CS 5430
Information-Flow Policies

Elisavet Kozyri

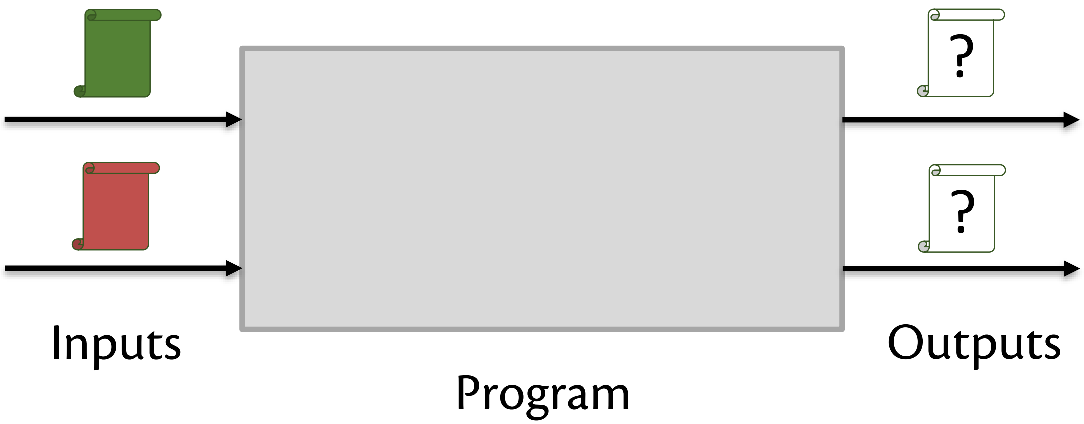
Spring 2019


RESTRICTIONS
APPLY

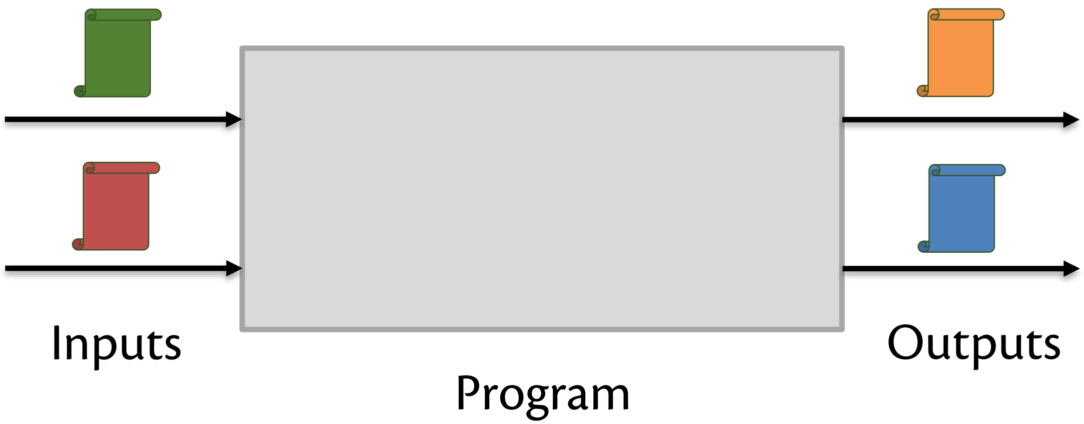
Restrictions on data

- Confidentiality
  - Who is trusted with information.
- Integrity
  - Who trusts the information.
    - Depends on trusting past writers.

Problem: Given restrictions on inputs, what are the restrictions on outputs?

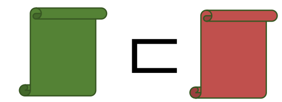


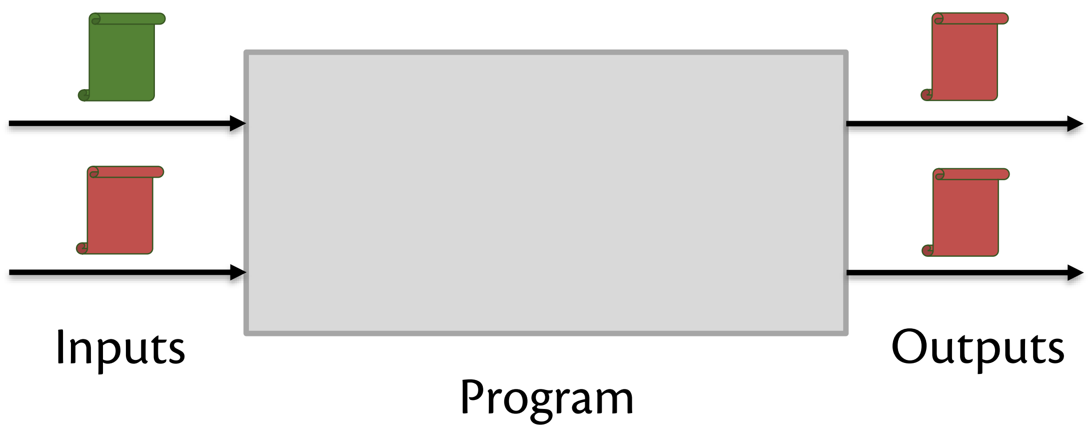
1^{st} solution



- Manual assignment of restrictions to output data.
- Does not scale to rich data ecosystems.

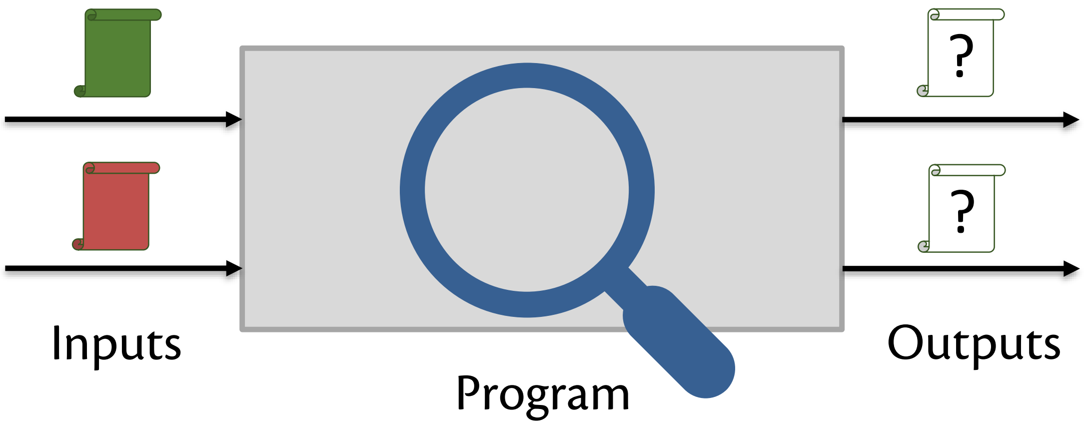
2nd solution





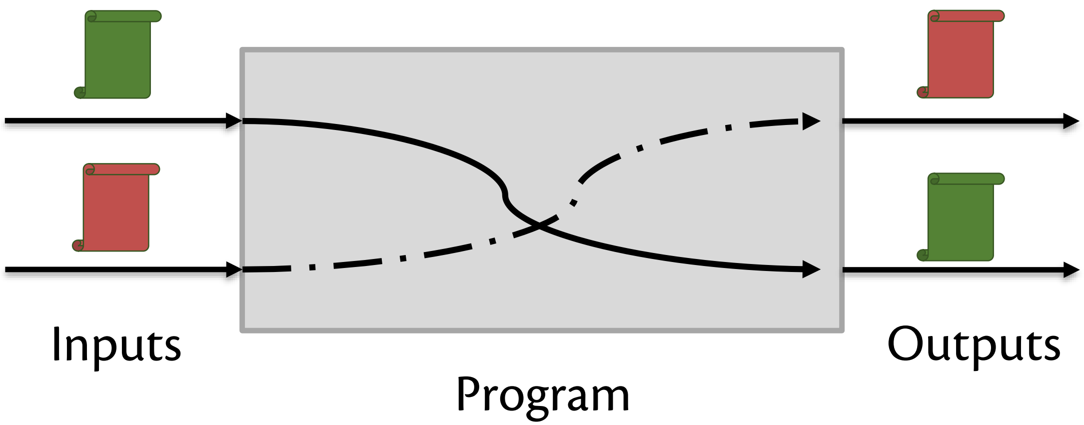
- Can be automated.
- Independent of the program code.
- Produces conservative restrictions.

3rd solution: Information Flow Control



- Program analysis to deduce information flows from inputs to outputs.

Information Flow Control



- Program analysis to deduce information flows from inputs to outputs.
- Restrictions are propagated along the flow.
- More permissive than 2nd solution.

When does a program cause a flow?

$x := y \bmod 2$

y flows to x

$x := y * 0$

no flow

$z := y + 2; \quad x := z$

$z := y + 2; \quad x := z - y$

Flow is not always transitive!

When does a program cause a flow?

$$\text{if } y > 0 \text{ then } x \coloneqq 1 \text{ else } x \coloneqq 2$$

$$\text{if } y > 0 \text{ then } x \coloneqq 0 \text{ else } x \coloneqq 0$$

$$\begin{aligned}\text{if } y > 0 \text{ then } & x \coloneqq 1;\ x \coloneqq 0 \\ \text{else } & x \coloneqq 2;\ x \coloneqq 0\end{aligned}$$

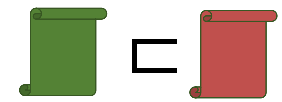
When does a program cause a flow?

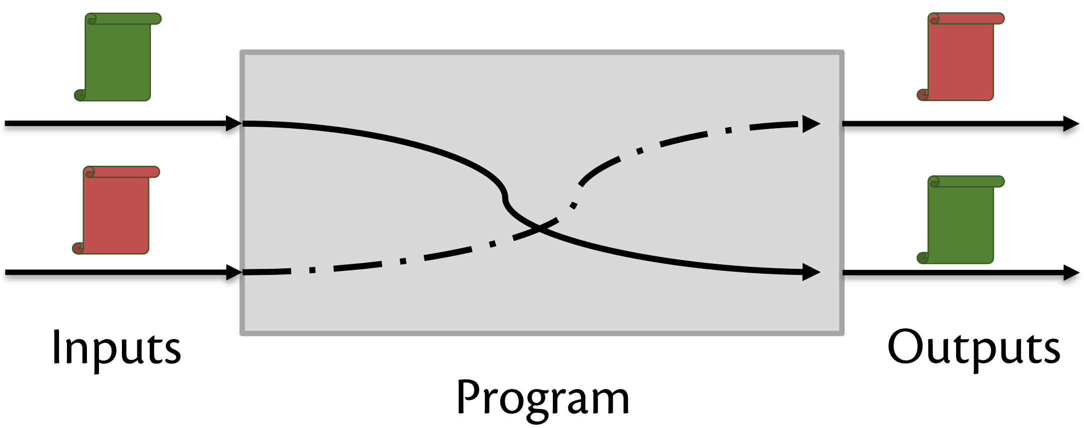
$$\text{while } y > 0 \text{ do x} \coloneqq \text{x} + 1; \text{y} \coloneqq \text{y} - 1 \text{ end}$$

?

$$\text{while } y > 0 \text{ do C end}; \quad x \coloneqq 1$$

Information Flow Control





- Program analysis to deduce information flows from inputs to outputs.
- Restrictions are propagated along the flow.

Information Flow (IF) Policies

- An IF policy specifies **restrictions** on the associated data, and on all its derived data.
- IF policy for confidentiality:
  - Value v *and all its derived values* are allowed to be read at most by Alice.

Information Flow (IF) Policies

- An IF policy specifies **restrictions** on the associated data, and on all its derived data.
- IF policy for confidentiality:
  - Value v *and all its derived values* are allowed to be read at most by Alice.
  - Equivalently, v is allowed to **flow** only to Alice.

Labels to represent IF policies

Examples for confidentiality:

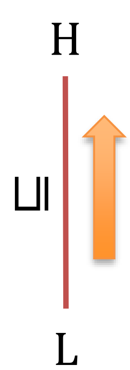
- Classifications
  - Unclassified (**U**), Confidential (**C**), Secret (**S**), Top Secret (**TS**)
  - Low confidentiality (**L**), High confidentiality (**H**)
- Sets of principals:
  - {Alice, Bob}, {Alice}, {Bob}, {}

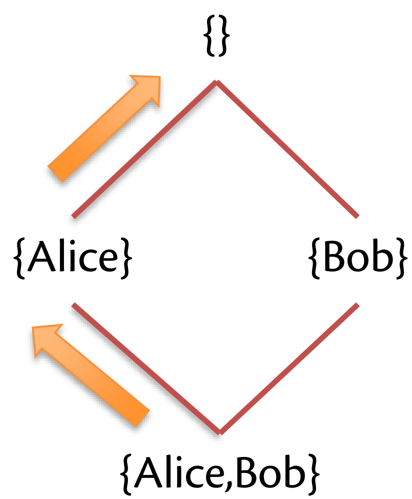
Information flow labels

- They form a *lattice* $\langle L, \sqsubseteq \rangle$ with join operation $\sqcup$.
- For ℓ, ℓ'∈ L, if ℓ $\sqsubseteq$ ℓ', then:

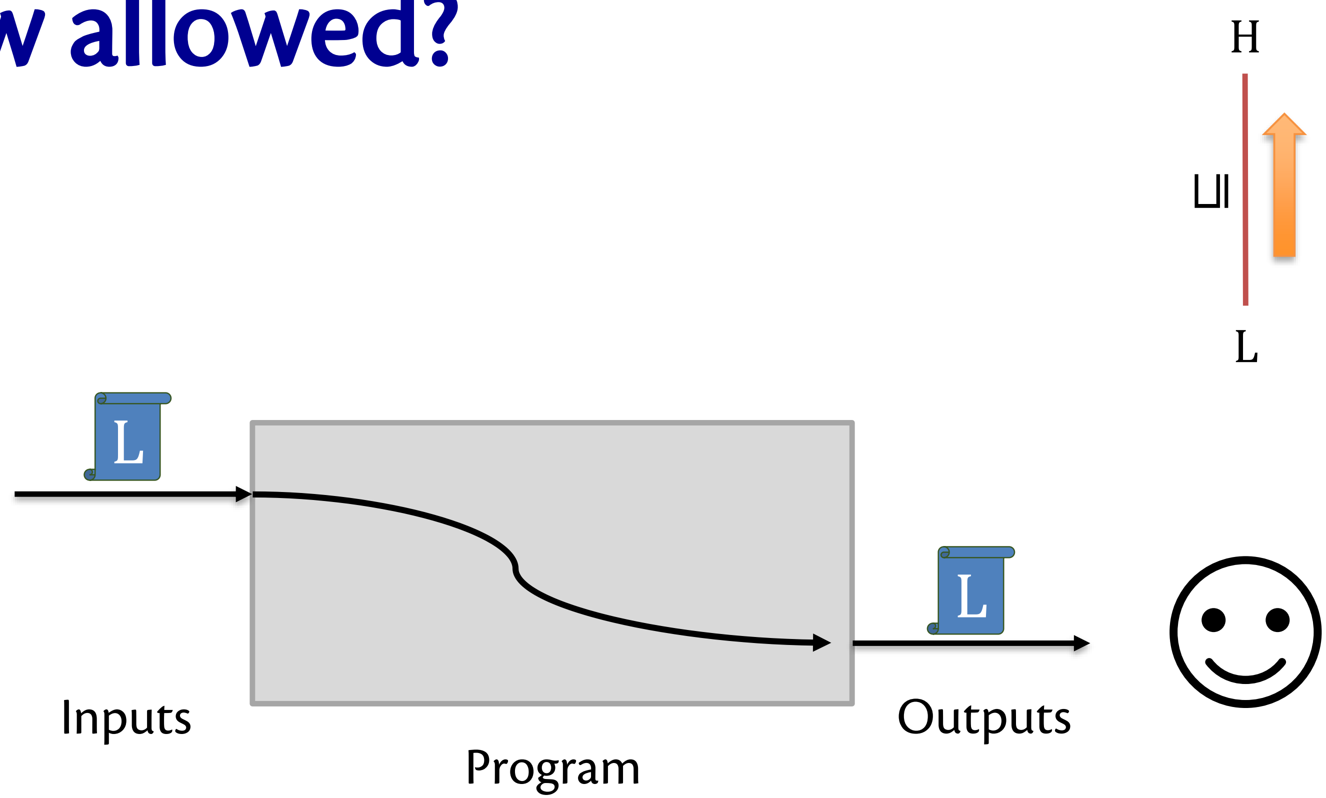
  ℓ' is *at least as restrictive as* ℓ, and thus,

  information flow from ℓ to ℓ' is allowed.

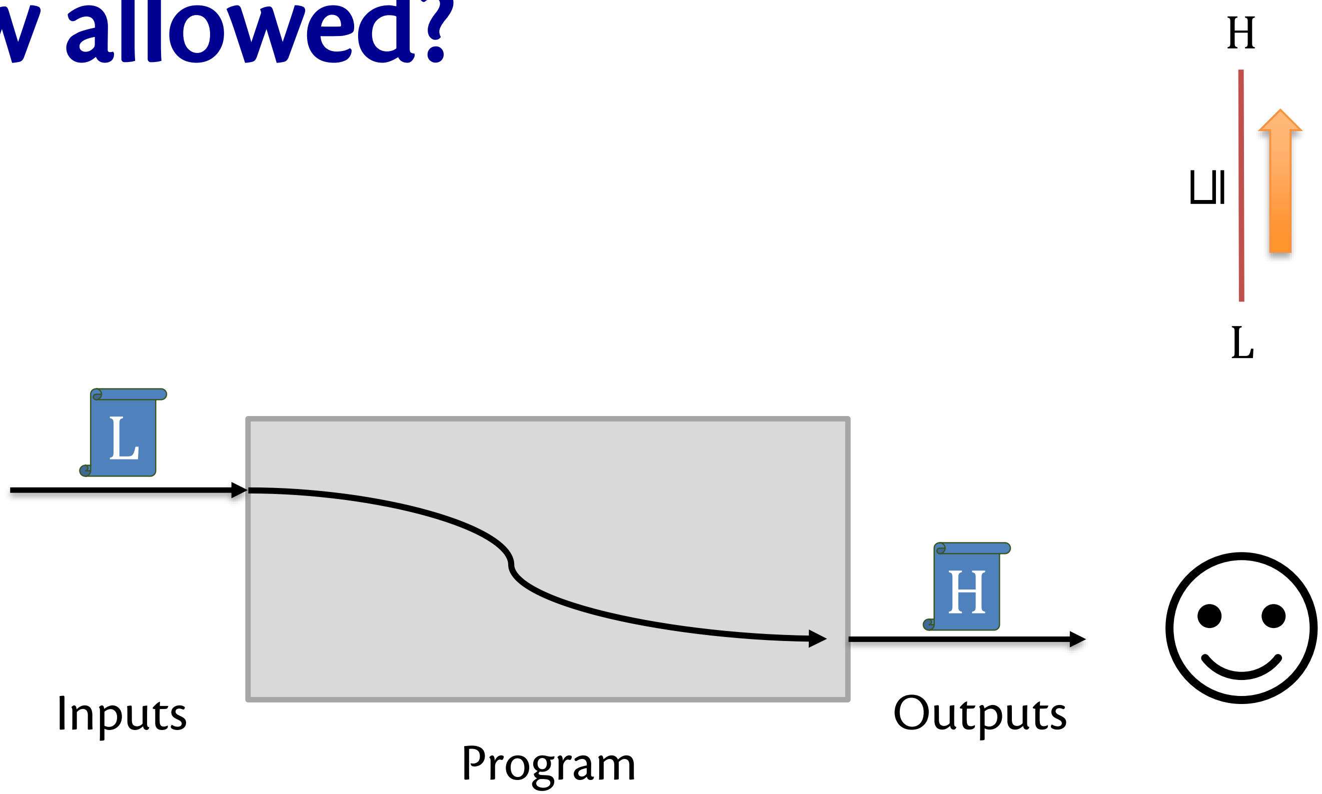




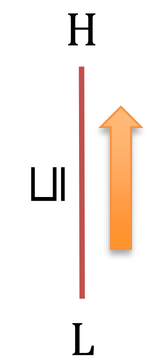
Is a flow allowed?

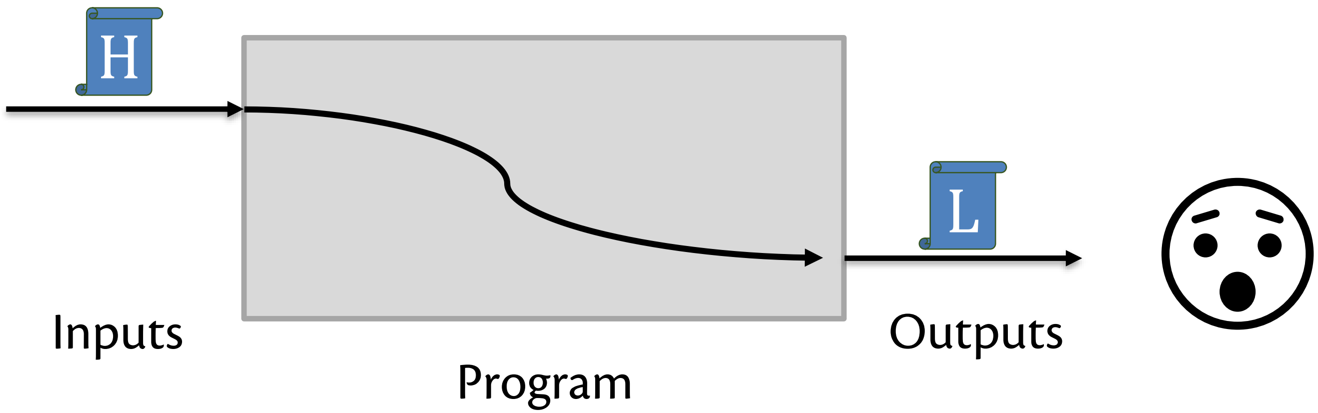


Is a flow allowed?

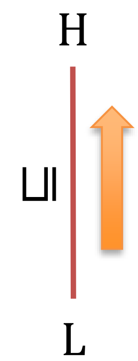


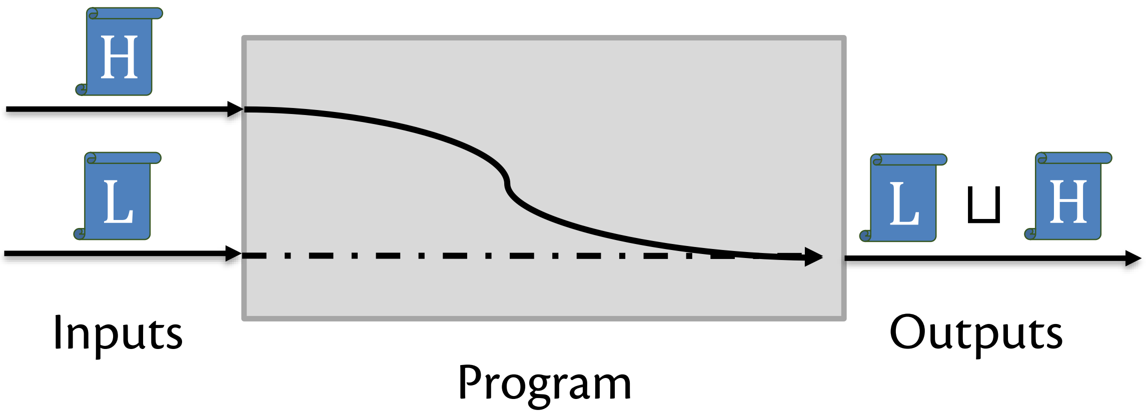
Is a flow allowed?





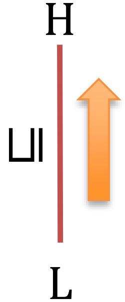
Is a flow allowed?



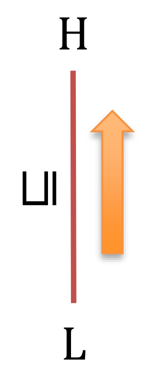


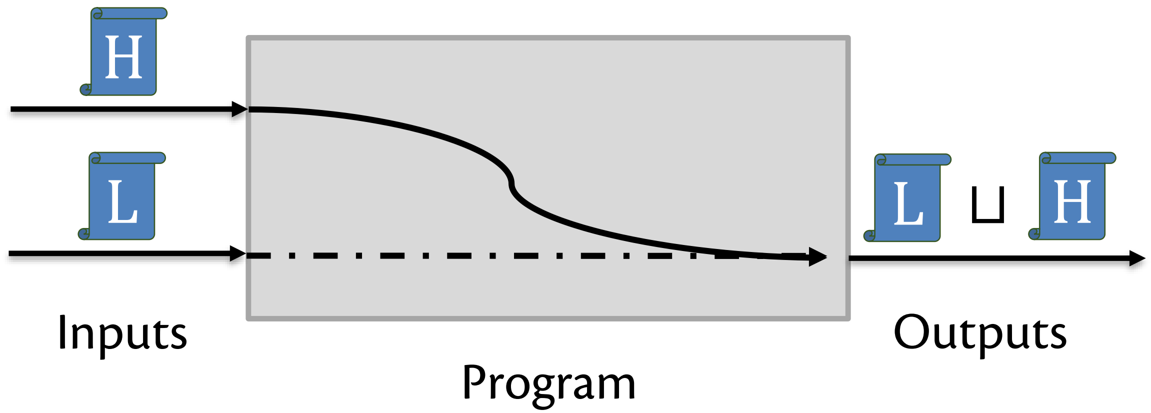
Operator ⊔ for combining labels

- For each ℓ and ℓ', there should exist label $\ell \sqcup \ell'$, such that:
  - $\ell \sqsubseteq \ell \sqcup \ell'$, $\ell' \sqsubseteq \ell \sqcup \ell'$, and
  - if $\ell \sqsubseteq \ell''$ and $\ell' \sqsubseteq \ell''$, then $\ell \sqcup \ell' \sqsubseteq \ell''$.
- $\ell \sqcup \ell'$ is called the **join** of ℓ and ℓ'.
- Examples: L ⊔ L= L, H ⊔ H= H, L ⊔ H= H

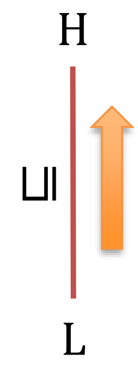


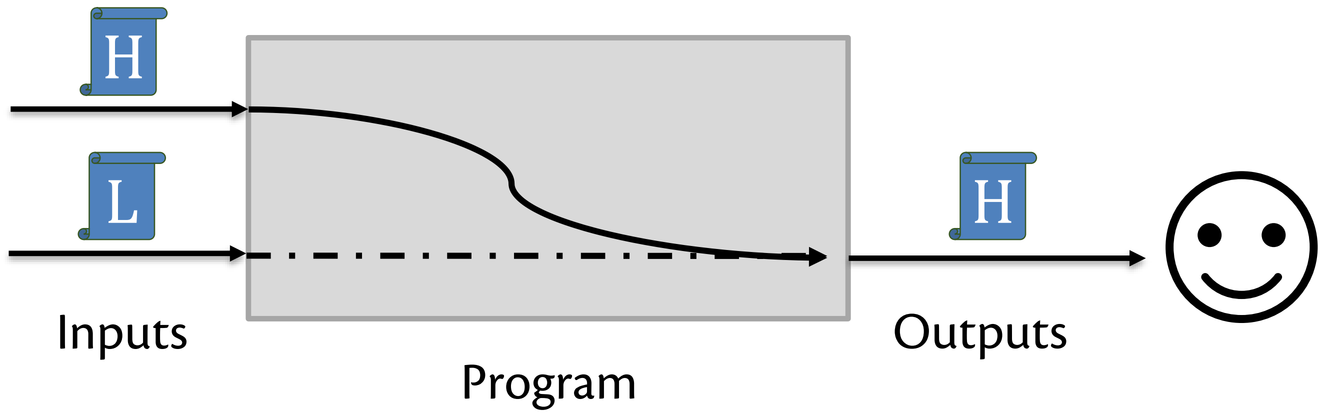
Is a flow allowed?





Is a flow allowed?





Is a flow allowed?

Given

- a lattice $\langle \{\mathrm{L}, \mathrm{H}\}, \sqsubseteq \rangle$ of labels,
- a program C, and
- labels on program inputs and outputs,

are all the flows from inputs to outputs that are caused by executing C allowed?

Noninterference (NI)

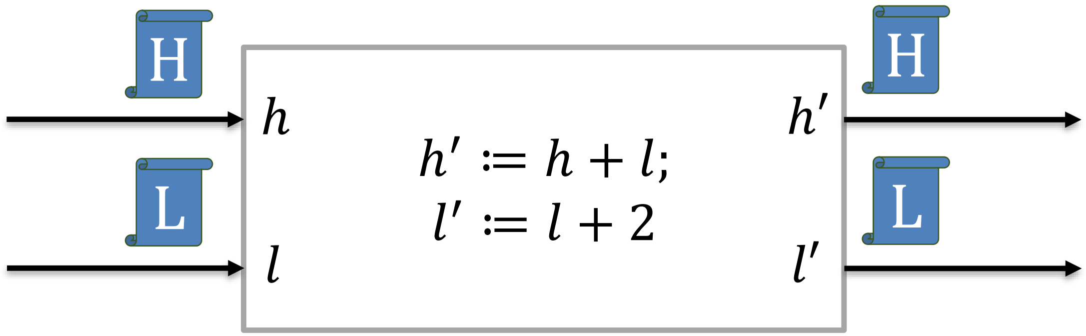
[Goguen and Meseguer 1982]

Noninterference for a program C:

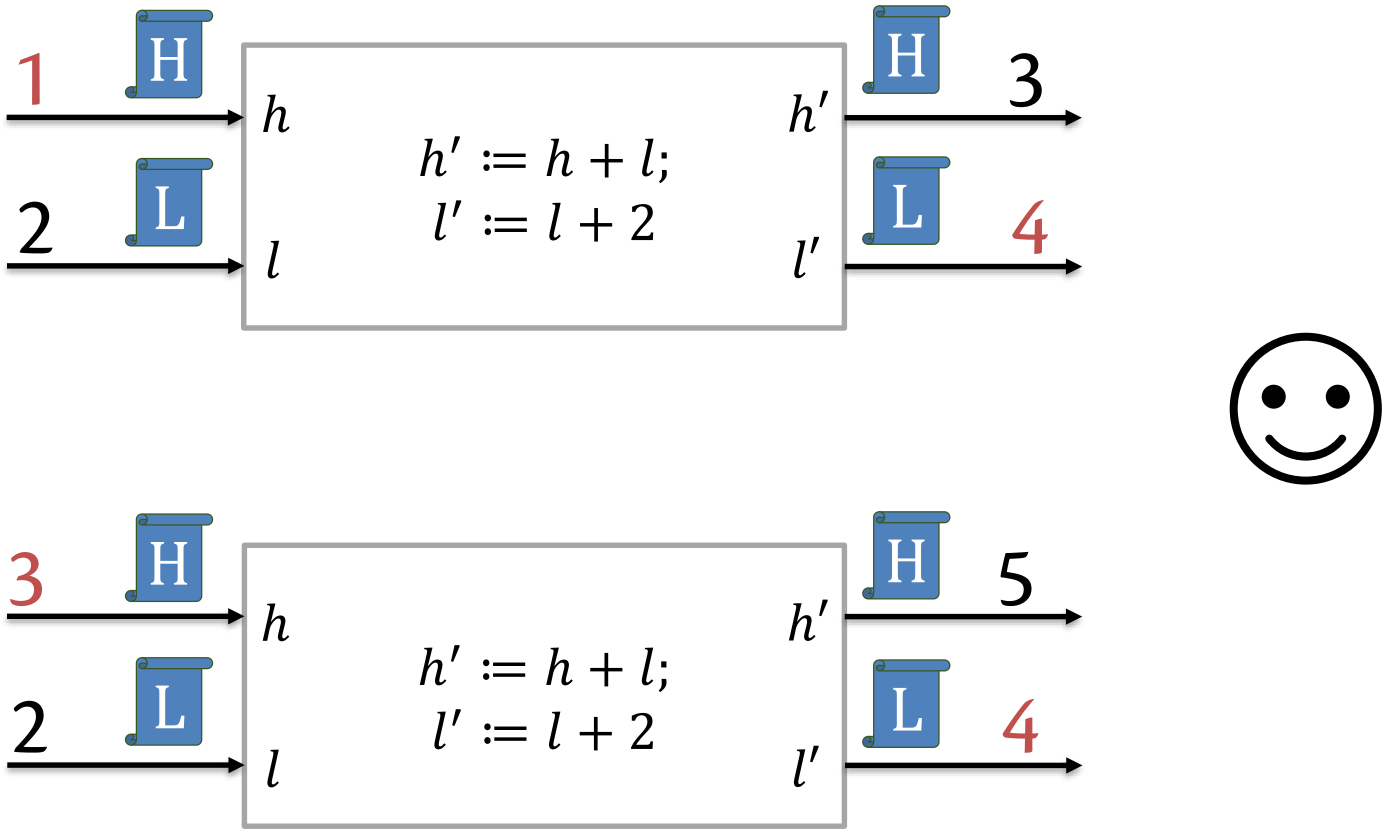
- Different H inputs, keeping L inputs fixed, should not cause different L outputs.

If a program C satisfies NI, then all flows from inputs to outputs are allowed.

Noninterference: Example

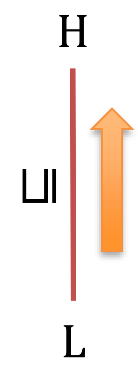


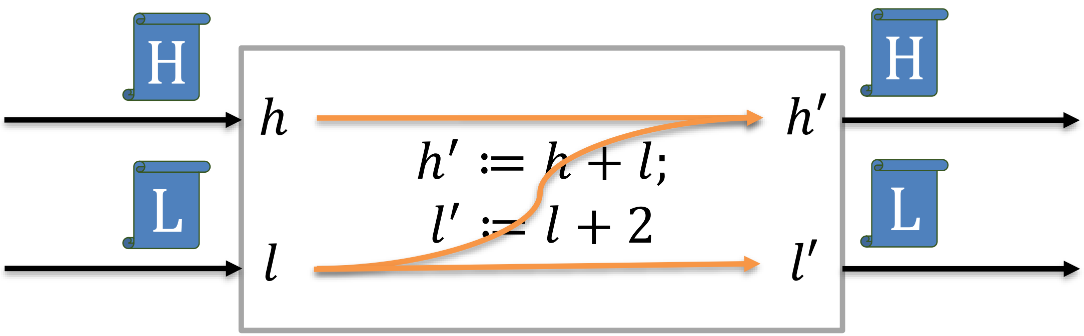
Noninterference: Example



The program satisfies noninterference!

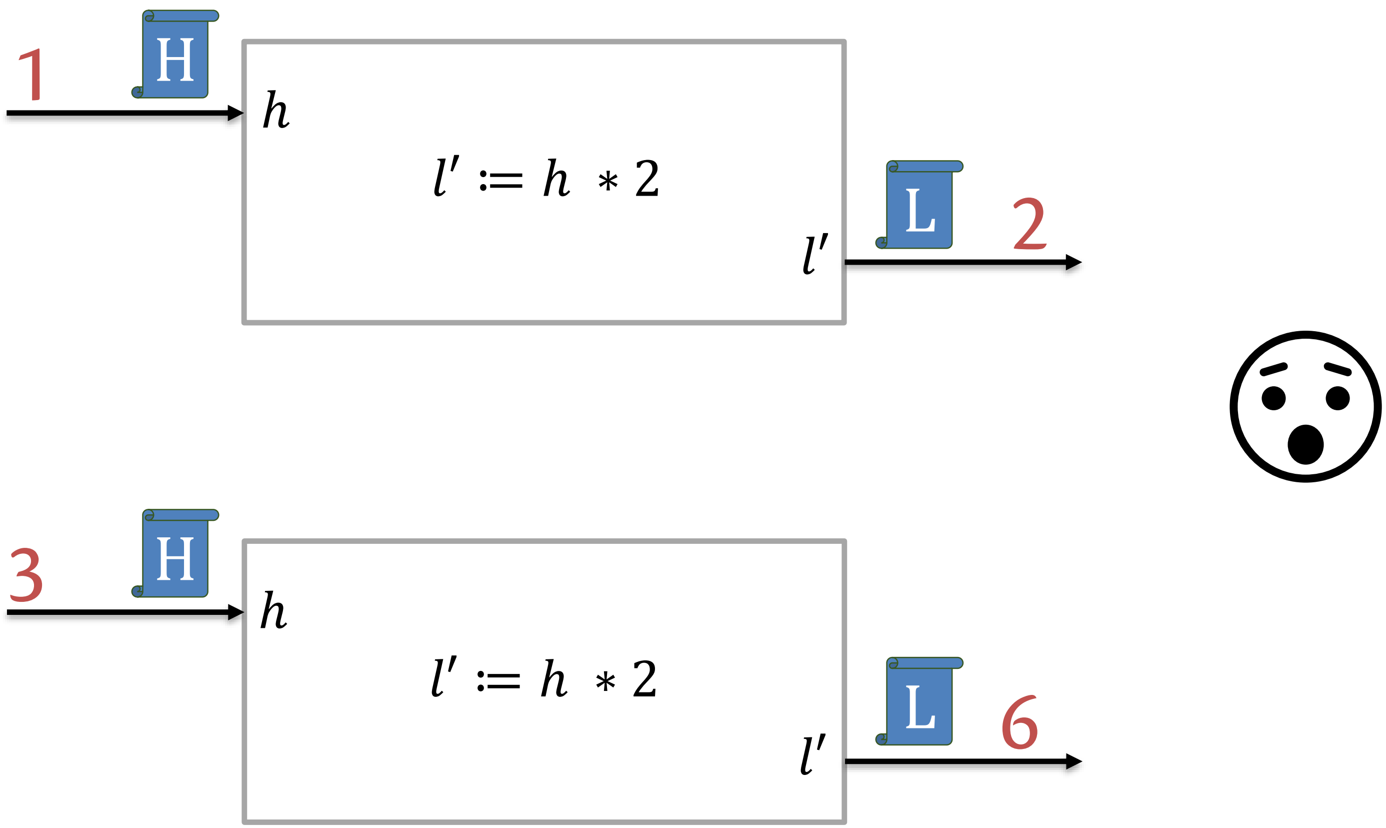
Noninterference: Example





The program causes only allowed flows!

Noninterference: Example



The program does not satisfy noninterference!

Noninterference (NI)

- Consider a program C.
- Variables in the program can model inputs and outputs.
- Consider two memories M_1 and M_2, such that
  - they agree on values of variables tagged with L:
  - $M_1 =_{\mathrm{L}} M_2$.

M_1 and M_2 may not agree on values of variables tagged with H.

Noninterference

- Consider a program C.
- Variables in the program can model inputs and outputs.
- Consider two memories M_1 and M_2, such that
  - they agree on values of variables tagged with L:
  - $M_1 =_{\mathrm{L}} M_2$.
- $C(M_i)$ are the observations produced by executing C to termination on initial memory M_i.
  - Observations are assignments to variables that are modeling outputs.
- For NI to hold, observations tagged with L should be the same, even if H inputs might differ:
  - $C(M_1) =_{\mathrm{L}} C(M_2)$.

Noninterference formalized

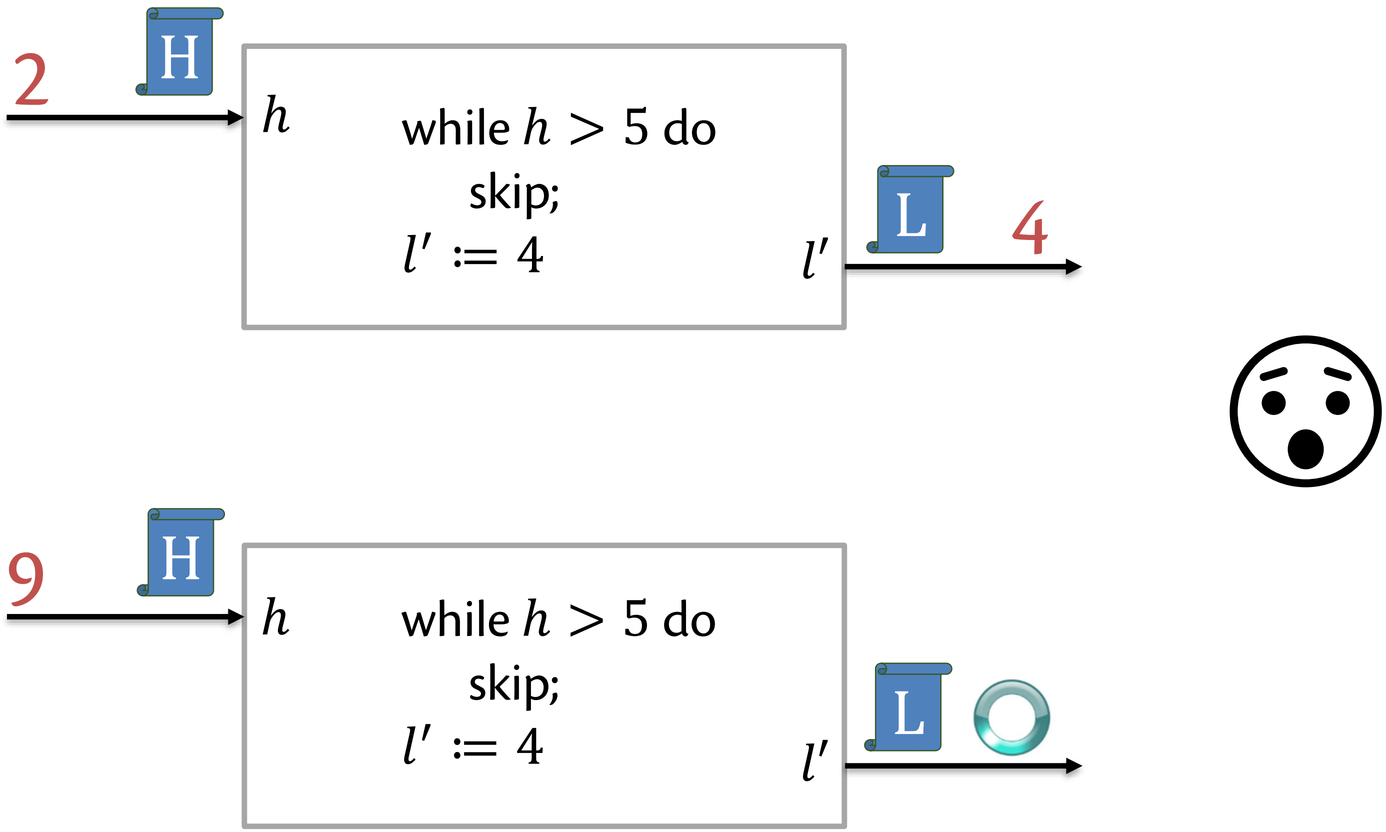
For a program C and a mapping from variables to labels in $\{\mathrm{L}, \mathrm{H}\}$:

$$\forall M_1, M_2\text{:}\ \ \text{if}\ \ M_1 =_{\mathrm{L}} M_2,\ \text{then}\ \ C(M_1) =_{\mathrm{L}} C(M_2).$$

Threat model

- Up until now an attacker could only observe outputs tagged with L.
- What if the attacker can also sense nontermination?

Termination sensitive noninterference



Termination sensitive noninterference

$\forall M_1, M_2$:

- If
  - $M_1 =_\mathrm{L} M_2$,
- then
  - $\boldsymbol{C}$ **terminates on** $\boldsymbol{M_1}$ **iff** $\boldsymbol{C}$ **terminates on** $\boldsymbol{M_2}$**, and**
  - $C(M_1) =_\mathrm{L} C(M_2)$.

Covert channels

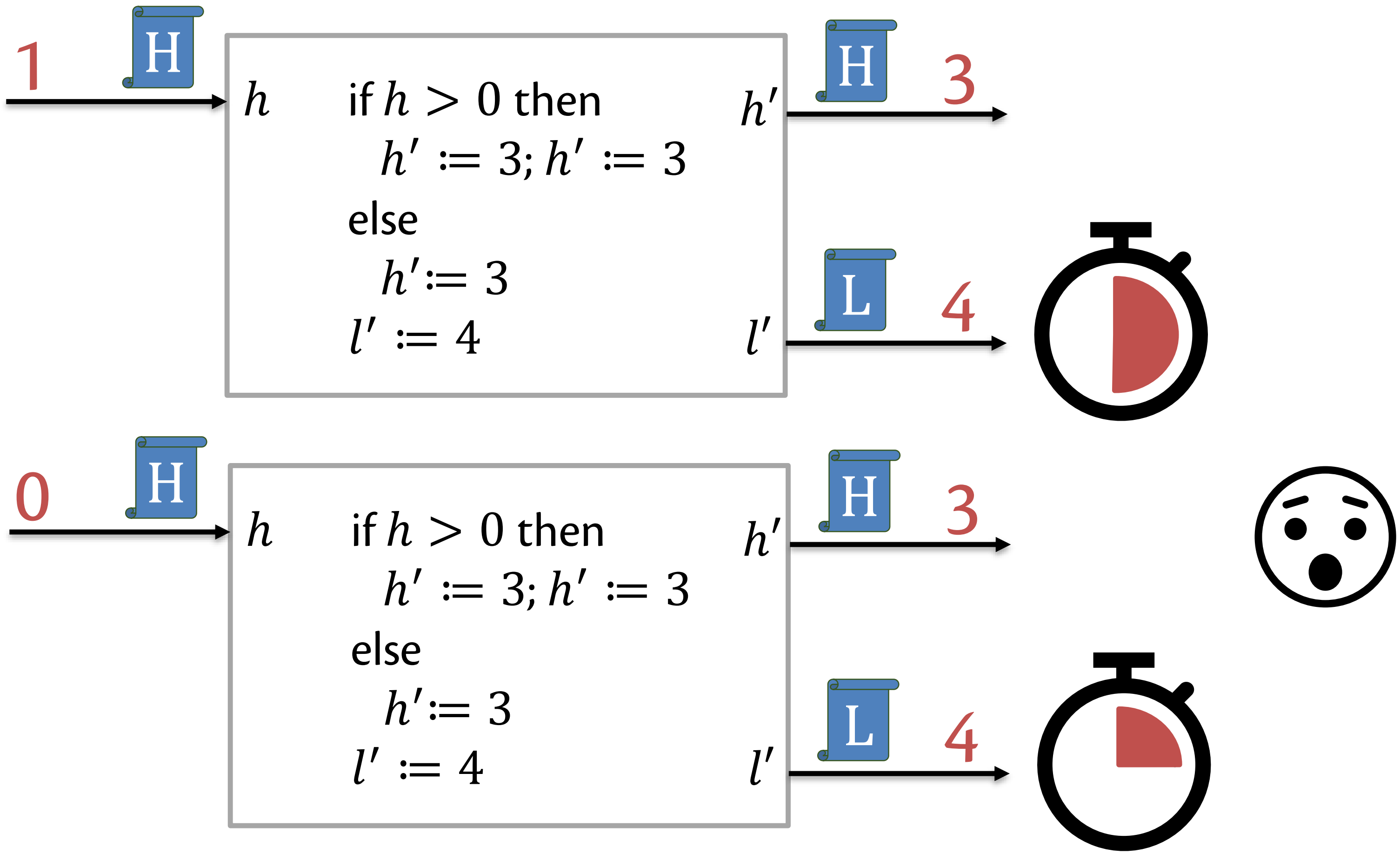
[Lampson 1973, Sabelfeld and Myers 2003]

- Termination channel is a *covert channel*:
  - not intended for information transfer, yet exploitable for that purpose.
- Other covert channels:
  - timing, heat emission, metadata.
- Information flow control can address covert channels:
  - treat covert channels as program outputs.
- Variations of noninterference can proscribe flows to covert channels.

Threat model

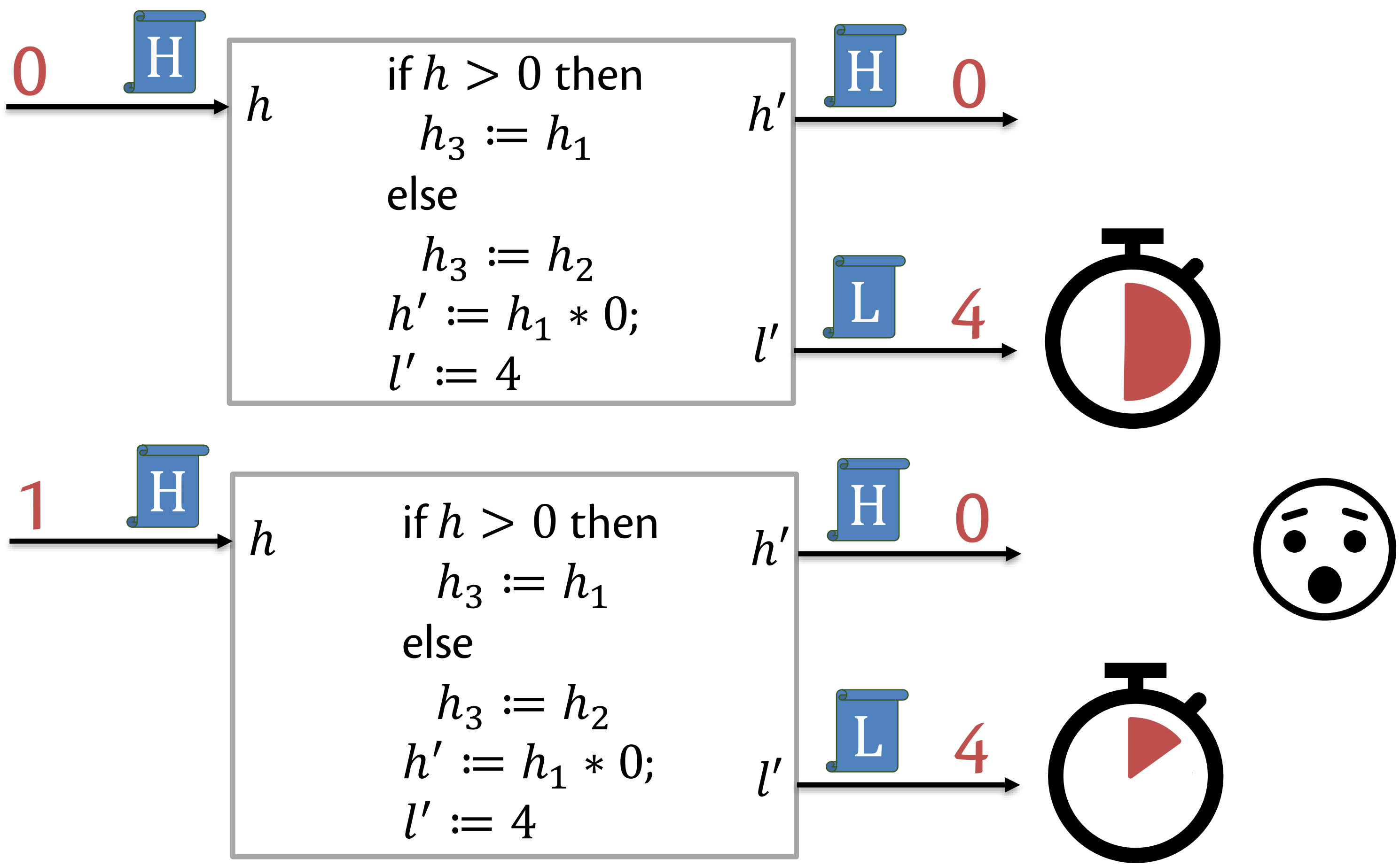
- What if the attacker can also measure execution time?

Timing channel



Timing channel: cache attack

Assume h_1, h_2, h_3 are high memory addresses that can be cached.



The stronger the threat model the more covert channels need to be considered, to prevent information leaking to attackers.

How can we ensure that a program causes only allowed flows and no leaks?

NEXT LECTURE