CS 5430:

Information Flow

Part I: Static Enforcement

(rev Fall 2023)

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: $x := y;$ … $z := x$ Access control with:

- containers: value in y can be leaked by reading z
- values: restrictions on z include all restrictions on y

  … no need to trust clients who access y.

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

x := y mod 2

x := y * 0

z := y+2; x:= z

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

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

x := y mod 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

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

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

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

Toy Language

e ::= x | n | e1+e2 | …

c ::= x := e

| **if** e **then** c1 **else** c2 **fi**

| **while** e **do** c **end**

| c1; c2

$\Gamma(x)$: label associated with variable x

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:

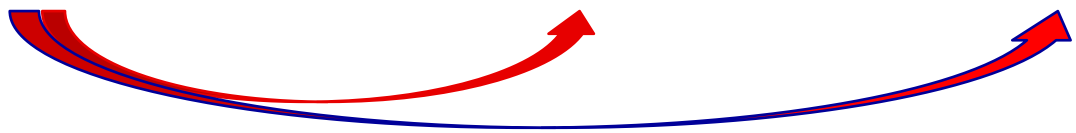
If Statements

$$\textbf{if}\ \ y > 0\ \ \textbf{then}\ \ x \coloneqq 1\ \ \textbf{else}\ \ x \coloneqq 2\ \ \textbf{fi}$$

Restrictions for:

If Statements

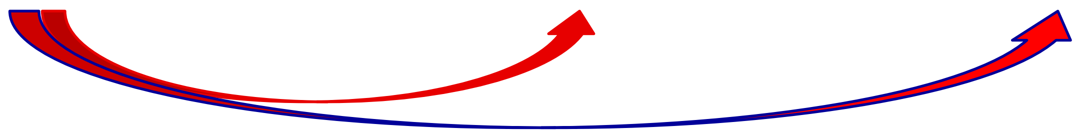
$$\mathbf{if}\ \ y > 0\ \ \mathbf{then}\ \ x := 1\ \ \mathbf{else}\ \ x := 2\ \ \mathbf{fi}$$



Restrictions for:

If Statements

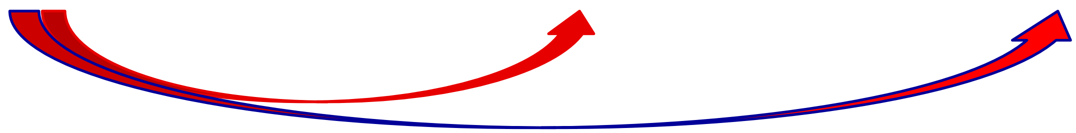
if $y > 0$ **then** $x := 1$ **else** $x := 2$ **fi**



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

Restrictions for:
If Statements

$\textbf{if}\ \ y > 0\ \ \textbf{then}\ \ x := 1\ \ \textbf{else}\ \ x := 2\ \ \textbf{fi}$



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

$y > 0 \rightarrow x \quad \text{requires} \quad \Gamma(y > 0) \sqsubseteq \Gamma(x)$

Restrictions for:
If Statements

$$\textbf{if} \;\; y > 0 \;\; \textbf{then} \;\; x := 1 \;\; \textbf{else} \;\; x := 2 \;\; \textbf{fi}$$

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

Restrictions for:

If Statements

$$\textbf{if} \;\; \mathrm{B} \;\; \textbf{then} \;\; \mathrm{x} \coloneqq \mathrm{E} \;\; \textbf{else} \;\; \ldots \;\; \textbf{fi}$$

$$B \rightarrow x, \qquad E \rightarrow x$$

Restrictions for:

If Statements

$$\textbf{if}\ \ \mathrm{B}\ \ \textbf{then}\ \ \mathrm{x} \coloneqq \mathrm{E}\ \ \textbf{else}\ \ \ldots\ \ \textbf{fi}$$

$$B \rightarrow x, \qquad E \rightarrow x$$

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

Restrictions for:

If Statements

$$\mathbf{if}\ \mathrm{B}\ \ \mathbf{then}\ \mathrm{x} \coloneqq \mathrm{E}\ \mathbf{else}\ \ \dots\ \ \mathbf{fi}$$

$$B \rightarrow x, \qquad E \rightarrow x$$

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

implied by:

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

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

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
```

Restrictions for: Nested If Statements

```
if z>0
        then  y := 23
              if y> 0
                   then  x:= 1 --- ctx = Γ(y)
                   else  u:= 2 --- ctx = Γ(y)
              fi
        else
              w:=3
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 Noninterference 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 nonintereference"

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

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

Rules for Expressions

- Constant: $$\frac{}{\Gamma, \text{ctx} \vdash \text{n} : \text{L}}$$

- Variable: $$\frac{\Gamma(\text{x})=\lambda}{\Gamma, \text{ctx} \vdash \text{x} : \lambda}$$

- Expression: $$\frac{\Gamma, \text{ctx} \vdash \text{e} : \lambda \quad \Gamma, \text{ctx} \vdash \text{e}' : \lambda'}{\Gamma, \text{ctx} \vdash \text{e}+\text{e}' : \lambda \sqcup \lambda'}$$

A Proof (1/3)

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

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

A Proof (2/3)

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

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

A Proof (3/3)

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

$$\dfrac{\dfrac{\Gamma(x) = L}{\Gamma, ctx \vdash x : L} \qquad \dfrac{\Gamma(y) = H}{\Gamma, ctx \vdash y : H}}{\Gamma, ctx \vdash x + y : L \sqcup H}$$

Conclusion: $x{+}y : H \quad (\text{since } L \sqcup H = H)$

skip Rule

skip

- Does nothing
- Changes nothing

skip: $$\frac{}{\Gamma, \text{ctx} \vdash \textbf{skip}}$$

Assignment Rule

$x := E$

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

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

if Rule

$$\frac{\Gamma, \mathrm{ctx} \vdash \mathrm{e} : \lambda, \quad \Gamma, \lambda \sqcup \mathrm{ctx} \vdash \mathrm{C}_1, \quad \Gamma, \lambda \sqcup \mathrm{ctx} \vdash \mathrm{C}_2}{\Gamma, \mathrm{ctx} \vdash \mathbf{if}\ \mathrm{e}\ \mathbf{then}\ \mathrm{C}_1\ \mathbf{else}\ \mathrm{C}_2\ \mathbf{fi}}$$

if Rule Example Proof

1. Constant:

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

2. Assign:

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

3. if

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

while Rule

while:

$$\frac{\Gamma, \mathrm{ctx} \ \vdash \mathrm{e}: \lambda \qquad \Gamma, \lambda \sqcup \mathrm{ctx} \vdash \mathrm{C}}{\Gamma, \mathrm{ctx} \ \vdash \ \textbf{while}\ \mathrm{e}\ \ \textbf{do}\ \ \mathrm{c}\ \ \textbf{end}}$$

; (sequence) rule

$$\frac{\Gamma, \mathrm{ctx} \vdash C_1, \qquad \Gamma, \mathrm{ctx} \vdash C_2}{\Gamma, \mathrm{ctx} \vdash C_1\ ; C_2}$$

secL Type System Retrospective

- Soundness
  - Type correct programs satisfy
    - $v \rightarrow w \implies \Gamma(v) \sqsubseteq \Gamma(w)$
    - Termination insensitive noninterference (TINI)
  - If program doesn't satisfy TINI 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 TINI.

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

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

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

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

- Type checking fails
- TINI satisfied.

Eliminate Incompleteness?

Sequence rule

$$\frac{\Gamma, \text{ctx} \vdash \mathrm{C}_1, \qquad \Gamma, \text{ctx} \vdash \mathrm{C}_2}{\Gamma, \text{ctx} \vdash \mathrm{C}_1 \;;\; \mathrm{C}_2}$$

Consider:

$$\textbf{if } \mathrm{h} > 0 \textbf{ then } C;\ \mathrm{v_L} := 2 \textbf{ else skip fi}$$

- Satisfies TINI if C diverges.
- Sequence rule must predict that C_1 diverges.
  - Predicting divergence requires solving the halting problem.

Program with Termination Channel

$$\textbf{while } v_H > 0 \textbf{ do skip end};\ \ v_L := 2$$

- Program is secL type correct.
- Program satisfies TINI.
- 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.

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

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