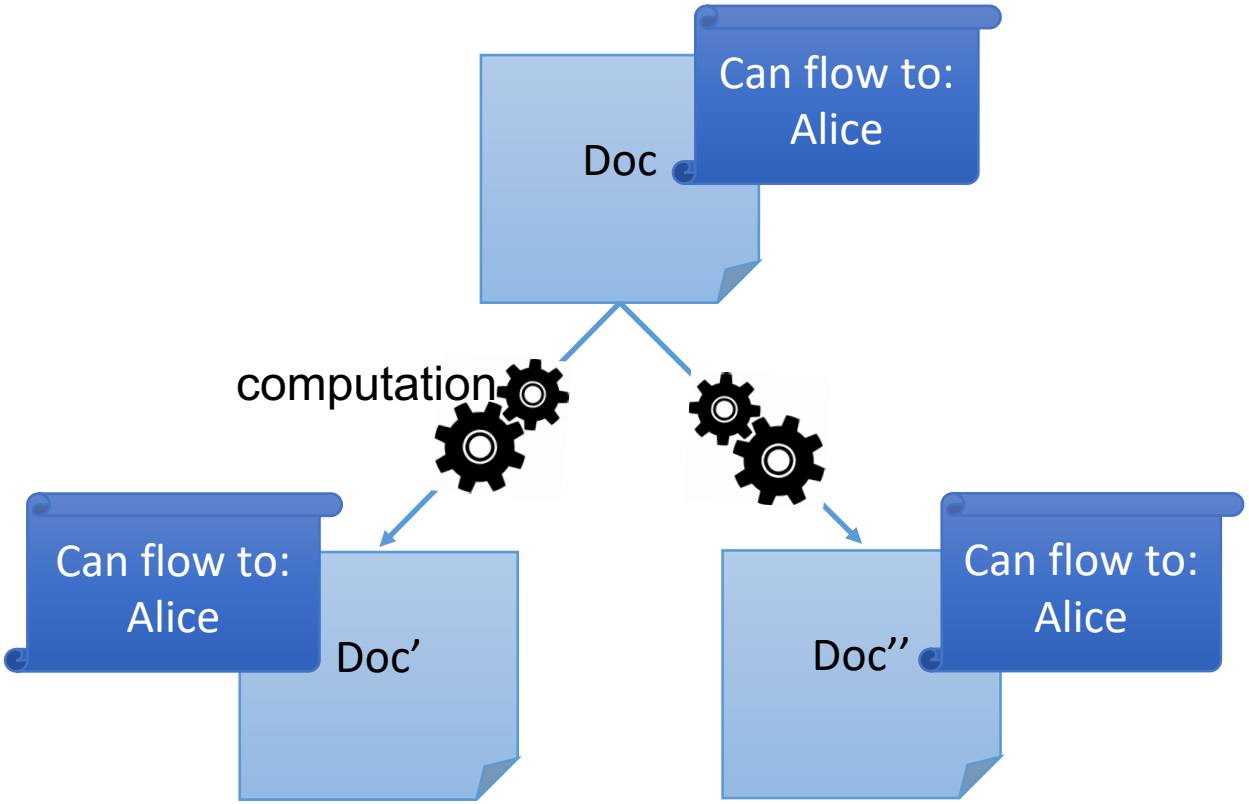
Lecture 18: Information Flow Control

CS 181S

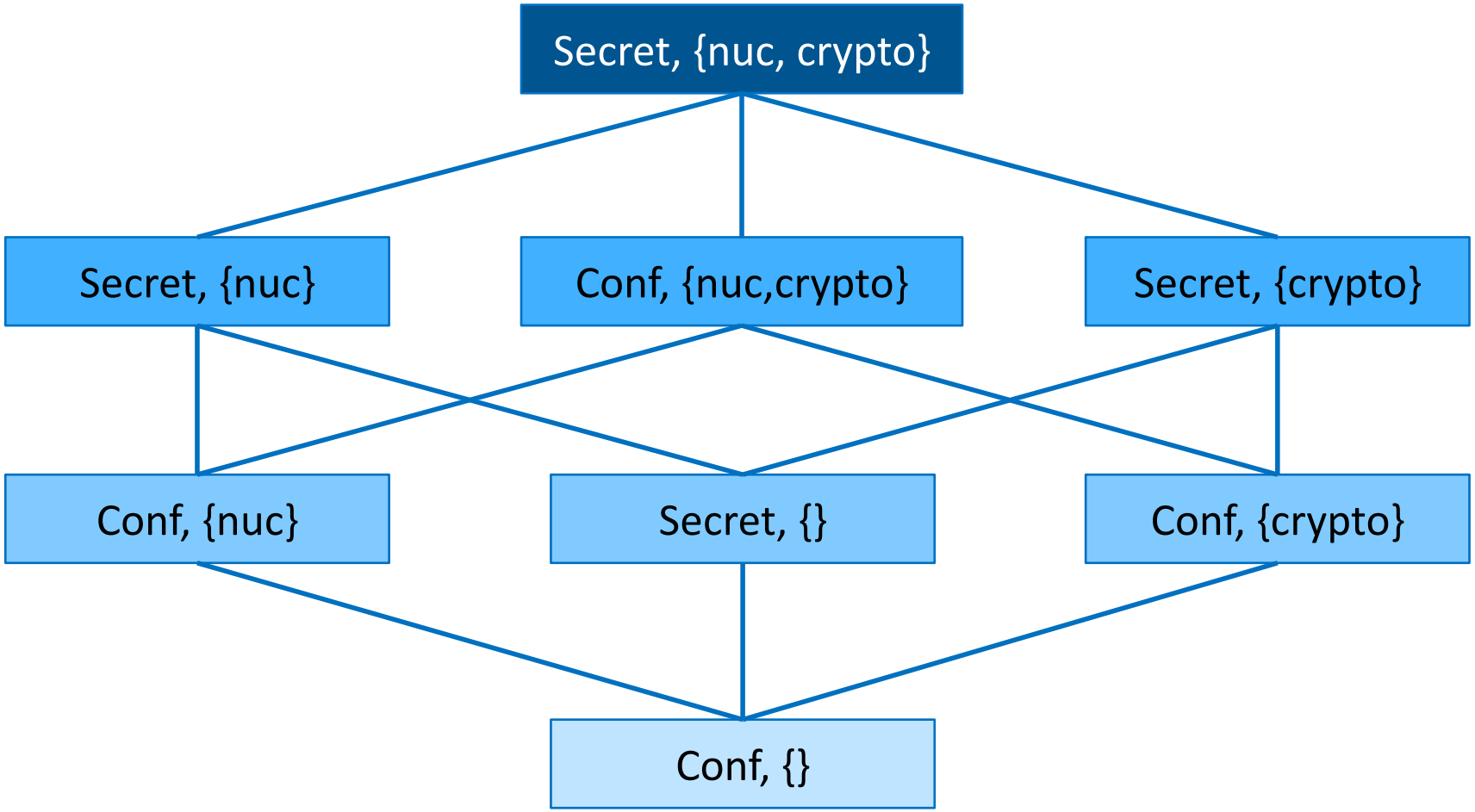
November 14, 2018

Information flow policies

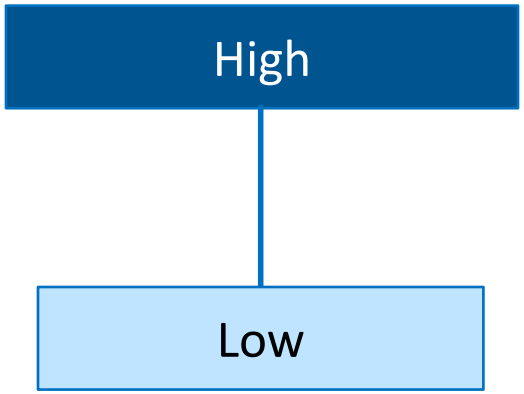


Automatic deduction of policies!

Labels represent policies



Labels represent policies

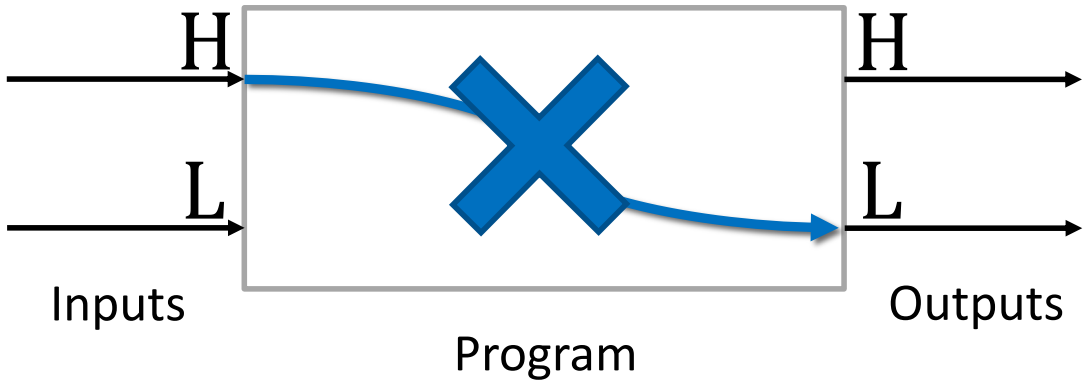


Noninterference

[Goguen and Meseguer 1982]

An interpretation of noninterference for a program:

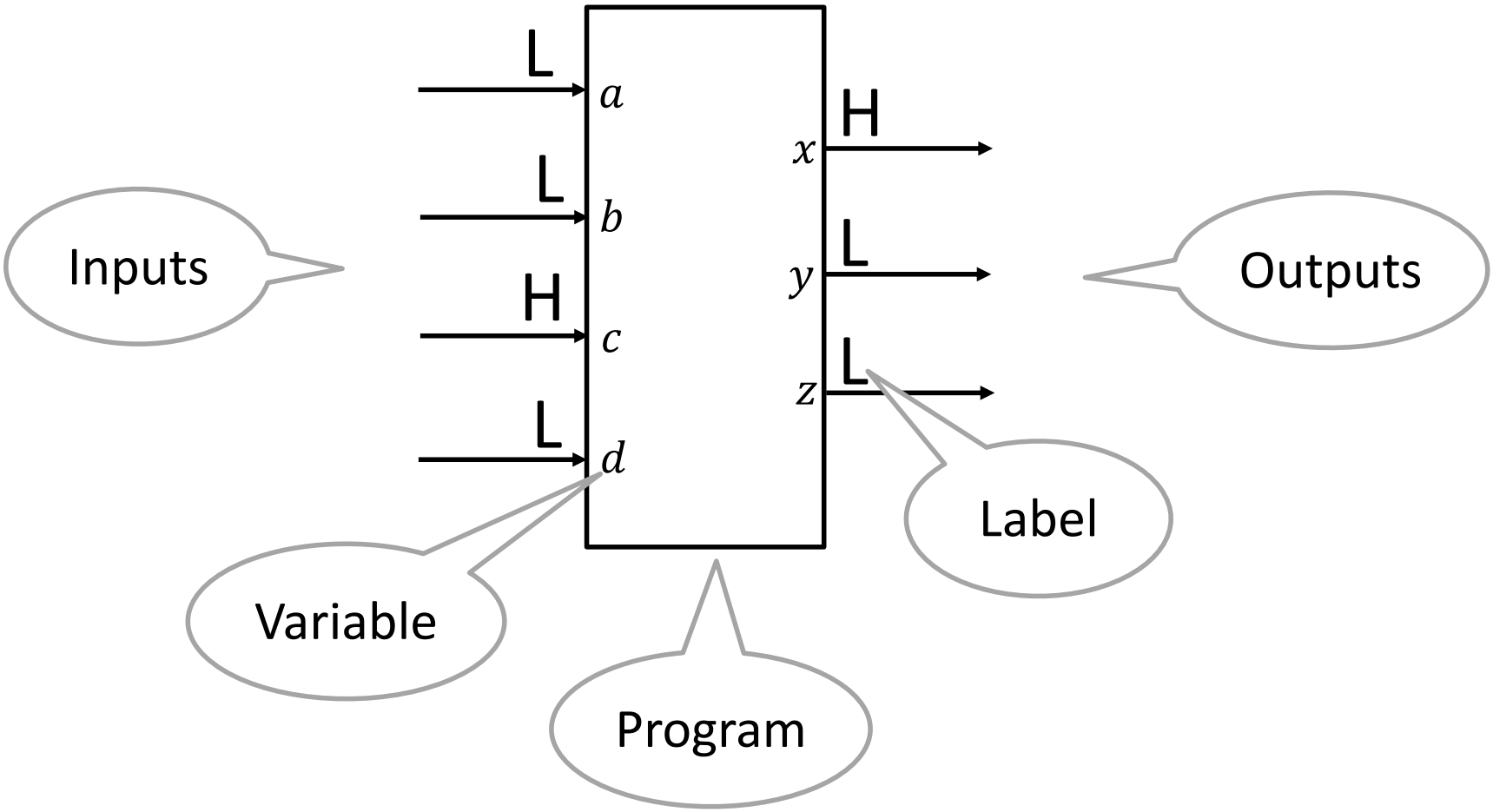
- Changes on H inputs should not cause changes on L outputs.



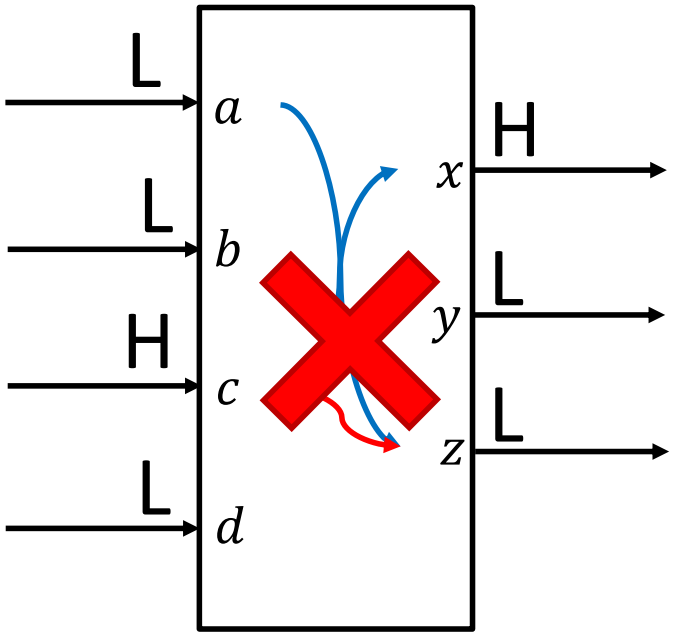
Today: Information Flow Control

- **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

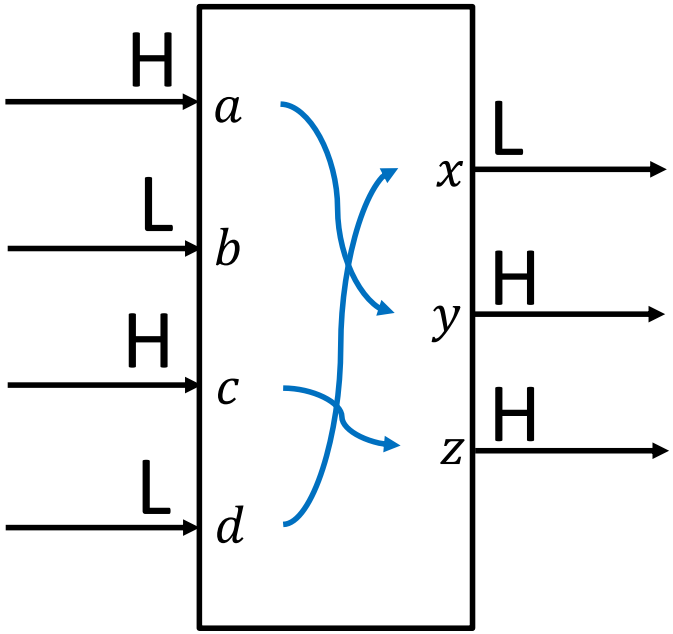


Information Flow Control: fixed Γ



- 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

```
x := y
```

Examples for confidentiality

Γ(**x**) is L.
Γ(**y**) is L.
Does this assignment satisfy NI?

Γ(**x**) is H.
Γ(**y**) is L.
Does this assignment satisfy NI?

Γ(**x**) is L.
Γ(**y**) is H.
Does this assignment satisfy NI?

Checking an assignment

Assignments cause **explicit** information flows.

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

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

Checking an assignment

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

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

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

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

MLS for confidentiality

"no read up":

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

"no write down":

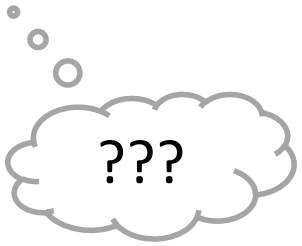
C may write $\mathtt{x}$ iff Label(C) $\sqsubseteq$ Label ($\mathtt{x}$)

Checking an assignment

$$\mathtt{x := y + z}$$

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

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

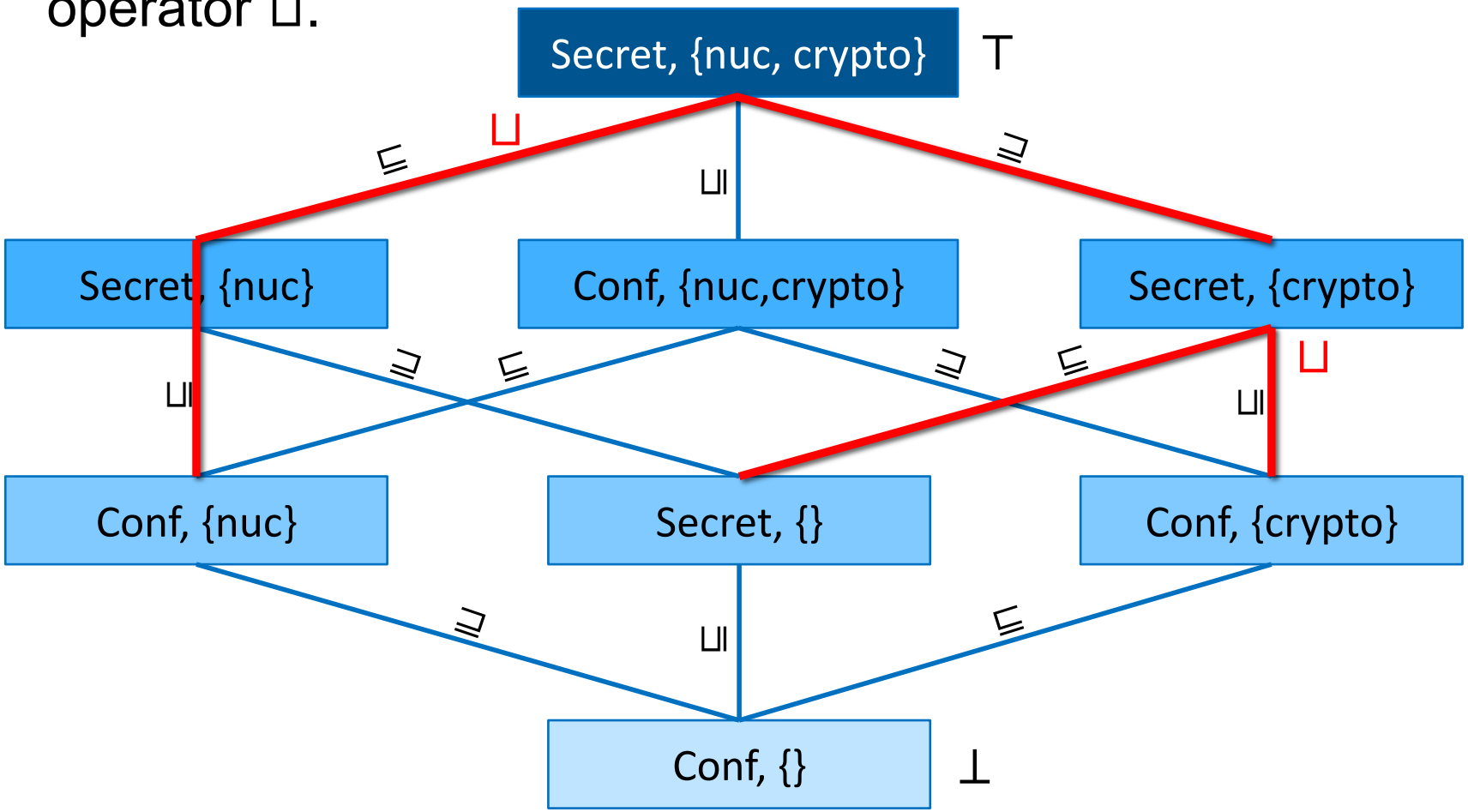


Operator for combining labels

- 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.

Lattice of labels

- The set of labels and relation $\sqsubseteq$ define a lattice, with join operator $\sqcup$.



Checking an assignment

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

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

Checking an if-statement

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

Examples for confidentiality

$\Gamma(\mathbf{x})$ 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
```

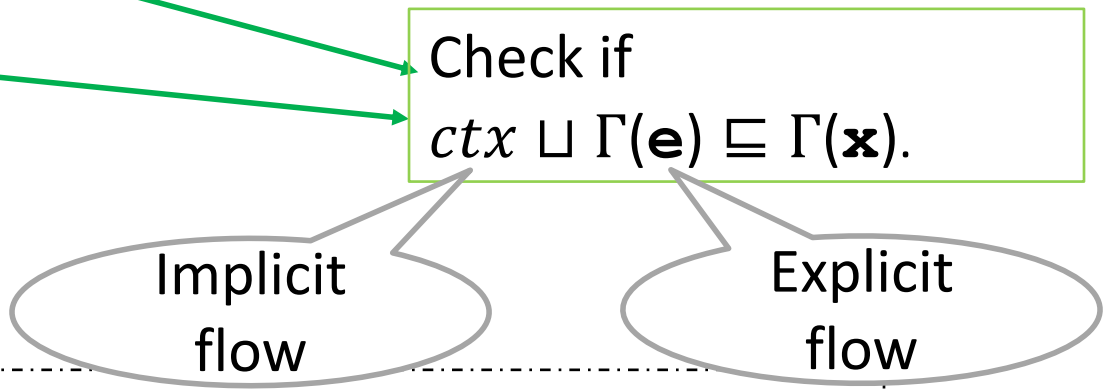
They reveal information about **z>0**.

Introduce a context label ctx

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

Context

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



Introduce a context label ctx

Its ctx is $\Gamma(\mathbf{z} > \mathbf{0})$.

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 **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 $\Gamma \vdash \mathbf{e'} : \ell'$

Example

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

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

Exercise

- Let $\Gamma(\mathbf{x}) = \mathrm{L}$ and $\Gamma(\mathbf{y}) = \mathrm{H}$.
- What is the type of **y>x+5**?
- *Proof tree:*

$$\dfrac{\dfrac{\Gamma(\mathbf{y}) = \mathrm{H}}{\Gamma \vdash \mathbf{y} : \mathrm{H}} \qquad \dfrac{\dfrac{\Gamma(\mathbf{x}) = \mathrm{L}}{\Gamma \vdash \mathbf{x} : \mathrm{L}} \qquad \dfrac{}{\Gamma \vdash \mathbf{5} : \mathrm{L}}}{\Gamma \vdash \mathbf{x + 5} : \mathrm{L}}}{\Gamma \vdash \mathbf{y > x + 5} : \mathrm{H}}$$

Typing rules for commands

Judgement $\Gamma, ctx \vdash \mathtt{c}$

According to mapping Γ, and context label ctx, command $\mathtt{c}$ is type correct.

If-rule (example)

$$\dfrac{\Gamma \vdash \mathtt{z>0} : \Gamma(\mathtt{z}) \qquad \dfrac{\Gamma \vdash \mathtt{1} : \bot \quad \bot \sqcup \Gamma(\mathtt{z}) \sqcup \mathsf{L} \sqsubseteq \Gamma(\mathtt{x})}{\Gamma, \Gamma(\mathtt{z}) \sqcup \mathrm{L} \vdash \mathtt{x:=1}} \qquad \dfrac{\Gamma \vdash \mathtt{0} : \bot, \quad \bot \sqcup \Gamma(\mathtt{z}) \sqcup \mathsf{L} \sqsubseteq \Gamma(\mathtt{x})}{\Gamma, \Gamma(\mathtt{z}) \sqcup \mathrm{L} \vdash \mathtt{x:=0}}}{\Gamma, \mathrm{L} \vdash \mathtt{if\ z>0\ then\ x:=1\ else\ x:=0}}$$

Static type system

Assignment-Rule:
$$\frac{\Gamma \vdash \mathtt{e} : \ell \qquad \ell \sqcup ctx \sqsubseteq \Gamma(\mathtt{x})}{\Gamma, ctx \vdash \mathtt{x:=e}}$$

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

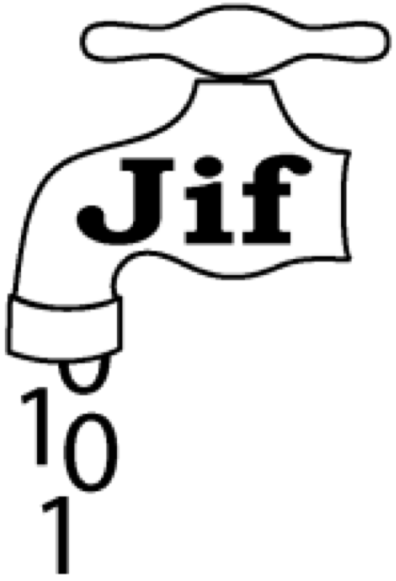
While-Rule:
$$\frac{\Gamma \vdash \mathtt{e} : \ell \qquad \Gamma, \ell \sqcup ctx \vdash \mathtt{c}}{\Gamma, ctx \vdash \mathtt{while\ e\ do\ c}}$$

Sequence-Rule:
$$\frac{\Gamma, ctx \vdash \mathtt{c1} \qquad \Gamma, ctx \vdash \mathtt{c2}}{\Gamma, ctx \vdash \mathtt{c1;c2}}$$

Soundness of type system

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

Languages for Information Flow Control



- Declare variables with information flow labels
  int {Alice→Bob} x;

- FlowCAML
- LMonad (Haskell)
- SPARK dependency contracts