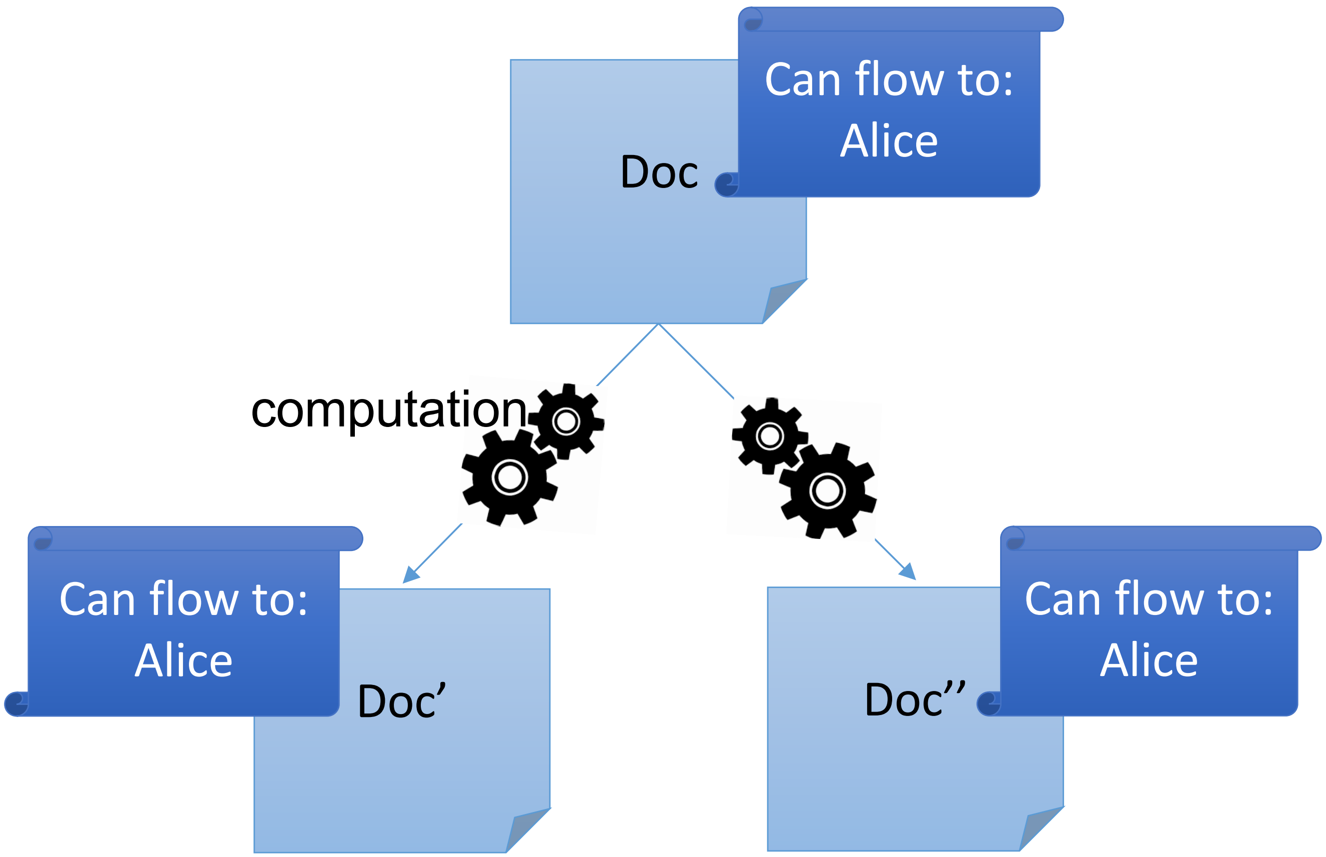
Lecture 20: Information Flow Control

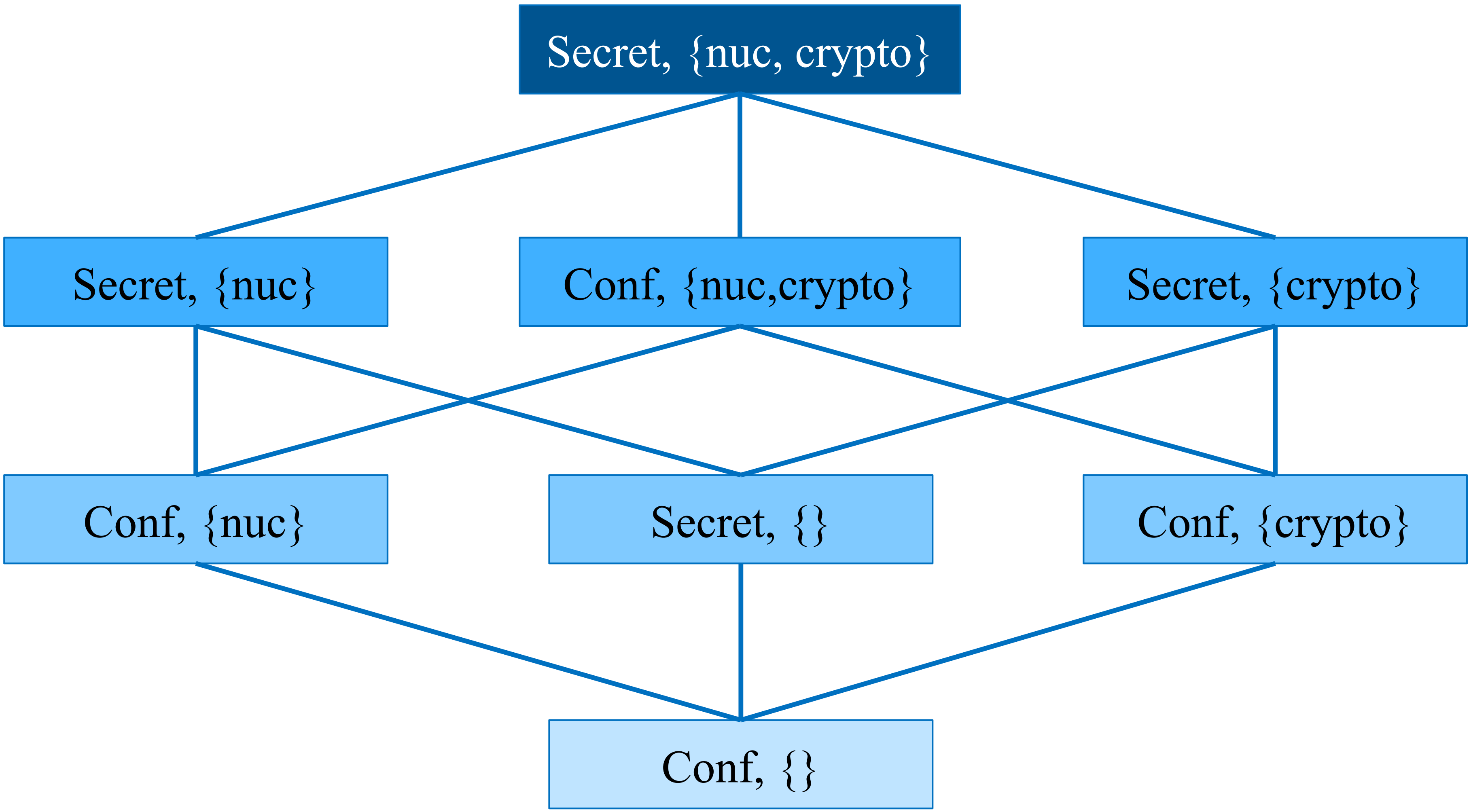
CS 5430 4/11/2018

Information flow policies

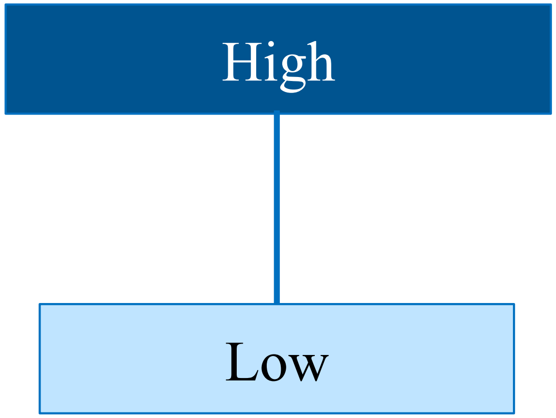


Automatic deduction of policies!

Labels represent policies



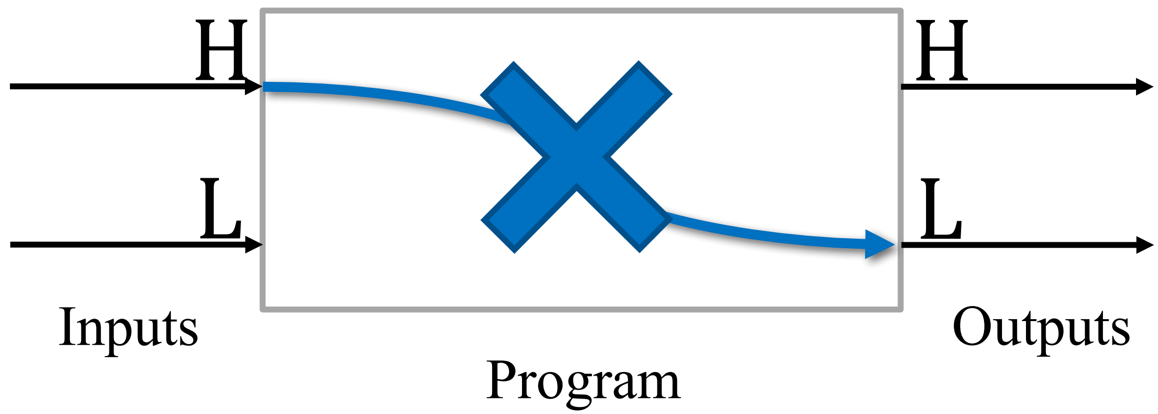
Labels represent policies



Noninterference
[Goguen and Meseguer 1982]

An interpretation of noninterference for a program:

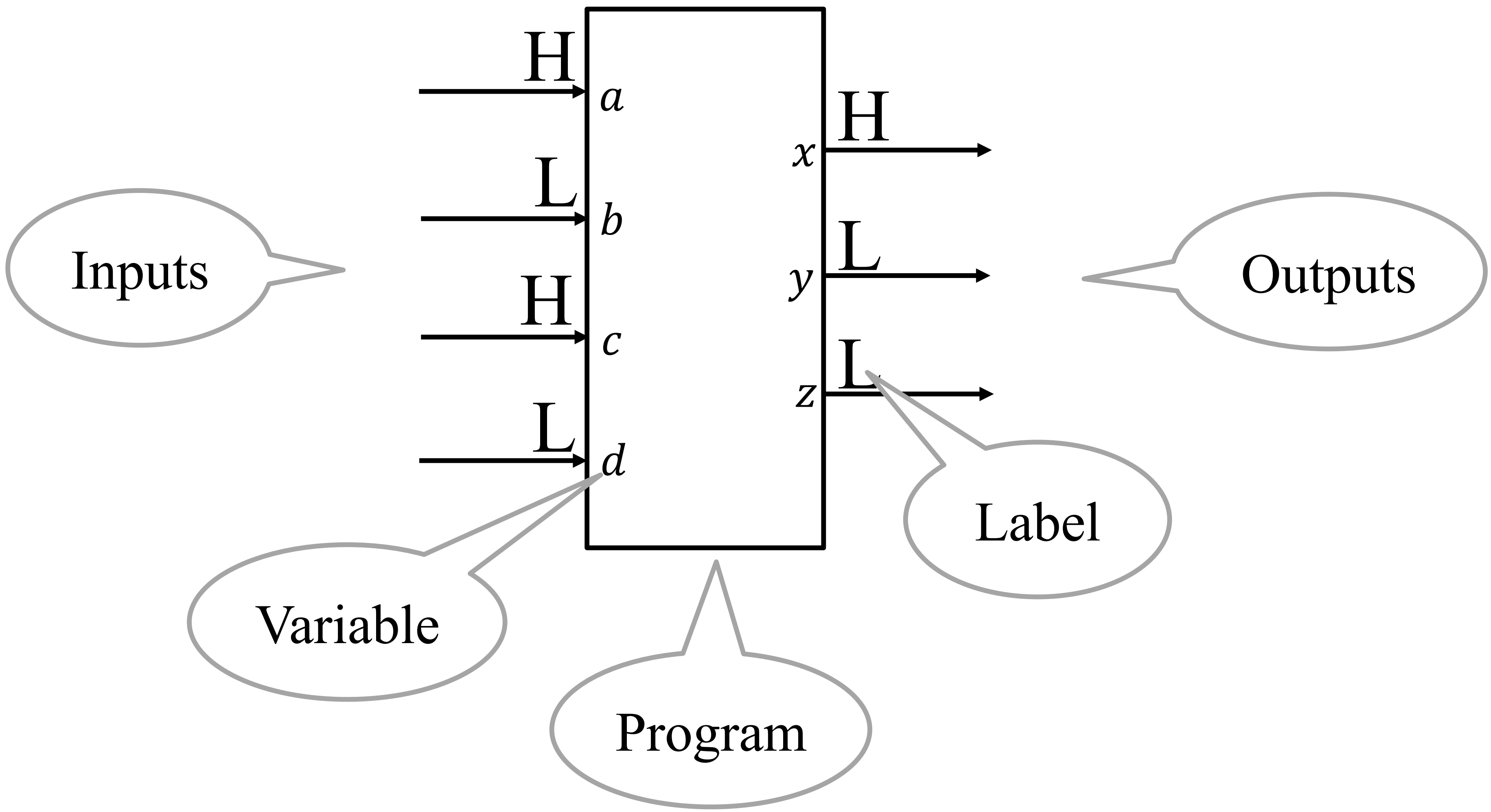
- Changes on H inputs should not cause changes on L outputs.



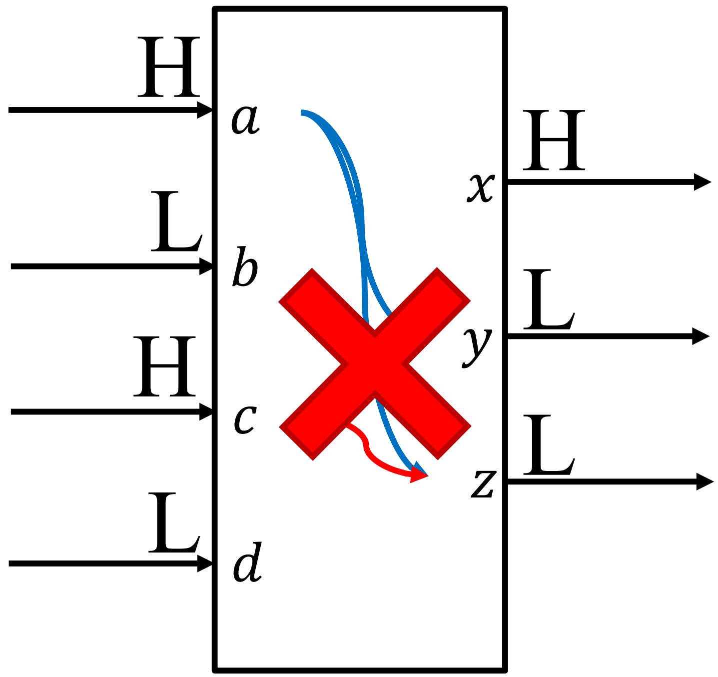
Today: Information Flow Control

- **Goal**: Enforce IF policies that tag variables in a program.
- There is a mapping Γ from variables to labels, which represent desired IF policies.
- The enforcement mechanism should ensure that a given program and a given Γ satisfy noninterference.

Information Flow Control

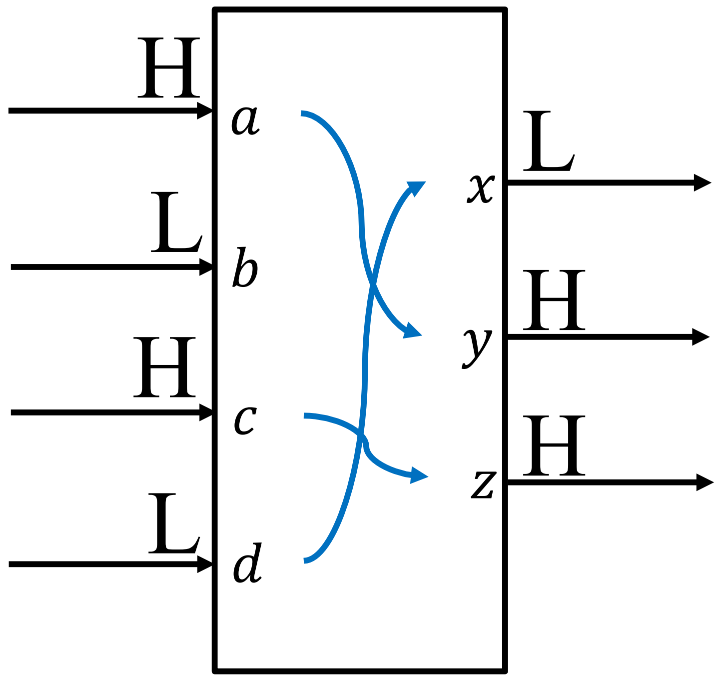


Information Flow Control: fixed Γ



- Γ remains the same during the analysis of the program.
- The mechanism checks that Γ satisfies noninterference.
- The program is rejected, if at least one red arrow appears in the program.

Information Flow Control: flow-sensitive Γ



- Γ may change during the analysis of the program.
- The mechanism deduces Γ(x), Γ(y), Γ(z) such that noninterference is satisfied.
- The program is never rejected.

Enforcing IF policies

- Static mechanism
  - Checking and/or deduction of labels before execution.
- Dynamic mechanism
  - Checking and/or deduction of labels during execution.
- Hybrid mechanism
  - Combination of static and dynamic.

STATIC TYPE CHECKING

fixed Γ

A simple programming language

```
e ::= x | n | e1+e2 | ...

c ::= x := e
    | if e then c1 else c2
    | while e do c
    | c1; c2
```

Checking an assignment

$$\texttt{x := y}$$

Examples for confidentiality

$\Gamma(\mathbf{x})$ is L.
$\Gamma(\mathbf{y})$ is L.
Does this assignment satisfy NI?

$\Gamma(\mathbf{x})$ is H.
$\Gamma(\mathbf{y})$ is L.
Does this assignment satisfy NI?

$\Gamma(\mathbf{x})$ is L.
$\Gamma(\mathbf{y})$ is H.
Does this assignment satisfy NI?

Checking an assignment

Assignments cause **explicit** information flows.

```
x := y
```

It satisfies NI, if $\Gamma(\mathtt{y}) \sqsubseteq \Gamma(\mathtt{x})$.

Checking an assignment

```
x := y
```

It satisfies NI, if $\Gamma(\mathtt{y}) \sqsubseteq \Gamma(\mathtt{x})$.

MLS for confidentiality

"no read up":

S may read O iff Label(O) $\sqsubseteq$ Label (S)

"no write down":

S may write O' iff Label(S) $\sqsubseteq$ Label (O')

Checking an assignment

$$\mathbf{x} := \mathbf{y}$$

It satisfies NI, if $\Gamma(\mathbf{y}) \sqsubseteq \Gamma(\mathbf{x})$.

<u>MLS for confidentiality</u>

"no read up":

C may read $\mathbf{y}$ iff $\text{Label}(\mathbf{y}) \sqsubseteq \text{Label}(C)$

"no write down":

C may write $\mathbf{x}$ iff $\text{Label}(C) \sqsubseteq \text{Label}(\mathbf{x})$

Checking an assignment

```
x := y + z
```

It satisfies NI, if $\Gamma(\mathbf{y}) \sqsubseteq \Gamma(\mathbf{x})$ and $\Gamma(\mathbf{z}) \sqsubseteq \Gamma(\mathbf{x})$.

It satisfies NI, if $\Gamma(\mathbf{y+z}) \sqsubseteq \Gamma(\mathbf{x})$.

???

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'.
- Operator $\sqcup$ is associative and commutative.

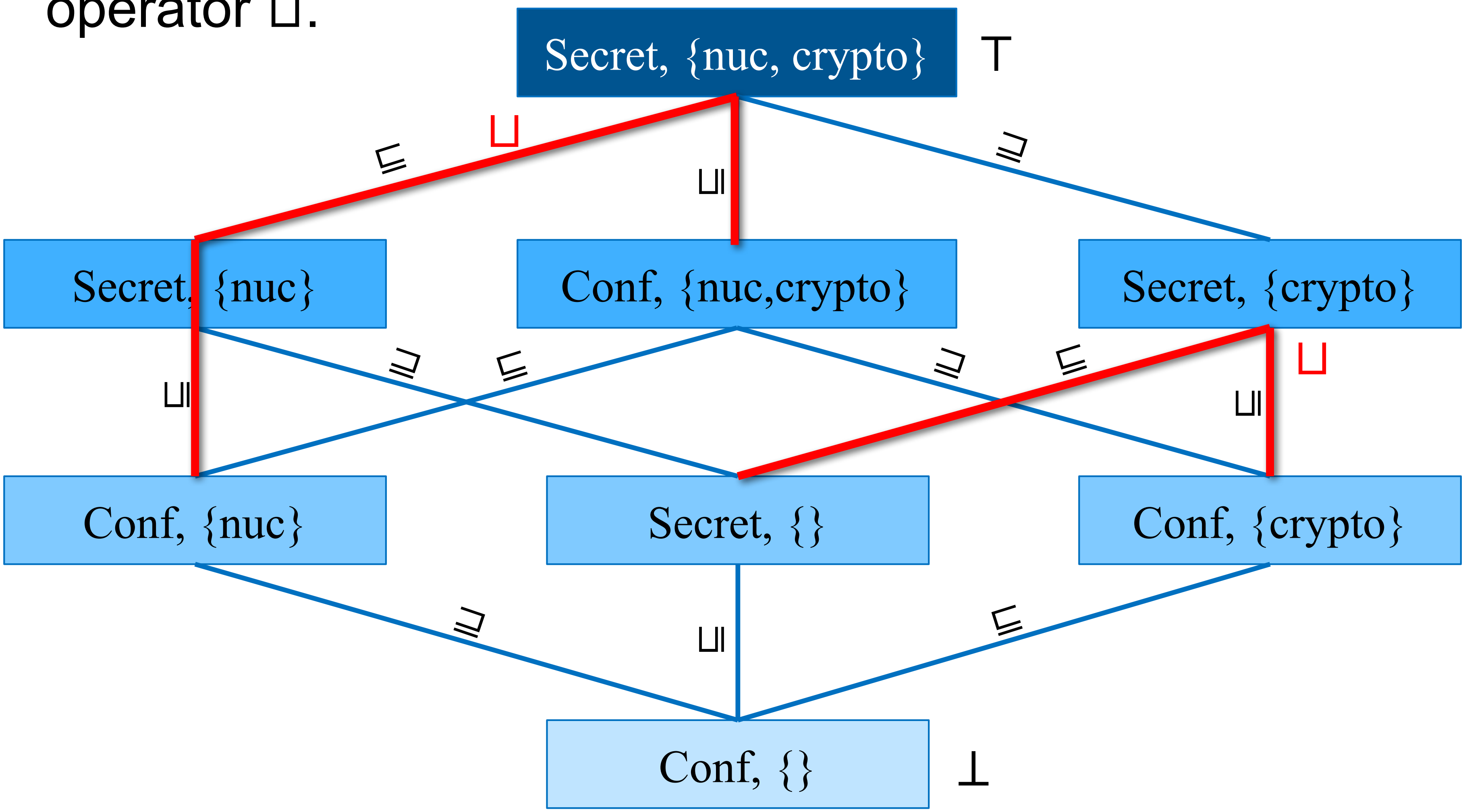
Checking an assignment

```
x := y + z
```

It satisfies NI, if $\Gamma(\mathtt{y}) \sqcup \Gamma(\mathtt{z}) \sqsubseteq \Gamma(\mathtt{x})$.

Lattice of labels

- The set of labels and relation ⊑ define a lattice, with join operator ⊔.



Checking an if-statement

```
if z > 0 then
    x:= 1
else
    x:= 0
```

<u>Examples for confidentiality</u>

$\Gamma(\mathbf{x})$ is L.
$\Gamma(\mathbf{z})$ is L.
Does this if-statement satisfy NI?

$\Gamma(\mathbf{x})$ is H.
$\Gamma(\mathbf{z})$ is L.
Does this if-statement satisfy NI?

$\Gamma(\mathbf{x})$ is L.
$\Gamma(\mathbf{z})$ is H.
Does this if-statement satisfy NI?

Checking an if-statement

```
if z > 0 then
    x:= 1
else
    x:= 0
```

Conditional commands (e.g., if-statements and while-statements) cause **implicit** information flows.

Context

```
if z > 0 then
     x:= 1
else
     x:= 0
```

They reveal information about **z>0**.

Introduce a context label ctx

Its ctx is $\Gamma(\mathbf{z})$.

Context

```
if z > 0 then
    x:= 1
else
    x:= 0
```

Check if
$ctx \sqcup \Gamma(\mathbf{e}) \sqsubseteq \Gamma(\mathbf{x})$.

Implicit flow

Explicit flow

Introduce a context label ctx

Its ctx is $\Gamma(\mathbf{z})$.

Typing system for IF control

- Static
- Fixed Γ
- Labels as types
  - Label $\Gamma(\mathbf{x})$ is the type of $\mathbf{x}$.
- Typing rules for all possible commands.
- **Goal**: type-correctness $\Rightarrow$ noninterference

We are already familiar with typing systems!

Example of typing rule from Java or OCaml:

```
x + y : int
  if x : int
  and y : int
```

Typing rules for expressions

Judgement $\Gamma \vdash \mathbf{e} : \ell$

According to mapping Γ, expression $\mathbf{e}$ has type (i.e., label) ℓ.

Constant: $\Gamma \vdash \mathbf{n} : \bot$

Variable: $\Gamma \vdash \mathbf{x} : \Gamma(\mathbf{x})$

Expression: $\Gamma \vdash \mathbf{e+e'} : \ell \sqcup \ell'$
if $\Gamma \vdash \mathbf{e} : \ell$
and $\Gamma \vdash \mathbf{e'} : \ell'$

Typing rules for expressions

Expression: $\Gamma \vdash \texttt{e+e}' : \ell \sqcup \ell'$
$\quad$ **if** $\Gamma \vdash \texttt{e} : \ell$
$\quad$ **and** $\Gamma \vdash \texttt{e}' : \ell'$

Inference rule:

Premises $\longrightarrow$ $\Gamma \vdash \texttt{e} : \ell \qquad \Gamma \vdash \texttt{e}' : \ell'$

Conclusion $\longrightarrow$ $\Gamma \vdash \texttt{e+e}' : \ell \sqcup \ell'$

$$\frac{\Gamma \vdash \texttt{e} : \ell \qquad \Gamma \vdash \texttt{e}' : \ell'}{\Gamma \vdash \texttt{e+e}' : \ell \sqcup \ell'}$$

Example

- Let $\Gamma(\mathbf{x}) = \mathrm{L}$ and $\Gamma(\mathbf{y}) = \mathrm{H}$.
- What is the type of **x+y+1**?
- *Proof tree:*

$$\dfrac{\dfrac{\Gamma(\mathbf{x}) = \mathrm{L}}{\Gamma \vdash \mathbf{x} : \mathrm{L}} \qquad \dfrac{\Gamma(\mathbf{y}) = \mathrm{H}}{\Gamma \vdash \mathbf{y} : \mathrm{H}} \qquad \dfrac{}{\Gamma \vdash \mathbf{1} : \mathrm{L}}}{\Gamma \vdash \mathbf{x} + \mathbf{y} + \mathbf{1} : \mathrm{H}}$$

Typing rules for commands

Judgement $\Gamma, ctx \vdash$ **`c`**

According to mapping Γ, and context label ctx, command **`c`** is type correct.

Assignment rule

$\Gamma, ctx \vdash \texttt{x:=e}$

  `if` $\Gamma \vdash \texttt{e} : \ell$

  `and` $\ell \sqcup ctx \sqsubseteq \Gamma(\texttt{x})$

$$\frac{\Gamma \vdash \texttt{e} : \ell \qquad \ell \sqcup ctx \sqsubseteq \Gamma(\texttt{x})}{\Gamma, ctx \vdash \texttt{x:=e}}$$

If-rule

$$\frac{\Gamma \vdash \mathbf{e} : \ell \qquad \Gamma, \ell \sqcup ctx \vdash \mathbf{c1} \qquad \Gamma, \ell \sqcup ctx \vdash \mathbf{c2}}{\Gamma, ctx \vdash \mathtt{\mathbf{if\ e\ then\ c1\ else\ c2}}}$$

If-rule (example)

$$
\dfrac{\Gamma \vdash \texttt{z>0} : \Gamma(\texttt{z}) \qquad \dfrac{\Gamma \vdash \texttt{1} : \bot \quad \bot \sqcup \Gamma(\texttt{z}) \sqcup L \sqsubseteq \Gamma(\texttt{x})}{\Gamma, \Gamma(\texttt{z}) \sqcup L \vdash \texttt{x:=1}} \qquad \dfrac{\begin{array}{c}\Gamma \vdash \texttt{0} : \bot, \\ \bot \sqcup \Gamma(\texttt{z}) \sqcup L \sqsubseteq \Gamma(\texttt{x})\end{array}}{\Gamma, \Gamma(\texttt{z}) \sqcup L \vdash \texttt{x:=0}}}{\Gamma, L \vdash \texttt{if z>0 then x:=1 else x:=0}}
$$

Static type system

Assignment-Rule:

$$\frac{\Gamma \vdash \mathbf{e} : \ell \qquad \ell \sqcup ctx \sqsubseteq \Gamma(\mathbf{x})}{\Gamma, ctx \vdash \mathbf{x:=e}}$$

If-Rule:

$$\frac{\Gamma \vdash \mathbf{e} : \ell \qquad \Gamma, \ell \sqcup ctx \vdash \mathbf{c1} \qquad \Gamma, \ell \sqcup ctx \vdash \mathbf{c2}}{\Gamma, ctx \vdash \mathbf{if\ e\ then\ c1\ else\ c2}}$$

While-Rule:

$$\frac{\Gamma \vdash \mathbf{e} : \ell \qquad \Gamma, \ell \sqcup ctx \vdash \mathbf{c}}{\Gamma, ctx \vdash \mathbf{while\ e\ do\ c}}$$

Sequence-Rule:

$$\frac{\Gamma, ctx \vdash \mathbf{c1} \qquad \Gamma, ctx \vdash \mathbf{c2}}{\Gamma, ctx \vdash \mathbf{c1;c2}}$$

Soundness of type system

$$\Gamma, ctx \vdash \mathbf{c} \quad \Rightarrow \quad \mathbf{c} \text{ satisfies NI}$$

Limitations of the type system

This type system does not prevent leaks through covert channels.

Example of covert channel:

```
while s != 0 do { //nothing };
p:=1
```

where **`s`** is a secret variable (i.e., $\Gamma($**`s`**$)=H$) and **`p`** is a public variable (i.e., $\Gamma($**`p`**$)=L$).

A solution

- To prevent covert channels due to infinite loops,
- strengthen the typing rule for while-statement, to allow only low guard expression:

$$\frac{\Gamma \vdash \mathbf{e} : \bot \qquad \Gamma, ctx \vdash \mathbf{c}}{\Gamma, ctx \vdash \mathbf{while\ e\ do\ c}}$$

- Now, type correctness implies termination sensitive NI.
- But, the enforcement mechanism becomes overly conservative.
- Another solution? Research!

This type system is not complete.

- **`c`** satisfies noninterference $\not\Rightarrow$ $\Gamma\,,\,ctx \vdash$ **`c`**
  - There is a command **`c`**, such that noninterference is satisfied, but **`c`** is not type correct.
- Example 1:
  - $\Gamma(\mathbf{x}) = \mathrm{H}, \Gamma(\mathbf{y}) = \mathrm{L}$
  - **`c`** is **`if x>0 then y:=1 else y:=1`**
  - **`c`** satisfies noninterference, because **`x`** does not leak to **`y`**.
  - **`c`** is not type correct, because $\Gamma(\mathrm{x}) \not\sqsubseteq \Gamma(\mathrm{y})$.

This type system is not complete.

- Example 2:
  - $\Gamma(x) = H$, $\Gamma(y) = L$
  - **`c`** is **`if 1=1 then y:=1 else y:=x`**
  - **`c`** satisfies noninterference, because **`x`** does not leak to **`y`**.
  - **`c`** is not type correct, because $\Gamma(x) \not\sqsubseteq \Gamma(y)$.
- So, this type system is *conservative*. It has *false negatives:*
  - There are programs that are not type correct, but that satisfy noninterference.

Can we build a complete mechanism?

- Is there an enforcement mechanism for information flow control that has no false negatives?
  - A mechanism that rejects only programs that do not satisfy noninterference?
- No! [Sabelfeld and Myers, 2003]
  - “The general problem of confidentiality for programs is undecidable.”
  - The halting problem can be reduced to the information flow control problem.
  - Example:

```
if h>1 then c; l:=2 else skip
```

    - If we could precisely decide whether this program is secure, we could decide whether `c` terminates!

Can we build a mechanism with fewer false positives?

Switch from static to dynamic mechanisms!