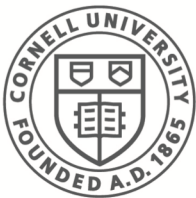


CS 5430:
Information Flow
Static Enforcement

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

Access Control

Access control associates restrictions with:

- Containers
 - access control lists, capabilities
- Values
 - information flow control

Example: $x := y; \dots z := x$

- container: value in y can be leaked by reading z
- value: restrictions on z include all restrictions on y
... no need to trust clients who access y .

Flow-based Access Control (FBAC)

- Labels propagate with flow.
- Labels restrict allowed info flow.

Flow-Label Invariant (FLI):

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

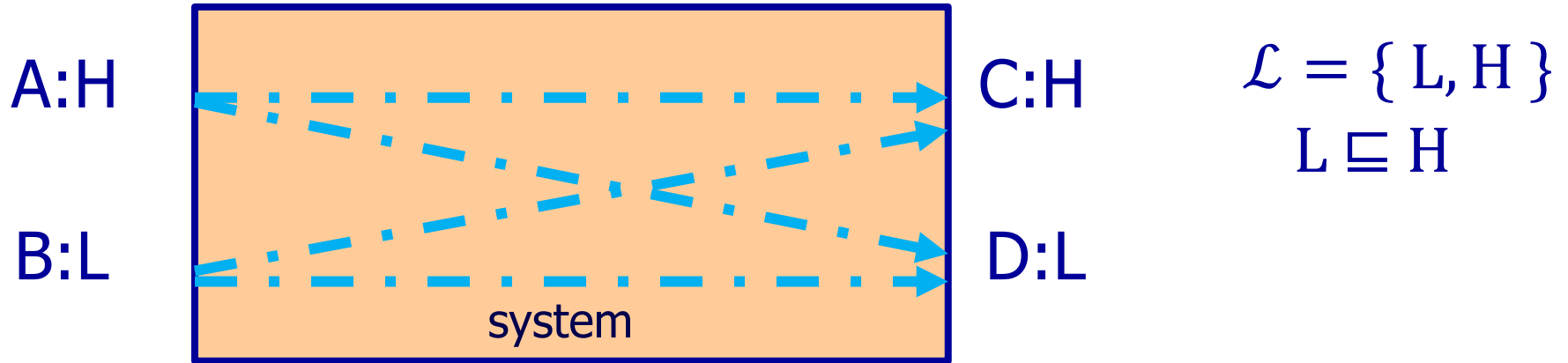
$v \rightarrow w$: v flows to w . *NB really \rightarrow_S for flow in S .*

$\Gamma(v)$: label assoc with v --- gives restrictions on use of v

\sqsubseteq reflexive and transitive relation on a set \mathcal{L} of labels.

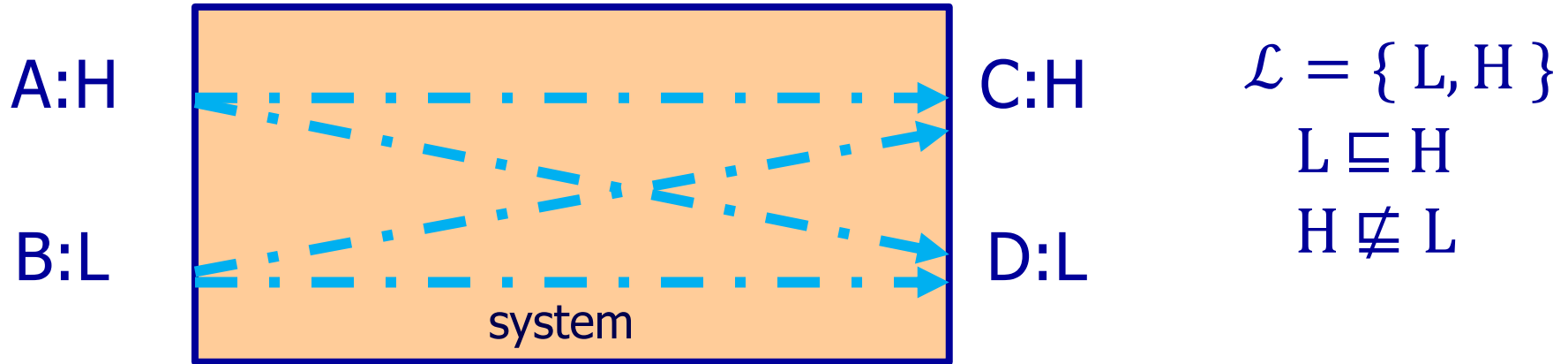
$\lambda_1 \sqsubseteq \lambda_2$: λ_2 includes all restrictions in λ_1

An application of FBAC



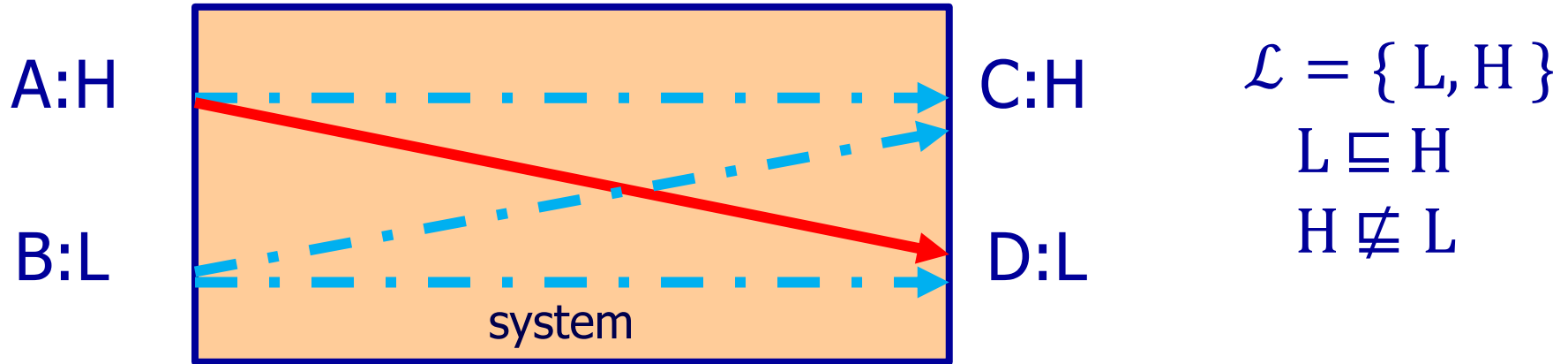
$$v \rightarrow w \implies \Gamma(v) \sqsubseteq \Gamma(w)$$

An application of FBAC



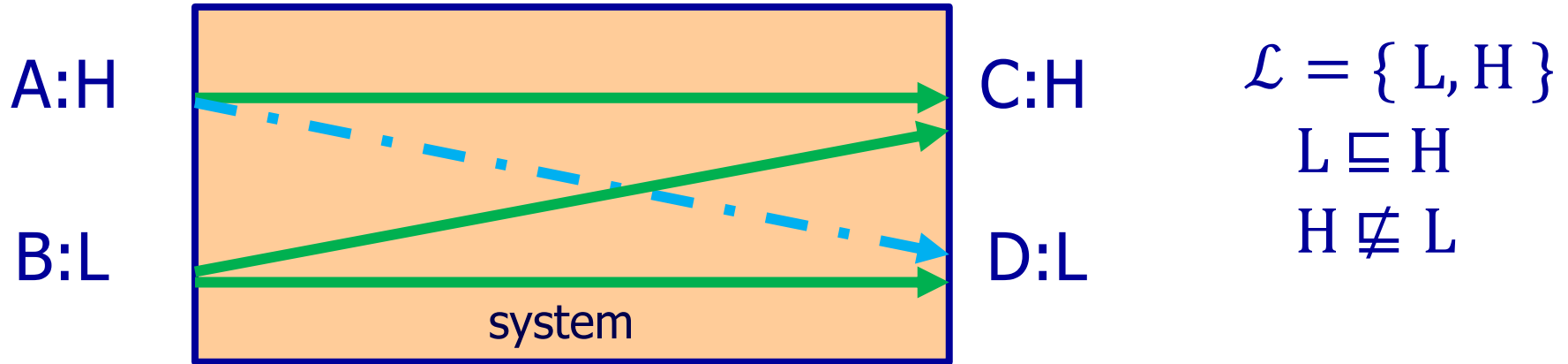
$$v \rightarrow w \implies \Gamma(v) \subseteq \Gamma(w)$$
$$= \Gamma(v) \not\subseteq \Gamma(w) \implies \neg(v \rightarrow w)$$

An application of FBAC



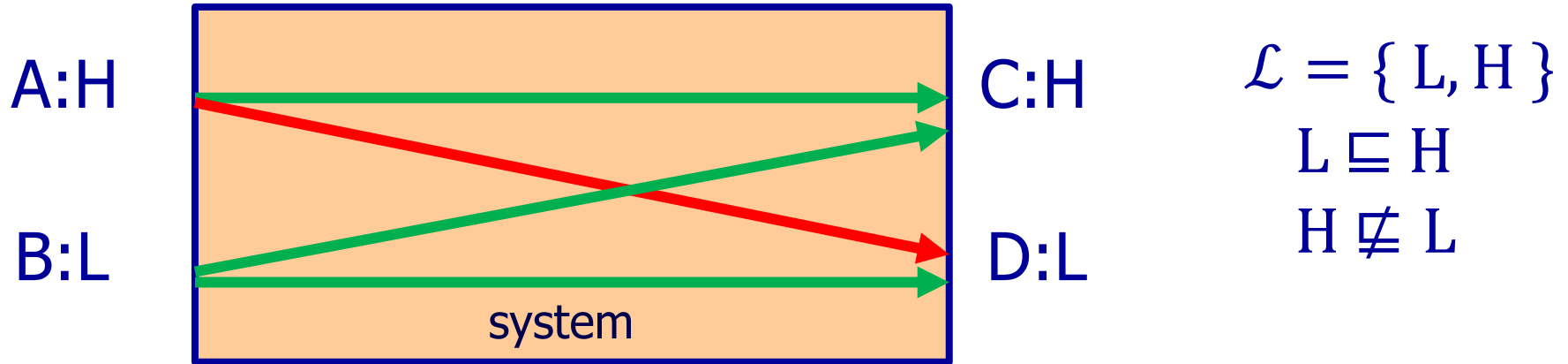
$$\begin{aligned} v \rightarrow w &\implies \Gamma(v) \sqsubseteq \Gamma(w) \\ &= \Gamma(v) \not\sqsubseteq \Gamma(w) \implies \neg(v \rightarrow w) \end{aligned}$$

An application of FBAC



$$v \rightarrow w \implies \Gamma(v) \sqsubseteq \Gamma(w)$$
$$= \Gamma(v) \not\sqsubseteq \Gamma(w) \implies \neg(v \rightarrow w)$$

An application of FBAC



$$v \rightarrow w \implies \Gamma(v) \subseteq \Gamma(w)$$
$$= \Gamma(v) \not\subseteq \Gamma(w) \implies \neg(v \rightarrow w)$$

- Confidentiality: L: public and H: secret
- Integrity: L: trusted and H: untrusted

FBAC in General

Possible source/destination of flows:

- ports
- people
- variables

FBAC in Programs

Example: $x := y; \dots z := x$

- $y \rightarrow x, \quad x \rightarrow z$
- $\Gamma(y) \sqsubseteq \Gamma(x), \quad \Gamma(x) \sqsubseteq \Gamma(z).$

FBAC in Programs

Example: $x := y; \dots z := x$

- $y \rightarrow x, \quad x \rightarrow z$
- $\Gamma(y) \sqsubseteq \Gamma(x), \quad \Gamma(x) \sqsubseteq \Gamma(z).$
 - Conclude: If $y \rightarrow z$ then $\Gamma(y) \sqsubseteq \Gamma(z)$ also must hold.
 - Nb. \rightarrow is not necessarily transitive.

Agenda

- Formalize Flow: $v \rightarrow v'$
 - Examples for intuition
 - Formal definitions
- Derive policies FBAC enforces:
 - Confidentiality
 - Integrity
- Means of enforcement
 - Static
 - Dynamic

$v \rightarrow v'$? Direct Flows in Programs

$x := y \bmod 2$

$x := y * 0$

$z := y + 2; x := z$

$z := y + 2; x := z - y$

$v \rightarrow v'$? Direct Flows in Programs

$x := y \bmod 2$

$y \rightarrow x$

$x := y * 0$

$\neg (y \rightarrow x)$

$z := y + 2; x := z$

$y \rightarrow x$

$z := y + 2; x := z - y$

$\neg (y \rightarrow x)$

... Illustrates intransitive flow

$v \rightarrow v'$? Indirect Flows in Programs

if $y > 0$ then $x := 1$ else $x := 2$ $y \rightarrow x$

while $y > 0$ do $x := x + 1$; $y := y - 1$ end $y \rightarrow x$

Definitions for Flow

$v \rightarrow w?$

Satisfied if there exist two executions

- that differ only in the initial value of v –and–
- terminate having different final values of w .

$v \rightarrow w?$ Formal Definition

Let

$\text{dom}(m) = D$ for $m \in \text{Mem}$

$\llbracket S \rrbracket: \text{Mem} \rightarrow \text{Mem} \cup \{\perp\}$

$m =_V m': (\forall v \in V: m(v) = m'(v))$

Define $v \rightarrow w$:

$(\exists m, m': m =_{D-\{v\}} m' \wedge \llbracket S \rrbracket m \neq \perp \wedge \llbracket S \rrbracket m' \neq \perp$
 $\wedge \llbracket S \rrbracket m \neq_{\{w\}} \llbracket S \rrbracket m')$

FBAC in action

Partition the set of all program variables: V_H and V_L

- $V_H = \{v \mid \Gamma(v) = H\}$ $V_L = \{v \mid \Gamma(v) = L\}$,
- $L \subseteq H$.

For all $v_H \in V_H$, $v_L \in V_L$ FBAC requires

$$\begin{aligned} & v_H \rightarrow v_L \Rightarrow \Gamma(v_H) \subseteq \Gamma(v_L) \\ = & v_H \rightarrow v_L \Rightarrow H \subseteq L \\ = & v_H \rightarrow v_L \Rightarrow \text{false} \\ = & \neg (v_H \rightarrow v_L) \end{aligned}$$

FBAC in action

$$\begin{aligned} & \neg (v_H \rightarrow v_L) \\ &= \neg (\exists m, m': m =_{D-\{v\}} m' \wedge \llbracket S \rrbracket m \neq \perp \wedge \llbracket S \rrbracket m' \neq \perp \\ & \quad \wedge \llbracket S \rrbracket m \neq_{\{w\}} \llbracket S \rrbracket m') \\ &= (\forall m, m': m =_{D-\{v\}} m' \wedge \llbracket S \rrbracket m \neq \perp \wedge \llbracket S \rrbracket m' \neq \perp \\ & \quad \Rightarrow \llbracket S \rrbracket m =_{\{w\}} \llbracket S \rrbracket m') \end{aligned}$$

Conclusion: Changes to v_H do not cause changes to v_L in terminating executions.

- Confidentiality: H is secret; L is public
- Integrity: H is untrusted; L is trusted.

Non-interference

Generalize variables v_H, v_L to sets V_H, V_L .

$$(\forall m, m': m =_{D-V_H} m' \wedge \llbracket S \rrbracket m \neq \perp \wedge \llbracket S \rrbracket m' \neq \perp \\ \Rightarrow \llbracket S \rrbracket m =_{V_L} \llbracket S \rrbracket m')$$

Changes to variables in V_H do not affect the final values of variables in V_L . Property (with **terms** often left implicit) is called :

- Termination **i**nsensitive non-interference (TINI)
- Goguen-Meseguer non-interference
- Relational non-interference (RNI)

Additional Leaks: Termination

```
if  $h > 0$   
  then while true do skip end  
  else skip  
fi
```

Termination leaks value of $h > 0$.

Value of h flows to termination: $h \rightarrow \perp$

$v \rightarrow \perp$? Formal Definition

$v \rightarrow \perp$:

$$\begin{aligned} & (\exists m, m': m =_{D-\{v\}} m' \\ & \quad \wedge (\llbracket S \rrbracket m = \perp) \neq (\llbracket S \rrbracket m' = \perp)) \end{aligned}$$

Define $\Gamma(\perp)$: Label needed by a principal in order to ascertain whether execution has terminated.

Usually $\Gamma(\perp) = L$.

Derive: Termination Sensitive NI 1/3

Flow-Label Invariant:

$$\begin{aligned} & (v \rightarrow w \implies \Gamma(v) \sqsubseteq \Gamma(w)) \wedge (v \rightarrow \perp \implies \Gamma(v) \sqsubseteq \Gamma(\perp)) \\ = & (v \rightarrow w \implies \Gamma(v) \sqsubseteq \Gamma(w)) \wedge (v \rightarrow \perp \implies \Gamma(v) \sqsubseteq L) \\ = & (\Gamma(v) = L) \\ & \vee (\neg(v \rightarrow \perp) \wedge (v \rightarrow w \implies \Gamma(v) \sqsubseteq \Gamma(w))) \end{aligned}$$

$$\Gamma(v) = L \vee (\neg(v \rightarrow \perp) \wedge (v \rightarrow w \Rightarrow \Gamma(v) \sqsubseteq \Gamma(w)))$$

$$\neg(v \rightarrow \perp):$$

$$= \neg(\exists m, m': m =_{D-\{v\}} m' \wedge (\llbracket S \rrbracket m = \perp) \neq (\llbracket S \rrbracket m' = \perp))$$

$$= (\forall m, m': m =_{D-\{v\}} m' \Rightarrow (\llbracket S \rrbracket m = \perp) = (\llbracket S \rrbracket m' = \perp))$$

$$\neg(v \rightarrow w): \quad \{ \text{since } \llbracket S \rrbracket m \neq \perp \Rightarrow \llbracket S \rrbracket m' \neq \perp \}$$

$$= \neg(\exists m, m': m =_{D-\{v\}} m' \wedge \llbracket S \rrbracket m \neq \perp \wedge (\llbracket S \rrbracket m \neq_{\{w\}} \llbracket S \rrbracket m'))$$

$$= (\forall m, m': m =_{D-\{v\}} m' \wedge \llbracket S \rrbracket m \neq \perp \Rightarrow (\llbracket S \rrbracket m =_{\{w\}} \llbracket S \rrbracket m'))$$

Define: TSNI

$$\begin{aligned} & (\forall m, m': m =_{D-\{V_H\}} m' \Rightarrow \\ & \quad ((\llbracket S \rrbracket m = \perp) = (\llbracket S \rrbracket m' = \perp)) \\ & \quad \wedge (\llbracket S \rrbracket m \neq \perp \Rightarrow (\llbracket S \rrbracket m =_{\{V_L\}} \llbracket S \rrbracket m'))) \end{aligned}$$

Other Generalizations of $v \rightarrow w$

Let $\text{dom}(m) = D$ for $m \in \text{Mem}$

$$\llbracket S \rrbracket: \text{Mem} \rightarrow \text{Mem}^* \cup \{\perp\}$$

$$m =_V m': (\forall v \in V: m(v) = m'(v))$$

$$(m_1 m_2 \dots m_i \dots) \approx_V (m'_1 m'_2 \dots m'_i):$$

$$(m_1|_V m_2|_V \dots m_i|_V \dots) =^* (m'_1|_V m'_2|_V \dots m'_i|_V)$$

where: $=^*$ is equality of de-stuttered sequences.

Define $v \rightarrow w$:

$$\begin{aligned} & (\exists m, m': m =_{D-\{v\}} m' \wedge \llbracket S \rrbracket m \neq \perp \wedge \llbracket S \rrbracket m' \neq \perp \\ & \quad \wedge \neg (\llbracket S \rrbracket m \approx_{\{w\}} \llbracket S \rrbracket m')) \end{aligned}$$

Enforcement of FBAC

FLI potentially imposes restrictions on statements.

- **Static Enforcement**

- Compiler ensures program is type correct.
- Type correct programs will satisfy restrictions.

- **Dynamic Enforcement**

- Insert run-time checks that halt program execution about to violate restrictions.
- Change labels to satisfy restrictions as program execution proceeds.

Toy Language

$e ::= x \mid n \mid e_1 + e_2 \mid \dots$

$c ::=$ $x := e$
| **if** e **then** c_1 **else** c_2 **fi**
| **while** e **do** c **end**
| $c_1; c_2$

Restrictions for:

Assignment $x := e$

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

$x := y$ causes $y \rightarrow x$

- requires $\Gamma(y) \sqsubseteq \Gamma(x)$

$x := y+z$ causes $y \rightarrow x$ and $z \rightarrow x$

- requires: $\Gamma(y+z) \sqsubseteq \Gamma(x)$
- implied by: $\Gamma(y) \sqcup \Gamma(z) \sqsubseteq \Gamma(x)$

Restrictions for:

Assignment $x := E$

$x := E$ causes $E \rightarrow x$

define $E \rightarrow x$: $(\forall v \in E: v \rightarrow x)$

define $\Gamma(E)$: $(\sqcup \Gamma(v) \in E)$

where $\lambda \sqcup \lambda'$ is smallest label satisfying

$\lambda \sqsubseteq \lambda \sqcup \lambda'$ and $\lambda' \sqsubseteq \lambda \sqcup \lambda'$

Restrictions for:

Assignment $x := E$

$x := E$ causes $E \rightarrow x$

define $E \rightarrow x$: $(\forall v \in E: v \rightarrow x)$

define $\Gamma(E)$: $(\sqcup \Gamma(v) \in E)$

where $\lambda \sqcup \lambda'$ is smallest label satisfying

$\lambda \sqsubseteq \lambda \sqcup \lambda'$ and $\lambda' \sqsubseteq \lambda \sqcup \lambda'$

$x := E$ causes $E \rightarrow x$

– requires $(\sqcup \Gamma(v) \in E) \sqsubseteq \Gamma(x)$

Restrictions for: If Statements

if $y > 0$ then $x := 1$ else $x := 2$ fi

Restrictions for: If Statements

if $y > 0$ **then** $x := 1$ **else** $x := 2$ **fi**



Restrictions for: If Statements

if $y > 0$ **then** $x := 1$ **else** $x := 2$ **fi**

$y > 0 \rightarrow pc, \quad pc \rightarrow x,$



Restrictions for: If Statements

if $y > 0$ **then** $x := 1$ **else** $x := 2$ **fi**

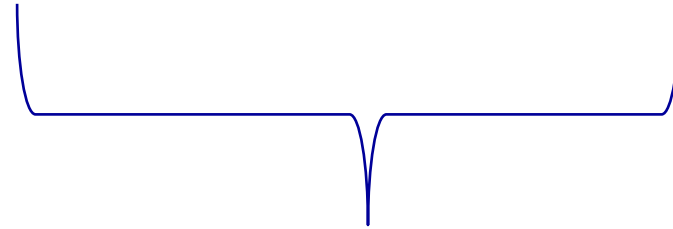


$y > 0 \rightarrow pc, \quad pc \rightarrow x, \quad y > 0 \rightarrow x$

$y > 0 \rightarrow x$ requires $\Gamma(y > 0) \sqsubseteq \Gamma(x)$

Restrictions for: If Statements

if $y > 0$ **then** $x := 1$ **else** $x := 2$ **fi**



$$\begin{aligned} ctx &= \Gamma(y > 0) \\ &= \Gamma(y) \sqcup \Gamma(0) \\ &= \Gamma(y) \end{aligned}$$

Restrictions for: If Statements

if B then $x := E$ else ... fi
 $B \rightarrow x, \quad E \rightarrow x$

Restrictions for: If Statements

if B **then** $x := E$ **else** ... **fi**

$B \rightarrow x, \quad E \rightarrow x$

requires: $\Gamma(B) \sqsubseteq \Gamma(x), \quad \Gamma(E) \sqsubseteq \Gamma(x)$

Restrictions for: If Statements

if B **then** $x := E$ **else** ... **fi**

$B \rightarrow x, \quad E \rightarrow x$

requires: $\Gamma(B) \sqsubseteq \Gamma(x), \quad \Gamma(E) \sqsubseteq \Gamma(x)$

implied by:

$\text{ctx} = \Gamma(B)$

$\text{ctx} \sqcup \Gamma(E) \sqsubseteq \Gamma(x)$

Restrictions for: Nested If Statements

```
if z > 0
  then y := 23
      if y > 0
          then x := 1
          else u := 2
      fi
  else
      w := 3
  fi
fi
```


Restrictions for: Nested If Statements

```
if z > 0
  then y := 23
        if y > 0
          then x := 1 --- ctx =  $\Gamma(y)$ 
          else u := 2 --- ctx =  $\Gamma(y)$ 
        fi
  else
    w := 3
  fi
fi
```

Restrictions for: Nested if Statements

if $z > 0$

then

$y := 23$ --- $ctx = \Gamma(z)$

if $y > 0$

then $x := 1$ --- $ctx = \Gamma(y) \sqcup \Gamma(z)$

else $u := 2$ --- $ctx = \Gamma(y) \sqcup \Gamma(z)$

fi

else

$w := 3$ --- $ctx = \Gamma(z)$

fi

A Type System

- Fixed label assignment Γ
- Goal:
 - Type correctness implies Flow-Label invariant will hold throughout executions.
$$v \rightarrow w \implies \Gamma(v) \sqsubseteq \Gamma(w)$$
 - Flow-Label invariant implies RNI will hold throughout executions.

Type Systems: Big Picture

“Program S is type correct” is a theorem in a logic (say) secl .

- Logic is decidable.
 - Compiler’s type checker “proves” these theorems.
- Logic is sound with respect to:
 - “Program S satisfies FLI invariant”

Formulas of `secL`

Formulas of `secL` are called judgements.

Formulas of `secL` are given as sequents:

- $\Gamma, ctx \vdash Expr: \lambda$ for expression $Expr$, label λ
- $\Gamma, ctx \vdash S$ for statement S

Inference rules give premises and conclusion

$$\frac{P_1, P_2, \dots, P_n}{C}$$

Rules for Expressions

- Constant:
$$\frac{}{\Gamma, \text{ctx} \vdash n : L}$$
- Variable:
$$\frac{\Gamma(x) = \lambda}{\Gamma, \text{ctx} \vdash x : \lambda}$$
- Expression:
$$\frac{\Gamma, \text{ctx} \vdash e : \lambda \quad \Gamma, \text{ctx} \vdash e' : \lambda'}{\Gamma, \text{ctx} \vdash e + e' : \lambda \sqcup \lambda'}$$

A Proof

(1/3)

Given $\Gamma(x) = L$ and $\Gamma(y) = H$:

$$\frac{\Gamma(x) = L}{\Gamma, \text{ctx} \vdash x : L}$$

A Proof

(2/3)

Given $\Gamma(x) = L$ and $\Gamma(y) = H$:

$$\frac{\Gamma(x) = L}{\Gamma, \text{ctx} \vdash x : L} \quad \frac{\Gamma(y) = H}{\Gamma, \text{ctx} \vdash y : H}$$

A Proof

(3/3)

Given $\Gamma(x) = L$ and $\Gamma(y) = H$:

$$\frac{\frac{\Gamma(x) = L}{\Gamma, \text{ctx} \vdash x : L} \quad \frac{\Gamma(y) = H}{\Gamma, \text{ctx} \vdash y : H}}{\Gamma, \text{ctx} \vdash x + y : L \sqcup H}$$

Conclusion: $x+y : H$ (since $L \sqcup H = H$)

Assignment Rule

$x := E$

- causes: $E \rightarrow x$
- requires: $\Gamma(E) \sqsubseteq \Gamma(x)$

$$\text{Assign: } \frac{\Gamma, \text{ctx} \vdash E : \lambda, \lambda \sqcup \text{ctx} \sqsubseteq \Gamma(x)}{\Gamma, \text{ctx} \vdash x := E}$$

if Rule

$$\frac{\Gamma, \text{ctx} \vdash e : \lambda, \quad \Gamma, \lambda \sqcup \text{ctx} \vdash C_1, \quad \Gamma, \lambda \sqcup \text{ctx} \vdash C_2}{\Gamma, \text{ctx} \vdash \mathbf{if\ e\ then\ } C_1 \mathbf{\ else\ } C_2 \mathbf{\ fi}}$$

if Rule Example Proof

1. Constant:

$$\frac{}{\Gamma, (L \sqcup \text{ctx}) \vdash 1:L}$$

2. Assign:

$$\frac{\Gamma, (L \sqcup \text{ctx}) \vdash 1:L, \quad L \sqcup (L \sqcup \text{ctx}) \sqsubseteq \Gamma(x)}{\Gamma, (L \sqcup \text{ctx}) \vdash x:=1}$$

3. if

$$\frac{\Gamma, \text{ctx} \vdash y>0 : L \quad \Gamma, L \sqcup \text{ctx} \vdash x:=1 \quad \Gamma, L \sqcup \text{ctx} \vdash x:=2}{\Gamma, \text{ctx} \vdash \text{if } y>0 \text{ then } x:=1 \text{ else } x:=2 \text{ fi}}$$

while Rule

while:

$$\frac{\Gamma, \text{ctx} \vdash e : \lambda \quad \Gamma, \lambda \sqcup \text{ctx} \vdash C}{\Gamma, \text{ctx} \vdash \mathbf{while\ e\ do\ c\ end}}$$

; (sequence) rule

$$\frac{\Gamma, \text{ctx} \vdash C_1, \quad \Gamma, \text{ctx} \vdash C_2}{\Gamma, \text{ctx} \vdash C_1 ; C_2}$$

secL Type System Retrospective

- Soundness

- Type correct programs satisfy
 - Flow-Label Invariant: $v \rightarrow w \Rightarrow \Gamma(v) \sqsubseteq \Gamma(w)$
 - Relational non-interference (RNI)
- If program doesn't satisfy
 - Flow-Label invariant or
 - RNI

then program won't be type correct.

secL Type System Retrospective

- (in)Completeness
 - The type system is incomplete.
 - If a program is not type correct then that program might still satisfy Flow-Label invariant and RNI.

$$\frac{\Gamma, \text{ctx} \vdash y * 0 : H, \quad H \sqcup L \sqsubseteq \Gamma(x)}{\Gamma, L \vdash x := y * 0}$$

If $\Gamma(x) = L \dots$

- Type checking fails

secL Type System Retrospective

- (in)Completeness
 - The type system is incomplete.
 - If a program is not type correct then that program might still satisfy Flow-Label invariant and RNI.

$$\frac{\Gamma, \text{ctx} \vdash y * 0 : H, \quad H \sqcup L \sqsubseteq \Gamma(x)}{\Gamma, L \vdash x := y * 0}$$

If $\Gamma(x) = L \dots$

- Type checking fails
- FLI invariant valid
- RNI satisfied.

Eliminate Incompleteness?

Sequence rule

$$\frac{\Gamma, \text{ctx} \vdash C_1, \quad \Gamma, \text{ctx} \vdash C_2}{\Gamma, \text{ctx} \vdash C_1 ; C_2}$$

Consider:

if $h > 0$ **then** C ; $v_L := 2$ **else skip fi**

- Satisfies RNI (=termination insensitive) if C diverges.
- Sequence rule must predict that C_1 diverges.
 - Predicting divergence requires solving the halting problem.

Program with Termination Channel

while $v_H > 0$ **do skip end;** $v_L := 2$

- Program is secL type correct.
- Program satisfies RNI.
- Program does not satisfy termination sensitive non interference (TSNI): $v_H \rightarrow \perp$

Type system for TSNI

Prevent channel arising from infinite loops.

- Allow only L terms in **while** guards.
 - Loop termination does not depend of H values.

$$\frac{\Gamma, \text{ctx} \vdash e:L \quad \Gamma, \text{ctx} \vdash C}{\Gamma, \text{ctx} \vdash \mathbf{while\ e\ do\ C\ end}}$$

- Type correct programs now exhibit TSNI.
- What about loops involving H terms?