Lecture 20: Information Flow Control

CS 181S

Spring 2024

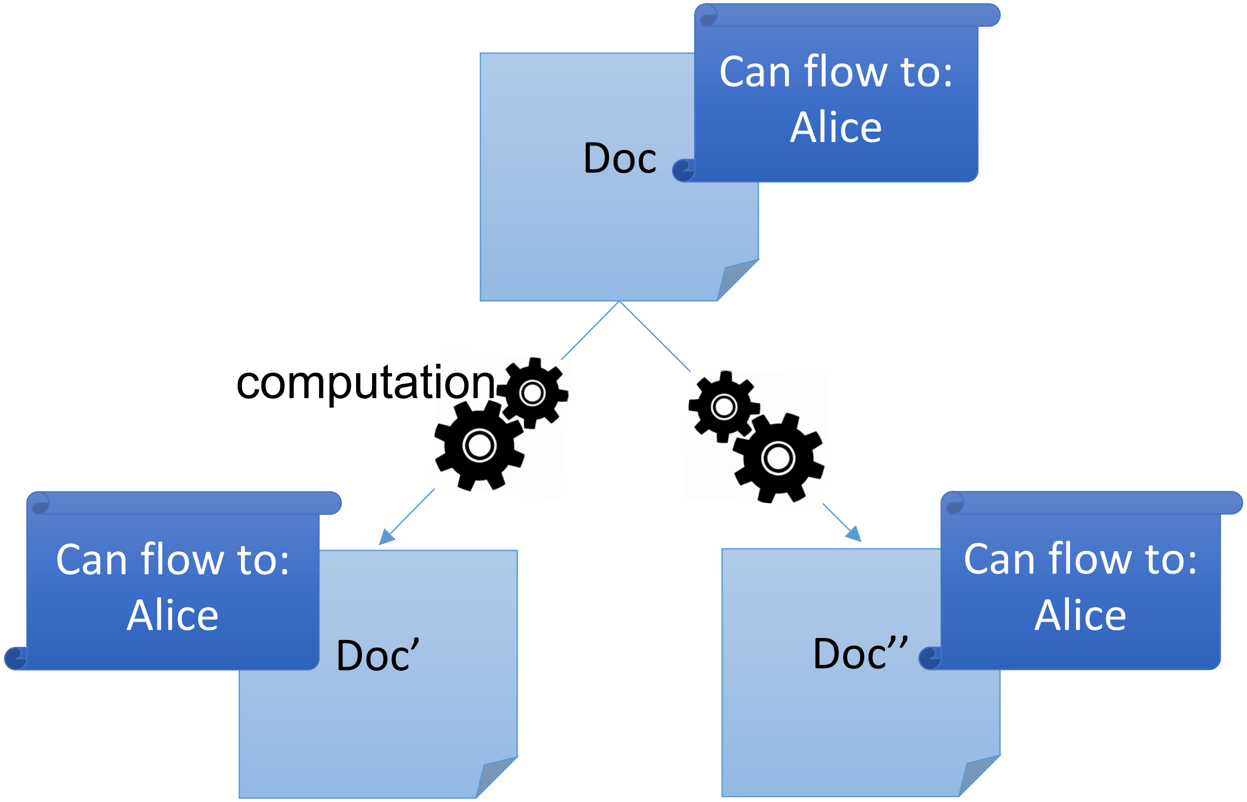
Where we were…

- **Authentication:** mechanisms that bind principals to actions

- **Authorization:** mechanisms that govern whether actions are permitted

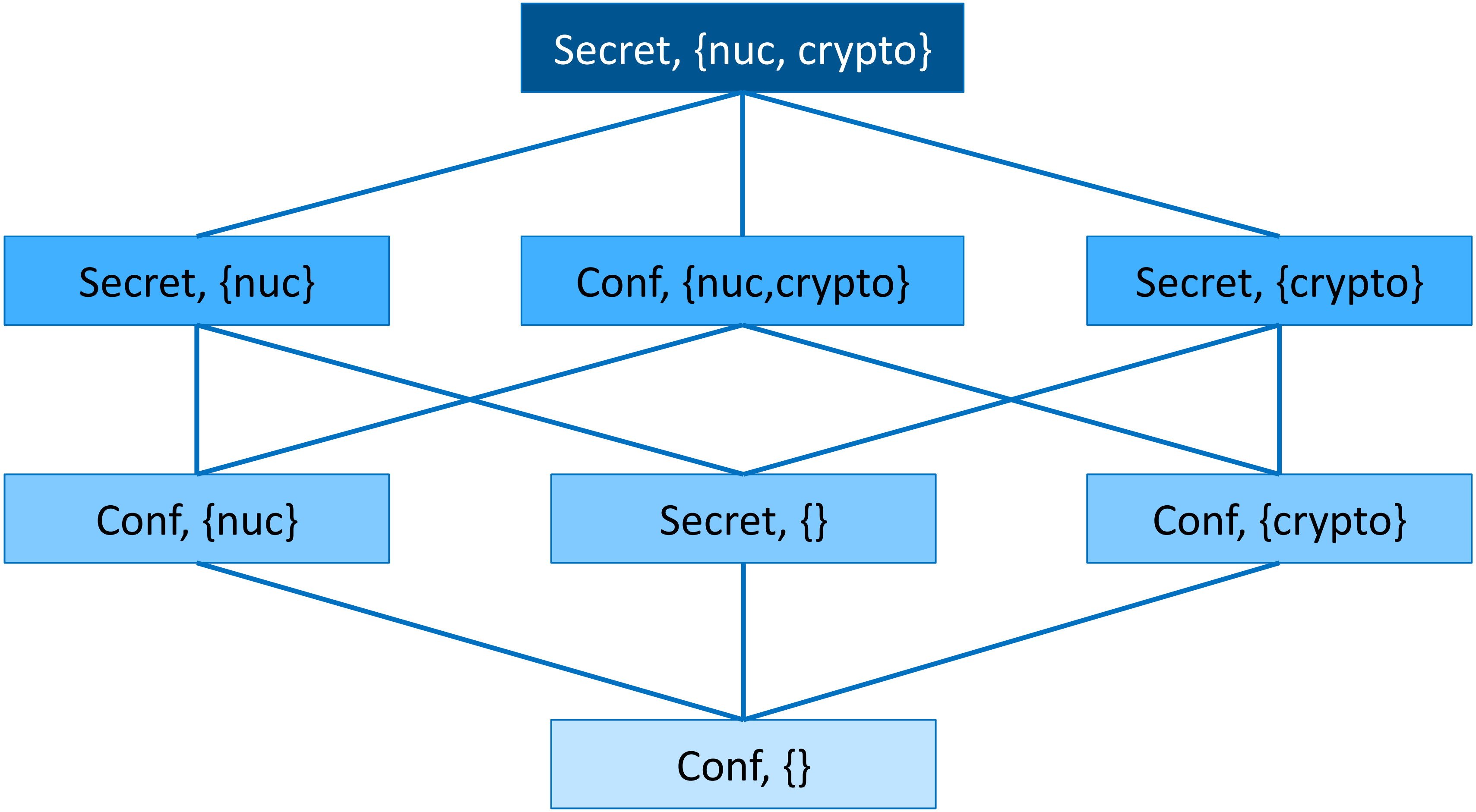
- **Audit:** mechanisms that record and review actions

Information flow policies

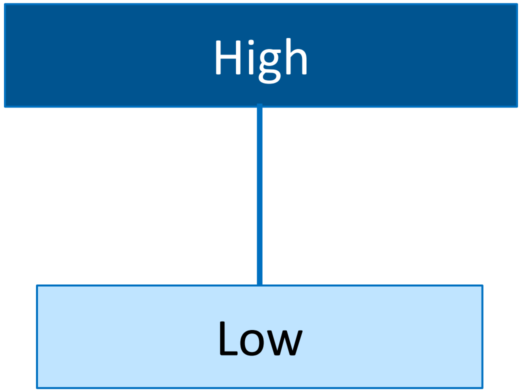


Automatic deduction of policies!

Labels represent policies



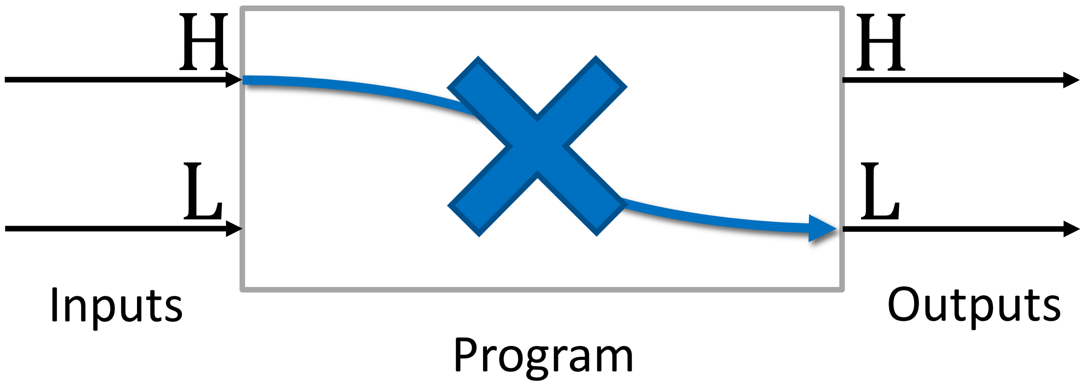
Labels represent policies



Noninterference
[Goguen and Meseguer 1982]

An interpretation of noninterference for a program:

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



Enforcing Information Flow

- Goal: Enforce that only programs that satisfy NonInterference can run in our system.

- Goal: Design a type system such that

$$\Gamma \vdash \mathbf{p} \;\Rightarrow\; \mathbf{p} \text{ satisfies NonInterference}$$

Review: Type Inference (Expressions)

```
int x;
bool y;
```

$\Gamma(x) = \textbf{int};$

$\Gamma(y) = \textbf{bool};$

- Type environment Γ maps variables to type
- Goal: Judgement (aka proof that) $\Gamma \vdash$ **e** : t

  According to mapping Γ, expression **e** has type t

- Constants: $\dfrac{}{\Gamma \vdash n{::}int}$ $\dfrac{}{\Gamma \vdash True{::}bool}$ $\dfrac{}{\Gamma \vdash False{::}bool}$
- Variables: $\dfrac{\Gamma(\mathrm{x}) = t}{\Gamma \vdash x{::}\mathrm{t}}$
- Expressions: $\dfrac{\Gamma \vdash e1{::}int, \Gamma \vdash e2{::}int}{\Gamma \vdash e1 + e2{::}int}$ $\dfrac{\Gamma \vdash e1{::}int, \Gamma \vdash e2{::}int}{\Gamma \vdash e1 < e2{::}bool}$ …

Review: Static type system

Assignment-Rule:

$$\frac{\Gamma \vdash \mathbf{e} : \mathrm{t} \qquad \mathrm{t} \sqsubseteq \Gamma(\mathbf{x})}{\Gamma \vdash \mathtt{x=e;}}$$

Sequence-Rule:

$$\frac{\Gamma \vdash \mathtt{p1} \qquad \Gamma \vdash \mathtt{p2}}{\Gamma \vdash \mathtt{p1\ p2}}$$

If-Rule:

$$\frac{\Gamma \vdash \mathbf{e} : \mathrm{bool} \qquad \Gamma \vdash \mathtt{p1} \qquad \Gamma \vdash \mathtt{p2}}{\Gamma \vdash \mathtt{if(e)\ then\{\ p1\ \}\ else\{\ p2\ \}}}$$

While-Rule:

$$\frac{\Gamma \vdash \mathbf{e} : \mathrm{bool} \qquad \Gamma \vdash \mathtt{p}}{\Gamma \vdash \mathtt{while(e)\{\ p\ \}}}$$

Skip-Rule:

$$\frac{}{\Gamma \vdash \mathtt{nop;}}$$

Label Inference (Expressions)

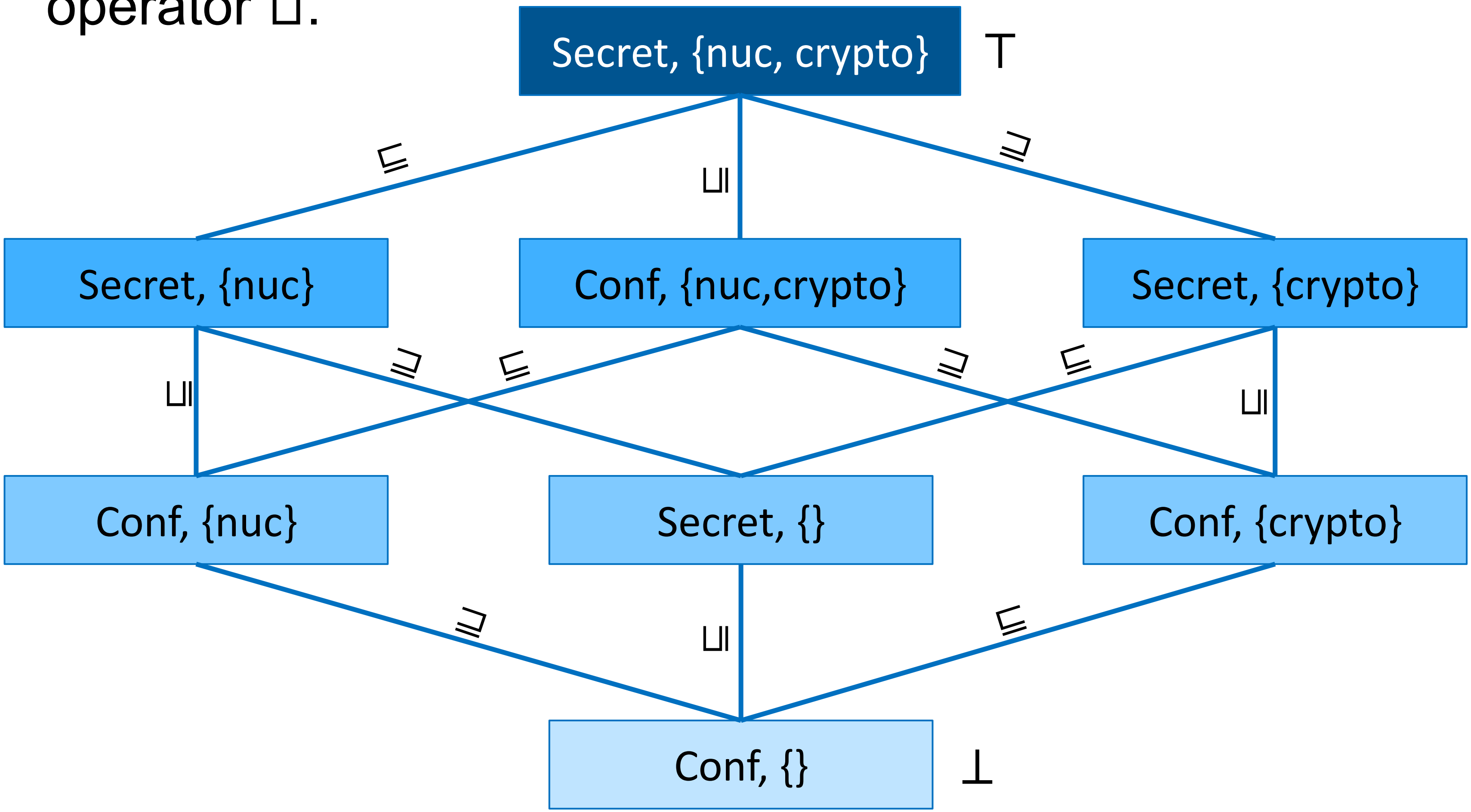
$\Gamma(x) = \boldsymbol{L};$

$\Gamma(y) = \mathbf{H};$

- Type environment Γ maps variables to ~~type~~ label
- Goal: Judgement (aka proof that) $\Gamma \vdash \mathbf{e} : \ell$
  According to mapping Γ, expression **e** has label ℓ
- Constants: $\dfrac{}{\Gamma \vdash n{::}L}$
- Variables: $\dfrac{\Gamma(\mathrm{x})=\ell}{\Gamma \vdash x{::}\ell}$
- Unary Operations: $\dfrac{\Gamma \vdash e{::}\ell}{\Gamma \vdash not\ e{::}\ell}$
- Binary Operations:

Lattice of labels

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

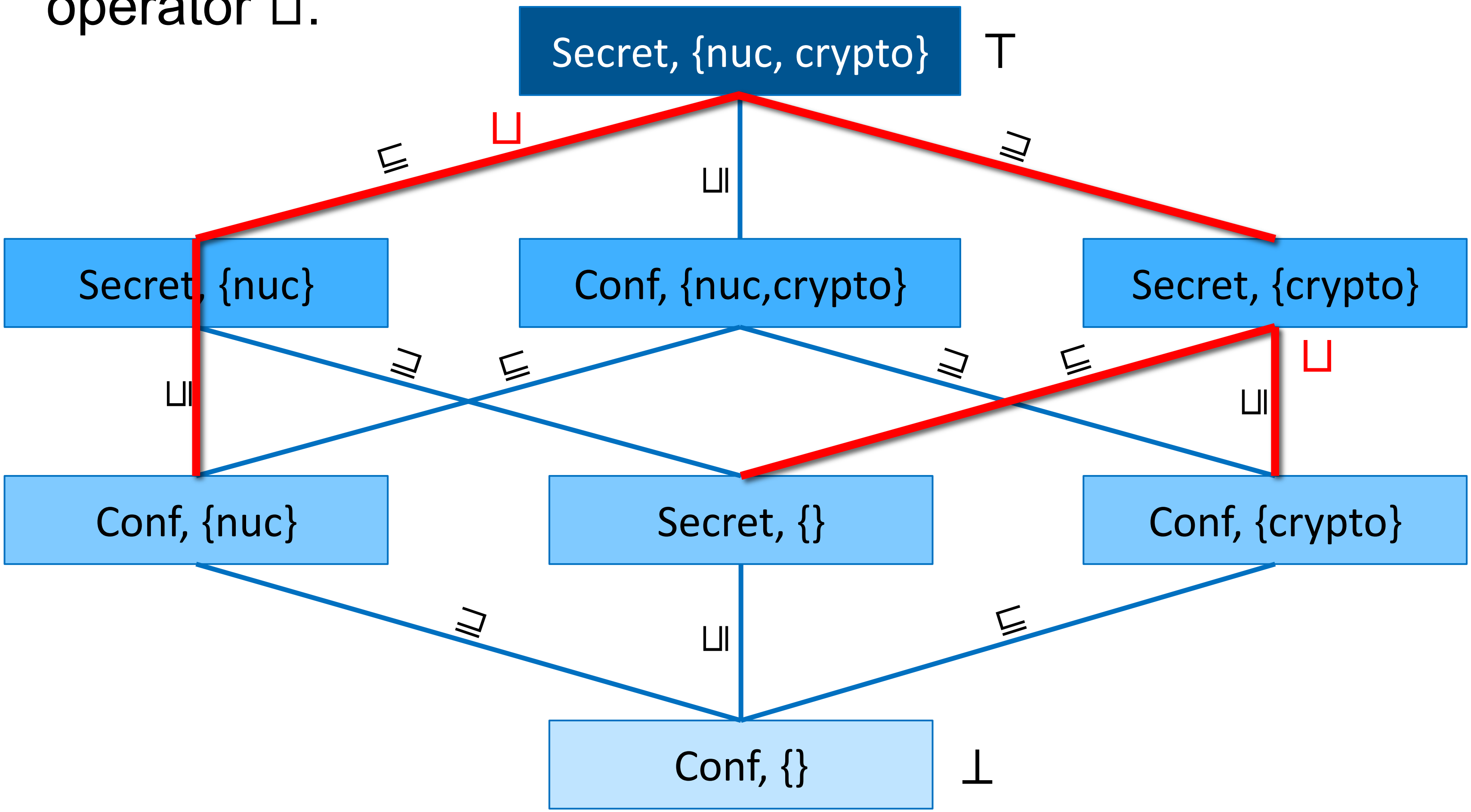


Join Operator for combining labels

- For each $\ell 1$ and $\ell 2$, there exists a label $\ell 3$, such that:
  - $\ell 1 \sqsubseteq \ell 3$
  - $\ell 2 \sqsubseteq \ell 3$
  - for all $\ell 4$ such that $\ell \sqsubseteq \ell 4$ and $\ell 2 \sqsubseteq \ell 4$, then $\ell 3 \sqsubseteq \ell 4$.
- $\ell 3$ is called the **join** of ℓ and $\ell 2$ and denoted $\ell 1 \sqcup \ell 2$
- Operator $\sqcup$ is associative and commutative.

Lattice of labels

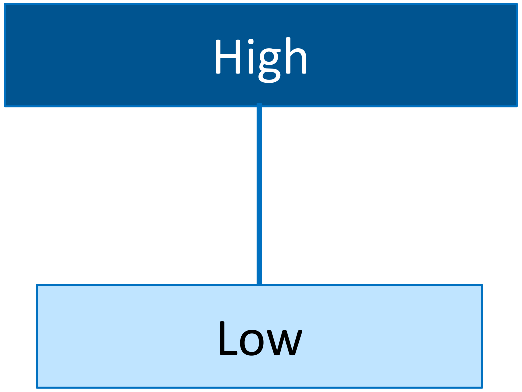
- The set of labels and relation ⊑ define a lattice, with join operator ⊔.



Exercise: Join

- What are the following labels (H or L)?

1. $H \sqcup H$
2. $H \sqcup L$
3. $L \sqcup H$
4. $L \sqcup L$



Label Inference (Expressions)

- Type environment Γ maps variables to ~~type~~ label

$\Gamma(x) = \boldsymbol{L};$
$\Gamma(y) = \mathbf{H};$

- Goal: Judgement (aka proof that) $\Gamma \vdash \mathbf{e} : \ell$
  According to mapping Γ, expression **e** has label ℓ

- Constants: $\dfrac{}{\Gamma \vdash\ n{::}L}$

- Variables: $\dfrac{\Gamma(\mathrm{x}) = \ell}{\Gamma \vdash\ x{::}\ell}$

- Unary Operations: $\dfrac{\Gamma \vdash e{::}\ell}{\Gamma \vdash\ not\ e{::}\ell}$

- Binary Operations: $\dfrac{\Gamma \vdash e1{::}\ell 1,\ \Gamma \vdash e2{::}\ell 2}{\Gamma \vdash\ e1 + e2{::}\ell 1 \sqcup \ell 2}$ …

Example

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

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

Exercise

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

Exercise: Checking an assignment

```
x = y;
```

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

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

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

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

Checking an assignment

```
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 $\Gamma(\mathtt{y}) \sqcup \Gamma(\mathtt{z}) \sqsubseteq \Gamma(\mathtt{x})$.

It satisfies NI, if $\Gamma \vdash \mathtt{y\ +\ z} :: \ell$ and $\ell \sqsubseteq \Gamma(\mathtt{x})$

Exercise: Checking a conditional assignment

```
if(z > 0){
    x = 1;
}else{
    x = 0;
}
```

Examples for confidentiality

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

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

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

$\Gamma(\mathbf{x})$ is H.
$\Gamma(\mathbf{z})$ is H.
Does the assignment satisfy NI?

Checking an if-statement

```
if(z > 0){
    x = 1;
} else {
    x = 0;
}
```

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

Context

```
if(z > 0){
    x = 1;
} else {
    x = 0;
}
```

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

Introduce a context label ctx

Its ctx is the type of the expression **z > 0**

Context

```
if(z > 0){
    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 the label of the expression `z > 0`.

Static type system

Assignment-Rule:

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

If-Rule:

$$\frac{\Gamma \vdash \mathbf{e} : \ell \qquad \Gamma, \ell \sqcup ctx \vdash \mathbf{p1} \qquad \Gamma, \ell \sqcup ctx \vdash \mathbf{p2}}{\Gamma, ctx \vdash \mathtt{if(e)\{\ p1\ \}\ else\{\ p2\ \}}}$$

While-Rule:

$$\frac{\Gamma \vdash \mathbf{e} : \ell \qquad \Gamma, \ell \sqcup ctx \vdash \mathbf{p}}{\Gamma, ctx \vdash \mathtt{while(e)\{\ p\ \}}}$$

Sequence-Rule:

$$\frac{\Gamma, ctx \vdash \mathbf{p1} \qquad \Gamma, ctx \vdash \mathbf{p2}}{\Gamma, ctx \vdash \mathtt{p1\ p2}}$$

Skip-Rule:

$$\frac{}{\Gamma, ctx \vdash \mathtt{nop;}}$$

Soundness of type system

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

Exercise: Type Checking

- Assume $\Gamma(x) = H$ and $\Gamma(z) = H$. Prove that the program `if(z>0){x = 1;} else {x = 0;}` type checks (in a L context).

Exercise: Type Checking

- Assume $\Gamma(x) = L$ and $\Gamma(z) = H$. Try to prove that the program **`if(z>0){x = 1;} else {x = 0;}`** type checks (in a L context).

Languages for Information Flow Control

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

- FlowCAML
- LMonad (Haskell)
- SPARK dependency contracts

```
class passwordFile authority(root) {
    public boolean
      check (String user, String password)
      where authority(root) {
        // Return whether password is correct
        boolean match = false;
        try {
            for (int i = 0; i < names.length; i++) {
                if (names[i] == user &&
                passwords[i] == password) {
                    match = true;
                    break;
                }
            }
        }
            catch (NullPointerException e) {}
            catch (IndexOutOfBoundsException e) {}
        return declassify(match, {user; password});
    }
    private String [ ] names;
    private String { root: } [ ] passwords;
}
```

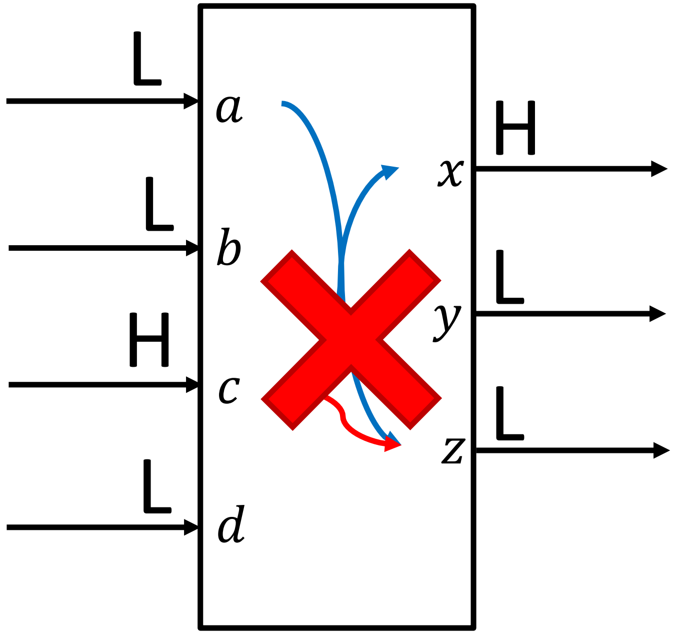
```
class passwordFile authority(root) {
  public boolean
    check (String user, String password)
    where authority(root) {
      // Return whether password is correct
      boolean match = false;
      try {
        for (int i = 0; i < names.length; i++) {
          if (names[i] == user &&
          passwords[i] == password) {
            match = true;
            break;
          }
        }
      }
        catch (NullPointerException e) {}
        catch (IndexOutOfBoundsException e) {}
      turn declassify(match, {user; password});
  }
  private [ ] names;
  private String { root: } [ ] passwords;
}
```

Security type: only **root** may learn information in this field

Declassification: okay to leak whether password matches

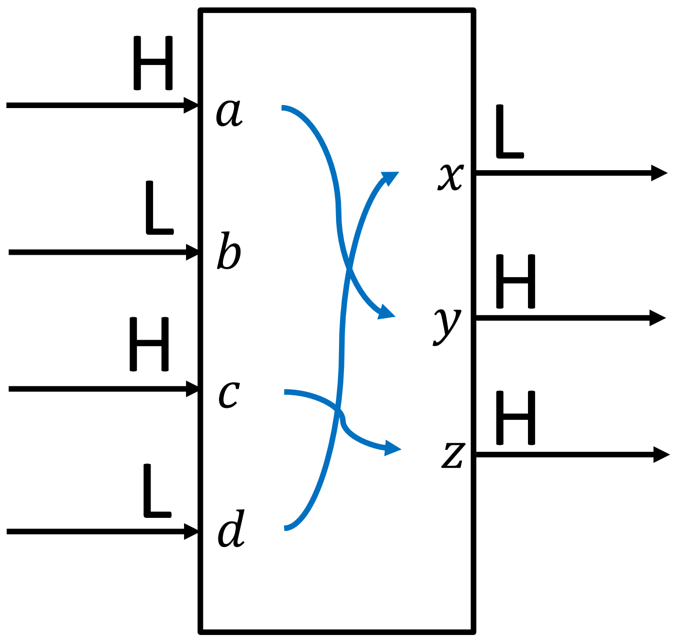
```
class passwordFile authority(root) {
  public boolean
    check (String user, String password)
    where authority(root) {
      // Return whether password is correct
      boolean match = false;
      try {
        for (int i = 0; i < names.length; i++) {
          if (names[i] == user &&
          passwords[i] == password) {
            match = true;
            break;
          }
        }
      }
      catch (NullPointerException e) {}
      catch (IndexOutOfBoundsException e) {}
      return declassify(match, {user; password});
  }
  private String [ ] names;
  private String { root: } [ ] passwords;
}
```

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 any flow violates noninterference

Information Flow Control: flow-sensitive Γ



- Γ may change during the analysis of the program.
- The mechanism deduces $\Gamma(x)$, $\Gamma(y)$, $\Gamma(z)$ 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.

- Also have to deal with declassification…