CS 591: Formal Methods in Security and Privacy
Example in Hoare Logic and Non-interference

Marco Gaboardi
gaboardi@bu.edu

Alley Stoughton
stough@bu.edu
From the previous classes
Hoare triple

Precondition (a logical formula)

\[ c : P \Rightarrow Q \]

Program

Postcondition (a logical formula)
Programming Language

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

x, y, z, ...  program variables

e_1, e_2, ...  expressions

c_1, c_2, ...  commands
Summary of the Semantics of Commands

\{\text{abort}\}_m = \perp

\{\text{skip}\}_m = m

\{x:=e\}_m = m[x\leftarrow\{e\}_m]

\{c; c'\}_m = \{c'\}_m \quad \text{if} \quad \{c\}_m = m'

\{c; c'\}_m = \perp \quad \text{if} \quad \{c\}_m = \perp

\{\text{if } e \text{ then } c_t \text{ else } c_f\}_m = \{c_t\}_m \text{ if } \{e\}_m = \text{true}

\{\text{if } e \text{ then } c_t \text{ else } c_f\}_m = \{c_f\}_m \text{ if } \{e\}_m = \text{false}

\{\text{while } e \text{ do } c\}_m = \sup_{n \in \mathbb{N}} \{\text{while }_n e \text{ do } c\}_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 $Q(m')$.

Is this condition easy to check?
Rules of Hoare Logic

Skip

⊢ skip: P \Rightarrow P
Rules of Hoare Logic

Assignment

\[ \vdash x := e : P[e/x] \Rightarrow P \]
Rules of Hoare Logic Composition

\[ \vdash c : P \Rightarrow R \quad \vdash c' : R \Rightarrow Q \]

\[ \vdash c ; c' : P \Rightarrow Q \]
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

\[\vdash \text{if } e \text{ then } c_1 \text{ else } c_2 : P \Rightarrow Q\]
Rules of Hoare Logic

While

\[ \vdash c : e \land P \implies P \]

\[ \vdash \text{while } e \text{ do } c : P \implies P \land \neg e \]

Invariant
Today 1: More Hoare Logic
Some examples

\[
\begin{align*}
\text{x := 3;} \\
\text{y := 1;} \\
\text{while x > 1 do} \\
\text{\quad y := y+1;} \\
\text{\quad x := x-1;}
\end{align*}
\]

\[\downarrow \quad : \{true\} \Rightarrow \{y = 3\}\]

How can we derive this?
Some examples

\[ \text{true} \Rightarrow 3 = 3 \quad \vdash x := 3 : \{3 = 3\} \Rightarrow \{x = 3\} \]

\[ \vdash x := 3 : \{\text{true}\} \Rightarrow \{x = 3\} \]

\[ x = 3 \Rightarrow x = 3 \land 1 = 1 \quad \vdash y := 1 : \{x = 3 \land 1 = 1\} \Rightarrow \{x = 3 \land y = 1\} \]

\[ \vdash y := 1 : \{x = 3\} \Rightarrow \{x = 3 \land y = 1\} \]

\[ \vdash x := 3; y := 1 : \{\text{true}\} \Rightarrow \{x = 3 \land y = 1\} \]

\[ x = 3 \land y = 1 \Rightarrow x = 3 \land 1 = 1 \land y = 4 - x \]

\[ \vdash x := 3; y := 1 : \{\text{true}\} \Rightarrow \{x = 3 \land 1 = 1 \land y = 4 - x\} \]
Some examples

\[ y := y+1; \{ y + 1 = 4 - (x - 1) \land x - 1 \geq 1 \} \Rightarrow \{ y = 4 - (x - 1) \land x - 1 \geq 1 \} \]

\[ x := x-1; \{ y = 4 - (x - 1) \land x - 1 \geq 1 \} \Rightarrow \{ y = 4 - x \land x \geq 1 \} \]

\[ y := y+1; \]
\[ x := x-1 \]
\[ y = 4 - x \land x \geq 1 \land x > 1 \Rightarrow y + 1 = 4 - (x - 1) \land x - 1 \geq 1 \]

\[ y := y+1; \]
\[ x := x-1 \]
\[ \{ y = 4 - x \land x \geq 1 \land x > 1 \} \Rightarrow \{ y = 4 - x \land x \geq 1 \} \]

while \( x > 1 \) do:

\[ y := y+1; \]
\[ x := x-1 \]
\[ \{ y = 4 - x \land x \geq 1 \land \neg(x > 1) \} \Rightarrow \{ y = 4 - x \land x = 1 \} \]

while \( x > 1 \) do:

\[ y := y+1; \]
\[ x := x-1 \]
\[ \{ y = 4 - x \land x \geq 1 \land \neg(x > 1) \} \Rightarrow \{ y = 4 - x \land x = 1 \} \]
Some examples

while \( x > 1 \) do
\[
\begin{align*}
\quad & \quad \downarrow \quad y := y + 1; \quad \quad : \{y = 4 - x \land x \geq 1\} \Rightarrow \{y = 4 - x \land x = 1\} \\
\quad & \quad x := x - 1; \quad \quad \vdash \quad x = 3 \land y = 1 \land y = 4 - x \Rightarrow y = 4 - x \land x \geq 1 \\
x &= 3 \land y = 1 \land y = 4 - x \Rightarrow y = 4 - x \land x \geq 1 \\
\vdash \quad y = 4 - x \land x = 1 \Rightarrow y = 3
\end{align*}
\]

\begin{align*}
\quad & \quad \downarrow \quad y := y + 1; \quad \quad : \{x = 3 \land y = 1 \land y = 4 - x\} \Rightarrow \{y = 3\} \\
\quad & \quad x := x - 1; \quad \quad \vdash \quad x \land y = 1 \land y = 4 - x \Rightarrow \{y = 3\}
\end{align*}
Some examples

\[ \begin{align*}
\text{while } x > 1 \text{ do} \\
\quad y & := y + 1; & \{x = 3 \land y = 1 \land y = 4 - x\} \Rightarrow \{y = 3\} \\
\quad x & := x - 1;
\end{align*} \]

\[ \begin{align*}
x & := 3; \\
y & := 1; \\
\text{while } x > 1 \text{ do} \\
\quad y & := y + 1; & \{true\} \Rightarrow \{y = 3\} \\
\quad x & := x - 1;
\end{align*} \]
How do we know that these are the right rules?
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 $\langle c : P \Rightarrow Q \rangle$ is valid, then we can derive $\langle \neg c : P \Rightarrow Q \rangle$ through the rules of the logic.
Are the rules we presented complete?
Relative Completeness

\[
\begin{align*}
\vdash c : & \quad P \Rightarrow S \\
\vdash c : & \quad S \Rightarrow R \\
\vdash & \quad R \Rightarrow Q \\
\hline
\vdash c : & \quad P \Rightarrow Q
\end{align*}
\]

If a triple \( c : P \Rightarrow Q \) is valid, and we have an oracle to derive all the true statements of the form \( P \Rightarrow S \) and of the form \( R \Rightarrow Q \), then we can derive \( \vdash c : P \Rightarrow Q \) through the rules of the logic.
Today 2: security as 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.
Is this program secure?

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

Secure
Is this program secure?

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

Insecure
Is this program secure?

```
x: private
y: public

y := x;
y := 5
```

Secure
Is this program secure?

\begin{verbatim}
x: private
y: public

if y mod 3 = 0 then
  x:=1
else
  x:=0
\end{verbatim}
Is this program secure?

\[ x:private \]
\[ y:public \]

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

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