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

Program

Postcondition

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

$\{\texttt{abort}\}_m = \bot$

$\{\texttt{skip}\}_m = m$

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

$\{c;c'\}_m = \{c'\}_{m'}$ If $\{c\}_m = m'$

$\{c;c'\}_m = \bot$ If $\{c\}_m = \bot$

$\{\texttt{if e then } c_t \texttt{ else } c_f\}_m = \{c_t\}_m$ If $\{e\}_m = \texttt{true}$

$\{\texttt{if e then } c_t \texttt{ else } c_f\}_m = \{c_f\}_m$ If $\{e\}_m = \texttt{false}$

$\{\texttt{while e do c}\}_m = \sup_{n \in Nat}\{\texttt{while}_n \texttt{ e do c}\}_m$

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

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

Rules of Hoare Logic
Assignment

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

Rules of Hoare Logic
Composition

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

Rules of Hoare Logic
Consequence

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

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

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

Rules of Hoare Logic
While

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

Invariant

Today 1: More Hoare Logic

Some examples

$\vdash$

```
x:=3;
y:=1;
while x > 1 do
  y := y+1;
  x := x-1;
```

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

How can we derive this?

Some examples

$$\frac{true \Rightarrow 3 = 3 \qquad \vdash x := 3 : \{3 = 3\} \Rightarrow \{x = 3\}}{\vdash x := 3 : \{true\} \Rightarrow \{x = 3\}}$$

$$\frac{x = 3 \Rightarrow x = 3 \land 1 = 1 \qquad \vdash y := 1 : \{x = 3 \land 1 = 1\} \Rightarrow \{x = 3 \land y = 1\}}{\vdash y := 1 : \{x = 3\} \Rightarrow \{x = 3 \land y = 1\}}$$

$$\frac{}{\vdash x := 3; y := 1 : \{true\} \Rightarrow \{x = 3 \land y = 1\}}$$

$$\frac{x = 3 \land y = 1 \Rightarrow x = 3 \land 1 = 1 \land y = 4 - x}{\vdash x := 3; y := 1 : \{true\} \Rightarrow \{x = 3 \land 1 = 1 \land y = 4 - x\}}$$

Some examples

$\vdash$ `y := y+1` $: \{y+1 = 4-(x-1) \wedge x-1 \geq 1\} \Rightarrow \{y = 4-(x-1) \wedge x-1 \geq 1\}$

$\vdash$ `x := x-1` $: \{y = 4-(x-1) \wedge x-1 \geq 1\} \Rightarrow \{y = 4-x \wedge x \geq 1\}$

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

$\vdash$ $: \{y+1 = 4-(x-1) \wedge x-1 \geq 1\} \Rightarrow \{y = 4-x \wedge x \geq 1\}$

$y = 4-x \wedge x \geq 1 \wedge x > 1 \Rightarrow y+1 = 4-(x-1) \wedge x-1 \geq 1$

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

$\vdash$ $: \{y = 4-x \wedge x \geq 1 \wedge x > 1\} \Rightarrow \{y = 4-x \wedge x \geq 1\}$

```
while x > 1 do
  y := y+1;
  x := x-1
```

$\vdash$ $: \{y = 4-x \wedge x \geq 1\} \Rightarrow \{y = 4-x \wedge x \geq 1 \wedge \neg(x > 1)\}$

$\{y = 4-x \wedge x \geq 1 \wedge \neg(x > 1)\} \Rightarrow \{y = 4-x \wedge x = 1\}$

```
while x > 1 do
  y := y+1;
  x := x-1
```

$\vdash$ $: \{y = 4-x \wedge x \geq 1\} \Rightarrow \{y = 4-x \wedge x = 1\}$

Some examples

```
  while x > 1 do
⊢   y := y+1;
    x := x-1;
```

$: \{y = 4 - x \wedge x \geq 1\} \Rightarrow \{y = 4 - x \wedge x = 1\}$

$x = 3 \wedge y = 1 \wedge y = 4 - x \Rightarrow y = 4 - x \wedge x \geq 1$

$y = 4 - x \wedge x = 1 \Rightarrow y = 3$

```
  while x > 1 do
⊢   y := y+1;
    x := x-1;
```

$: \{x = 3 \wedge y = 1 \wedge y = 4 - x\} \Rightarrow \{y = 3\}$

Some examples

$\vdash$

```
x :=3;
y :=1;
```

$\{true\} \Rightarrow \{x = 3 \wedge 1 = 1 \wedge y = 4 - x\}$

$\vdash$

```
while x > 1 do
 y := y+1;
 x := x-1;
```

$: \{x = 3 \wedge y = 1 \wedge y = 4 - x\} \Rightarrow \{y = 3\}$

$\vdash$

```
x:=3;
y:=1;
while x > 1 do
  y := y+1;
  x := x-1;
```

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

How do we know that these are the right rules?

Soundness

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

Are the rules we presented sound?

Completeness

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

Are the rules we presented complete?

Relative Completeness

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

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

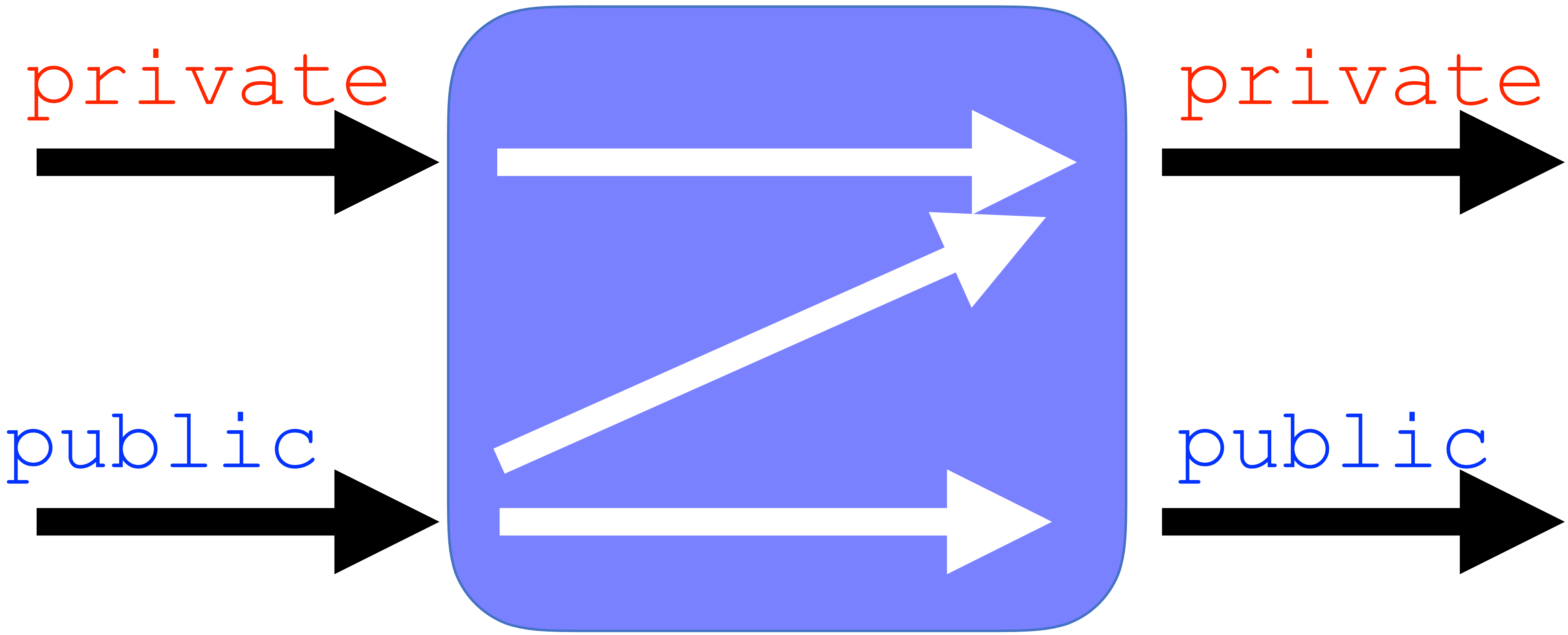
We want to distinguish confidential information that need to be kept secret from nonconfidential information that can be accessed by everyone.

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

`x:public` `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:private
y:public

x:=y
```

Secure

Is this program secure?

```
x:private
y:public


y:=x
```

Insecure

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
```

Secure

Is this program secure?

```
x:private
y:public

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

Insecure

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