## Rules of Hoare Logic:

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \vdash \text{skip} : P \Rightarrow P )</td>
<td>Empty statement preserves the precondition</td>
</tr>
<tr>
<td>( \vdash \text{abort} : P \Rightarrow Q )</td>
<td>Aborting a program changes the precondition</td>
</tr>
<tr>
<td>( \vdash x := e : P[e/x] \Rightarrow P )</td>
<td>Assignment changes the precondition based on the expression</td>
</tr>
<tr>
<td>( \vdash c; c' : P \Rightarrow Q )</td>
<td>Sequential execution preserves the precondition</td>
</tr>
<tr>
<td>( \vdash c : P \Rightarrow R )</td>
<td>Conditional preserves the precondition if the condition is true</td>
</tr>
<tr>
<td>( \vdash c' : R \Rightarrow Q )</td>
<td>Conditional preserves the precondition if the condition is false</td>
</tr>
<tr>
<td>( \vdash c ; c' : P \Rightarrow Q )</td>
<td>Sequential execution preserves the precondition</td>
</tr>
<tr>
<td>( \vdash c : P \Rightarrow Q )</td>
<td>Precondition preservation for a compound statement</td>
</tr>
</tbody>
</table>

- If \( e \) then \( c_1 \) else \( c_2 \) : \( P \Rightarrow Q \)

- \( \vdash c : e \land P \Rightarrow P \)

- \( \vdash \text{while} \ e \ \text{do} \ c : P \Rightarrow P \land \neg e \)
Rules of Hoare Logic:

- \( \vdash \text{skip} : P \Rightarrow P \)
- \( \vdash \text{abort} : P \Rightarrow Q \)
- \( \vdash x := e : P[e/x] \Rightarrow P \)
- \( \vdash c ; c' : P \Rightarrow Q \)
- \( \vdash c : P \Rightarrow R \)
- \( \vdash c' : R \Rightarrow Q \)
- \( \vdash c ; c' : P \Rightarrow Q \)
- \( \vdash c : P \Rightarrow Q \)
- \( \vdash \text{if } e \text{ then } c_1 \text{ else } c_2 : P \Rightarrow Q \)
- \( \vdash c : e \land P \Rightarrow P \)
- \( \vdash \text{while } e \text{ do } c : P \Rightarrow P \land \neg e \)
Rules of Hoare Logic:

\[ \begin{align*}
\vdash \text{skip} : P & \Rightarrow P \\
\vdash \text{abort} : P & \Rightarrow Q \\
\vdash x := e : P[e/x] & \Rightarrow P \\
\vdash c; c' & : P \\
\vdash c & : R \\
\vdash c' & : R \\
\vdash c; c' & : P \\
\vdash P & \Rightarrow S \\
\vdash c & : S \\
\vdash c & : R \\
\vdash c & : P \\
\vdash \text{if } e \text{ then } c_1 \text{ else } c_2 & : P \\
\vdash \text{while } e \text{ do } c & : P \\
\end{align*} \]
Weakest precondition calculus
Predicate Transformer Semantics

Given a program \( c \) and an assertion \( P \) we can define an assertion \( \text{wp}(c, P) \) which is the weakest precondition of \( c \) and \( P \), i.e. \( c : \text{wp}(c, P) \Rightarrow P \) is a valid triple, and for every triple \( c : Q \Rightarrow P \) we have \( Q \Rightarrow \text{wp}(c, P) \)
Weakest precondition

This is defined on the structure of commands:

\[
\begin{align*}
wp(\text{abort}, P) & = \text{false} \\
wp(\text{skip}, P) & = P \\
wp(x:=e, P) & = P[x\leftarrow\{e\}_m] \\
wp(c;c', P) & = wp(c, wp(c', P)) \\
wp(\text{if } e \text{ then } c_t \text{ else } c_f, P) & = (e \Rightarrow wp(c_t, P)) \land (\neg e \Rightarrow wp(c_t, P)) \\
wp(\text{while } e \text{ do } c, P) & = \exists n \in \text{Nat} \ P_n \text{ where}
\end{align*}
\]
Weakest precondition

This is defined on the structure of commands:

\[ wp(\text{abort}, P) = \text{false} \]
\[ wp(\text{skip}, P) = P \]
\[ wp(x := e, P) = P[x \leftarrow \{e\}_m] \]
\[ wp(c; c', P) = wp(c, wp(c', P)) \]
\[ wp(\text{if } e \text{ then } c_t \text{ else } c_f, P) = (e \Rightarrow wp(c_t, P)) \land (\neg e \Rightarrow wp(c_f, P)) \]
\[ wp(\text{while } e \text{ do } c, P) = \exists n \in \mathbb{Nat} \; P_n \text{ where} \]
\[ P_0 = \neg e \land P \]
\[ P_{n+1} = e \land wp(c, P_n) \]
Security as information flow control
Some Examples of Security Properties

• Access Control
• Encryption
• Malicious Behavior Detection
• Information Filtering
• Information Flow Control
Some Examples of Security Properties

• Access Control
• Encryption
• Malicious Behavior Detection
• Information Filtering
• Information Flow Control
Private vs Public

We want to distinguish confidential information that need to be kept secret from nonconfidential information that can be accessed by everyone.

We assume that every variable is tagged with one either public or private.

\[
\text{x: public} \quad \text{x: private}
\]
Information Flow Control

We want to guarantee that confidential information do not flow in what is considered nonconfidential.
We want to guarantee that confidential information do not flow in what is considered nonconfidential.
Is this program secure?

\[x: \text{private}\]
\[y: \text{public}\]
\[x := y\]
Is this program secure?

\[
x: \text{private} \\
y: \text{public} \\
x := y
\]
Is this program secure?

\[
\begin{align*}
x : & \text{private} \\
y : & \text{public} \\
y : = x
\end{align*}
\]
Is this program secure?

```
x: private
y: public

y := x
```

Insecure
Is this program secure?

x: private
y: public

y := x;
y := 5
Is this program secure?

x: private
y: public

y := x;
y := 5

Secure
Is this program secure?

x: private
y: public

if y mod 3 = 0 then
  x := 1
else
  x := 0
Is this program secure?

x: private
y: public

if y mod 3 = 0 then
  x := 1
else
  x := 0

Secure
Is this program secure?

\[
\begin{align*}
x & : \text{private} \\
y & : \text{public} \\
\text{if } x \mod 3 = 0 \text{ then } & y := 1 \\
\text{else } & y := 0
\end{align*}
\]
Is this program secure?

\[
\begin{align*}
x & : \text{private} \\
y & : \text{public}
\end{align*}
\]

\[
\text{if } x \mod 3 = 0 \text{ then } \\
y := 1 \\
\text{else} \\
y := 0
\]

Insecure
How can we formulate a policy that forbids flows from private to public?
Low equivalence

Two memories $m_1$ and $m_2$ are low equivalent if and only if they coincide in the value that they assign to public variables.

In symbols: $m_1 \sim_{\text{low}} m_2$
Noninterference

A program \textit{prog} is \textit{noninterferent} if and only if, whenever we run it on two memories \( m_1 \) and \( m_2 \) that are \textit{low equivalent}, we obtain two memories \( m_1' \) and \( m_2' \) which are also \textit{low equivalent}. 
Noninterference

In symbols

\[ m_1 \sim_{\text{low}} m_2 \text{ and } \{c\} m_1 = m_1' \text{ and } m_2' \{c\} m_2 = m_2' \]

implies \[ m_1' \sim_{\text{low}} m_2' \]
Does this program satisfy noninterference?

\[
\begin{align*}
x & : \text{private} \\
y & : \text{public} \\
x & := y
\end{align*}
\]
Does this program satisfy noninterference?

\[
\begin{array}{c}
x: \text{private} \\
y: \text{public} \\
x := y
\end{array}
\]

Yes
Does this program satisfy noninterference?

\[
\begin{align*}
x &: \text{private} \\
y &: \text{public} \\
x &= y
\end{align*}
\]

\(m_{\text{in}_1} = [x = n_1, y = k]\)

Yes
Does this program satisfy noninterference?

\[ x : \text{private} \]
\[ y : \text{public} \]
\[ x := y \]

\[ m^{in_1} = [x = n_1, y = k] \]
\[ m^{in_2} = [x = n_2, y = k] \]
Does this program satisfy noninterference?

\[
\begin{align*}
x & : \text{private} \\
y & : \text{public} \\
x & := y
\end{align*}
\]

Yes

\[
\begin{align*}
m_{\text{in}1} & = [x=n_1, y=k] \\
m_{\text{in}2} & = [x=n_2, y=k] \\
m_{\text{out}1} & = [x=k, y=k] \\
m_{\text{out}2} & = [x=k, y=k]
\end{align*}
\]
Does this program satisfy noninterference?

\[
\begin{align*}
x &: \text{private} \\
y &: \text{public} \\
y &= x
\end{align*}
\]
Does this program satisfy noninterference?

\[
\begin{array}{l}
x : \text{private} \\
y : \text{public} \\
y := x
\end{array}
\]

No
Does this program satisfy noninterference?

\[
x: \text{private} \\
y: \text{public} \\
y := x
\]

\[m_{i_1} = [x = n_1, y = k]\]

No
Does this program satisfy noninterference?

\begin{align*}
\text{x: private} \\
\text{y: public} \\
\text{y:=x}
\end{align*}

\begin{align*}
\text{m}^{\text{in}_1} &= [x=n_1, y=k] \\
\text{m}^{\text{in}_2} &= [x=n_2, y=k]
\end{align*}
Does this program satisfy noninterference?

\[
x: \text{private} \\
y: \text{public} \\
y := x
\]

\[
m_{\text{in}1} = [x=n_1, y=k] \\
m_{\text{out}1} = [x=n_1, y=n_1] \\
m_{\text{in}2} = [x=n_2, y=k] \\
m_{\text{out}2} = [x=n_2, y=n_2]
\]

No
Does this program satisfy noninterference?

\[
\begin{array}{l}
x: \text{private} \\
y: \text{public} \\
y := x \\
y := 5
\end{array}
\]
Does this program satisfy noninterference?

\[
\begin{align*}
x & : \text{private} \\
y & : \text{public} \\
y & := x \\
y & := 5
\end{align*}
\]

Yes
Does this program satisfy noninterference?

\[ x: \text{private} \]
\[ y: \text{public} \]
\[ y := x \]
\[ y := 5 \]

\[ m^1_{in} = [x = n_1, y = k] \]

Yes
Does this program satisfy noninterference?

\[
\begin{align*}
x &: \text{private} \\
y &: \text{public} \\
y &= x \\
y &= 5
\end{align*}
\]

\[
\begin{align*}
m^{\text{in}}_{1} &= [x=n_1, y=k] \\
m^{\text{in}}_{2} &= [x=n_2, y=k]
\end{align*}
\]

Yes
Does this program satisfy noninterference?

\[
\begin{align*}
x & : \text{private} \\
y & : \text{public} \\
y & := x \\
y & := 5
\end{align*}
\]

\[
\begin{align*}
m_{\text{in}1} & = [x = n_1, y = k] \\
m_{\text{out}1} & = [x = n_1, y = 5] \\
m_{\text{in}2} & = [x = n_2, y = k] \\
m_{\text{out}2} & = [x = n_2, y = 5]
\end{align*}
\]

Yes
Does this program satisfy noninterference?

\[
x: \text{private} \\
y: \text{public} \\
\text{if } y \mod 3 = 0 \text{ then} \\
\hspace{1cm} x := 1 \\
\text{else} \\
\hspace{1cm} x := 0
\]
Does this program satisfy noninterference?

\[
x: \text{private} \\
y: \text{public} \\
\text{if } y \mod 3 = 0 \text{ then} \\
\quad x := 1 \\
\text{else} \\
\quad x := 0
\]

Yes
Does this program satisfy noninterference?

\[
x : \text{private} \\
y : \text{public} \\
\text{if } y \mod 3 = 0 \text{ then} \\
\quad x := 1 \\
\text{else} \\
\quad x := 0
\]

\[
\text{min}_{1} = [x=n_1, y=6]
\]

Yes
Does this program satisfy noninterference?

\[
\begin{align*}
x & : \text{private} \\
y & : \text{public} \\
\text{if } y \mod 3 = 0 \text{ then} & \\
\quad x & := 1 & \text{Yes} \\
\text{else} & \\
\quad x & := 0
\end{align*}
\]

\[m_{in_1} = [x = n_1, y = 6] \quad m_{in_2} = [x = n_2, y = 6]\]
Does this program satisfy noninterference?

\[
\begin{align*}
x &: \text{private} \\
y &: \text{public} \\
\text{if } y \mod 3 = 0 & \text{ then} \\
& \quad x := 1 \\
\text{else} \\
& \quad x := 0
\end{align*}
\]

\[
\begin{align*}
m_{\text{in}1} &= [x=n_1, y=6] \\
m_{\text{in}2} &= [x=n_2, y=6] \\
m_{\text{out}1} &= [x=1, y=6] \\
m_{\text{out}2} &= [x=1, y=6]
\end{align*}
\]

Yes
Does this program satisfy noninterference?

x: private
y: public
if \( x \mod 3 = 0 \) then
  y := 1
else
  y := 0
Does this program satisfy noninterference?

x:private
y:public
if x mod 3 = 0 then
    y:=1
else
    y:=0

No
Does this program satisfy noninterference?

\[
\begin{align*}
x & : \text{private} \\
y & : \text{public} \\
\text{if } x \mod 3 = 0 & \text{ then } y := 1 \\
\text{else } & y := 0
\end{align*}
\]

\(m_{in_1} = [x=6, y=k]\)
Does this program satisfy noninterference?

```
x:private
y:public
if x mod 3 = 0 then
    y := 1
else
    y := 0
```

m^{in_1} = [x=6, y=k]  \hspace{1cm} m^{in_2} = [x=5, y=k]

No
Does this program satisfy noninterference?

\[ x: \text{private} \]
\[ y: \text{public} \]
\[ \text{if } x \mod 3 = 0 \text{ then} \]
\[ \quad y := 1 \]
\[ \text{else} \]
\[ \quad y := 0 \]

\[ m_{\text{in}}^1 = [x=6, y=k] \]
\[ m_{\text{in}}^2 = [x=5, y=k] \]
\[ m_{\text{out}}^1 = [x=6, y=1] \]
\[ m_{\text{out}}^2 = [x=5, y=0] \]
Does this program satisfy noninterference?

s1: public
s2: private
r: private
i: public

proc Compare (s1: list[n] bool, s2: list[n] bool)
i := 0;
r := 0;
while i < n ∧ r = 0 do
  if not (s1[i] = s2[i]) then
    r := 1
  i := i + 1
Does this program satisfy noninterference?

s1: public
s2: private
r: private
i: public

proc Compare (s1: list[n] bool, s2: list[n] bool)
i := 0;
r := 0;
while i < n ∧ r = 0 do
  if not (s1[i] = s2[i]) then
    r := 1
  i := i + 1

No
How can we prove our programs noninterferent?
Noninterference

In symbols, c is noninterferent if and only if for every $m_1 \sim_{\text{low}} m_2$:

1) $\{c\}_{m_1} = \bot$ iff $\{c\}_{m_2} = \bot$

2) $\{c\}_{m_1} = m_1'$ and $\{c\}_{m_2} = m_2'$ implies $m_1' \sim_{\text{low}} m_2'$

Is this condition easy to check?
Can we use the tool we studied so far?

\[ c : P \Rightarrow Q \]
Validity of Hoare triple

We say that the triple $c : P \Rightarrow Q$ is valid if and only if for every memory $m$ such that $P(m)$ and memory $m'$ such that $\{c\}_m = m'$ we have $Q(m')$. 
Validity of Hoare triple
We say that the triple $c : P \Rightarrow Q$ is valid if and only if for every memory $m$ such that $P(m)$ and memory $m'$ such that $\{c\}_{m} = m'$ we have $Q(m')$.

Validity talks only about one memory. How can we manage two memories?
Relational Property

In symbols, c is noninterferent if and only if for every $m_1 \sim_{\text{low}} m_2$:

1) $\{c\}_{m_1} = \bot$ iff $\{c\}_{m_2} = \bot$

2) $\{c\}_{m_1} = m_1'$ and $\{c\}_{m_2} = m_2'$ implies $m_1' \sim_{\text{low}} m_2'$
Relational Property

In symbols, c is noninterferent if and only if for every $m_1 \sim_{low} m_2$:

1) $\{c\}_{m_1} = \bot$ iff $\{c\}_{m_2} = \bot$

2) $\{c\}_{m_1} = m_1'$ and $\{c\}_{m_2} = m_2'$ implies $m_1' \sim_{low} m_2'$
Relational Property

In symbols, c is noninterferent if and only if for every $m_1 \sim_{low} m_2$:

1) $\{c\}_{m_1} = \bot$ iff $\{c\}_{m_2} = \bot$

2) $\{c\}_{m_1} = m_1'$ and $\{c\}_{m_2} = m_2'$ implies $m_1' \sim_{low} m_2'$
Relational Property

In symbols, c is noninterferent if and only if for every $m_1 \sim_{\text{low}} m_2$:

1) $\{c\}_{m_1} = \bot$ iff $\{c\}_{m_2} = \bot$

2) $\{c\}_{m_1} = m_1'$ and $\{c\}_{m_2} = m_2'$ implies $m_1' \sim_{\text{low}} m_2'$
Relational Property

In symbols, c is noninterferent if and only if for every $m_1 \sim_{\text{low}} m_2$:

1) $\{c\}_{m_1} = \perp$ iff $\{c\}_{m_2} = \perp$

2) $\{c\}_{m_1} = m_1'$ and $\{c\}_{m_2} = m_2'$ implies $m_1' \sim_{\text{low}} m_2'$
Relational Property

In symbols, c is noninterferent if and only if for every $m_1 \sim_{\text{low}} m_2$:

1) $\{c\}_{m_1} = \perp$ iff $\{c\}_{m_2} = \perp$

2) $\{c\}_{m_1} = m_1'$ and $\{c\}_{m_2} = m_2'$ implies $m_1' \sim_{\text{low}} m_2'$
Relational Property

In symbols, $c$ is **noninterferent** if and only if for every $m_1 \sim_{\text{low}} m_2$:

1) $\{c\}_{m_1} = \bot$ iff $\{c\}_{m_2} = \bot$

2) $\{c\}_{m_1} = m_1'$ and $\{c\}_{m_2} = m_2'$ implies $m_1' \sim_{\text{low}} m_2'$

![Diagram showing the relational property with nodes and arrows labeled as private or public, and the envelope $C$.]
Relational Hoare Logic - RHL

$\text{Precondition} \quad \text{Program}_1 \sim \text{Program}_2 \quad \text{Postcondition}$

$c_1 \sim c_2 : P \Rightarrow Q$

Precondition (a logical formula)

Program

Program

Postcondition (a logical formula)
Relational Assertions

\[ c_1 \sim c_2 : P \Rightarrow Q \]

Need to talk about variables of the two memories
Relational Assertions

\[ c_1 \sim c_2 : P \Rightarrow Q \]

Need to talk about variables of the two memories

\[ c_1 \sim c_2 : x\langle 1 \rangle \leq x\langle 2 \rangle \Rightarrow x\langle 1 \rangle \geq x\langle 2 \rangle \]
Relational Assertions

\[ c_1 \sim c_2 : P \Rightarrow Q \]

Need to talk about variables of the two memories

\[ c_1 \sim c_2 : x^{(1)} \leq x^{(2)} \Rightarrow x^{(1)} \geq x^{(2)} \]

Tags describing which memory we are referring to.
Validity of Hoare quadruple

We say that the quadruple $c_1 \sim c_2 : P \Rightarrow Q$ is valid if and only if for every pair of memories $m_1, m_2$ such that $P(m_1, m_2)$ we have:

1) $\{c_1\}_{m_1} = \bot$ iff $\{c_2\}_{m_2} = \bot$

2) $\{c_1\}_{m_1} = m_1'$ and $\{c_2\}_{m_2} = m_2'$ implies $Q(m_1', m_2')$. 
Validity of Hoare quadruple

We say that the quadruple $c_1 \sim c_2 : P \Rightarrow Q$ is valid if and only if for every pair of memories $m_1, m_2$ such that $P(m_1, m_2)$ we have:

1) $\{c_1\}_{m_1} = \bot$ iff $\{c_2\}_{m_2} = \bot$
2) $\{c_1\}_{m_1} = m_1'$ and $\{c_2\}_{m_2} = m_2'$ implies $Q(m_1', m_2')$.

Is this easy to check?
Rules of Relational Hoare Logic

Skip

\[ \vdash \text{skip} \sim \text{skip} : P \Rightarrow P \]
Correctness of an axiom

We say that an axiom is correct if we can prove the validity of each instance of the conclusion.
Correctness of an axiom

We say that an axiom is correct if we can prove the validity of each instance of the conclusion.

Is this still good for RHL?
Correctness of Skip Rule

\[ \vdash \text{skip}\sim\text{skip}: P \Rightarrow P \]

To show this rule correct we need to show the validity of the quadruple \textit{skip}\sim\text{skip}: P \Rightarrow P.
Correctness of Skip Rule

\[ \neg \text{skip} \rightarrow \text{skip} : P \Rightarrow P \]

To show this rule correct we need to show the validity of the quadruple \( \text{skip} \rightarrow \text{skip} : P \Rightarrow P \).

For every \( m_1, m_2 \) such that \( P(m_1, m_2) \) and \( m_1', m_2' \) such that \( \{ \text{skip} \}_{m_1} = m_1' \) and \( \{ \text{skip} \}_{m_2} = m_2' \), we need \( P(m_1', m_2') \).
Correctness of Skip Rule

\[ \vdash \text{skip~skip}: P \Rightarrow P \]

To show this rule correct we need to show the validity of the quadruple \( \text{skip~skip}: P \Rightarrow P \).

For every \( m_1, m_2 \) such that \( P(m_1, m_2) \) and \( m_1', m_2' \) such that \( \{\text{skip}\}_{m_1} = m_1' \) and \( \{\text{skip}\}_{m_2} = m_2' \) we need \( P(m_1', m_2') \).

Follow easily by our semantics:

\( \{\text{skip}\}_m = m \)
Rules of Relational Hoare Logic

Abort

\[ \vdash \neg \text{abort} \Rightarrow \neg \text{abort} : \text{true} \Rightarrow \text{false} \]
Rules of Relational Hoare Logic
Abort

\[ \vdash \text{abort} \sim \text{abort} : \text{true} \Rightarrow \text{false} \]

To show this rule correct we need to show the validity of the quadruple \( \text{abort} \sim \text{abort} : T \Rightarrow F \).
Rules of Relational Hoare Logic

Abort

\[ \frac{}{\text{abort} \leadsto \text{abort} : \text{true} \Rightarrow \text{false}} \]

To show this rule correct we need to show the validity of the quadruple \( \text{abort} \leadsto \text{abort} : \text{true} \Rightarrow \text{false} \).

For every \( m_1, m_2 \) such that \( P(m_1, m_2) \) we can show \( \{ \text{abort} \}_{m_1} = \bot \) iff \( \{ \text{abort} \}_{m_2} = \bot \).
Rules of Relational Hoare Logic

Abort

⊢ abort~abort : true ⇒ false

To show this rule correct we need to show the validity of the quadruple abort~abort : T ⇒ F.

For every $m_1, m_2$ such that $P(m_1, m_2)$ we can show \( \{\text{abort}\}_{m_1} = \bot \) iff \( \{\text{abort}\}_{m_2} = \bot \).

Follow easily by our semantics:

\( \{\text{abort}\}_{m} = \bot \)
Rules of Relational Hoare Logic
Assignment

\[ \vdash x_1 := e_1 \sim x_2 := e_2 : \\
P[e_1<1>/x_1<1>, e_2<2>/x_2<2>] \Rightarrow P \]
Rules of Relational Hoare Logic Composition

\[ \vdash c_1 \sim c_2 : P \Rightarrow R \quad \vdash c_1' \sim c_2' : R \Rightarrow S \]

\[ \vdash c_1 ; c_1' \sim c_2 ; c_2' : P \Rightarrow S \]
We can weaken $P$, i.e. replace it by something that is implied by $P$. In this case $S$.

We can strengthen $Q$, i.e. replace it by something that implies $Q$. In this case $R$.  

---
Rules of Hoare Logic

If then else

\[ \vdash c_1 \sim c_2 : e_1<1> \land e_2<2> \land P \Rightarrow Q \]
\[ \vdash c_1' \sim c_2' : \neg e_1<1> \land \neg e_2<2> \land P \Rightarrow Q \]

\[ \begin{array}{c}
\text{if } e_1 \text{ then } c_1 \text{ else } c_1' \\
\vdash \sim : P \Rightarrow Q \\
\text{if } e_2 \text{ then } c_2 \text{ else } c_2'
\end{array} \]
Rules of Hoare Logic
If then else

$\vdash c_1 \neg c_2 : e_1 <1> \land e_2 <2> \land P \Rightarrow Q$

$\vdash c_1' \neg c_2' : \neg e_1 <1> \land \neg e_2 <2> \land P \Rightarrow Q$

\[
\begin{align*}
\text{if } e_1 \text{ then } c_1 \text{ else } c_1' \\
\vdash \quad \neg \\
\text{if } e_2 \text{ then } c_2 \text{ else } c_2'
\end{align*}
\]

Is this correct?
An example

if true then skip else $x := x + 1$

if false then $x := x + 1$ else skip

⊢ $\{x=n\} \Rightarrow \{x=n+1\}$

Is this a valid quadruple?
An example

if true then skip else x := x + 1

\[ \vdash \neg \quad \{x=n\} \Rightarrow \{x=n+1\} \]

if false then x := x + 1 else skip

Is this a valid quadruple? ✗
An example

if true then skip else x:=x+1

if false then x:=x+1 else skip

\[ \vdash \{ x=n \} \Rightarrow \{ x=n+1 \} \]

Is this a valid quadruple? ✗

Can we prove it with the rule above? ✗
An example

```
if true then skip else x:=x+1
~
:{x=n} ⇒ {x=n+1}

if false then x:=x+1 else skip
```

Is this a valid quadruple? ✗

Can we prove it with the rule above? ✓
Rules of Relational Hoare Logic

If then else

\[ P \Rightarrow e_1 <1> = e_2 <2> \]

\[ \vdash c_1 \sim c_2 : e_1 <1> \land P \Rightarrow Q \]

\[ \vdash c'_1 \sim c'_2 : \neg e_1 <1> \land P \Rightarrow Q \]

\[ \begin{array}{c}
\dfrac{
\begin{array}{c}
\vdash e_1 \text{ then } c_1 \text{ else } c_1' \\
\vdash e_2 \text{ then } c_2 \text{ else } c_2'
\end{array}
}{
P \Rightarrow Q}
\end{array} \]
Rules of Hoare Logic

While

\( P \Rightarrow e_1 \langle 1 \rangle = e_2 \langle 2 \rangle \)

\[ \vdash c_1 \sim c_2 : e_1 \langle 1 \rangle \wedge P \Rightarrow P \]

\[ \vdash \text{while } e_1 \text{ do } c_1 \]

\[ \vdash \sim : P \Rightarrow P \wedge \neg e_1 \langle 1 \rangle \]

Invariant
How can we prove this?

\[ x \text{: private} \]
\[ y \text{: public} \]
\[ x := y \]
\[ \therefore \quad =_{\text{low}} \Rightarrow =_{\text{low}} \]
How can we prove this?

\[
x: \text{private} \\
y: \text{public} \\
y := x
\]

\[
\vdash \quad _{\text{low}} = \Rightarrow \neg ( =_{\text{low}} )
\]
How can we prove this?

\[
x: \text{private}\\y: \text{public}\\y := x\\\therefore \quad =_{\text{low}} \Rightarrow \neg (=_{\text{low}})
\]

Can we prove it?
How can we prove this?

\[
\begin{align*}
x & : \text{private} \\
y & : \text{public} \\
y & := x \\
y & := 5 \\
\therefore & \ = \text{low} \ \Rightarrow \ = \text{low}
\end{align*}
\]
How can we prove this?

\[
x: \text{private}
y: \text{public}
\]

if \( y \mod 3 = 0 \) then
  \( x := 1 \)
else
  \( x := 0 \)

\[ \Rightarrow \quad =_{\text{low}} \Rightarrow =_{\text{low}} \]
How can we prove this?

\[
\begin{align*}
x & : \text{private} \\
y & : \text{public}
\end{align*}
\]

if \( x \mod 3 = 0 \) then
\[
\begin{align*}
y & := 1
\end{align*}
\]
else
\[
\begin{align*}
y & := 1
\end{align*}
\]

\[
\vdash \text{low} \Rightarrow \text{low}
\]
How can we prove this?

\[ x : \text{private} \]
\[ y : \text{public} \]

if \( x \mod 3 = 0 \) then
  \[ y := 1 \]
else
  \[ y := 1 \]
\[ \Rightarrow \quad = \text{low} \Rightarrow = \text{low} \]

Can we prove it?
Rules of Relational Hoare Logic

If then else

\[ \text{if } e_1 \text{ then } c_1 \text{ else } c_1' \]
\[ \vdash \sim c_1' \sim c_2' : \neg e_1 \land P \Rightarrow Q \]
\[ \vdash c_1 \sim c_2 : e_1 = e_2 \]
\[ \vdash P \Rightarrow Q \]
Rules of Relational Hoare Logic

If then else - left

\[ \vdash c_1 \neg c_2 : e < 1 > \land P \Rightarrow Q \]

\[ \vdash c_1' \neg c_2 : \neg e < 1 > \land P \Rightarrow Q \]

\[ \begin{array}{c}
    \vdash \\
    \sim \\
    C_2 \\
    \vdash \neg \vdash : P \Rightarrow Q
\end{array} \]
Rules of Relational Hoare Logic
If then else - left

\[ \vdash c_1 \sim c_2 : e < 2 > \land P \Rightarrow Q \]

\[ \vdash c_1 \sim c_2' : \neg e < 2 > \land P \Rightarrow Q \]

\[ \vdash \sim \quad \begin{array}{c} c_1 \\ \text{if } e \text{ then } c_2 \text{ else } c_2' \end{array} : P \Rightarrow Q \]
How can we prove this?

\[
\begin{align*}
x & : \text{private} \\
y & : \text{public} \\
\text{if } x \mod 3 = 0 \text{ then} & \\
\quad y & := 1 \\
\text{else} & \\
\quad y & := 1 \\
\therefore & =_{\text{low}} \Rightarrow =_{\text{low}}
\end{align*}
\]
How can we prove this?

x: public
z: public
y: private

y:=0
z:=0
if x=0 then z:=1;
if z=0 then y:=1

: =low \Rightarrow =\text{low}
How can we prove this?

\[
x: \text{private} \\
z: \text{public} \\
y: \text{private}
\]

\[
y := 0 \\
z := 0 \\
\text{if } x = 0 \text{ then } z := 1; \\
\text{if } z = 0 \text{ then } y := 1
\]

\[
\vdash =_{\text{low}} \Rightarrow \neg ( =_{\text{low}} )
\]
How can we prove this?

s1: public
s2: private
r: private
i: public

proc Compare (s1:list[n] bool, s2:list[n] bool)
i := 0;
r := 0;
while i < n \( \land \) r = 0 do
  if not (s1[i] = s2[i]) then
    r := 1
  i := i + 1
: n > 0 \( \land \) =low \( \Rightarrow \) \( \neg \) (=low)
How can we prove this?

s1:public
s2:private
r:private
i:public

proc Compare (s1:list[n] bool,s2:list[n] bool)
i:=0;
r:=0;
while i<n do
  if not(s1[i]=s2[i]) then
    r:=1
    i:=i+1
: n>0 /
   =low \rightarrow \neg (=low)