CS 5430: Information Flow

Part II: Dynamic Enforcement

Fred B. Schneider
Samuel B Eckert Professor of Computer Science

Department of Computer Science
Cornell University
Ithaca, New York 14853
U.S.A.

Cornell CIS
Computer Science

Enforcement of FBAC

FLI imposes restrictions on each statement.

$$v \rightarrow w \quad \Longrightarrow \quad \Gamma(v) \sqsubseteq \Gamma(w)$$

- Static Enforcement
  - Compiler ensures type-correct programs satisfy restrictions.
- Dynamic Enforcement
  - run-time checks ensure program execution satisfies restrictions.
  - changes to labels mean program execution satisfies restrictions.

Why Dynamic Enforcement?

- Static enforcement: Rejects program if any execution could violate Flow-Label invariant.
- Dynamic enforcement: Blocks after partial execution when Flow-Label invariant could be violated.

$$\mathbf{if}\ 0 = 0\ \mathbf{then}\ x_L \coloneqq 2\ \mathbf{else}\ x_L \coloneqq x_H\ \mathbf{fi}$$

Why Dynamic Enforcement?

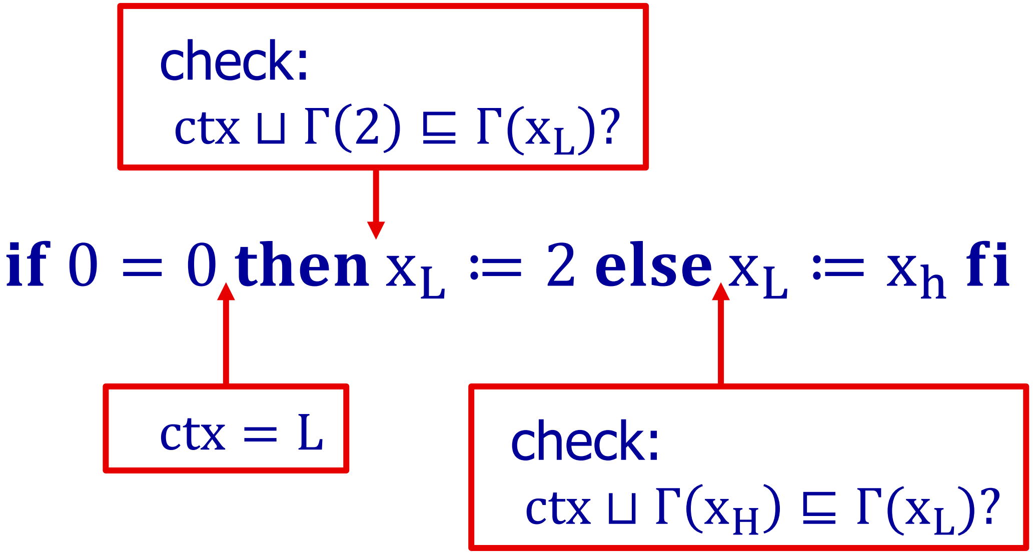
- Static enforcement: Rejects program if any execution could violate Flow-Label invariant.
- Dynamic enforcement: Blocks after partial execution when Flow-Label invariant could be violated.

Type error!

$$\textbf{if } 0 = 0 \textbf{ then } \mathrm{x_L} := 2 \textbf{ else } \boxed{\mathrm{x_L} := \mathrm{x_H}} \textbf{ fi}$$

Why Dynamic Enforcement?

- Static enforcement: Rejects program if execution could violates Flow-Label invariant
- Dynamic enforcement: Blocks after partial execution when Flow-Label invariant could be violated.



check:

$ctx \sqcup \Gamma(2) \sqsubseteq \Gamma(x_L)$?

$\mathbf{if}\ 0 = 0\ \mathbf{then}\ x_L := 2\ \mathbf{else}\ x_L := x_h\ \mathbf{fi}$

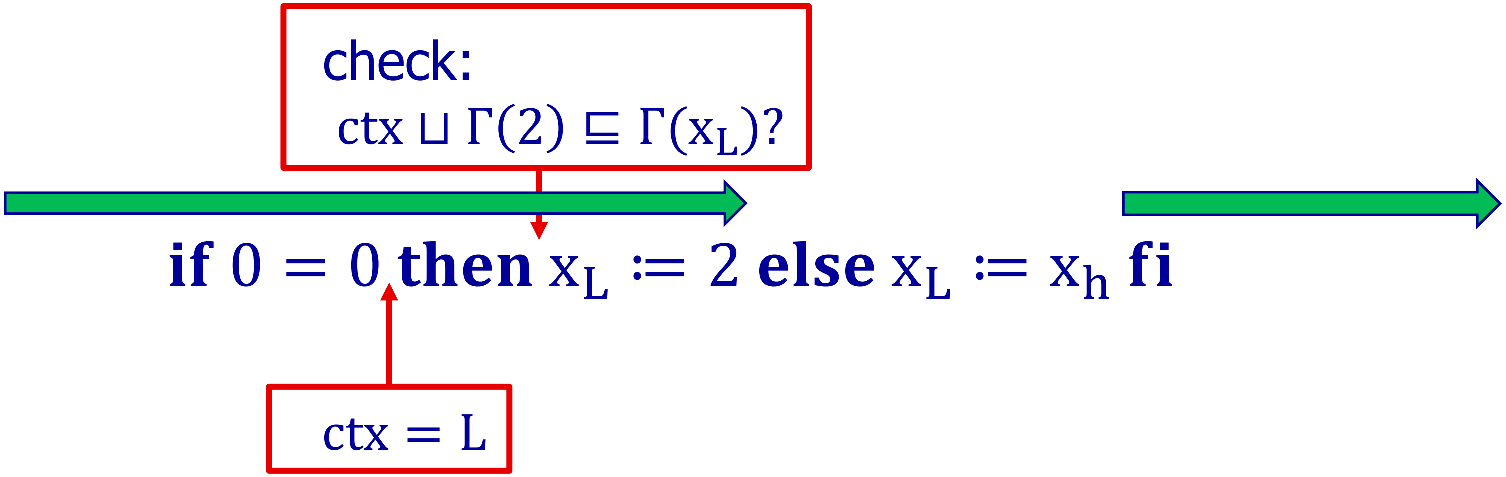
$ctx = L$

check:

$ctx \sqcup \Gamma(x_H) \sqsubseteq \Gamma(x_L)$?

Why Dynamic Enforcement?

- Static enforcement: Rejects program if execution could violates Flow-Label invariant
- Dynamic enforcement: Blocks after partial execution when Flow-Label invariant could be violated.



check:

$\mathrm{ctx} \sqcup \Gamma(2) \sqsubseteq \Gamma(x_L)$?

$\textbf{if } 0 = 0 \textbf{ then } x_L := 2 \textbf{ else } x_L := x_h \textbf{ fi}$

$\mathrm{ctx} = L$

Implementing Dynamic Enforcement

Conjecture: To implement dynamic enforcement:

- Precede $x := \mathrm{Expr}$ with check: "$\mathrm{ctx} \sqcup \Gamma(\mathrm{Expr}) \sqsubseteq \Gamma(x)$?"
- Block execution if check fails.

Implementing Dynamic Enforcement

Conjecture: To implement dynamic enforcement:

- Precede $x := \mathrm{Expr}$ with check: "$\mathrm{ctx} \sqcup \Gamma(\mathrm{Expr}) \sqsubseteq \Gamma(x)$?"
- Block execution if check fails

$\Gamma(B) \sqcup \Gamma(\mathrm{Expr}) \sqsubseteq \Gamma(x_L)$?

```
$x_L := 0$
if B then $x_L := \mathrm{Expr}$
     else skip
fi
```

Implementing Dynamic Enforcement

Conjecture:

- Precede $x := \mathrm{Expr}$ with check: "$\mathrm{ctx} \sqcup \Gamma(\mathrm{Expr}) \sqsubseteq \Gamma(x)$?"
- Block execution if check fails

$\Gamma(B) \sqcup \Gamma(\mathrm{Expr}) \sqsubseteq \Gamma(x_L)$?

```
$x_L := 0$
**if** B **then** $x_L := \mathrm{Expr}$
      **else** **skip**
**fi**
```

But… when stop on check:

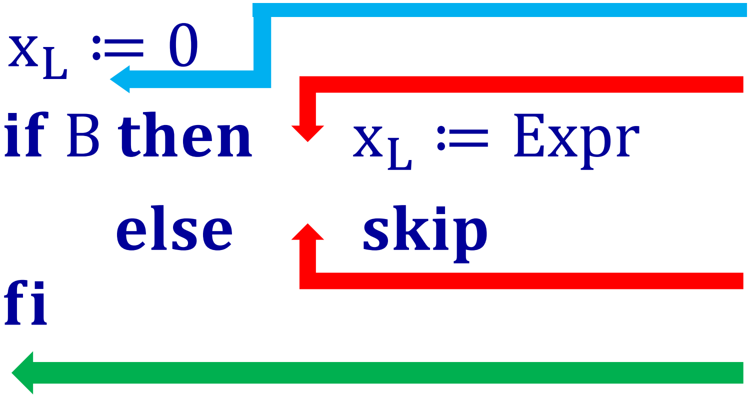
- … B=true leaks!
- Result: implemented RNI (=termination insensitive) only

Solution: Hybrid Enforcement

```
$x_L$ := 0
if B then    $x_L$ := Expr
     else    skip
fi
```

- $B \rightarrow x_L$ whether or not $x_L := \mathrm{Expr}$ executes.
  - For $\Gamma(B) = H$, could exist memories M and M′ with different H values causing termination with x_L having different values.

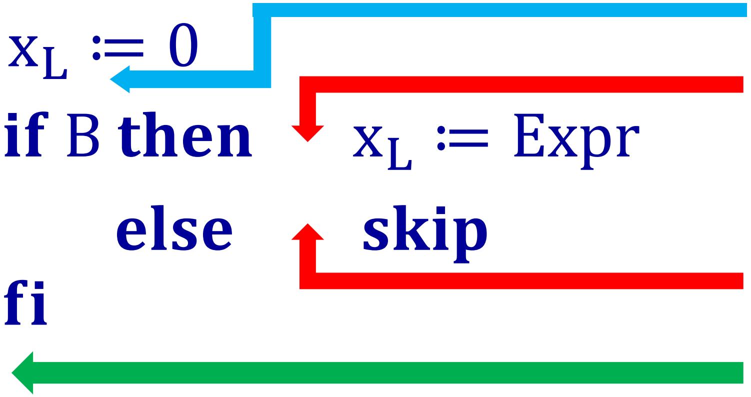
Solution: Hybrid Enforcement



```
$x_L := 0$
if B then  $x_L := Expr$
     else  skip
fi
```

- $B \rightarrow x_L$ whether or not $x_L := Expr$ executes.
  - For $\Gamma(B) = H$, could exist memories M and M′ with different H values causing termination with x_L having different values.
  - FLI requires $\Gamma(B) \sqsubseteq x_L$
    - Before **if** **-or-** Within **then** and within **else** **-or-** After **if**
  - FLI also requires $\Gamma(Expr) \sqsubseteq x_L$ before $x_L := Expr$

Solution: Hybrid Enforcement



```
$x_L := 0$
if B then   $x_L := \text{Expr}$
     else   skip
fi
```

- $B \rightarrow x_L$ whether or not $x_L := \text{Expr}$ executes.
  - For $\Gamma(B) = H$, could exist memories M and M′ with different H values causing termination with x_L having different values.
  - FLI requires $\Gamma(B) \sqsubseteq x_L$
    - Before **if** **-or-** Within **then** and within **else** **-or-** After **if**
  - FLI also requires $\Gamma(\text{Expr}) \sqsubseteq x_L$ before $x_L := \text{Expr}$
- What if B is $x_H \neq x_H$?

Hybrid Enforcement: Summary

$$\textbf{if}\ \ \mathrm{B}\ \ \textbf{then}\ \ \mathrm{C}_1\ \ \textbf{else}\ \ \mathrm{C}_2\ \ \textbf{fi}$$

- Insert check $\Gamma(\mathrm{Expr}) \sqsubseteq \Gamma(\mathrm{x})$ before execution of each "$\mathrm{x} := \mathrm{Expr}$" in C_1 or C_2.

- Insert check $\Gamma(\mathrm{B}) \sqsubseteq \Gamma(\mathrm{x})$ within execution of both C_1 and C_2 if "$\mathrm{x} := \ldots$" appears anywhere within C_1 or within C_2.

Flow-Sensitive Labels

A given variable might be given different **flow-sensitive** labels during execution.

Example:

$$x := \text{Hval}; \quad x := 0; \quad x_L := x$$

Observe:

- If $\Gamma(x) = H$ then program does not type check.

Flow-Sensitive Labels

A given variable might be given different **flow-sensitive** labels during execution.

Example:

$$x := \mathrm{Hval}; \quad x := 0; \quad x_L := x$$

red given label H; green given label L

Program does type check and satisfies:

$$v \to w \Rightarrow \Gamma(v) \sqsubseteq \Gamma(w)$$

Flow Sensitive Labels + Dynamic?

```
x := 0  {Γ(x) = L}
if h > 0 then      x := 2;  {Γ(x) = Γ(h) = H}
         else      skip
fi
```

- $h > 0$ is true: After **fi** $\Gamma(x) = H$
- $h > 0$ is false: After fi $\Gamma(x) = L$

Problem: $h \rightarrow x$ but $\Gamma(h) \not\sqsubseteq \Gamma(x)$

Flow Sensitive + …   Soln 1

Rule: Block execution from entering conditional commands with high guards and lower targets.

Stop here!

```
x := 0
if h > 0 then      x := 2
         else      skip
fi
```

Flow Sensitive + … Soln 2

Rule: Update labels of target variables in untaken branches to capture implicit flow.

```
x := 0
 if h > 0 then      x  := 2;  Γ(x) := Γ(h)
          else      skip;   Γ(x) := Γ(h)
 fi
```

Leaks thru Flow-Sensitive Labels

Suppose: $\Gamma(\mathrm{m}) = \mathrm{M}$ and $\mathrm{L} \sqsubseteq \mathrm{M} \sqsubseteq \mathrm{H}$

$$\textbf{if}\ \mathrm{m} > 0\ \textbf{then}\ \mathrm{w} := \mathrm{hi}\ \textbf{else}\ \mathrm{w} := \mathrm{lo}\ \textbf{fi}$$

Leaks thru Flow-Sensitive Labels

Suppose: $\Gamma(\mathrm{m}) = \mathrm{M}$ and $\mathrm{L} \sqsubseteq \mathrm{M} \sqsubseteq \mathrm{H}$

false M

$$\textbf{if}\ \mathrm{m} > 0\ \textbf{then}\ \mathrm{w} := \mathrm{hi}\ \textbf{else}\ \mathrm{w} := \mathrm{lo}\ \textbf{fi}$$

Leaks thru Flow-Sensitive Labels

Suppose: $\Gamma(\mathrm{m}) = \mathrm{M}$ and $\mathrm{L} \sqsubseteq \mathrm{M} \sqsubseteq \mathrm{H}$

$$\mathbf{if}\ \mathrm{m} \overset{\substack{\text{false}\\ \vdots}}{\underset{\substack{\vdots\\ \text{true}}}{>}} 0\ \mathbf{then}\ \underset{\substack{\vdots\\ \text{H}}}{\mathrm{w}} := \mathrm{hi}\ \mathbf{else}\ \overset{\substack{\text{M}\\ \vdots}}{\mathrm{w}} := \mathrm{lo}\ \mathbf{fi}$$

- Value of m leaks to label (M vs H) of w.

Avoiding Leaks thru Flow Sensitive 1

Rule: Use the same flow-sensitive label for an assignment target, independent of guard.

Example

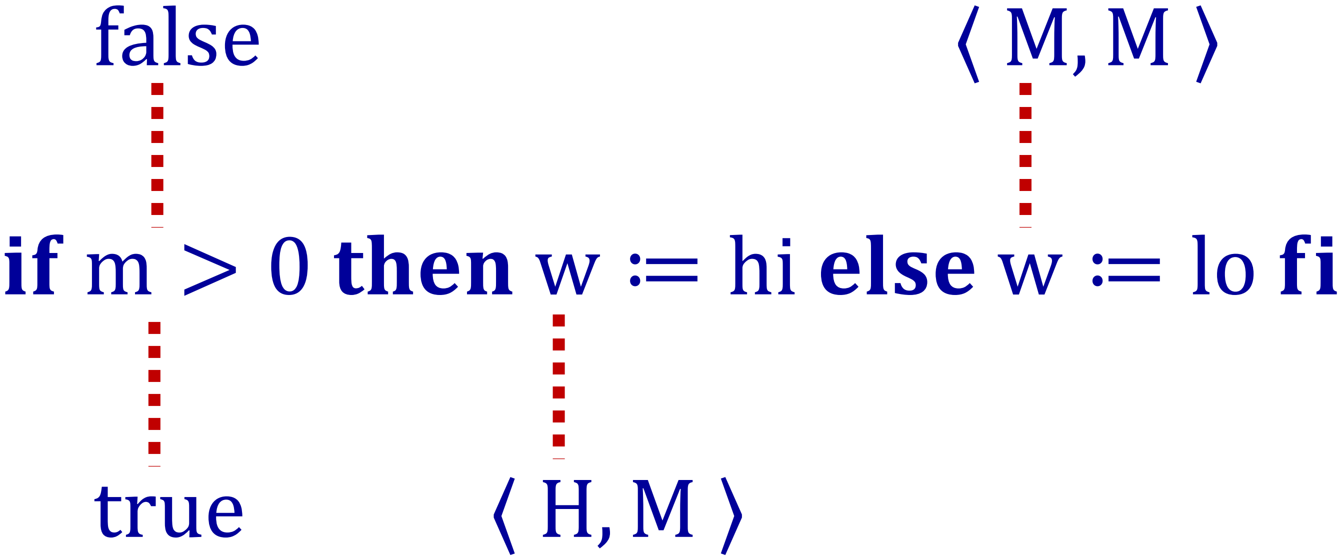
$$\textbf{if}\ m > 0\ \textbf{then}\ \underset{H}{w} := \mathrm{hi}\ \textbf{else}\ \underset{H}{w} := \mathrm{lo}\ \textbf{fi}$$

(Sound but conservative.)

Avoiding Leaks thru Flow Sensitive 2

Rule: Associate a metalabel with each label.

Example:



false

$\langle M, M \rangle$

$\textbf{if}\ m > 0\ \textbf{then}\ w := hi\ \textbf{else}\ w := lo\ \textbf{fi}$

true

$\langle H, M \rangle$

Labels for meta-labels?

Summary

FLI: $v \rightarrow w \implies \Gamma(v) \sqsubseteq \Gamma(w)$

- Static enforcement
  - Conservative
- Dynamic enforcement
  - Insert tests
    - Mind the untaken assignment!
  - Change labels
    - Static
    - Dynamic: Leaks thru labels