Differential Privacy

Beyond Global Sensitivity - verification

Marco Gaboardi
Boston University

The opinions expressed in this course are mine and they do not not reflect those of the National Science Foundation or the US. Census Bureau.

Differential privacy

Definition

Given $\varepsilon,\delta \geq 0$, a probabilistic query $Q: X^n \rightarrow R$ is (ε,δ)-differentially private iff for all adjacent database b_1, b_2 and for every $S \subseteq R$:

$$\Pr[Q(b_1) \in S] \leq \exp(\varepsilon)\Pr[Q(b_2) \in S] + \delta$$

(ε,δ)-Differential Privacy

This corresponds to a privacy loss of the form:

$$\mathcal{L}_{\mathcal{M}}^{D \to D'}(r) = \ln\left(\frac{\Pr[\mathcal{M}(D) = r | E]}{\Pr[\mathcal{M}(D') = r | E']}\right)$$

The (ϵ, δ)-differential privacy requirement corresponds to requiring that for every r and every adjacent D, D' we have:

$$\Pr\left[\left|\mathcal{L}_{\mathcal{M}}^{D \to D'}(r)\right| \leq \epsilon\right] \geq 1 - \delta$$

Not exactly!

Bounding the moments

A random variable can be described using its moments.

$$\mu_n = \mathbb{E}[X^n]$$

Here we consider central moments. For instance, the first central moment is the mean, the second is the variance, the third is the skewness, etc.

Can we bound the moments of the privacy loss?

Moment generating function

The probability distribution of a random variable X can be described by its moment generating function:

$$\mathtt{m}_X(\alpha) = \mathbb{E}[e^{\alpha X}]$$

This function can be used to compute, or give upper bounds on the moments of the random variable X.

$$m_X(\alpha) = 1 + \alpha\mu_1 + \frac{\alpha^2\mu_2}{2!} + \ldots + \frac{\alpha^n\mu_n}{n!} + \ldots$$

Can we do better than the global sensitivity?

Global Sensitivity

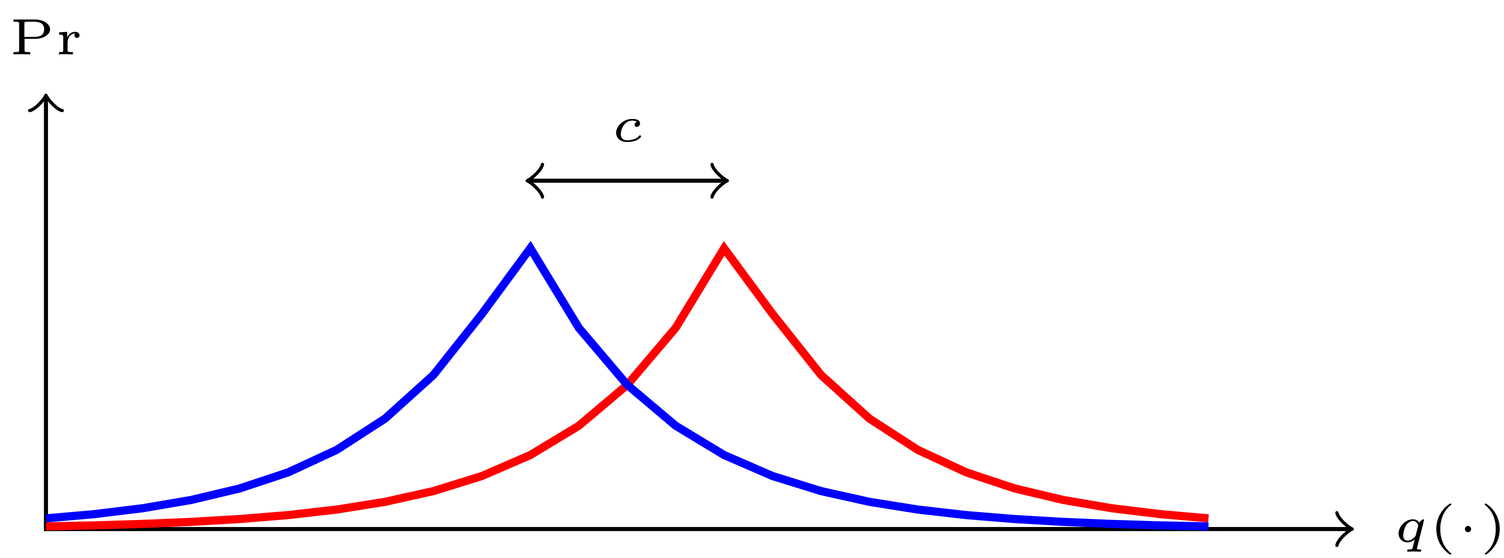
Definition 1.8 (Global sensitivity)**.** The *global sensitivity* of a function $q : \mathcal{X}^n \to \mathbb{R}$ is:

$$\Delta q = \max \left\{ |q(D) - q(D')| \;\middle|\; D \sim_1 D' \in \mathcal{X}^n \right\}$$

Laplace Mechanism

Algorithm 2 Pseudo-code for the Laplace Mechanism

function $\textsc{LapMech}(D, q, \epsilon)$
  $Y \xleftarrow{\$} \mathsf{Lap}(\frac{\Delta q}{\epsilon})(0)$
  return $q(D) + Y$
end function



Laplace Mechanism

Accuracy Theorem: let $r = \mathsf{LapMech}(D, q, \epsilon)$

$$\Pr\left[|q(D) - r| \geq \left(\frac{\Delta q}{\epsilon}\right) \ln\left(\frac{1}{\beta}\right)\right] = \beta$$

Local sensitivity

Definition 1.8 (Global sensitivity)**.** The *global sensitivity* of a function $q : \mathcal{X}^n \to \mathbb{R}$ is:

$$\Delta q = \max \left\{ |q(D) - q(D')| \;\middle|\; D \sim_1 D' \in \mathcal{X}^n \right\}$$

Definition 1.14 (Local sensitivity)**.** The *local sensitivity* of a function $q : \mathcal{X}^n \to \mathbb{R}$ at $D \in \mathcal{X}^n$ is:

$$\ell\Delta q(D) = \max \left\{ |q(D) - q(D')| \;\middle|\; D \sim_1 D',\ D' \in \mathcal{X}^n \right\}$$

Calibrating noise to the local sensitivity

We may add noise proportional to the local sensitivity (LS).

Unfortunately, this does not guarantee privacy.

Suppose that for a given D we have LS(D)=0 but that we also have D~D' with LS(D')=10^9.

Calibrating noise to the local sensitivity

We may add noise proportional to the local sensitivity (LS).

Unfortunately, this does not guarantee privacy.

Suppose that for a given D we have LS(D)=0 but that we also have D~D' with LS(D')=10^9.

We will see that we can do anyway better than GS.

Smooth Sensitivity

Definition 2.2 (Smooth sensitivity)**.** *For $\beta > 0$, the β-smooth sensitivity of f is*

$$S^*_{f,\beta}(x) = \max_{y \in D^n} \left(LS_f(y) \cdot e^{-\beta d(x,y)} \right).$$

[Nissim, Raskhodnikova, Smith '06]

Smooth Sensitivity

Definition 2.2 (Smooth sensitivity)**.** *For $\beta > 0$, the β-smooth sensitivity of f is*

$$S^*_{f,\beta}(x) = \max_{y \in D^n} \left(LS_f(y) \cdot e^{-\beta d(x,y)} \right).$$

Definition 2.1 (A Smooth Bound on LS)**.** *For $\beta > 0$, a function $S : D^n \to \mathbb{R}^+$ is a β-smooth upper bound on the local sensitivity of f if it satisfies the following requirements:*

$$\forall x \in D^n : \qquad S(x) \geq LS_f(x) \; ; \tag{1}$$

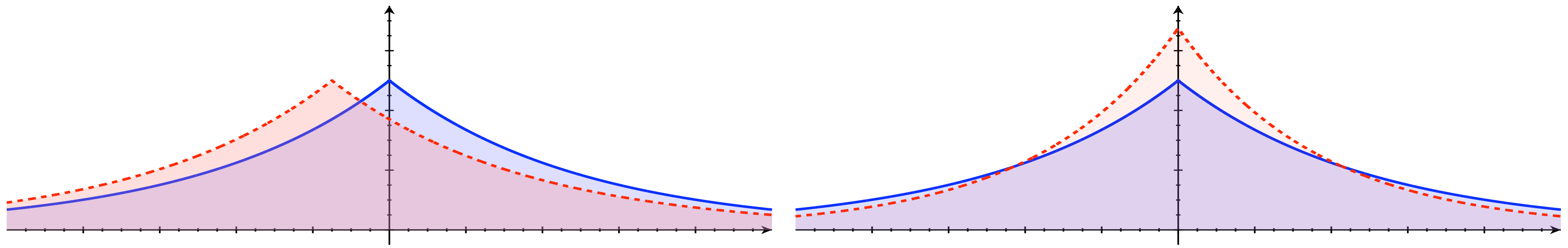
$$\forall x, y \in D^n, d(x,y) = 1 : \qquad S(x) \leq e^{\beta} \cdot S(y) \; . \tag{2}$$

[Nissim, Raskhodnikova, Smith '06]

Calibrating noise to the smooth sensitivity

Lemma 2.6. *Let h be an (α, β)-admissible noise probability density function, and let Z be a fresh random variable sampled according to h. For a function $f : D^n \to \mathbb{R}^d$, let $S : D^n \to \mathbb{R}$ be a β-smooth upper bound on the local sensitivity of f. Then algorithm $\mathcal{A}(x) = f(x) + \frac{S(x)}{\alpha} \cdot Z$ is (ϵ, δ)-differentially private.*

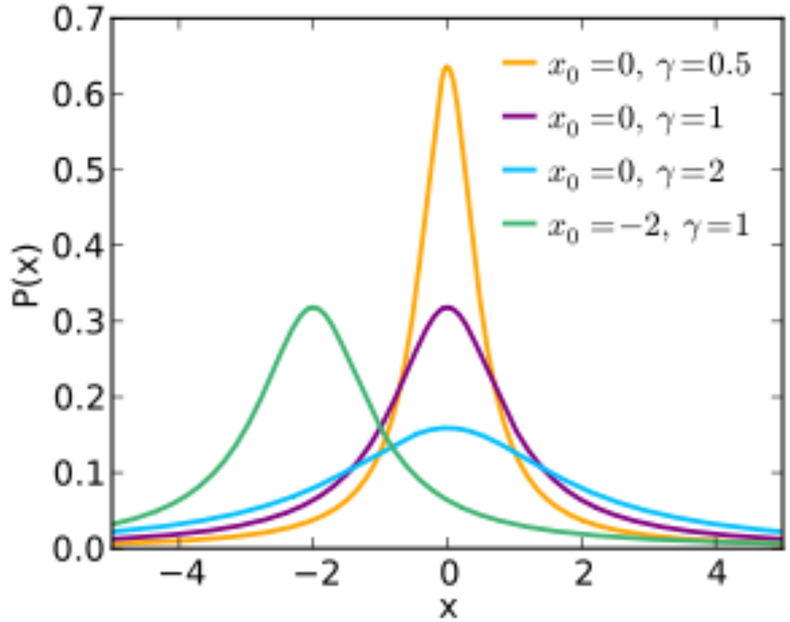
For two neighbor databases x and y, the output distribution $\mathcal{A}(y)$ is a shifted and scaled version of $\mathcal{A}(x)$. The sliding and dilation properties ensure that $\Pr[\mathcal{A}(x) \in \mathcal{S}]$ and $\Pr[\mathcal{A}(y) \in \mathcal{S}]$ are close for all sets $\mathcal{S}$ of outputs.



[Nissim, Raskhodnikova, Smith '06]

Admissible Noise

Adding noise $O(\mathrm{SS}_q^{\varepsilon}(x)/\varepsilon)$ (according to a Cauchy distribution) is sufficient for ε-differential privacy.

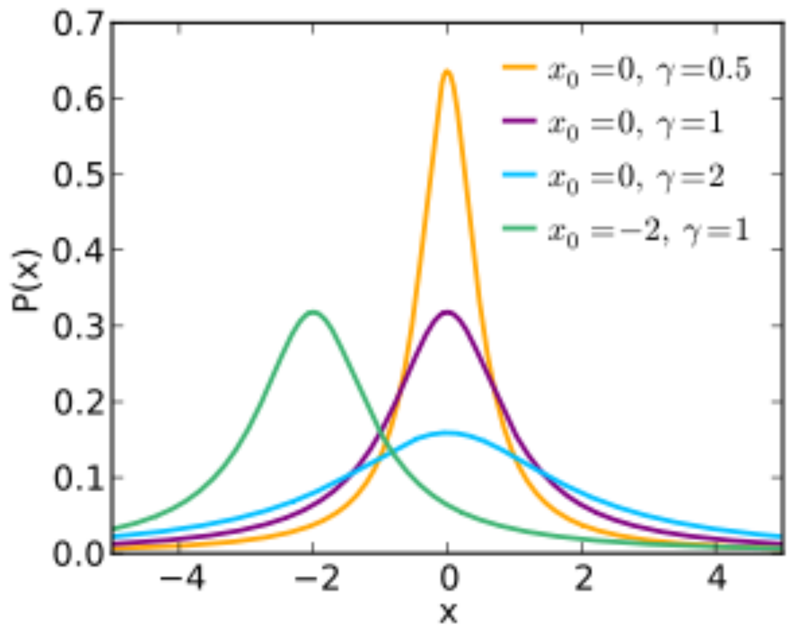


Laplace and Gauss give (ε,δ)-DP

[Nissim, Raskhodnikova, Smith '06]

Admissible Noise

Adding noise $O(\mathrm{SS}_q^{\varepsilon}(x)/\varepsilon)$ (according to a Cauchy distribution) is sufficient for ε-differential privacy.



Laplace and Gauss give (ε,δ)-DP

Computing the Smooth Sensitivity can be intractable.

[Nissim, Raskhodnikova, Smith '06]

Accuracy revisited

Accuracy Theorem (smooth sensitivity using Laplace):

$$||q(D) - r||_\infty \in O\left(\frac{S(D)}{\epsilon}\right)$$

Where

$$||\overrightarrow{v}||_\infty = \max_{i=0} |v_i|$$

Propose Test Release

Propose-test-release Given $q : \mathcal{X}^n \to \mathbb{R}$, $\epsilon, \delta, \beta \geq 0$

1. Propose a target bound β on local sensitivity.
2. Let $\hat{d} = d(x, \{x' : \mathrm{LS}_q(x') > \beta\}) + \mathrm{Lap}(1/\varepsilon)$, where d denotes Hamming distance.
3. If $\hat{d} \leq \ln(1/\delta)/\varepsilon$, output $\perp$.
4. If $\hat{d} > \ln(1/\delta)/\varepsilon$, output $q(x) + \mathrm{Lap}(\beta/\varepsilon)$.

Stability-based algorithms

Releasing stable values Given $q : \mathcal{X}^n \to \mathbb{R}$, $\epsilon, \delta \geq 0$

1. Let $\hat{d} = d(x, \{x' : q(x') \neq q(x)\}) + \mathrm{Lap}(1/\varepsilon)$, where d denotes Hamming distance.
2. If $\hat{d} \leq 1 + \ln(1/\delta)/\varepsilon$, output $\perp$.
3. Otherwise output $q(x)$.

Stability-based algorithms

Releasing stable values Given $q : \mathcal{X}^n \to \mathbb{R}$, $\epsilon, \delta \geq 0$

1. Let $\hat{d} = d(x, \{x' : q(x') \neq q(x)\}) + \mathrm{Lap}(1/\varepsilon)$, where d denotes Hamming distance.
2. If $\hat{d} \leq 1 + \ln(1/\delta)/\varepsilon$, output $\perp$.
3. Otherwise output $q(x)$.

Proposition 3.3 (releasing stable values)**.** *For every query $q : \mathcal{X}^n \to \mathcal{Y}$ and $\varepsilon, \delta > 0$, the above algorithm is (ε, δ)-differentially private.*

Stability-based algorithms

Stability-based algorithms

Consider, for example, the *mode* function $q : \mathcal{X}^n \to \mathcal{X}$, where $q(x)$ is defined to be the most frequently occurring data item in x (breaking ties arbitrarily). Then $d(x, \{x' : q(x') \neq q(x)\})$ equals half of the gap in the number of occurrences between the mode and the second-most frequently occurring item (rounded up). So we have:

Proposition 3.4 (stability-based mode)**.** *For every data universe $\mathcal{X}$, $n \in \mathbb{N}$, $\varepsilon, \delta \geq 0$, there is an (ε, δ)-differentially private algorithm $\mathcal{M} : \mathcal{X}^n \to \mathcal{X}$ such that for every dataset $x \in \mathcal{X}^n$ where the difference between the number of occurrences of the mode and the 2nd most frequently occurring item is larger than $4\lceil \ln(1/\delta)/\varepsilon \rceil$, $\mathcal{M}(x)$ outputs the mode of x with probability at least $1 - \delta$.*

Stability-based Histogram

1. For every point $y \in \mathcal{X}$:
   (a) If $q_y(x) = 0$, then set $a_y = 0$.
   (b) If $q_y(x) > 0$, then:
      i. Set $a_y \leftarrow q_y(x) + \mathrm{Lap}(2/\varepsilon n)$.
      ii. If $a_y < 2\ln(2/\delta)/\varepsilon n + 1/n$, then set $a_y \leftarrow 0$.
2. Output $(a_y)_{y \in \mathcal{X}}$.

Stability-based Histogram

1. For every point $y \in \mathcal{X}$:
   (a) If $q_y(x) = 0$, then set $a_y = 0$.
   (b) If $q_y(x) > 0$, then:
      i. Set $a_y \leftarrow q_y(x) + \mathrm{Lap}(2/\varepsilon n)$.
      ii. If $a_y < 2\ln(2/\delta)/\varepsilon n + 1/n$, then set $a_y \leftarrow 0$.
2. Output $(a_y)_{y \in \mathcal{X}}$.

Stability-based Histogram

Utility: The algorithm gives exact answers for queries q_y where $q_y(x) = 0$. There are at most n queries q_y with $q_y(x) > 0$ (namely, ones where $y \in \{x_1, \ldots, x_n\}$). By the tails of the Laplace distribution and a union bound, with high probability all of the noisy answers $q_y(x)+\text{Lap}(2/\varepsilon n)$ computed in Step 1(b)i have error at most $O((\log n)/\varepsilon n) \leq O(\log(1/\delta)/\varepsilon n)$. Truncating the small values to zero in Step 1(b)ii introduces an additional error of up to $2\ln(1/\delta)/\varepsilon n + 1/n = O(\log(1/\delta)/\varepsilon n)$.

Stability-based Histogram

Utility: The algorithm gives exact answers for queries q_y where $q_y(x) = 0$. There are at most n queries q_y with $q_y(x) > 0$ (namely, ones where $y \in \{x_1, \ldots, x_n\}$). By the tails of the Laplace distribution and a union bound, with high probability all of the noisy answers $q_y(x)+\mathrm{Lap}(2/\varepsilon n)$ computed in Step 1(b)i have error at most $O((\log n)/\varepsilon n) \leq O(\log(1/\delta)/\varepsilon n)$. Truncating the small values to zero in Step 1(b)ii introduces an additional error of up to $2\ln(1/\delta)/\varepsilon n + 1/n = O(\log(1/\delta)/\varepsilon n)$.

Histogram

Accuracy with the standard histogram DP algorithm:

$$|q_h(D) - r_h| \leq O\left(\frac{log(|\mathcal{X}|)}{n}\right)$$

Accuracy with the stable histogram DP algorithm:

$$|q_h(D) - r_h| \leq O\left(\frac{log(1/\delta)}{n}\right)$$

How can we make this reasoning mathematically precise?

Formal Semantics

We need to assign a formal meaning to the different components:

```
Precondition
Program
Postcondition
```

Formal Semantics

We need to assign a formal meaning to the different components:

`Precondition`

`Program`

`Postcondition`

formal semantics of programs

Formal Semantics

We need to assign a formal meaning to the different components:

Precondition

Program

Postcondition

formal semantics of specification conditions

formal semantics of programs

formal semantics of specification conditions

Formal Semantics

We need to assign a formal meaning to the different components:

Precondition

Program

Postcondition

formal semantics of specification conditions

formal semantics of programs

formal semantics of specification conditions

We also need to describe the rules which combine program and specifications.

An example

```
FastExponentiation(n, k : Nat) : Nat
 n′:= n; k′:= k; r := 1;
 if k′ > 0 then
   while k′ > 1 do
     if even(k′) then
       n′ := n′ ∗ n′;
       k′ := k′/2;
     else
       r := n′ ∗ r;
       n′ := n′ ∗ n′;
       k′ := (k′ − 1)/2;
   r := n′ ∗ r;
 (* result is r *)
```

Programming Language

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

x,y,z,...	program variables
$e_1, e_2, \ldots$	expressions
$c_1, c_2, \ldots$	commands

Expressions

We want to be able to write complex programs with our language.

```
e::= x
   | f(e₁,…,eₙ)
```

Where `f` can be any arbitrary operator.

Some expression examples

`x+5`   `x mod k`   `x[i]`   `(x[i+1] mod 4)+5`

Memories

We can formalize a memory as a map $\mathtt{m}$ from variables to values.

$$\mathtt{m} = [\mathtt{x}_1 \longmapsto \mathtt{v}_1, \ldots, \mathtt{x}_n \longmapsto \mathtt{v}_n]$$

We consider only maps that respect types.

Memories

We can formalize a memory as a map $\mathtt{m}$ from variables to values.

$$\mathtt{m} = [\mathtt{x}_1 \longmapsto \mathtt{v}_1, \ldots, \mathtt{x}_n \longmapsto \mathtt{v}_n]$$

We consider only maps that respect types.

We want to read the value associated to a particular variable:

$$\mathtt{m}(\mathtt{x})$$

We want to update the value associated to a particular variable:

$$\mathtt{m}[\mathtt{x} \leftarrow \mathtt{v}]$$

This is defined as

$$\mathtt{m}[\mathtt{x} \leftarrow \mathtt{v}](\mathtt{y}) = \begin{cases} \mathtt{v} & \text{If} \quad \mathtt{x} = \mathtt{y} \\ \mathtt{m}(\mathtt{y}) & \text{Otherwise} \end{cases}$$

Semantics of Expressions

What is the meaning of the following expressions?

`x+5`   `x mod k`   `x[i]`   `(x[i+1] mod 4)+5`

Semantics of Expressions

What is the meaning of the following expressions?

`x+5` `x mod k` `x[i]` `(x[i+1] mod 4)+5`

We can give the semantics as a relation between expressions, memories and values.

```
Exp * Mem * Val
```

We will denote this relation as:

$$\{e\}_m = v$$

Semantics of Expressions

What is the meaning of the following expressions?

`x+5`   `x mod k`   `x[i]`   `(x[i+1] mod 4)+5`

We can give the semantics as a relation between expressions, memories and values.

```
Exp * Mem * Val
```

We will denote this relation as:

`{e}`$_{\texttt{m}}$`=v`

This is commonly typeset as:

$$[\![e]\!]_m = v$$

Semantics of Commands

What is the meaning of the following command?

```
k:=2; z:=x mod k; if z=0 then r:=1 else r:=2
```

Semantics of Commands

What is the meaning of the following command?

```
k:=2; z:=x mod k; if z=0 then r:=1 else r:=2
```

We can give the semantics as a relation between command, memories and memories or failure.

$$\texttt{Exp * Mem * (Mem | } \perp\texttt{)}$$

We will denote this relation as:

$\{\texttt{c}\}_{\texttt{m}}=\texttt{m}'$ Or $\{\texttt{c}\}_{\texttt{m}}=\perp$

Semantics of Commands

What is the meaning of the following command?

```
k:=2; z:=x mod k; if z=0 then r:=1 else r:=2
```

We can give the semantics as a relation between command, memories and memories or failure.

```
Exp * Mem * (Mem | ⊥)
```

We will denote this relation as:

$\{c\}_m = m'$ Or $\{c\}_m = \perp$

This is commonly typeset as:

$$[\![c]\!]_m = m'$$

Semantics of Commands

This is defined on the structure of commands:

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

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

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

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

$$\{\texttt{c;c'}\}_m = \bot \quad \text{If} \quad \{\texttt{c}\}_m = \bot$$

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

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

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

where

$$\texttt{while}_n \texttt{ e do c} = \texttt{while}^n \texttt{ e do c;if e then abort else skip}$$

and

Semantics of Commands

This is defined on the structure of commands:

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

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

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

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

$$\{\texttt{c;c'}\}_m = \bot \quad \text{If} \quad \{\texttt{c}\}_m = \bot$$

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

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

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

where

$$\texttt{while}_n \texttt{ e do c} = \texttt{while}^n \texttt{ e do c;if e then abort else skip}$$

and

$$\texttt{while}^0 \texttt{ e do c} = \texttt{skip}$$

$$\texttt{while}^{n+1} \texttt{ e do c} = \texttt{if e then (c;while}^n \texttt{ e do c) else skip}$$

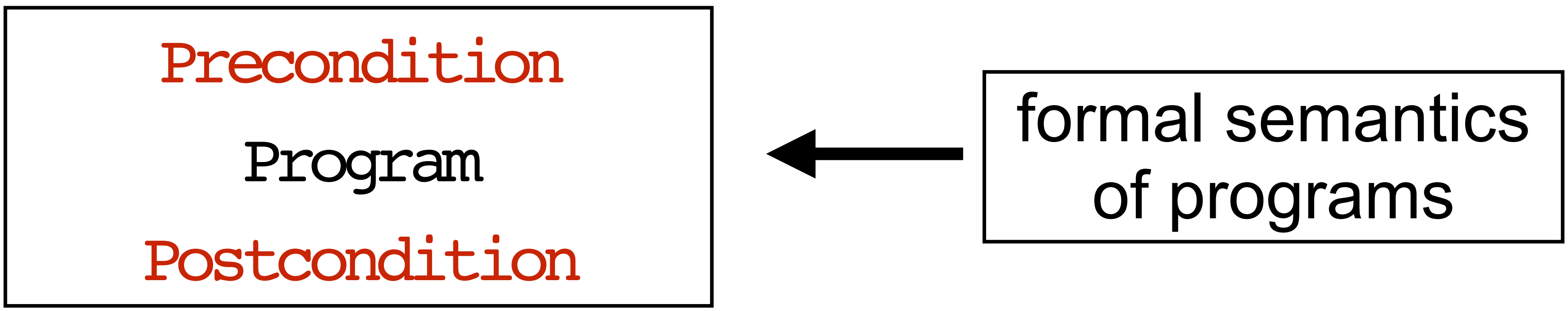
Formal Semantics

We need to assign a formal meaning to the different components:

Precondition
Program
Postcondition

Formal Semantics

We need to assign a formal meaning to the different components:



Formal Semantics

We need to assign a formal meaning to the different components:

`Precondition`

`Program`

`Postcondition`

formal semantics of specification conditions

formal semantics of programs

formal semantics of specification conditions

Formal Semantics

We need to assign a formal meaning to the different components:

formal semantics of specification conditions

Precondition

Program

Postcondition

formal semantics of programs

formal semantics of specification conditions

We also need to describe the rules which combine program and specifications.

Hoare triple

Precondition
(a logical formula)

Precondition

`Program`

Postcondition

$$c : P \Rightarrow Q$$

Program

Postcondition
(a logical formula)

Some examples

```
i:=0;
r:=1;
while(i≤k)do
  r:=r * n;
  i:=i + 1
```

Precondition

$$: \{0 < k\} \Rightarrow \{r = n^k\}$$

Postcondition

Is it a good specification?

Some examples

```
i:=0;
r:=1;
while(i≤k)do
  r:=r * n;
  i:=i + 1
```

Precondition

$$: \{0 < k\} \Rightarrow \{r = n^k\}$$

Postcondition

Is it a good specification?

✗

Some examples

```
i:=0;
r:=1;
while(i≤k)do
  r:=r * n;
  i:=i + 1
```

Precondition

$$: \{0 < k\} \Rightarrow \{r = n^k\}$$

Postcondition

Is it a good specification?

✗

$$m_{in} = [k = 1, n = 2, i = 0, r = 0]$$

$$m_{out} = [k = 1, n = 2, i = 2, r = 4]$$

Some examples

```
i:=0;
r:=1;
while(i<k)do
  r:=r * n;
  i:=i + 1
```

Precondition

$$: \{0 \le k\} \Rightarrow \{r = n^k\}$$

Postcondition

Is it a good specification?

Some examples

```
i:=0;
r:=1;
while(i<k)do
  r:=r * n;
  i:=i + 1
```

Precondition

$$: \{0 \leq k\} \Rightarrow \{r = n^k\}$$

Postcondition

Is it a good specification?

✓

Some examples

```
i:=0;
r:=1;
while(i≤k)do
  r:=r * n;
  i:=i + 1
```

Precondition

$$: \{0 \leq k\} \Rightarrow \{r = n^i\}$$

Postcondition

Is it a good specification?

Some examples

```
i:=0;
r:=1;
while(i≤k)do
  r:=r * n;
  i:=i + 1
```

Precondition

$$: \{0 \leq k\} \Rightarrow \{r = n^i\}$$

Postcondition

Is it a good specification?

✓

How do we determine the validity of an Hoare triple?

Validity of Hoare triple

Precondition
(a logical formula)

$$c : P \Rightarrow Q$$

Program

Postcondition
(a logical formula)

Validity of Hoare triple

Precondition
(a logical formula)

We are interested only in inputs that meets P and we want to have outputs satisfying Q.

$$c : P \Rightarrow Q$$

Program

Postcondition
(a logical formula)

Validity of Hoare triple

Precondition
(a logical formula)

$$c : P \Rightarrow Q$$

Program

Postcondition
(a logical formula)

We are interested only in inputs that meets P and we want to have outputs satisfying Q.

How shall we formalize this intuition?

Validity of Hoare triple

We say that the triple $\mathtt{c:P \Rightarrow Q}$ is valid if and only if

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

and memory m' such that $\mathtt{\{c\}_m = m'}$

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

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 $Q(m')$.

Is this condition easy to check?

Rules of Hoare Logic:

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

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

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

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

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

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

Rules of Hoare Logic:

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

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

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

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

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

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

Rules of Hoare Logic:

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

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

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

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

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

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

Some examples

$$\vdash x := z * 2; z := x * 2$$

$$: \{z * 4 = 8\} \Rightarrow \{z = 8\}$$

Is this a valid triple?

Some examples

$$\vdash x := z * 2; z := x * 2 \\ : \{z * 4 = 8\} \Rightarrow \{z = 8\}$$

Is this a valid triple? ✓

Some examples

$$\vdash x := z * 2; z := x * 2$$
$$: \{z * 4 = 8\} \Rightarrow \{z = 8\}$$

Is this a valid triple? ✓

Can we prove it with the rules that we have?

Some examples

$$\vdash x := z * 2; z := x * 2 \\ : \{z * 4 = 8\} \Rightarrow \{z = 8\}$$

Is this a valid triple? ✓

Can we prove it with the rules that we have? ✓

Some Examples

$$\dfrac{\dfrac{\{z*4=8\} \Rightarrow \{(z*2)*2=8\} \qquad \dfrac{}{\vdash x := z*2\ \{(z*2)*2=8\} \Rightarrow \{x*2=8\}}}{\vdash x := z*2\!:\ \{z*4=8\} \Rightarrow \{x*2=8\}} \qquad \dfrac{}{\vdash z := x*2: \{x*2=8\} \Rightarrow \{z=8\}}}{\vdash x := z*2; z := x*2\!:\ \{z*4=8\} \Rightarrow \{z=8\}}$$

Soundness

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

Relative Completeness

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

Relative Completeness

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

If a triple $c : Pre \Rightarrow Post$ 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 $\vdash c : Pre \Rightarrow Post$ through the rules of the logic.

A logic for 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.

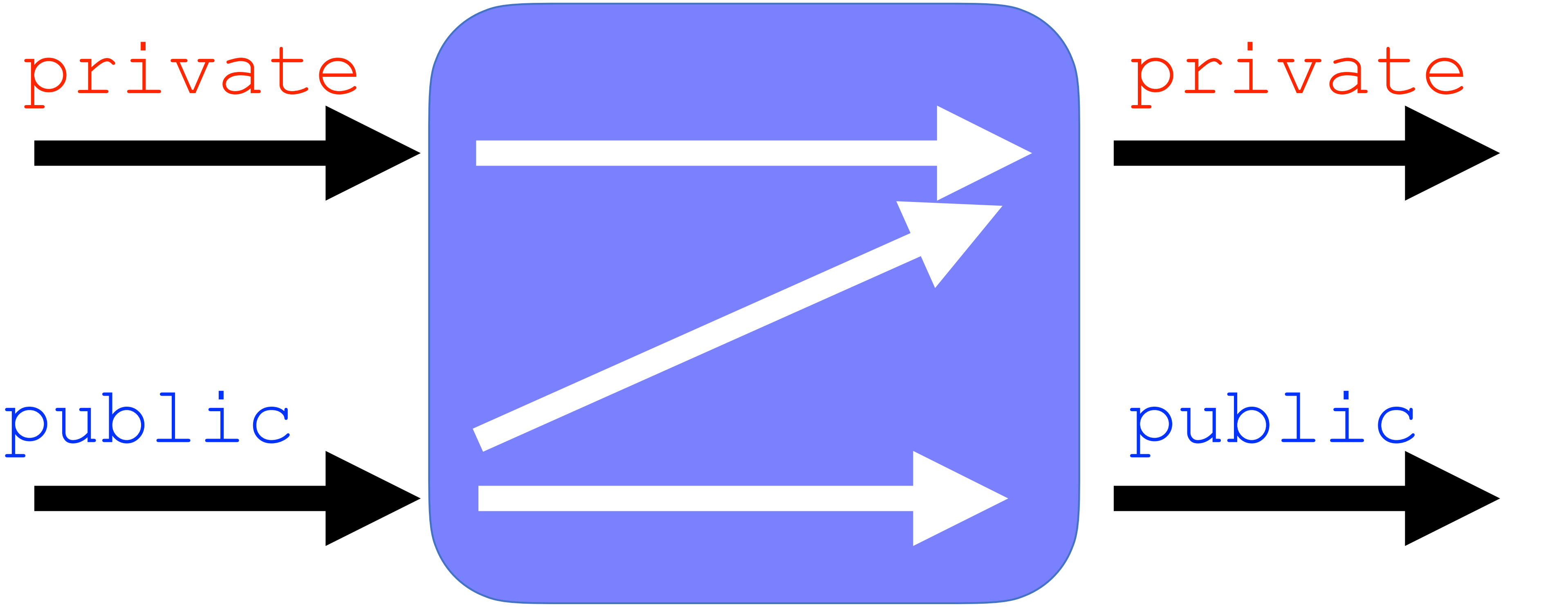
`x:public`   `x:private`

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.

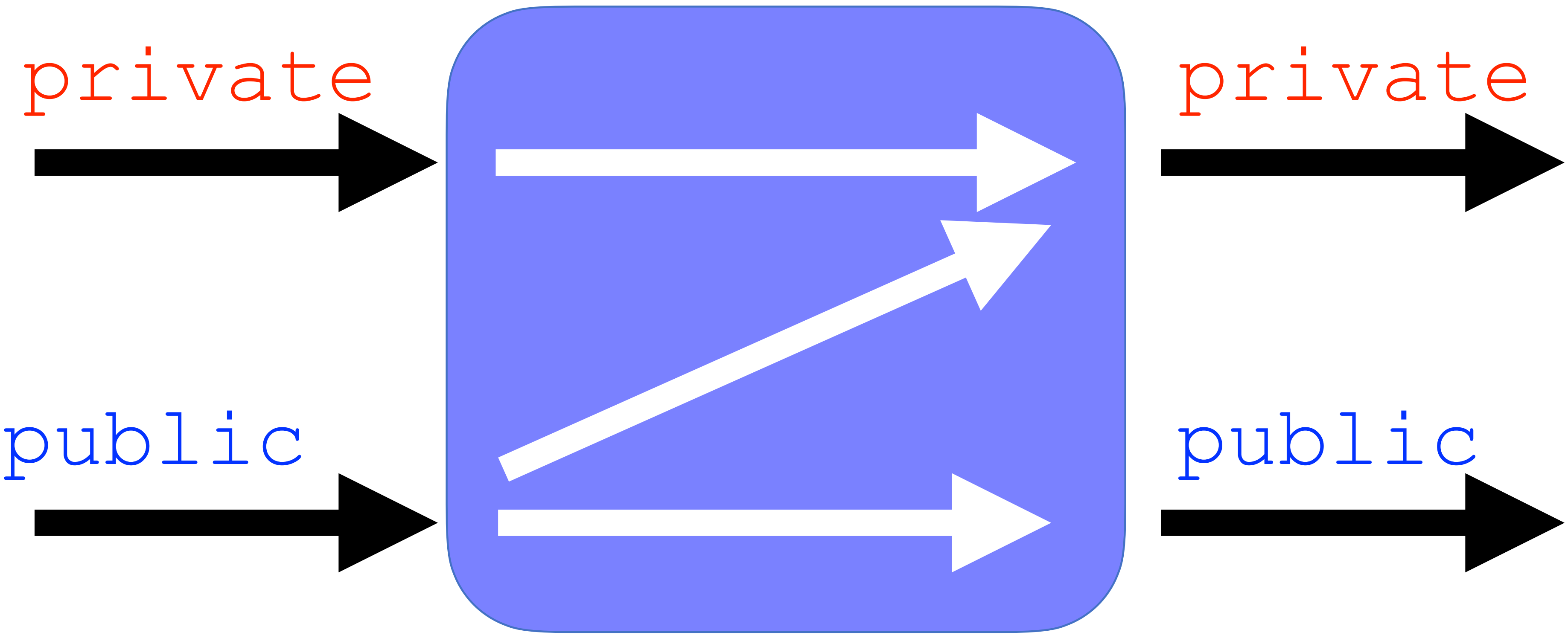
In symbols: $m_1 \sim_{low} m_2$

Noninterference

In symbols

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

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
```

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

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

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

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

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

Does this program

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

Does this program

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

Yes

Does this program

```
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

```
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

```
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

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

Does this program

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

No

Does this program

```
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

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

$\boxed{\text{No}}$

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

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

Does this program

```
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]

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

m^{out}_2=[x=5,y=0]

Does this program

```
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

```
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?

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 $Q(m')$.

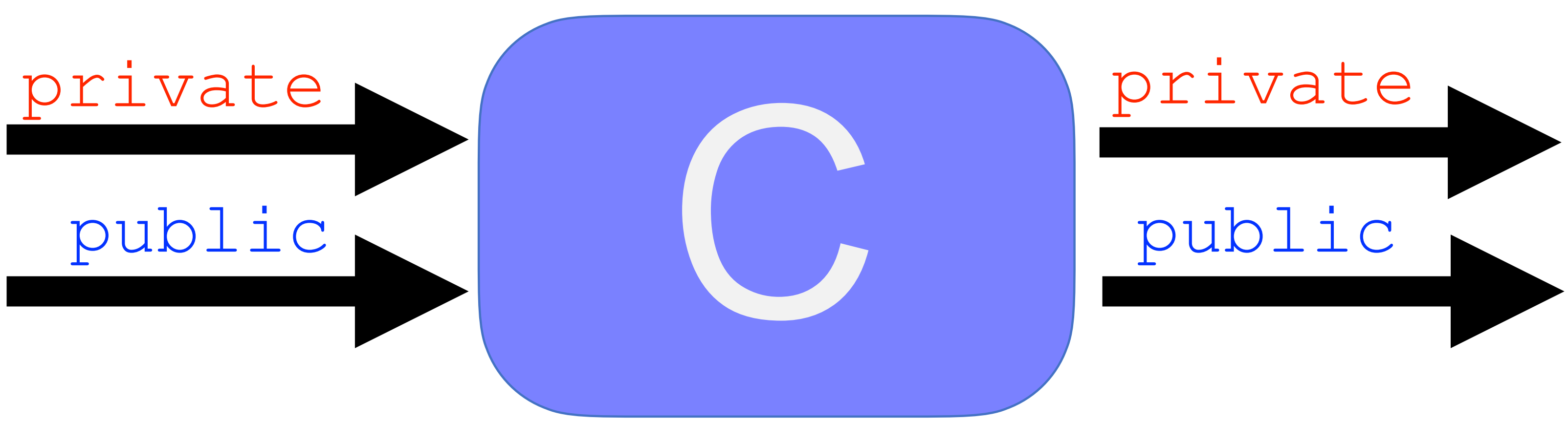
Validity talks only about one memory. How can we manage two memories?

Relational Property

In symbols, c is noninterferent if and only if for every $m_1 \sim_{low} m_2$, $\{c\}_{m1} = m_1'$ and $\{c\}_{m2} = m_2'$ implies $m_1' \sim_{low} m_2'$

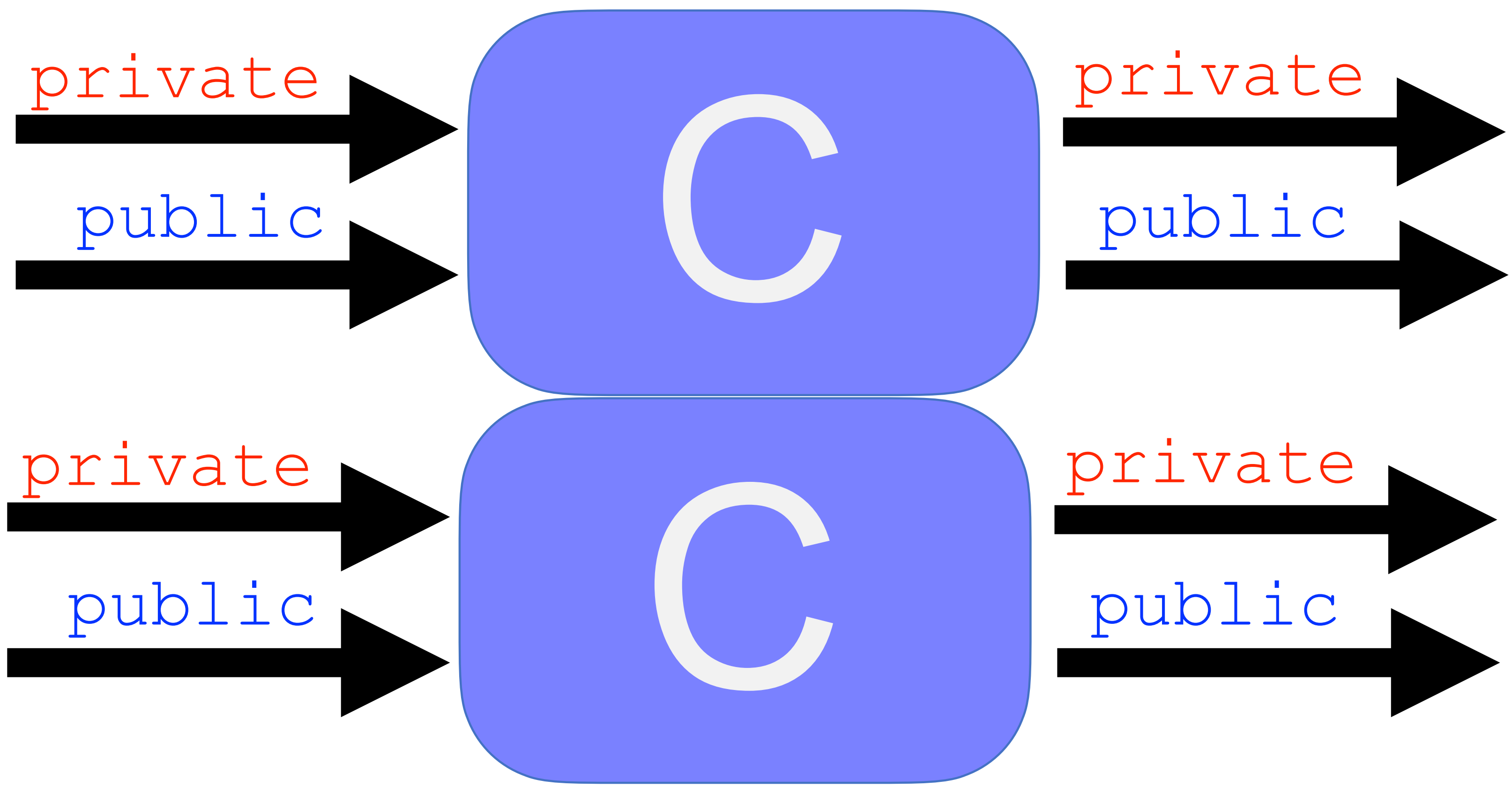
Relational Property

In symbols, c is noninterferent if and only if for every $m_1 \sim_{low} m_2$, $\{c\}_{m1} = m_1'$ and $\{c\}_{m2} = m_2'$ implies $m_1' \sim_{low} m_2'$



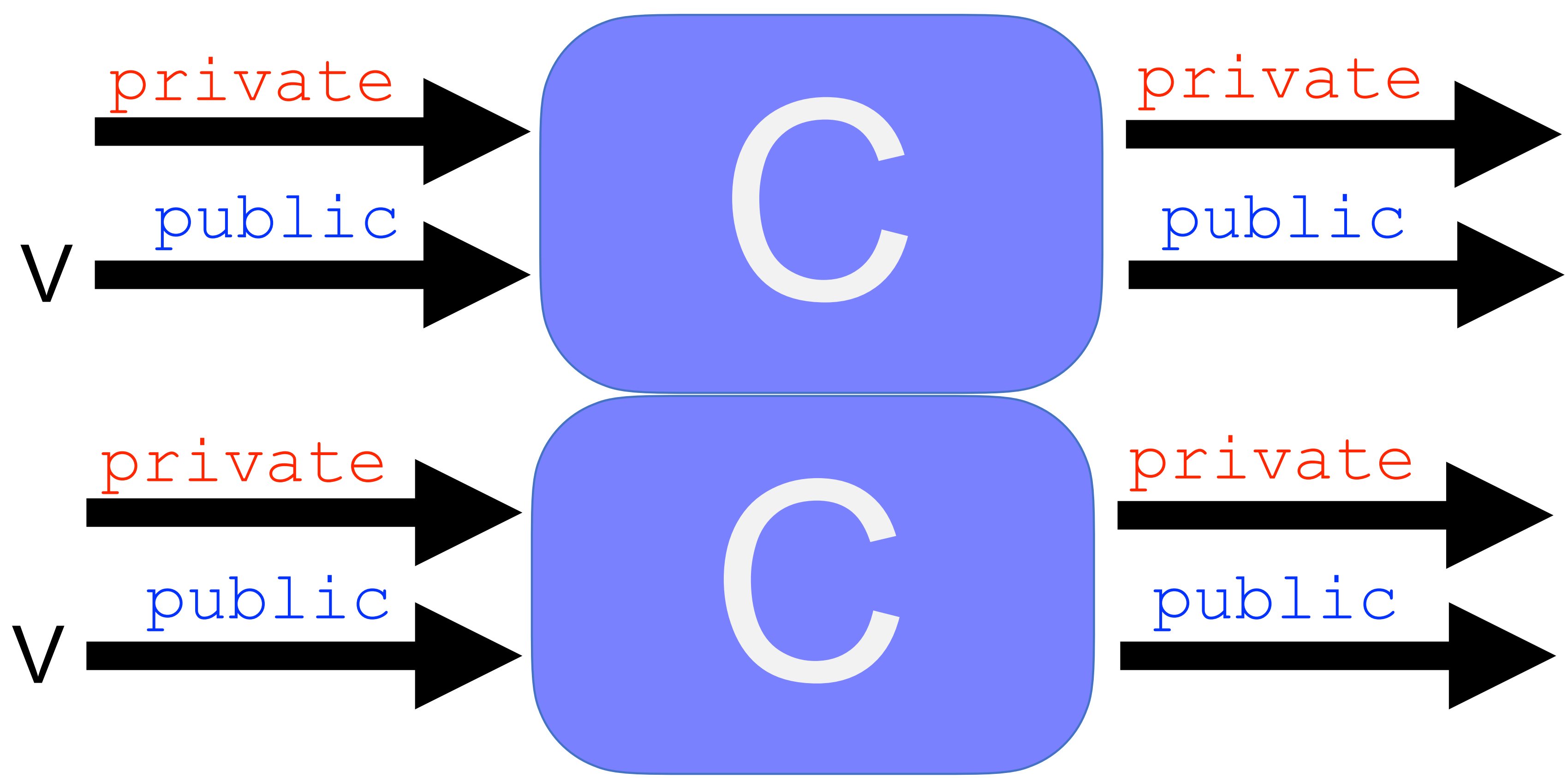
Relational Property

In symbols, c is noninterferent if and only if for every $m_1 \sim_{low} m_2$, $\{c\}_{m1}=m_1'$ and $\{c\}_{m2}=m_2'$ implies $m_1' \sim_{low} m_2'$



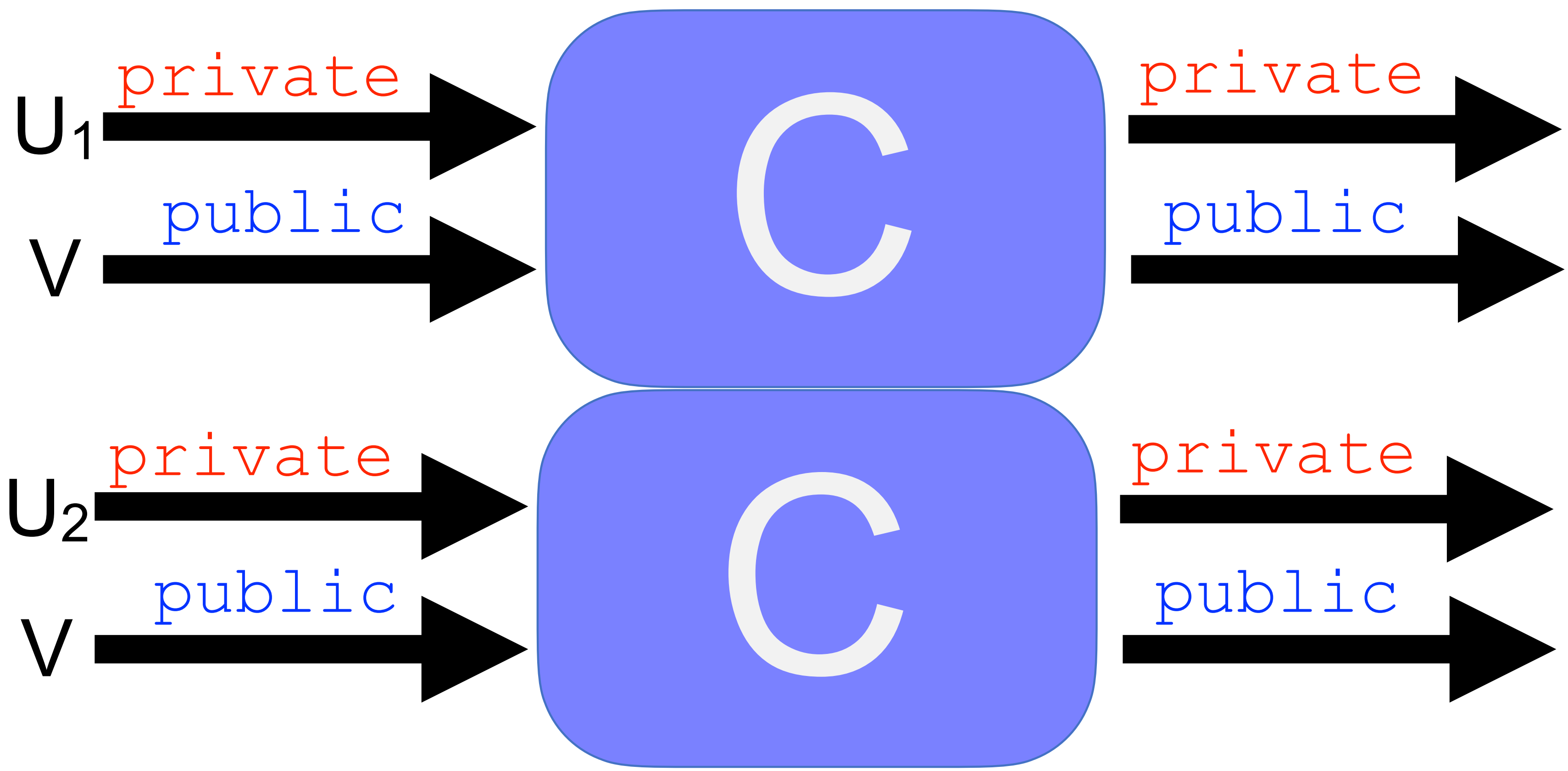
Relational Property

In symbols, c is noninterferent if and only if for every $m_1 \sim_{low} m_2$, $\{c\}_{m1} = m_1'$ and $\{c\}_{m2} = m_2'$ implies $m_1' \sim_{low} m_2'$



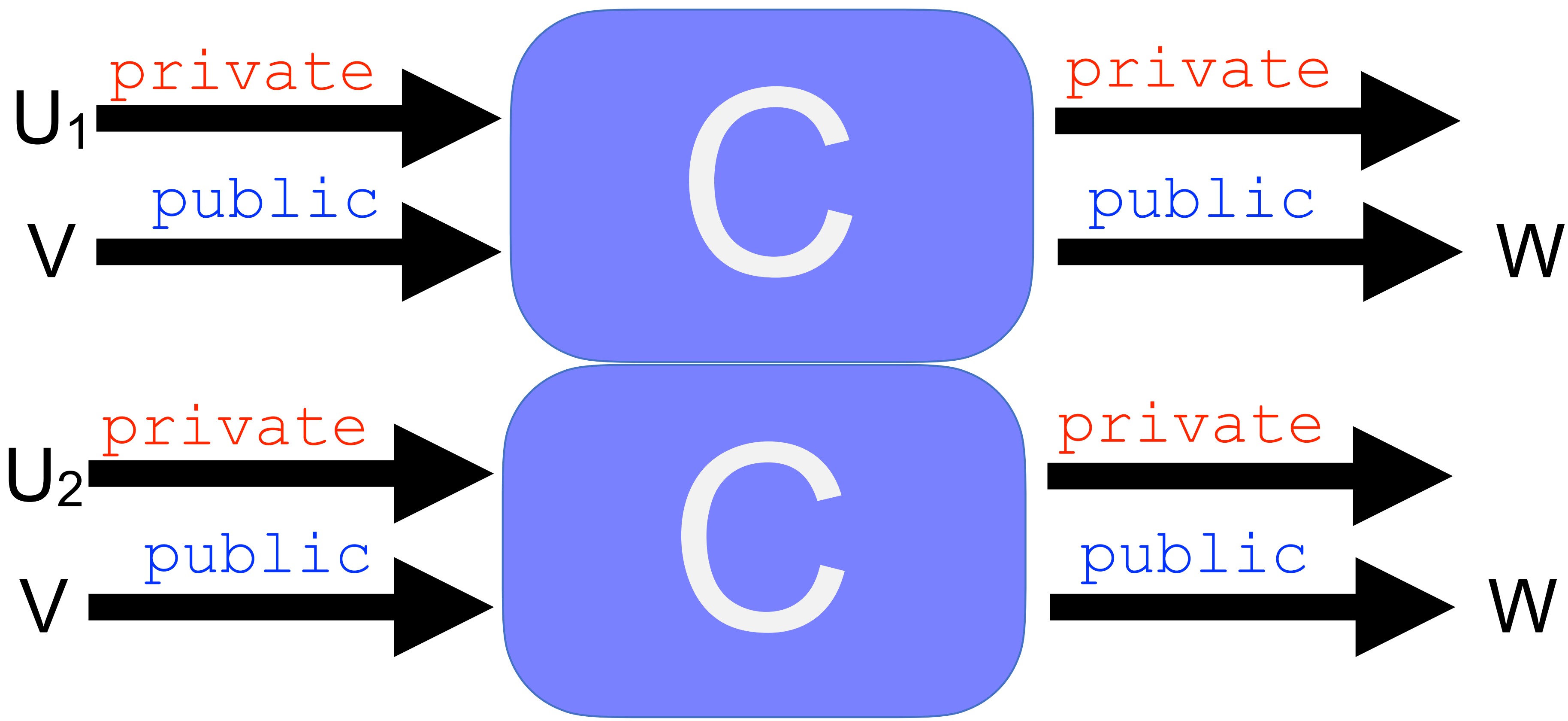
Relational Property

In symbols, c is noninterferent if and only if for every $m_1 \sim_{low} m_2$, $\{c\}_{m1}=m_1'$ and $\{c\}_{m2}=m_2'$ implies $m_1' \sim_{low} m_2'$



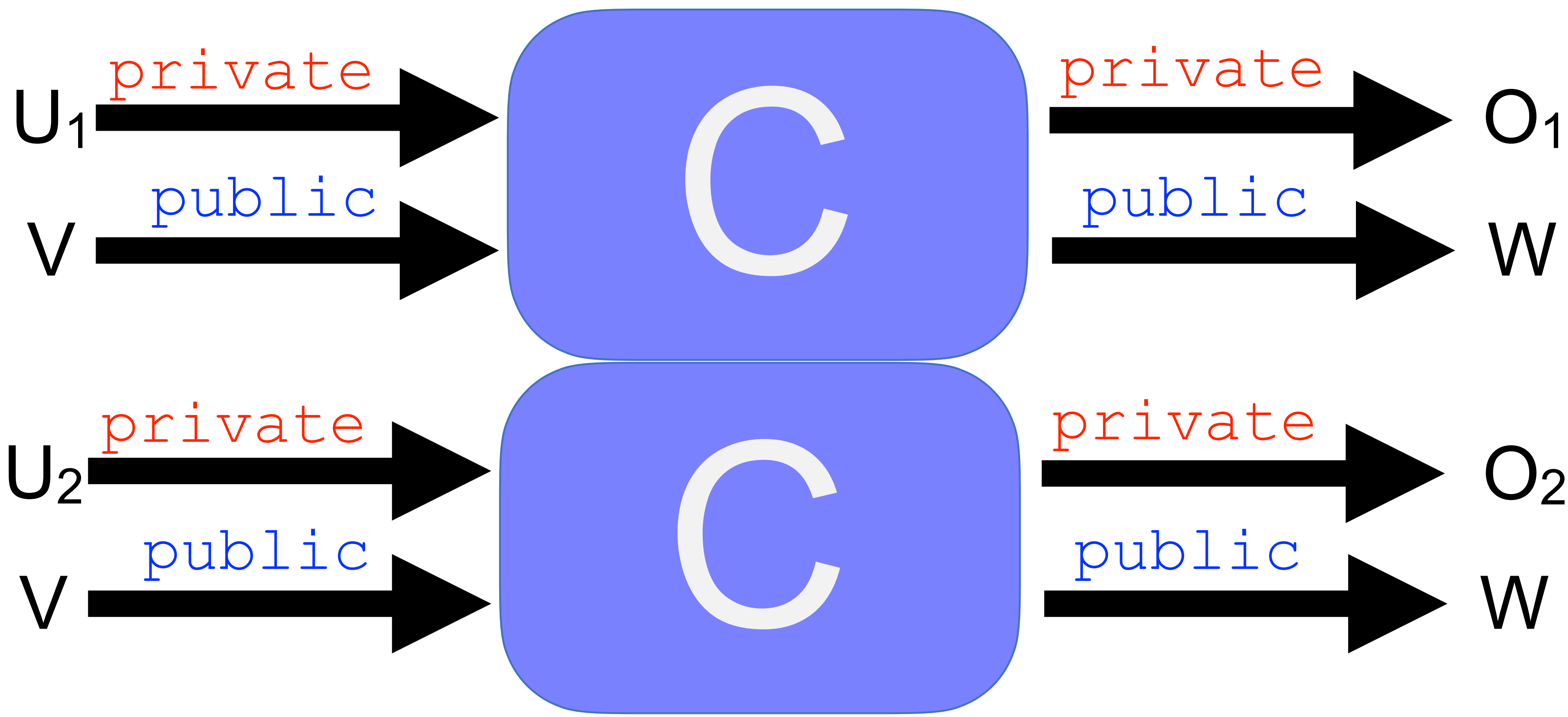
Relational Property

In symbols, c is noninterferent if and only if for every $m_1 \sim_{low} m_2$, $\{c\}_{m1}=m_1'$ 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 $m_1 \sim_{low} m_2$, $\{c\}_{m1}=m_1'$ and $\{c\}_{m2}=m_2'$ implies $m_1' \sim_{low} m_2'$



Relational Hoare Logic - RHL

Precondition
(a logical formula)

`Precondition`
`Program`$_1$ `~ Program`$_2$
`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.

Rules of Relational Hoare Logic
Skip

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

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 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
If then else

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

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

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

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

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

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

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

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

Soundness

If we can derive $\vdash c_1 \sim c_2 : P \Rightarrow Q$ through the rules of the logic, then the quadruple

$c_1 \sim c_2 : P \Rightarrow Q$ is valid.

Relative Completeness

If a quadruple $c_1 \sim c_2 : P \Rightarrow Q$ 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 $\vdash c_1 \sim c_2 : P \Rightarrow Q$ through the rules of the logic.

Soundness and completeness with respect to Hoare Logic

$$\vdash_{\mathtt{RHL}} \mathtt{c}_1 \sim \mathtt{c}_2 : \mathtt{P} \Rightarrow \mathtt{Q}$$

iff

$$\vdash_{\mathtt{HL}} \mathtt{c}_1 ; \mathtt{c}_2 : \mathtt{P} \Rightarrow \mathtt{Q}$$

Soundness and completeness with respect to Hoare Logic

$$\vdash_{\mathtt{RHL}} \mathtt{c_1 \sim c_2 : P \Rightarrow Q}$$

iff

$$\vdash_{\mathtt{HL}} \mathtt{c_1 ; c_2 : P \Rightarrow Q}$$

Under the assumption that we can partition the memory adequately, and that we have termination.

Questions?