i+1/i]$. **Hence preservation holds.**

Svendsen, *Hoare Logic and Model Checking*, Cambridge

# The Loop Invariant

$$I \;\equiv\; 0 \leq i \leq 10$$

- **Initialisation**: after $i := 0$ we have $0 \leq 0 \leq 10$ **holds**
- **Preservation**: $I[\,i+1/i\,] \equiv 0 \leq i+1 \leq 10 \equiv -1 \leq i \leq 9$.
  Since $I \wedge B \equiv (0 \leq i \leq 10) \wedge (i < 10)$, we have $I \wedge B \Rightarrow I[\,i+1/i\,]$. **Hence preservation holds.**
- **Exit**: $I \wedge \neg B \equiv (0 \leq i \leq 10) \wedge (i \geq 10) \equiv i = 10$ **holds**

The exit obligation yields exactly the fact we want after the loop: $i = 10$.

Svendsen, *Hoare Logic and Model Checking*, Cambridge