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.
Enforcement of FBAC

FLI imposes restrictions on each statement.

\[ v \to w \implies \Gamma(v) \subseteq \Gamma(w) \]

- **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.

\[
\text{if } 0 = 0 \text{ then } x_L := 2 \text{ else } x_L := x_H \text{ fi}
\]
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!

\[
\text{if } 0 = 0 \text{ then } x_L := 2 \text{ else } x_L := x_H \text{ fi}
\]
Why Dynamic Enforcement?

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

```
if 0 = 0 then x_L := 2 else x_L := x_h fi
```

check:
\[
\text{ctx} \sqcup \Gamma(2) \subseteq \Gamma(x_L)\
\]

check:
\[
\text{ctx} \sqcup \Gamma(x_H) \subseteq \Gamma(x_L)\
\]

\[
\text{ctx} = L
\]
Why Dynamic Enforcement?

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

```latex
\text{check:} \quad \text{ctx} \cup \Gamma(2) \subseteq \Gamma(x_L) ?
```

```latex
\text{if } 0 = 0 \text{ then } x_L := 2 \text{ else } x_L := x_h \text{ fi}
```

```latex
\text{ctx} = \text{L}
```
Implementing Dynamic Enforcement

Conjecture: To implement dynamic enforcement:
  - Precede $x := \text{Expr}$ with check: “$\text{ctx} \sqcup \Gamma(\text{Expr}) \subseteq \Gamma(x)$?”
  - Block execution if check fails.
Implementing Dynamic Enforcement

Conjecture: To implement dynamic enforcement:

- Precede $x := \text{Expr}$ with check: “$\text{ctx} \sqcup \Gamma(\text{Expr}) \subseteq \Gamma(x)$?”
- Block execution if check fails

\[
\begin{align*}
  x_L &:= 0 \\
  \text{if } B &\text{ then } x_L := \text{Expr} \\
  \text{else } &\text{ skip} \\
  \text{fi}
\end{align*}
\]
Implementing Dynamic Enforcement

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

\[
x_L := 0
\]

\[
\text{if } B \text{ then } x_L := \text{Expr}
\]

\[
\text{else } \text{skip}
\]

\[
\text{fi}
\]

But... when stop on check:
- ... \( B=\text{true} \) leaks!
- Result: implemented RNI (=termination insensitive) only
Solution: Hybrid Enforcement

\[ x_L := 0 \]
\[ \text{if } B \text{ then } x_L := \text{Expr} \]
\[ \quad \text{else skip} \]
\[ \text{fi} \]

- \( B \rightarrow x_L \) 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.
Solution: Hybrid Enforcement

\[ x_L := 0 \]

\begin{align*}
\text{if } B \text{ then } & \quad x_L := \text{Expr} \\
\text{else } & \quad \text{skip}
\end{align*}

- \( B \rightarrow x_L \) 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 \( \Gamma(B) \subseteq x_L \)
    - Before if  -or-  Within then and within else  -or-  After if
  - FLI also requires \( \Gamma(\text{Expr}) \subseteq x_L \) before \( x_L := \text{Expr} \)

What if \( B \) is \( x_D \neq x_D \)?

10
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 \( \Gamma(B) \subseteq x_L \)
    - Before if -or- Within then and within else -or- After if
  - FLI also requires \( \Gamma(\text{Expr}) \subseteq x_L \) before \( x_L := \text{Expr} \)

- What if \( B \) is \( x_H \neq x_H \)?
Hybrid Enforcement: Summary

\[
\text{if } B \text{ then } C_1 \text{ else } C_2 \text{ fi}
\]

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

- Insert check \( \Gamma(B) \subseteq \Gamma(x) \) within execution of both \( C_1 \) and \( C_2 \) if "\( x := \ldots \)" 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:
\[ \text{x := Hval; } \quad \text{x := 0; } \quad \text{x}_L := x \]

red given label H; green given label L

Program does type check and satisfies:
\[ v \rightarrow w \Rightarrow \Gamma(v) \subseteq \Gamma(w) \]
Flow Sensitive Labels + Dynamic?

\[
\begin{align*}
  x &:= 0 \ (\{\Gamma(x) = L\}) \\
  \text{if } h > 0 \text{ then } & x := 2; \ (\{\Gamma(x) = \Gamma(h) = H\}) \\
  \text{else } & \text{skip} \\
  \text{fi}
\end{align*}
\]

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

**Problem:** \( h \rightarrow x \) but \( \Gamma(h) \not\subseteq \Gamma(x) \)
Rule: Block execution from entering conditional commands with high guards and lower targets.

\[ x := 0 \]

\[ \text{if } h > 0 \text{ then } x := 2 \]
\[ \text{else skip} \]
\[ \text{fi} \]
**Rule:** Update labels of target variables in untaken branches to capture implicit flow.

\[
x := 0
\]

\[
\text{if } h > 0 \text{ then } x := 2; \Gamma(x) := \Gamma(h)
\]

\[
\text{else } \text{skip;} \Gamma(x) := \Gamma(h)
\]

\[
\text{fi}
\]
Leaks thru Flow-Sensitive Labels

Suppose: \( \Gamma(m) = M \) and \( L \subseteq M \subseteq H \)

\[
\text{if } m > 0 \text{ then } w := \text{hi} \text{ else } w := \text{lo} \text{ fi}
\]
Leaks thru Flow-Sensitive Labels

Suppose: \( \Gamma(m) = M \) and \( L \subseteq M \subseteq H \)

\[
\begin{align*}
\text{false} & \quad \quad \quad M \\
\vdots & \quad \quad \quad \vdots \\
\text{if } m > 0 \text{ then } w := \text{hi} \quad \text{else } w := \text{lo} \quad \text{fi}
\end{align*}
\]
Leaks thru Flow-Sensitive Labels

Suppose: \( \Gamma(m) = M \) and \( L \subseteq M \subseteq H \)

\[
\begin{align*}
\text{false} \quad & \quad M \\
\text{if } m > 0 \text{ then } w := \text{hi} \quad & \quad \text{else } w := \text{lo} \quad \text{fi} \\
\text{true} \quad & \quad H
\end{align*}
\]

\( m \) 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

```
if m > 0 then w := hi else w := lo fi
```

(Sound but conservative.)
Avoiding Leaks thru Flow Sensitive 2

**Rule:** Associate a metalabel with each label.

**Example:**

\[
\text{false} \quad \langle M, M \rangle \\
\text{if } m > 0 \text{ then } w := \text{hi} \quad \text{else } w := \text{lo fi} \\
\text{true} \quad \langle H, M \rangle
\]

Labels for meta-labels?
Summary

**FLI:** \( v \rightarrow w \Rightarrow \Gamma(v) \subseteq \Gamma(w) \)

- **Static enforcement**
  - Conservative

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