# CS 591: Formal Methods in Security and Privacy

## Non-interference

Marco Gaboardi
gaboardi@bu.edu

Alley Stoughton
stough@bu.edu

# Rules of Hoare Logic:

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

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

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

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

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

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

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

# Rules of Hoare Logic:

$$\frac{}{\vdash \texttt{skip}:\ P \Rightarrow P} \qquad \frac{}{\vdash \texttt{abort}:\ P \Rightarrow Q}$$

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

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

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

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

# Rules of Hoare Logic:

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

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

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

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

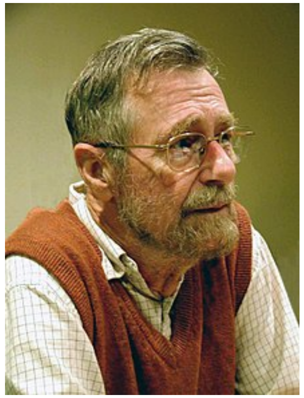
$$\frac{P \Rightarrow S \qquad \vdash c : S \Rightarrow R \qquad R \Rightarrow Q}{\vdash c:\ P \Rightarrow Q}$$

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

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

# 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)⇒P` is a valid triple, and for every triple `c:Q⇒P` we have `Q⇒wp(c,P)`

# Weakest precondition

This is defined on the structure of commands:

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

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

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

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

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

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

# Weakest precondition

This is defined on the structure of commands:

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

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

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

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

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

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

$$\mathtt{P_0} = \mathtt{\neg e \land P}$$

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

# 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.

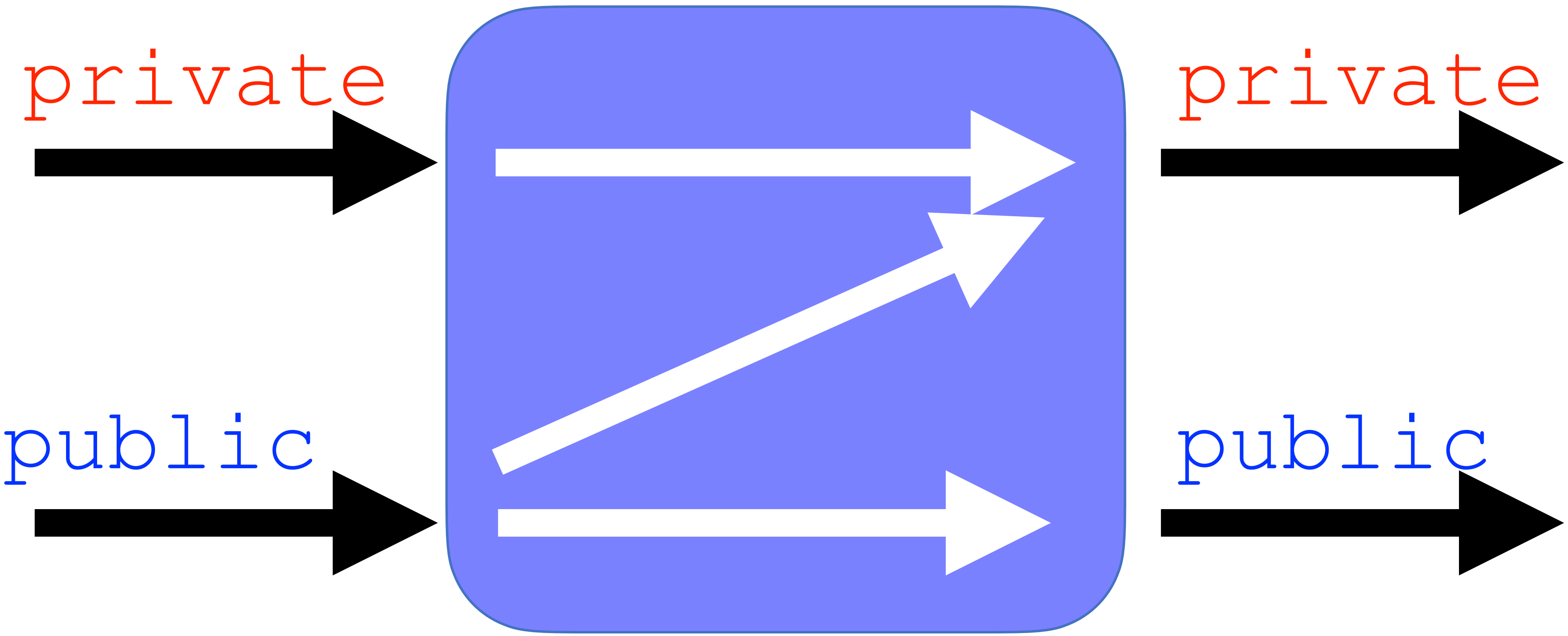
`x:public` `x:private`

# Information Flow Control

We want to guarantee that confidential information do not flow in what is considered nonconfidential.

# 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
```

# Is this program secure?

```
x:private
y:public

x:=y
```

Secure

# Is this program secure?

```
x:private
y:public

y:=x
```

# 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?

```
x:private
y:public

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

# 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?

# 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: $m_1 \sim_{low} m_2$

# Noninterference

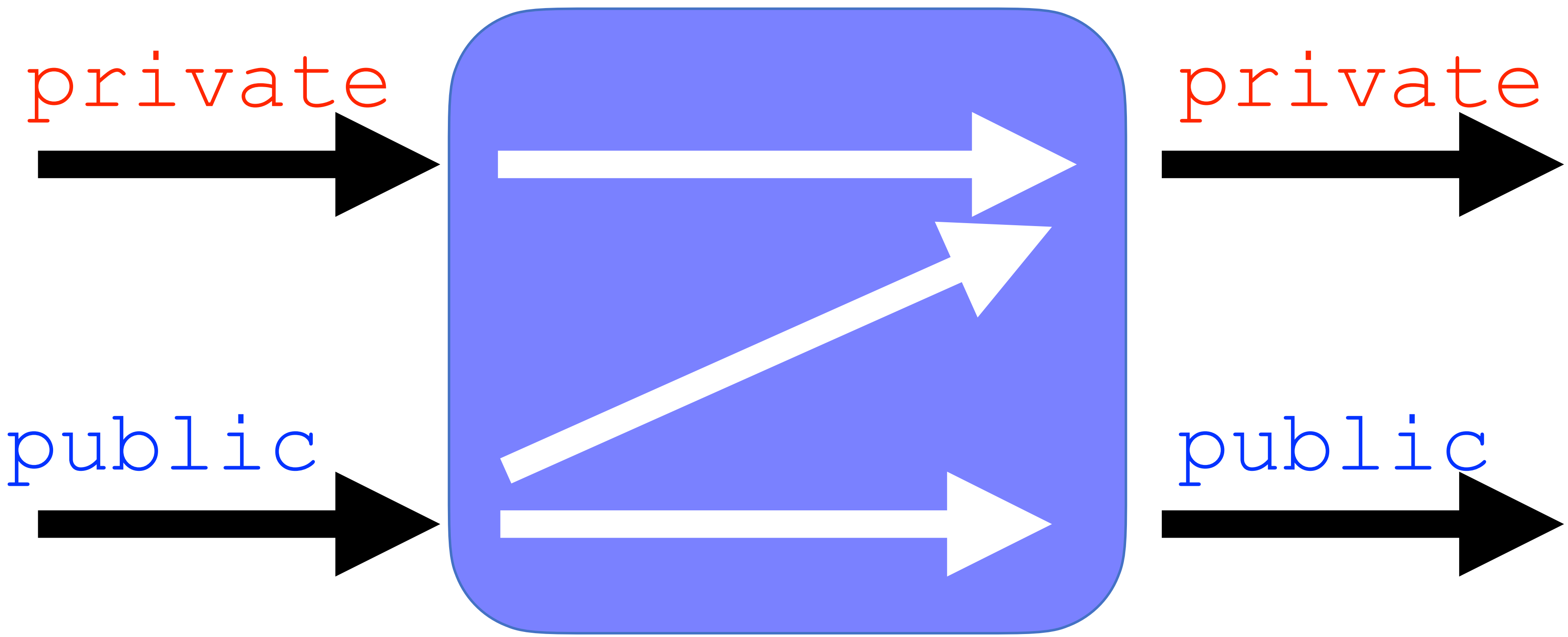
A program `prog` is noninterferent if and only if, whenever we run it on two memories $m_1$ and $m_2$ that are low equivalent, we obtain two memories $m_1'$ and $m_2'$ which are also low equivalent.

# Noninterference

In symbols

$m_1 \sim_{low} m_2$ and $\{c\}_{m1}=m_1'$ and $m_2'\{c\}_{m2}=m_2'$

implies $m_1' \sim_{low} m_2'$



# Does this program satisfy noninterference?

```
x:private
y:public


x:=y
```

# Does this program satisfy noninterference?

```
x:private
y:public

x:=y
```

Yes

# Does this program satisfy noninterference?

```
x:private
y:public

x:=y
```

Yes

$m^{in}_1=[x=n_1,y=k]$

# Does this program satisfy noninterference?

```
x:private
y:public

x:=y
```

Yes

$m^{in}_1$=[x=$n_1$,y=k]

$m^{in}_2$=[x=$n_2$,y=k]

# Does this program satisfy noninterference?

```
x:private
y:public

x:=y
```

Yes

$m^{in}_1=[x=n_1,y=k]$

$m^{out}_1=[x=k,y=k]$

$m^{in}_2=[x=n_2,y=k]$

$m^{out}_2=[x=k,y=k]$

# Does this program satisfy noninterference?

```
x:private
y:public


y:=x
```

# Does this program satisfy noninterference?

```
x:private
y:public


y:=x
```

No

# Does this program satisfy noninterference?

```
x:private
y:public

y:=x
```

$\boxed{\text{No}}$

$m^{in}_1=[x=n_1,y=k]$

# Does this program satisfy noninterference?

```
x:private
y:public


y:=x
```

$\boxed{\text{No}}$

$m^{in}{}_1=[x=n_1,y=k]$

$m^{in}{}_2=[x=n_2,y=k]$

# Does this program satisfy noninterference?

```
x:private
y:public

y:=x
```

$\boxed{\text{No}}$

$m^{in}_1=[x=n_1,y=k]$

$m^{out}_1=[x=n_1,y=n_1]$

$m^{in}_2=[x=n_2,y=k]$

$m^{out}_2=[x=n_2,y=n_2]$

# Does this program satisfy noninterference?

```
x:private
y:public

y:=x
y:=5
```

# Does this program satisfy noninterference?

```
x:private
y:public

y:=x
y:=5
```

Yes

# Does this program satisfy noninterference?

```
x:private
y:public

y:=x
y:=5
```

Yes

$m^{in}_1=[x=n_1,y=k]$

# Does this program satisfy noninterference?

```
x:private
y:public

y:=x
y:=5
```

Yes

$m^{in}_1=[x=n_1,y=k]$

$m^{in}_2=[x=n_2,y=k]$

# Does this program satisfy noninterference?

```
x:private
y:public

y:=x
y:=5
```

Yes

$m^{in}_1=[x=n_1,y=k]$

$m^{in}_2=[x=n_2,y=k]$

$m^{out}_1=[x=n_1,y=5]$

$m^{out}_2=[x=n_2,y=5]$

# Does this program satisfy noninterference?

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

# Does this program satisfy noninterference?

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

Yes

# Does this program satisfy noninterference?

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

Yes

$m^{in}_1$=[x=$n_1$,y=6]

# Does this program satisfy noninterference?

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

Yes

$m^{in}_1=[x=n_1,y=6]$

$m^{in}_2=[x=n_2,y=6]$

# Does this program satisfy noninterference?

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

Yes

$m^{in}_1=[x=n_1,y=6]$

$m^{in}_2=[x=n_2,y=6]$

$m^{out}_1=[x=1,y=6]$

$m^{out}_2=[x=1,y=6]$

# 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?

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

No

$m^{in}_1=[x=6,y=k]$

# Does this program satisfy noninterference?

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

No

$m^{in}_1$=[x=6,y=k]

$m^{in}_2$=[x=5,y=k]

# Does this program satisfy noninterference?

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

$\boxed{\text{No}}$

$m^{in}{}_1=[x=6,y=k]$ $m^{in}{}_2=[x=5,y=k]$

$m^{out}{}_1=[x=6,y=1]$ $m^{out}{}_2=[x=5,y=0]$

# 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 $m_1 \sim_{low} m_2$ :

1) $\{c\}_{m1} = \perp$ iff $\{c\}_{m2} = \perp$
2) $\{c\}_{m1} = m_1'$ and $\{c\}_{m2} = m_2'$ implies $m_1' \sim_{low} m_2'$

$\boxed{\text{Is this condition easy to check?}}$

# Can we use the tool we studied so far?

| |
|---|
| Precondition |
| Program |
| Postcondition |

Precondition
(a logical formula)

$$c : P \Rightarrow Q$$

Program

Postcondition
(a logical formula)

# Validity of Hoare triple

We say that the triple $\texttt{c:P}\Rightarrow\texttt{Q}$ is valid

if and only if

for every memory $\texttt{m}$ such that $\texttt{P(m)}$

and memory m' such that $\{\texttt{c}\}_{\texttt{m}}\texttt{=m'}$

we have $\texttt{Q(m')}$.

# 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 $m_1 \sim_{low} m_2$ :

1) $\{c\}_{m1}=\perp$ iff $\{c\}_{m2}=\perp$

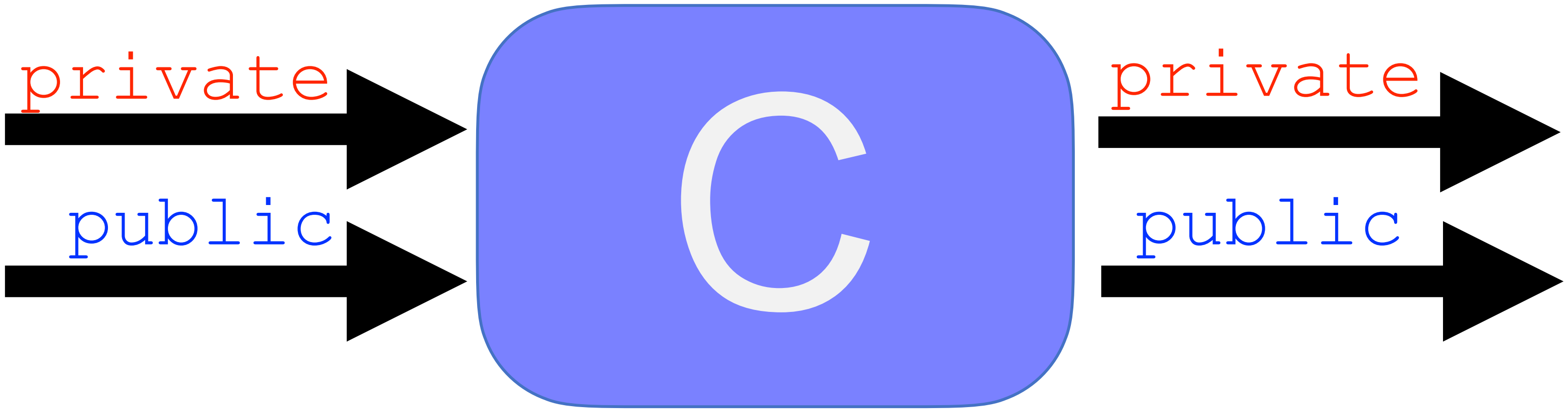
2) $\{c\}_{m1}=m_1'$ and $\{c\}_{m2}=m_2'$ implies $m_1' \sim_{low} m_2'$

# Relational Property

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

1) $\{c\}_{m1}=\perp$ iff $\{c\}_{m2}=\perp$

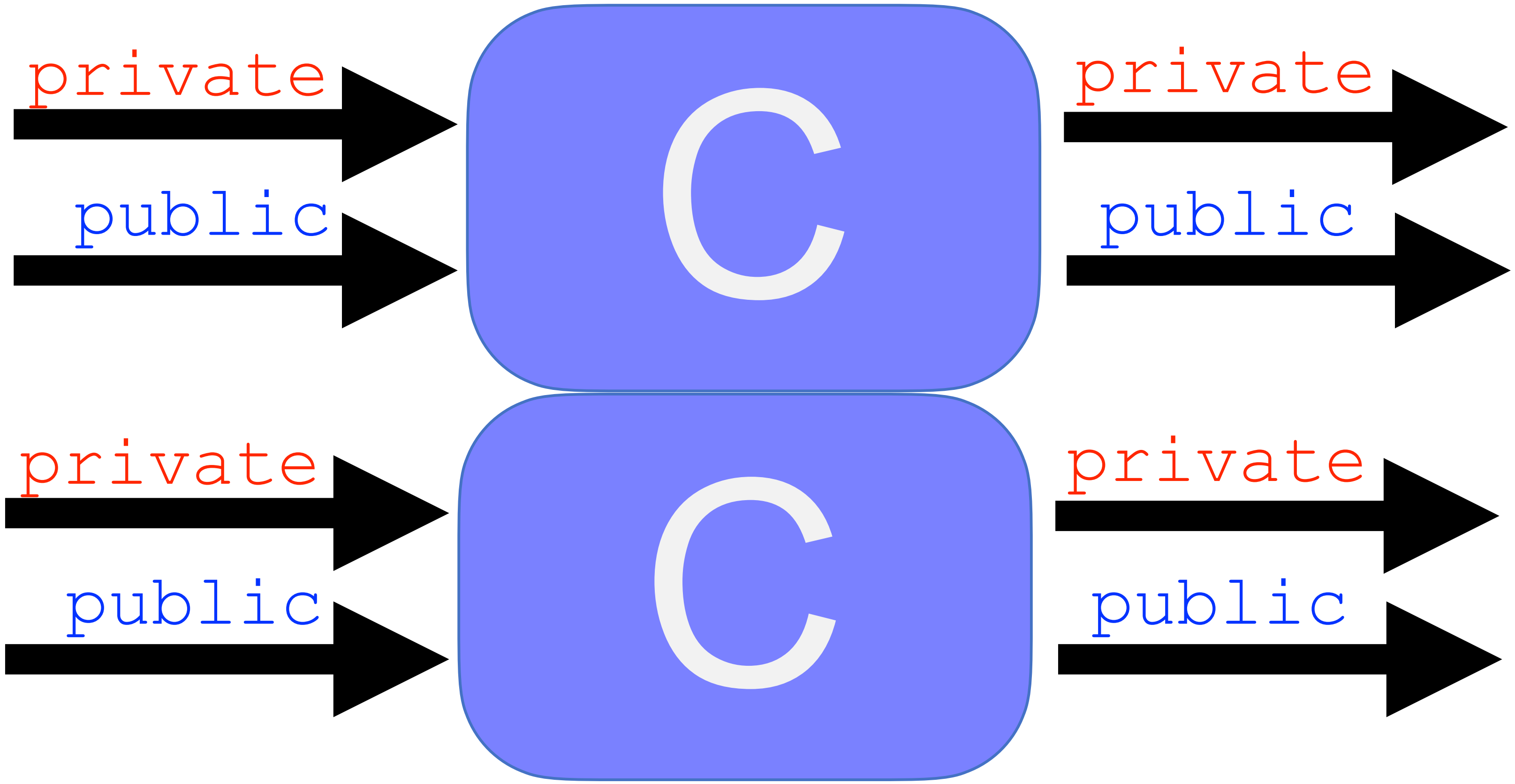
2) $\{c\}_{m1}=m_1'$ and $\{c\}_{m2}=m_2'$ implies $m_1' \sim_{low} m_2'$



# Relational Property

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

1) $\{c\}_{m1} = \perp$ iff $\{c\}_{m2} = \perp$
2) $\{c\}_{m1} = m_1'$ and $\{c\}_{m2} = m_2'$ implies $m_1' \sim_{low} m_2'$

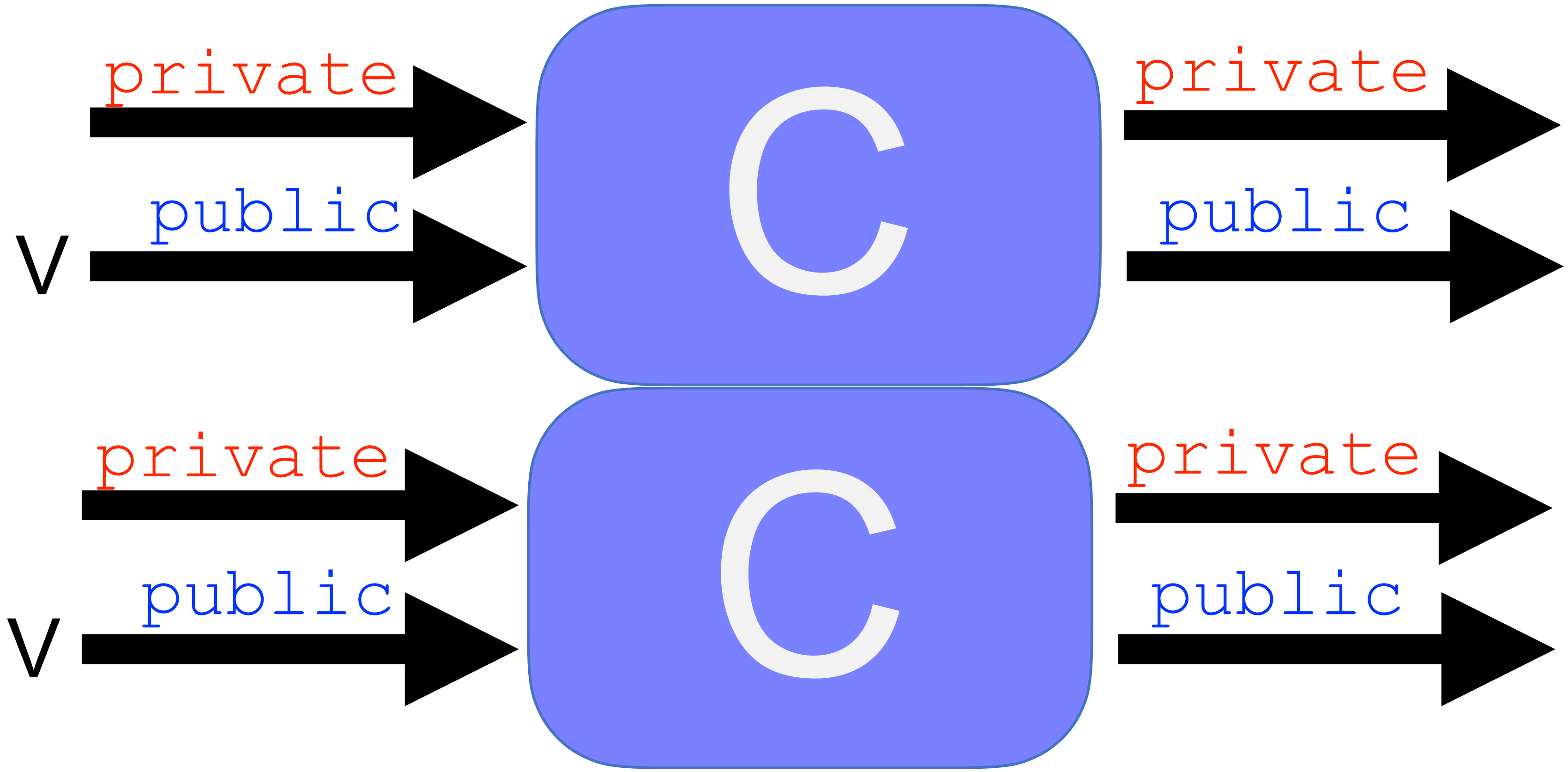


# Relational Property

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

1) $\{c\}_{m1} = \perp$ iff $\{c\}_{m2} = \perp$

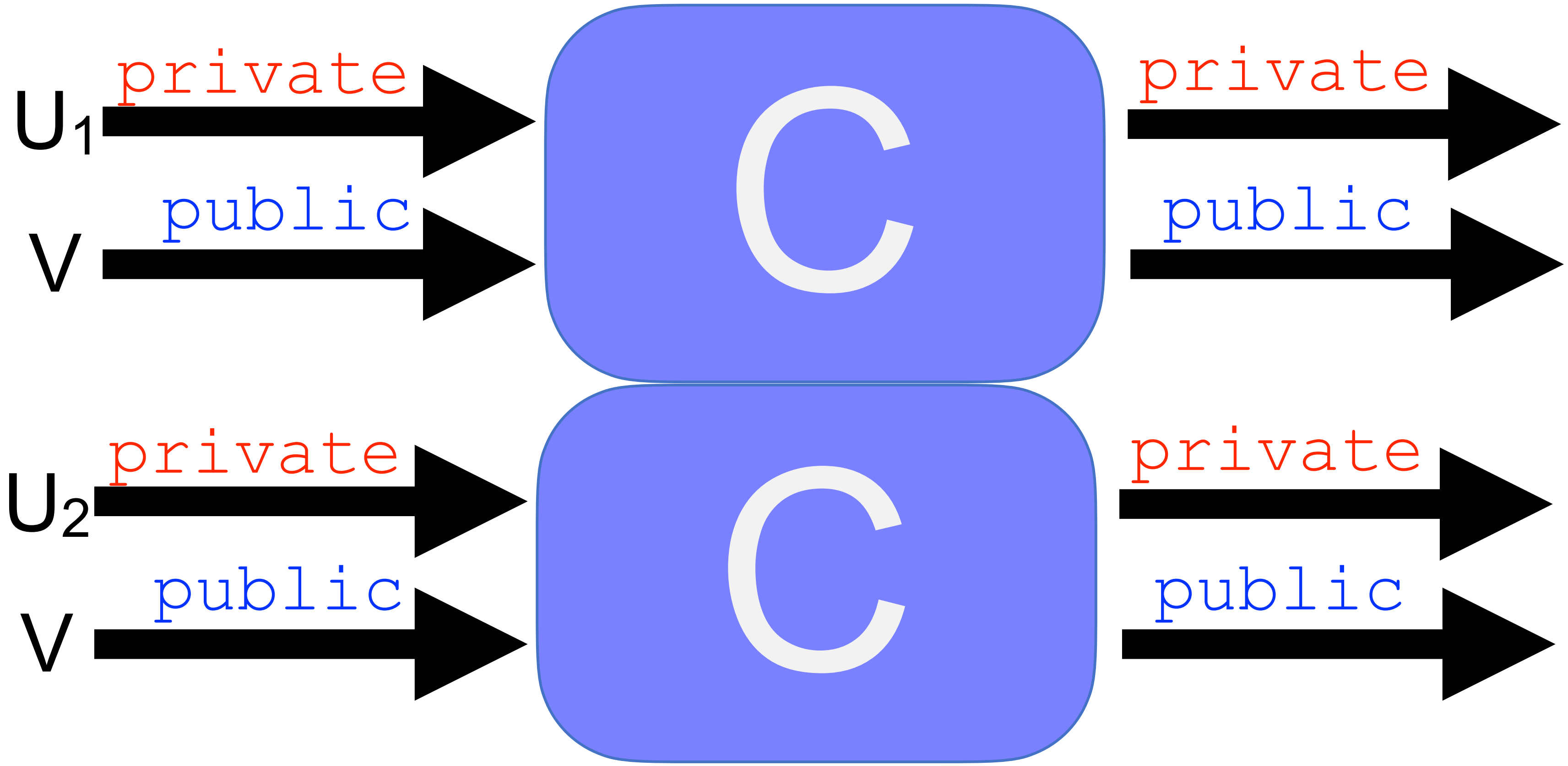
2) $\{c\}_{m1} = m_1'$ and $\{c\}_{m2} = m_2'$ implies $m_1' \sim_{low} m_2'$



# Relational Property

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

1) $\{c\}_{m1} = \perp$ iff $\{c\}_{m2} = \perp$
2) $\{c\}_{m1} = m_1'$ and $\{c\}_{m2} = m_2'$ implies $m_1' \sim_{low} m_2'$

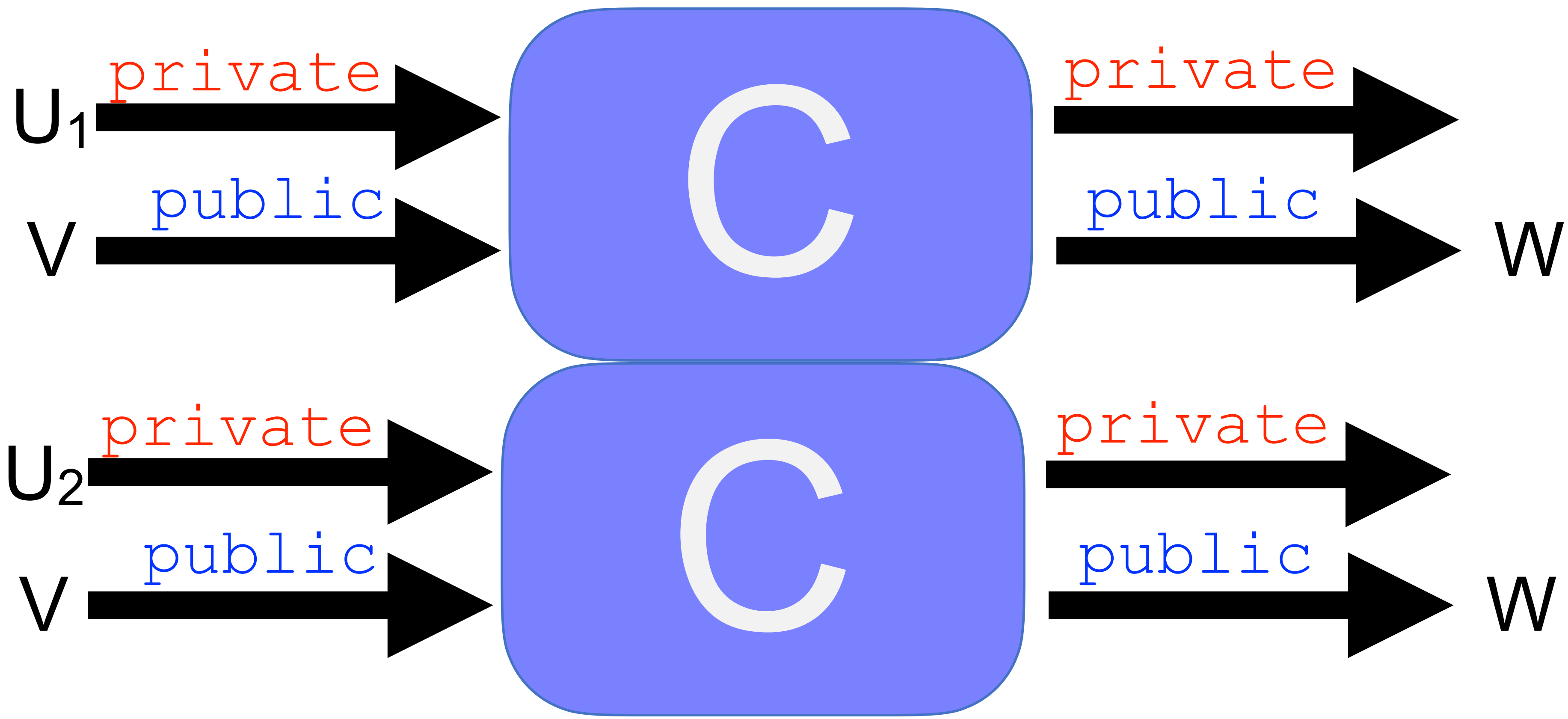


# Relational Property

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

1) $\{c\}_{m1}=\perp$ iff $\{c\}_{m2}=\perp$

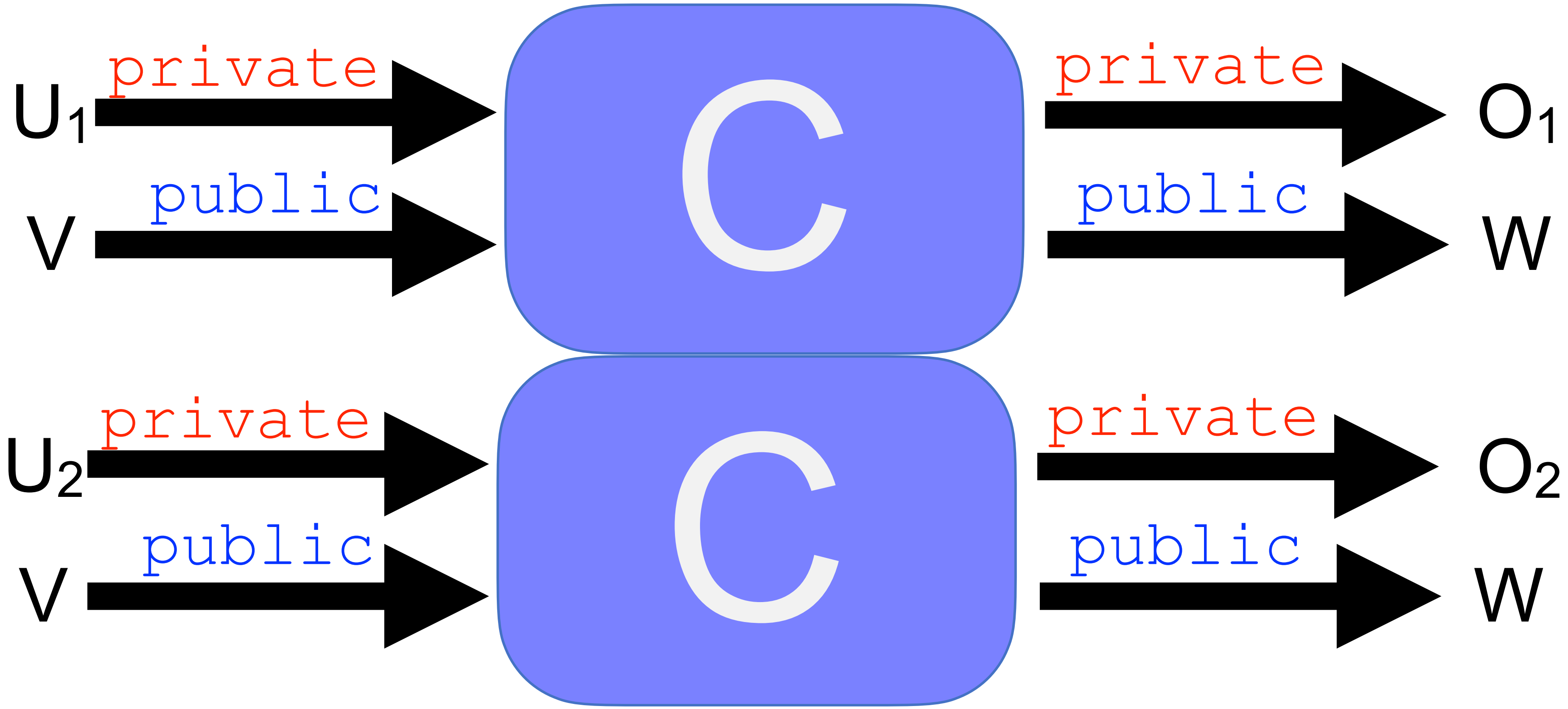
2) $\{c\}_{m1}=m_1'$ and $\{c\}_{m2}=m_2'$ implies $m_1' \sim_{low} m_2'$



# Relational Property

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

1) $\{c\}_{m1} = \perp$ iff $\{c\}_{m2} = \perp$
2) $\{c\}_{m1} = m_1'$ and $\{c\}_{m2} = m_2'$ implies $m_1' \sim_{low} m_2'$



# Relational Hoare Logic - RHL

Precondition
(a logical formula)

```
Precondition
Program₁ ~ Program₂
Postcondition
```

$$c_1 \sim c_2 : P \Rightarrow Q$$

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

$$c_1 \sim c_2 : x\langle 1 \rangle \leq x\langle 2 \rangle \Rightarrow x\langle 1 \rangle \geq x\langle 2 \rangle$$

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) $\{c_1\}_{m1} = \perp$ iff $\{c_2\}_{m2} = \perp$
2) $\{c_1\}_{m1} = m_1'$ and $\{c_2\}_{m2} = m_2'$ 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) $\{c_1\}_{m1} = \perp$ iff $\{c_2\}_{m2} = \perp$
2) $\{c_1\}_{m1} = m_1'$ and $\{c_2\}_{m2} = m_2'$ implies $Q(m_1', m_2')$.

Is this easy to check?

# Rules of Relational Hoare Logic

## Skip

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

# 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

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

To show this rule correct we need to show the validity of the quadruple $\texttt{skip} \sim \texttt{skip} : P \Rightarrow P$.

# Correctness of Skip Rule

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

To show this rule correct we need to show the validity of the quadruple $\texttt{skip} \sim \texttt{skip}: P \Rightarrow P$.

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

# Correctness of Skip Rule

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

To show this rule correct we need to show the validity of the quadruple $\texttt{skip} \sim \texttt{skip} : P \Rightarrow P$.

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

Follow easily by our semantics:
$\{\texttt{skip}\}_m = m$

# Rules of Relational Hoare Logic
## Abort

$\vdash \texttt{abort} \sim \texttt{abort} : \texttt{true} \Rightarrow \texttt{false}$

# Rules of Relational Hoare Logic
## Abort

---

$\vdash$ `abort~abort:true⇒false`

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

# Rules of Relational Hoare Logic
## Abort

$$\vdash \texttt{abort} \sim \texttt{abort} : \texttt{true} \Rightarrow \texttt{false}$$

To show this rule correct we need to show the validity of the quadruple $\texttt{abort} \sim \texttt{abort} : \texttt{T} \Rightarrow \texttt{F}$.

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

# Rules of Relational Hoare Logic
## Abort

$\vdash \texttt{abort} \sim \texttt{abort} : \texttt{true} \Rightarrow \texttt{false}$

To show this rule correct we need to show the validity of the quadruple $\texttt{abort} \sim \texttt{abort} : \texttt{T} \Rightarrow \texttt{F}$.

For every $m_1, m_2$ such that $P(m_1, m_2)$ we can show $\{\texttt{abort}\}_{m1} = \perp$ iff $\{\texttt{abort}\}_{m2} = \perp$.

Follow easily by our semantics:
$\{\texttt{abort}\}_m = \perp$

# Rules of Relational Hoare Logic

## Assignment

$$\frac{}{\vdash x_1 := e_1 \sim x_2 := e_2 : P[e_1\langle 1\rangle / x_1\langle 1\rangle, e_2\langle 2\rangle / x_2\langle 2\rangle] \Rightarrow P}$$

# Rules of Relational Hoare Logic
# Composition

$$\frac{\vdash \mathtt{c_1 \sim c_2 : P \Rightarrow R} \qquad \vdash \mathtt{c_1' \sim c_2' : R \Rightarrow S}}{\vdash \mathtt{c_1 ; c_1' \sim c_2 ; c_2' : P \Rightarrow S}}$$

# Rules of Relational Hoare Logic Consequence

$$\frac{P \Rightarrow S \qquad \vdash c_1 \sim c_2 : S \Rightarrow R \qquad R \Rightarrow Q}{\vdash c_1 \sim c_2 : P \Rightarrow Q}$$

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

$$\vdash c_1 \sim c_2 : e_1\langle 1\rangle \wedge e_2\langle 2\rangle \wedge P \Rightarrow Q$$

$$\vdash c_1' \sim c_2' : \neg e_1\langle 1\rangle \wedge \neg e_2\langle 2\rangle \wedge P \Rightarrow Q$$

$$\frac{}{\vdash \begin{array}{c}\texttt{if } e_1 \texttt{ then } c_1 \texttt{ else } c_1' \\ \sim \\ \texttt{if } e_2 \texttt{ then } c_2 \texttt{ else } c_2'\end{array} : P \Rightarrow Q}$$

# Rules of Hoare Logic
# If then else

$$\vdash c_1 \sim c_2 : e_1\langle 1\rangle \wedge e_2\langle 2\rangle \wedge P \Rightarrow Q$$

$$\vdash c_1' \sim c_2' : \neg e_1\langle 1\rangle \wedge \neg e_2\langle 2\rangle \wedge P \Rightarrow Q$$

$$\dfrac{}{\vdash \begin{array}{c}\texttt{if } e_1 \texttt{ then } c_1 \texttt{ else } c_1' \\ \sim \\ \texttt{if } e_2 \texttt{ then } c_2 \texttt{ else } c_2'\end{array} : P \Rightarrow Q}$$

Is this correct?

# An example

$$\vdash \begin{array}{c}\texttt{if true then skip else x:=x+1} \\ \sim \\ \texttt{if false then x:=x+1 else skip}\end{array} : \{\texttt{x=n}\} \Rightarrow \{\texttt{x=n+1}\}$$

Is this a valid quadruple?

# An example

$$\vdash \begin{array}{c} \texttt{if true then skip else x:=x+1} \\ \sim \\ \texttt{if false then x:=x+1 else skip} \end{array} : \{\texttt{x=n}\} \Rightarrow \{\texttt{x=n+1}\}$$

Is this a valid quadruple? ✗

# An example

$$\vdash \begin{array}{c}\texttt{if true then skip else x:=x+1} \\ \sim \\ \texttt{if false then x:=x+1 else skip}\end{array} : \{x=n\} \Rightarrow \{x=n+1\}$$

Is this a valid quadruple? ✗

Can we prove it with the rule above?

# An example

$$\vdash \begin{array}{l}\texttt{if true then skip else x:=x+1} \\ \qquad\qquad\sim \\ \texttt{if false then x:=x+1 else skip}\end{array} : \{\texttt{x=n}\} \Rightarrow \{\texttt{x=n+1}\}$$

Is this a valid quadruple? ✗

Can we prove it with the rule above? ✓

# Rules of Relational Hoare Logic If then else

$P \Rightarrow e_1\langle 1\rangle = e_2\langle 2\rangle$

$\vdash c_1 \sim c_2 : e_1\langle 1\rangle \wedge P \Rightarrow Q$

$\vdash c_1' \sim c_2' : \neg e_1\langle 1\rangle \wedge P \Rightarrow Q$

$$\frac{}{\vdash \begin{array}{c} \texttt{if } e_1 \texttt{ then } c_1 \texttt{ else } c_1' \\ \sim \\ \texttt{if } e_2 \texttt{ then } c_2 \texttt{ else } c_2' \end{array} : P \Rightarrow Q}$$

# Rules of Hoare Logic
# While

$$P \Rightarrow e_1\langle 1\rangle = e2\langle 2\rangle$$

$$\frac{\vdash c_1 \sim c_2 \ : \ e_1\langle 1\rangle \wedge P \Rightarrow P}{\vdash \begin{array}{c}\texttt{while } e_1 \texttt{ do } c_1 \\ \sim \\ \texttt{while } e_2 \texttt{ do } c_2\end{array} : P \Rightarrow P \wedge \neg e_1\langle 1\rangle}$$

Invariant

# How can we prove this?

```
x:private
y:public

x:=y
```

$: \quad =_{\mathtt{low}} \Rightarrow \quad =_{\mathtt{low}}$

# How can we prove this?

```
x:private
y:public

y:=x
```

$: \; =_{\texttt{low}} \Rightarrow \neg(=_{\texttt{low}})$

# How can we prove this?

```
x:private
y:public

y:=x
```

$: \; =_{\texttt{low}} \Rightarrow \neg(=_{\texttt{low}})$

Can we prove it?

# How can we prove this?

```
x:private
y:public

y:=x
y:=5
```

$: \quad =_{\mathtt{low}} \Rightarrow \; =_{\mathtt{low}}$

# How can we prove this?

```
x:private
y:public

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

$: \quad =_{low} \Rightarrow \; =_{low}$

# How can we prove this?

```
x:private
y:public

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

$: \quad =_{\mathtt{low}} \Rightarrow \; =_{\mathtt{low}}$

# How can we prove this?

```
x:private
y:public

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

Can we prove it?

$: \; =_{low} \Rightarrow \; =_{low}$

# Rules of Relational Hoare Logic
# If then else

$$\dfrac{\begin{array}{c} P \Rightarrow e_1\langle 1\rangle = e_2\langle 2\rangle \\ \vdash c_1 \sim c_2 : e_1\langle 1\rangle \wedge P \Rightarrow Q \\ \vdash c_1' \sim c_2' : \neg e_1\langle 1\rangle \wedge P \Rightarrow Q \end{array}}{\vdash \begin{array}{c} \texttt{if } e_1 \texttt{ then } c_1 \texttt{ else } c_1' \\ \sim \\ \texttt{if } e_2 \texttt{ then } c_2 \texttt{ else } c_2' \end{array} : P \Rightarrow Q}$$

# Rules of Relational Hoare Logic
# If then else - left

$$\vdash c_1 \sim c_2 : e\langle 1\rangle \wedge P \Rightarrow Q$$

$$\vdash c_1' \sim c_2 : \neg e\langle 1\rangle \wedge P \Rightarrow Q$$

---

$$\vdash \begin{array}{c} \texttt{if } e \texttt{ then } c_1 \texttt{ else } c_1' \\ \sim \\ c_2 \end{array} : P \Rightarrow Q$$

# Rules of Relational Hoare Logic
# If then else - left

$$\vdash c_1 \sim c_2 : e\langle 2\rangle \wedge P \Rightarrow Q$$

$$\vdash c_1 \sim c_2' : \neg e\langle 2\rangle \wedge P \Rightarrow Q$$

---

$$\vdash \begin{array}{c} c_1 \\ \sim \\ \texttt{if } e \texttt{ then } c_2 \texttt{ else } c_2' \end{array} : P \Rightarrow Q$$

# How can we prove this?

```
x:private
y:public

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

$: \quad =_{low} \Rightarrow \quad =_{low}$

# 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
```

$: \;\; =_{low} \Rightarrow \;\; =_{low}$

# How can we prove this?

```
x:private
z:public
y:private

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

$: \ =_{low} \Rightarrow \neg(=_{low})$

# 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 /\ r=0 do
 if not(s1[i]=s2[i]) then
      r:=1
 i:=i+1

: n>0 /\ =low ⇒ ¬(=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 ⇒ ¬(=low)
```