CS 599: Formal Methods in Security and Privacy
Noninterference and Relational Hoare Logic

Marco Gaboardi
gaboardi@bu.edu

Alley Stoughton
stough@bu.edu
From the previous classes
Information Flow Control

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

private → private
public → public
NonInterference

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

1) $\{c\}_{m_1} = \bot$ iff $\{c\}_{m_2} = \bot$

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

[Diagram of a system with inputs and outputs, labeled with private and public access]
Relational Hoare Logic - RHL

Program₁ \sim \text{Program₂}

Postcondition

\begin{align*}
& c₁ \sim c₂ : P \Rightarrow Q \\
& \text{Precondition (a logical formula)} \\
& \text{Postcondition (a logical formula)}
\end{align*}
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) $\{c_1\}_{m_1} = \bot$ iff $\{c_2\}_{m_2} = \bot$

2) $\{c_1\}_{m_1} = m_1'$ and $\{c_2\}_{m_2} = m_2'$ implies $Q(m_1', m_2')$. 
Some Rules of Relational Hoare Logic

\[ \vdash \text{skip} \leftarrow \text{skip} : P \Rightarrow P \]

\[ \vdash \text{abort} \leftarrow \text{abort} : \text{true} \Rightarrow \text{false} \]

\[ \vdash x_1 := e_1 \leftarrow x_2 := e_2 : P[e_1<1>/x_1<1>, e_2<2>/x_2<2>] \Rightarrow P \]

\[ \vdash c_1 \leftarrow c_2 : P \Rightarrow R \quad \vdash c_1' \leftarrow c_2' : R \Rightarrow S \]

\[ \vdash c_1 ; c_1' \leftarrow c_2 ; c_2' : P \Rightarrow S \]

\[ P \Rightarrow S \quad \vdash c_1 \leftarrow c_2 : S \Rightarrow R \quad R \Rightarrow Q \]

\[ \vdash c_1 \leftarrow c_2 : P \Rightarrow Q \]
Some Rules of Relational Hoare Logic

\[ \vdash c_1 \sim c_2 : e_1 < 1 > \wedge P \Rightarrow Q \quad P \Rightarrow e_1 < 1 > \triangleright e_2 < 2 > \]

\[ \vdash c_1' \sim c_2' : \neg e_1 < 1 > \wedge P \Rightarrow Q \]

\[ \vdash \text{if } e_1 \text{ then } c_1 \text{ else } c_1' : P \Rightarrow Q \]

\[ \vdash \text{if } e_2 \text{ then } c_2 \text{ else } c_2' \]

\[ \vdash c_1 \sim c_2 : e_1 < 1 > \wedge P \Rightarrow P \quad P \Rightarrow e_1 < 1 > \triangleright e_2 < 2 > \]

\[ \vdash \text{while } e_1 \text{ do } c_1 \]

\[ \vdash \text{while } e_2 \text{ do } c_2 : P \Rightarrow P \wedge \neg e_1 < 1 > \]
One-sided Rules

\[ \vdash c_1 \sim c_2 : e<1> \land P \Rightarrow Q \quad \vdash c_1' \sim c_2 : \neg e<1> \land P \Rightarrow Q \]

\[ \vdash \text{if } e \text{ then } c_1 \text{ else } c_1' : P \Rightarrow Q \]

\[ \vdash c_1 \sim c_2 : e<2> \land P \Rightarrow Q \quad \vdash c_1 \sim c_2' : \neg e<2> \land P \Rightarrow Q \]

\[ \vdash \text{if } e \text{ then } c_2 \sim c_2' : P \Rightarrow Q \]
Today: More Relational Hoare Logic
Assignment Example

⊢ \begin{align*}
x & := x + 1 \sim y := y - 1 : \\
& x<1> + 1 = -(y<2> - 1) \Rightarrow x<1> = -y<2>
\end{align*}
Assignment Example

\[ \vdash x := x + 1 \sim y := y - 1 : \\
(x <1> = -y<2>) \\
[ (x+1)<1>/x<1>, (y-1)<2>/y<2>] \\
\Rightarrow \\
x<1> = -y<2> \]
Assignment Example

⊢ x := x + 1 ~ y := y - 1:

(x<1> = -y<2>)

[(x<1>+1)/x<1>, (y<2>-1)/y<2>]

⇒

x<1> = -y<2>
Consequence + Assignment

Example

\[ x_{<1>} = -y_{<2>} \Rightarrow x_{<1>} + 1 = -(y_{<2>} - 1) \]

\[ \vdash x := x + 1 \sim y := y - 1: \]

\[ x_{<1>} + 1 = -(y_{<2>} - 1) \Rightarrow x_{<1>} = -y_{<2>} \]

\[ x_{<1>} = -y_{<2>} \Rightarrow x_{<1>} = -y_{<2>} \]

\[ \vdash x := x + 1 \sim y := y - 1: \]

\[ x_{<1>} = -y_{<2>} \Rightarrow x_{<1>} = -y_{<2>} \]
How can we prove this?

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

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

\[
\implies \quad =_{\text{low}} \Rightarrow =_{\text{low}}
\]
Rules of Relational Hoare Logic
If then else - right

\[ \vdash \neg c_1 \land P \Rightarrow Q \]

\[ \vdash \neg c_1 \land P \Rightarrow Q \]

\[ \vdash \neg \neg c_1 \land P \Rightarrow Q \]

\[ \vdash if e then c_2 else c_2' : P \Rightarrow Q \]
How can we prove this?

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

\[
y := 0 \\
z := 0 \\
\text{if } x = 0 \text{ then } z := 1; \\
\text{if } z = 0 \text{ then } y := 1
\]

\[
\Rightarrow \quad = \text{low} \quad \Rightarrow \quad = \text{low}
\]
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
  else
    i:=i+1

: n>0 /
  =low => =low
What do we do if our two programs have different forms? There are three pairs of one-sided rules.

\[
\begin{align*}
\text{if } e \text{ then } c_1 \text{ else } c_1' \\
\vdash \\
\sim \\
C_2 \\
: P \Rightarrow Q
\end{align*}
\]
Rules of Relational Hoare Logic
If-then-else — left

\[ \vdash c_1 \sim c_2 : e < 1> \land P \Rightarrow Q \]

\[ \vdash c_1' \sim c_2 : \neg e < 1> \land P \Rightarrow Q \]

\[ \vdash \sim c_2 : P \Rightarrow Q \]

if \( e \) then \( c_1 \) else \( c_1' \)
Rules of Relational Hoare Logic
If-then-else — right

\[ \vdash c_1 \sim c_2 : e<2> \land P \Rightarrow Q \]
\[ \vdash c_1 \sim c_2' : \neg e<2> \land P \Rightarrow Q \]

\[ \vdash \frac{c_1}{if \ e \ then \ c_2 \ else \ c_2'} : P \Rightarrow Q \]
Rules of Relational Hoare Logic
Assignment — left

\[ \vdash x:=e \sim \text{skip}: \quad P[e<1>/x<1>] \Rightarrow P \]
Assignment — left

\[\begin{align*}
\vdash & x := e \sim \text{skip:} \\
\quad P[e<1>/x<1>] & \Rightarrow P
\end{align*}\]
Assignment — right

⊢ skip \sim x := e :

P[e<2>/x<2>] \Rightarrow P

Also pair of one-sided rules for while — we’ll ignore for now
Rules of Relational Hoare Logic
Program Equivalence Rule

\[ \vdash P : c_1' \equiv c_1 \]
\[ \vdash P : c_2' \equiv c_2 \quad \vdash c_1' \sim c_2' : P \Rightarrow Q \]
\[ \vdash c_1 \sim c_2 : P \Rightarrow Q \]

\[ \vdash P : c_1 \equiv c_2 \text{ means } \{ c_1 \}_m = \{ c_2 \}_m \]
for all m such that \( P(m) \)
Rules of Relational Hoare Logic

Program Equivalences

\[ \vdash P : \text{skip}; c \equiv c \]

\[ \vdash P : c; \text{skip} \equiv c \]

\[ \vdash P : (c_1; c_2); c_3 \equiv c_1; (c_2; c_3) \]

...
We can combine the Composition and Program Equivalence Rules to split commands where we like:

\[ \vdash C_1; C_2 \sim C_1' : P \Rightarrow R \]

\[ \vdash C_3 \sim C_2' ; C_3' : R \Rightarrow Q \]

\[ \vdash C_1; C_2 ; C_3 \sim C_1' ; C_2' ; C_3' : P \Rightarrow Q \]
Rules of Relational Hoare Logic
Combining Composition and Equivalence

\[ \vdash c_1 \sim \text{skip}: \, P \Rightarrow R \]

\[ \vdash c_2 \sim c_1': \, R \Rightarrow Q \]

\[ \vdash c_1; c_2 \sim \text{skip}; c_1': \, P \Rightarrow Q \]

\[ \vdash c_1; c_2 \sim c_1': \, P \Rightarrow Q \]
Rules of Relational Hoare Logic
Combining Composition and Equivalence

\[ \vdash c_1 \sim c_1' : P \Rightarrow R \]
\[ \vdash c_2 \sim \text{skip} : R \Rightarrow Q \]

\[ \vdash c_1; c_2 \sim c_1' ; \text{skip} : P \Rightarrow Q \]

\[ \vdash c_1; c_2 \sim c_1' : P \Rightarrow Q \]
Relational Hoare Logic in EasyCrypt

• EasyCrypt’s implementation of Relational Hoare Logic has much in common with its implementation of Hoare Logic.

• Look for the pRHL tactics in Section 3.4 of the EasyCrypt Reference Manual (the “p” stands for “probabilistic”, but ignore that for now).
Soundness

If we can derive $\Gamma \vdash c_1 \sim c_2 : P \Rightarrow Q$ through the rules of the logic, then the quadruple $c_1 \sim c_2 : P \Rightarrow Q$ is valid.
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) $\{c_1\}_{m_1} = \bot$ iff $\{c_2\}_{m_2} = \bot$

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

How do we check this?
RelativeCompleteness

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

\[ \vdash_{RHL} c_1 \sim c_2 : P \Rightarrow Q \]

iff

\[ \vdash_{HL} c_1 ; c_2 : P \Rightarrow Q \]

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

In Easycrypt
• Look at how to guarantee trace-based noninterference.
• Look at how to guarantee side-channel free noninterference.
• Look at the relations between self-composition and relational logic.

Not related to Easycrypt
• Look at type systems for non-interference.
• Look at other methods for relational reasoning
• Look at declassification