

CS 511: Formal Methods in Security and Privacy

More Hoare Logic

Marco Gaboardi
gaboardi@bu.edu

Alley Stoughton
stough@bu.edu

Programming Language

```
c ::= abort
    | skip
    | x := e
    | c ; c
    | if e then c else c
    | while e do c
```

$x, y, z, \dots$ program variables

$e_1, e_2, \dots$ expressions

$c_1, c_2, \dots$ commands

Specifications - Hoare triple

Precondition
(a logical formula)



$$c : P \Rightarrow Q$$



Program



Postcondition
(a logical formula)

Precondition

Program

Postcondition

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 $Q(m')$.

Rules of Hoare Logic:

$$\frac{}{\vdash \text{skip} : P \Rightarrow P}$$

$$\frac{}{\vdash x := e : P[e/x] \Rightarrow P}$$

$$\frac{\vdash c : P \Rightarrow R \quad \vdash c' : R \Rightarrow Q}{\vdash c ; c' : P \Rightarrow Q}$$

$$\frac{P \Rightarrow S \quad \vdash c : S \Rightarrow R \quad R \Rightarrow Q}{\vdash c : P \Rightarrow Q}$$

$$\frac{\vdash c_1 : e \wedge P \Rightarrow Q \quad \vdash c_2 : \neg e \wedge P \Rightarrow Q}{\vdash \text{if } e \text{ then } c_1 \text{ else } c_2 : P \Rightarrow Q}$$

Some Instances

$$\vdash x := z * 2 : \{(z * 2) * 2 = 8\} \Rightarrow \{x * 2 = 8\}$$

$$\vdash z := x * 2 : \{x * 2 = 8\} \Rightarrow \{z = 8\}$$

$$\vdash x := z * 2; z := x * 2 : \{(z * 2) * 2 = 8\} \Rightarrow \{z = 8\}$$

Some Instances

$$\begin{array}{c}
 \overline{\{z * 4 = 8\} \Rightarrow \{(z * 2) * 2 = 8\} \quad \vdash x := z * 2 : \{(z * 2) * 2 = 8\} \Rightarrow \{x * 2 = 8\} \quad \{x * 2 = 8\} \Rightarrow \{x * 2 = 8\}} \\
 \hline
 \vdash x := z * 2 : \{z * 4 = 8\} \Rightarrow \{x * 2 = 8\} \qquad \overline{\vdash z := x * 2 : \{x * 2 = 8\} \Rightarrow \{z = 8\}} \\
 \hline
 \vdash x := z * 2; z := x * 2 : \{z * 4 = 8\} \Rightarrow \{z = 8\}
 \end{array}$$

Rules of Hoare Logic: Abort

$$\vdash \text{Abort} : \quad ? \Rightarrow ?$$

Rules of Hoare Logic: Abort

$$\vdash \text{Abort} : \quad ? \Rightarrow ?$$

What can be a good
specification?

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 $Q(m')$.

Rules of Hoare Logic:

Abort

$\vdash \text{Abort} : P \Rightarrow Q$

Rules of Hoare Logic: Abort

$$\vdash \text{Abort} : P \Rightarrow Q$$

To show this rule **correct** we need to show the **validity** $\text{Abort} : P \Rightarrow Q$ for every P, Q .

Rules of Hoare Logic: Abort

$$\vdash \text{Abort} : P \Rightarrow Q$$

To show this rule **correct** we need to show the **validity** $\text{Abort} : P \Rightarrow Q$ for every P, Q .

For every m such that $P(m)$ and m' such that $\{\text{Abort}\}_{m=m'}$ we need $Q(m')$.

Rules of Hoare Logic: Abort

$$\vdash \text{Abort} : P \Rightarrow Q$$

To show this rule **correct** we need to show the **validity** $\text{Abort} : P \Rightarrow Q$ for every P, Q .

For every m such that $P(m)$ and m' such that $\{\text{Abort}\}_{m=m'}$ we need $Q(m')$.

Vacuously True

Rules of Hoare Logic

While

$\vdash \text{while } e \text{ do } c : ??$

Rules of Hoare Logic

While

$$P \Rightarrow \neg e$$

$$\vdash \text{while } e \text{ do } c : P \Rightarrow P$$

Rules of Hoare Logic

While

$$P \Rightarrow e$$
$$\vdash c : P \Rightarrow P$$

$$\vdash \text{while } e \text{ do } c : P \Rightarrow P$$

Rules of Hoare Logic

While

$$\vdash c : e \wedge P \Rightarrow P$$

$$\vdash \text{while } e \text{ do } c : P \Rightarrow P \wedge \neg e$$

Invariant



An example

$\vdash \text{while } x = 0 \text{ do } x := x + 1$
 $\quad : \{x = 1\} \Rightarrow \{x = 1\}$

How can we derive this?

An example

$\vdash \text{while } x = 0 \text{ do } x := x + 1$
 $\quad : \{x = 1\} \Rightarrow \{x = 1\}$

What can be a good Invariant?

An example

$\vdash \text{while } x = 0 \text{ do } x := x + 1$
 $\quad : \{x = 1\} \Rightarrow \{x = 1\}$

What can be a good Invariant?

$\text{Inv} = \{x = 1\}$

An example

$\vdash \text{while } x = 0 \text{ do } x := x + 1 : \{x = 1\} \Rightarrow \{x = 1\}$

An example

$\vdash \text{while } x = 0 \text{ do } x := x + 1: \{x = 1\} \Rightarrow \{x = 1 \wedge x \neq 0\}$ $x = 1 \wedge x \neq 0 \Rightarrow x = 1$

$\vdash \text{while } x = 0 \text{ do } x := x + 1: \{x = 1\} \Rightarrow \{x = 1\}$

An example

$$\frac{x = 1 \wedge x = 0 \Rightarrow x + 1 = 1 \qquad \vdash x := x + 1 : \{x + 1 = 1\} \Rightarrow \{x = 1\}}{\vdash x := x + 1 : \{x = 1 \wedge x = 0\} \Rightarrow \{x = 1\}}$$

$$\frac{\vdash \text{while } x = 0 \text{ do } x := x + 1 : \{x = 1\} \Rightarrow \{x = 1 \wedge x \neq 0\} \qquad x = 1 \wedge x \neq 0 \Rightarrow x = 1}{\vdash \text{while } x = 0 \text{ do } x := x + 1 : \{x = 1\} \Rightarrow \{x = 1\}}$$

An example

$$\frac{x = 1 \wedge x = 0 \Rightarrow x + 1 = 1 \qquad \vdash x := x + 1 : \{x + 1 = 1\} \Rightarrow \{x = 1\}}{\vdash x := x + 1 : \{x = 1 \wedge x = 0\} \Rightarrow \{x = 1\}}$$

$$\vdash \text{while } x = 0 \text{ do } x := x + 1 : \{x = 1\} \Rightarrow \{x = 1 \wedge x \neq 0\} \qquad x = 1 \wedge x \neq 0 \Rightarrow x = 1$$

$$\vdash \text{while } x = 0 \text{ do } x := x + 1 : \{x = 1\} \Rightarrow \{x = 1\}$$

An example

$$x = 1 \wedge x = 0 \Rightarrow x + 1 = 1$$

$$\vdash x := x + 1 : \{x + 1 = 1\} \Rightarrow \{x = 1\}$$

$$\vdash x := x + 1 : \{x = 1\} \wedge x = 0 \Rightarrow \{x = 1\}$$

$$\vdash \text{while } x = 0 \text{ do } x := x + 1 : \{x = 1\} \Rightarrow \{x = 1\} \wedge x \neq 0 \quad x = 1 \wedge x \neq 0 \Rightarrow x = 1$$

$$\vdash \text{while } x = 0 \text{ do } x := x + 1 : \{x = 1\} \Rightarrow \{x = 1\}$$

Another example

$\vdash$

<pre>x := 3; y := 1; while x > 1 do y := y + 1; x := x - 1;</pre>
--

$: \{true\} \Rightarrow \{y = 3\}$

How can we derive this?

Another example

$\vdash$ `x := 3;
y := 1;
while x > 1 do
 y := y + 1;
 x := x - 1;` : $\{true\} \Rightarrow \{y = 3\}$

What can be a good Invariant?

Another example

$\vdash$ `x := 3;
y := 1;
while x > 1 do
 y := y + 1;
 x := x - 1;` : $\{true\} \Rightarrow \{y = 3\}$

What can be a good Invariant?

$$\text{Inv} = \{y = 4 - x \wedge x \geq 1\}$$

Rules of Hoare Logic:

$$\frac{}{\vdash \text{skip} : P \Rightarrow P}$$

$$\frac{}{\vdash \text{abort} : P \Rightarrow Q}$$

$$\frac{}{\vdash x := e : P[e/x] \Rightarrow P}$$

$$\frac{\vdash c : P \Rightarrow R \quad \vdash c' : R \Rightarrow Q}{\vdash c ; c' : P \Rightarrow Q}$$

$$\frac{P \Rightarrow S \quad \vdash c : S \Rightarrow R \quad R \Rightarrow Q}{\vdash c : P \Rightarrow Q}$$

$$\frac{\vdash c_1 : e \wedge P \Rightarrow Q \quad \vdash c_2 : \neg e \wedge P \Rightarrow Q}{\vdash \text{if } e \text{ then } c_1 \text{ else } c_2 : P \Rightarrow Q}$$

$$\frac{\vdash c : e \wedge P \Rightarrow P}{\vdash \text{while } e \text{ do } c : P \Rightarrow P \wedge \neg e}$$

How do we know that these
are the right rules?

Soundness

If we can derive $\vdash C : P \Rightarrow Q$ through the rules of the logic, then the triple $C : P \Rightarrow Q$ is valid.

Are the rules we presented
sound?

Completeness

If a triple $c : P \Rightarrow Q$ is valid, then
we can derive $\vdash c : P \Rightarrow Q$ through
the rules of the logic.

Are the rules we presented
complete?

Relative Completeness

 $P \Rightarrow S$ $\vdash c : S \Rightarrow R$ $R \Rightarrow Q$

 $\vdash c : P \Rightarrow Q$

Relative Completeness

$$P \Rightarrow S \qquad \vdash c : S \Rightarrow R \qquad R \Rightarrow Q$$

$$\vdash c : P \Rightarrow Q$$

If a triple $c : \text{Pre} \Rightarrow \text{Post}$ is valid, and we have an oracle to derive all the true statements of the form $P \Rightarrow S$ and of the form $R \Rightarrow Q$, which we can use in applications of the `conseq` rule, then we can derive $\vdash c : \text{Pre} \Rightarrow \text{Post}$ through the rules of the logic.

Rules of Hoare Logic:

$$\frac{}{\vdash \text{skip} : P \Rightarrow P}$$

$$\frac{}{\vdash \text{abort} : P \Rightarrow Q}$$

$$\frac{}{\vdash x := e : P[e/x] \Rightarrow P}$$

$$\frac{\vdash c : P \Rightarrow R \quad \vdash c' : R \Rightarrow Q}{\vdash c ; c' : P \Rightarrow Q}$$

$$\frac{P \Rightarrow S \quad \vdash c : S \Rightarrow R \quad R \Rightarrow Q}{\vdash c : P \Rightarrow Q}$$

$$\frac{}{\vdash \text{if } e \text{ then } c_1 \text{ else } c_2 : P \Rightarrow Q}$$

$$\frac{\vdash c : e \wedge P \Rightarrow P}{\vdash \text{while } e \text{ do } c : P \Rightarrow P \wedge \neg e}$$

Rules of Hoare Logic:

$$\frac{}{\vdash \text{skip} : P \Rightarrow P}$$

$$\frac{}{\vdash \text{abort} : P \Rightarrow Q}$$

$$\frac{}{\vdash x := e : P[e/x] \Rightarrow P}$$

$$\frac{\vdash c : P \Rightarrow R \quad \vdash c' : R \Rightarrow Q}{\vdash c ; c' : P \Rightarrow Q}$$

$$\frac{P \Rightarrow S \quad \vdash c : S \Rightarrow R \quad R \Rightarrow Q}{\vdash c : P \Rightarrow Q}$$

$$\frac{\vdash c_1 : e \wedge P \Rightarrow Q}{\vdash \text{if } e \text{ then } c_1 \text{ else } c_2 : P \Rightarrow Q}$$

$$\frac{\vdash c : e \wedge P \Rightarrow P}{\vdash \text{while } e \text{ do } c : P \Rightarrow P \wedge \neg e}$$

Rules of Hoare Logic:

$$\frac{}{\vdash \text{skip} : P \Rightarrow P}$$

$$\frac{}{\vdash \text{abort} : P \Rightarrow Q}$$

$$\frac{}{\vdash x := e : P[e/x] \Rightarrow P}$$

$$\frac{\vdash c : P \Rightarrow R \quad \vdash c' : R \Rightarrow Q}{\vdash c ; c' : P \Rightarrow Q}$$

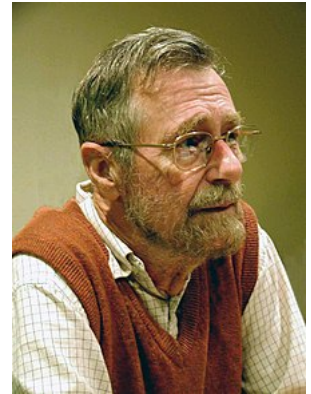
$$\frac{P \Rightarrow S \quad \vdash c : S \Rightarrow R \quad R \Rightarrow Q}{\vdash c : P \Rightarrow Q}$$

$$\frac{\vdash c_1 : e \wedge P \Rightarrow Q \quad \vdash c_2 : \neg e \wedge P \Rightarrow Q}{\vdash \text{if } e \text{ then } c_1 \text{ else } c_2 : P \Rightarrow Q}$$

$$\frac{\vdash c : e \wedge P \Rightarrow P}{\vdash \text{while } e \text{ do } c : P \Rightarrow P \wedge \neg e}$$

Weakest precondition calculus

Predicate Transformer Semantics



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

Weakest precondition

This is defined on the structure of commands:

where

Weakest precondition

This is defined on the structure of commands:

$$\text{wp}(\text{abort}, P) = \text{false}$$

where

Weakest precondition

This is defined on the structure of commands:

$$\text{wp}(\text{abort}, P) = \text{false}$$
$$\text{wp}(\text{skip}, P) = P$$

where

Weakest precondition

This is defined on the structure of commands:

$$\text{wp}(\text{abort}, P) = \text{false}$$

$$\text{wp}(\text{skip}, P) = P$$

$$\text{wp}(x := e, P) = P[x \leftarrow \{e\}_m]$$

where

Weakest precondition

This is defined on the structure of commands:

$$\text{wp}(\text{abort}, P) = \text{false}$$

$$\text{wp}(\text{skip}, P) = P$$

$$\text{wp}(x := e, P) = P[x \leftarrow \{e\}_m]$$

$$\text{wp}(c; c', P) = \text{wp}(c, \text{wp}(c', P))$$

where

Weakest precondition

This is defined on the structure of commands:

$$\text{wp}(\text{abort}, P) = \text{false}$$

$$\text{wp}(\text{skip}, P) = P$$

$$\text{wp}(x := e, P) = P[x \leftarrow \{e\}_m]$$

$$\text{wp}(c; c', P) = \text{wp}(c, \text{wp}(c', P))$$

$$\text{wp}(\text{if } e \text{ then } c_t \text{ else } c_f, P) = (e \Rightarrow \text{wp}(c_t, P)) \wedge (\neg e \Rightarrow \text{wp}(c_f, P))$$

where

Weakest precondition

This is defined on the structure of commands:

$$\text{wp}(\text{abort}, P) = \text{false}$$

$$\text{wp}(\text{skip}, P) = P$$

$$\text{wp}(x := e, P) = P[x \leftarrow \{e\}_m]$$

$$\text{wp}(c; c', P) = \text{wp}(c, \text{wp}(c', P))$$

$$\text{wp}(\text{if } e \text{ then } c_t \text{ else } c_f, P) = (e \Rightarrow \text{wp}(c_t, P)) \wedge (\neg e \Rightarrow \text{wp}(c_f, P))$$

$$\text{wp}(\text{while } e \text{ do } c, P) = \exists_{n \in \text{Nat}} P_n \text{ where}$$

$$P_0 = \neg e \wedge P$$

$$P_{n+1} = e \wedge \text{wp}(c, P_n)$$