A Sound Type System for Secure Flow Analysis

Dennis Volpana, Geoffrey Smith, Cynthia Irvine

Jason Belt CIS 890

09.20.2010

Secure Information Flow Analysis

- Static analysis is used to ensure sensitive information is not leaked
- Define a lattice of security levels and prove information only flows upwards
 e.g. if L ≤ H then L ~ L, H ~ H, L ~ H, H ≁ L

Secure Information Flow Analysis

- Static analysis is used to ensure sensitive information is not leaked
- Define a lattice of security levels and prove information only flows upwards
 e.g. if L ≤ H then L ~ L, H ~ H, L ~ H, H ≁ L

Definition (Noninterference) Program c satisifies noninterference if, for any memories μ and v that agree on L variables, the memories produced by running c on μ and on v also agree on L variables (provided both runs terminate successfully)

Type-Based Approach

- Security levels \approx Types
- Lattice order on security levels \approx Subtyping
- Program certification \approx Type checking

Type-Based Approach

- Security levels \approx Types
- Lattice order on security levels \approx Subtyping
- Program certification \approx Type checking

welltyped(P) \Rightarrow noninterference(P)

The Core Language

Phrases $p ::= e \mid c$

Expressions $e ::= \begin{array}{c} x \mid l \mid n \mid e + e' \mid e - e' \mid \\ e = e' \mid e < e' \end{array}$

Commands c ::= $e := e' \mid c; c' \mid if e then c else c' \mid$ while $e do c \mid letvar x := e in c$

The Core Language

Phrases $p ::= e \mid c$

Expressions $e ::= \begin{array}{c} x \mid l \mid n \mid e + e' \mid e - e' \mid \\ e = e' \mid e < e' \end{array}$

Commands $c ::= \begin{array}{c} e := e' \mid c; c' \mid \text{if } e \text{ then } c \text{ else } c' \mid \\ \text{while } e \text{ do } c \mid \text{letvar } x := e \text{ in } c \end{array}$

Security classes $s \in SC$ (partially ordered by \leq)

Type τ ::= s

Phrase types ρ ::= $\tau \mid \tau \ var \mid \tau \ cmd$

Typing Judgements

 $\lambda;\gamma \vdash p:\rho$

Typing Judgements

$\lambda;\gamma \vdash p:\rho$

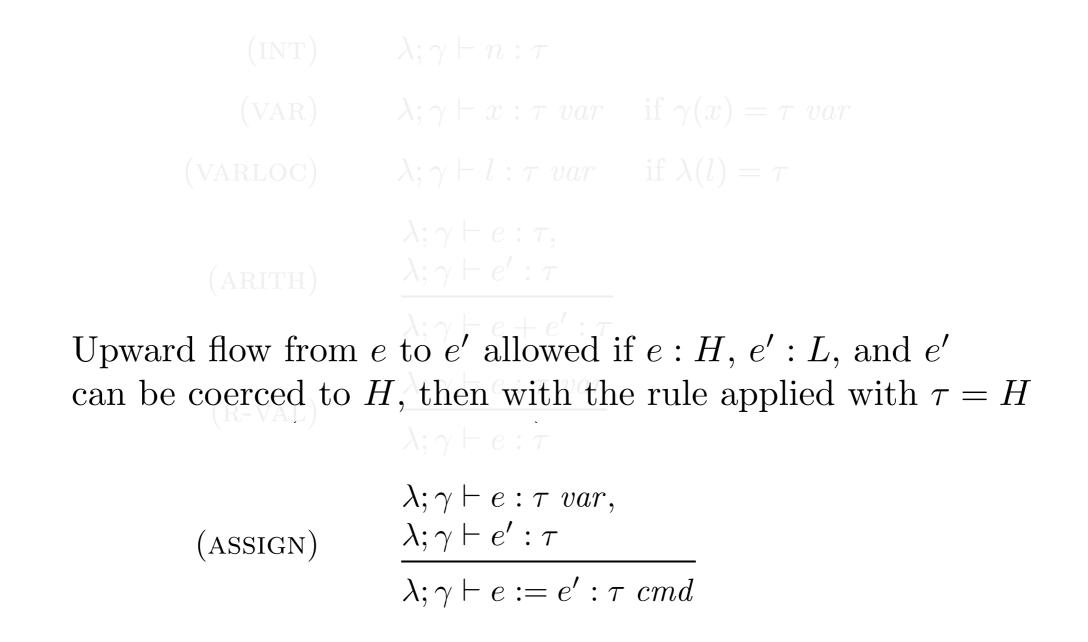
• $\lambda: l \to \tau$ location typing

Typing Judgements

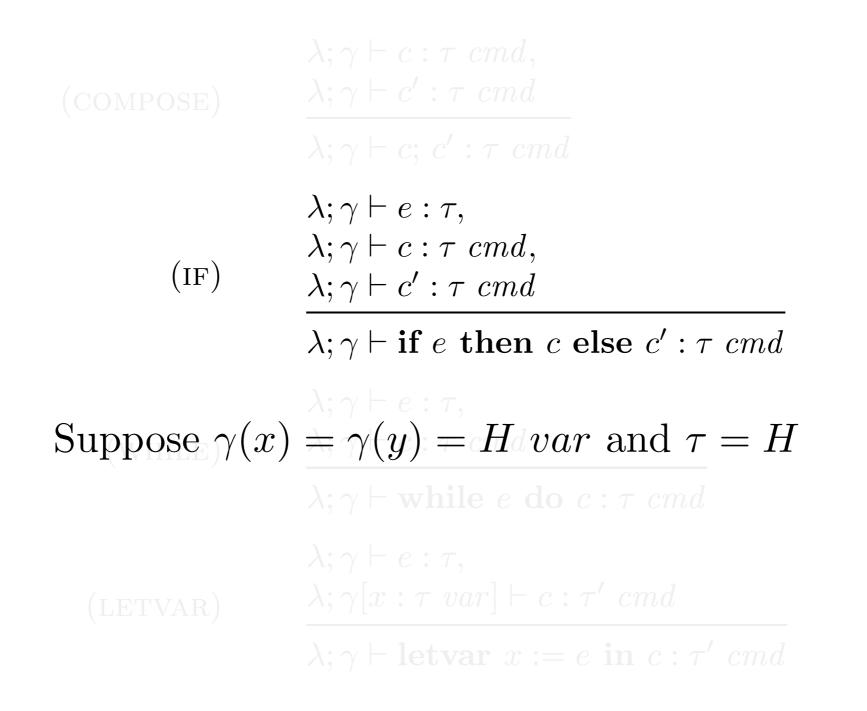
$\lambda;\gamma \vdash p:\rho$

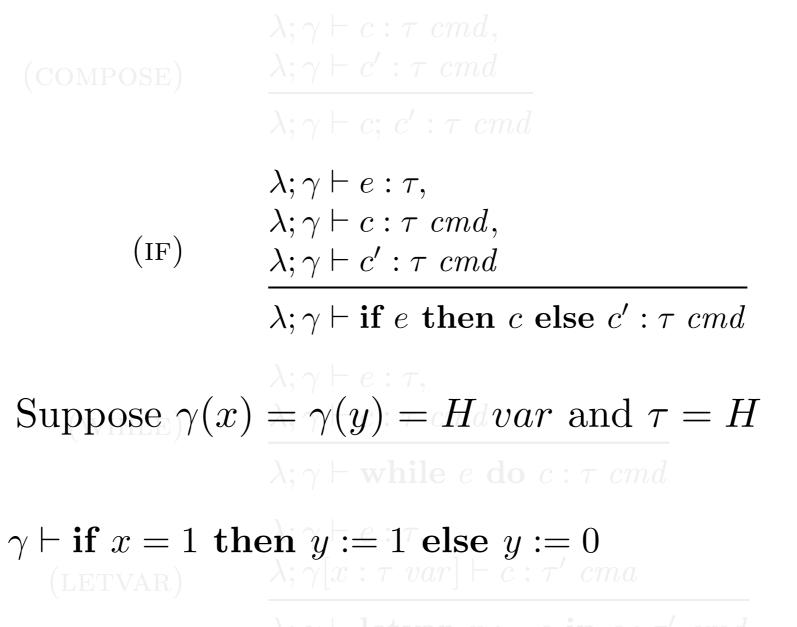
- $\lambda: l \to \tau$ location typing
- $\gamma: x \to \rho$ identifier typing

(INT)	$\lambda;\gamma \vdash n:\tau$	
(VAR)	$\lambda; \gamma \vdash x : \tau \ var$	if $\gamma(x) = \tau \ var$
(VARLOC)	$\lambda; \gamma \vdash l : \tau \ var$	if $\lambda(l) = \tau$
(ARITH)	$\begin{aligned} \lambda; \gamma \vdash e : \tau, \\ \lambda; \gamma \vdash e' : \tau \\ \overline{\lambda}; \gamma \vdash e + e' : \tau \end{aligned}$	
(R-VAL)	$\frac{\lambda; \gamma \vdash e : \tau \ var}{\lambda; \gamma \vdash e : \tau}$	
(ASSIGN)	$egin{aligned} \lambda; \gamma dash e : au \; var, \ \lambda; \gamma dash e' : au \ \lambda; \gamma dash e' : au \ \lambda; \gamma dash e := e' : au \end{aligned}$	cmd

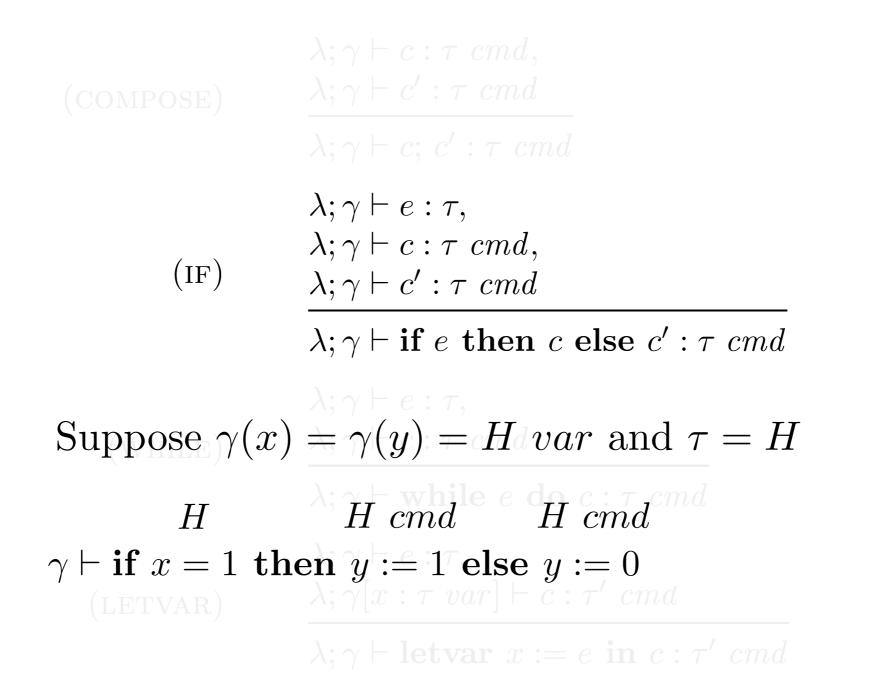


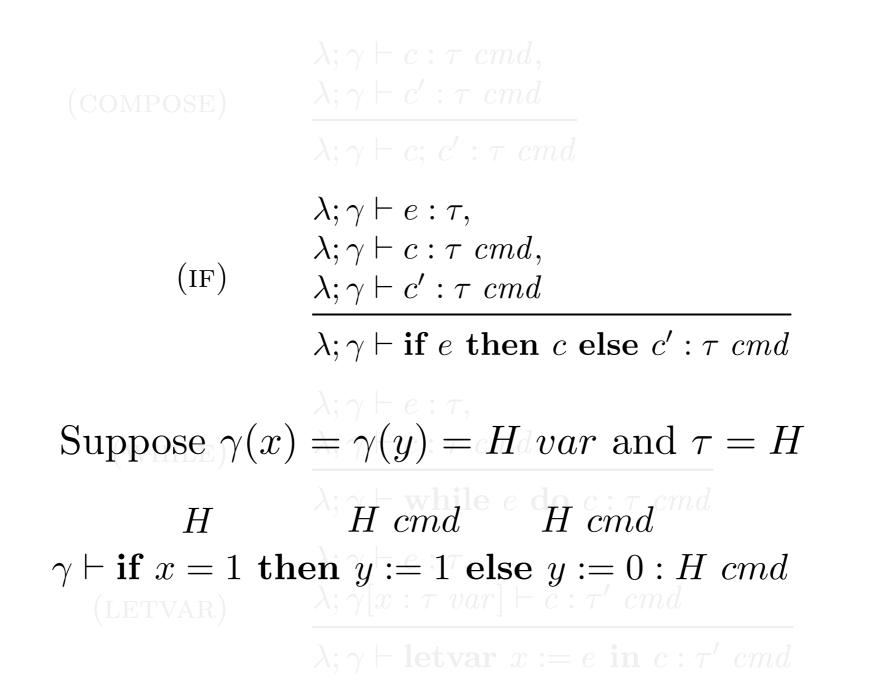
(COMPOSE)	$\begin{aligned} \lambda; \gamma \vdash c : \tau \ cmd, \\ \lambda; \gamma \vdash c' : \tau \ cmd \\ \overline{\lambda; \gamma \vdash c; \ c' : \tau \ cmd} \end{aligned}$
(IF)	$\begin{aligned} \lambda; \gamma \vdash e : \tau, \\ \lambda; \gamma \vdash c : \tau \ cmd, \\ \lambda; \gamma \vdash c' : \tau \ cmd \end{aligned}$
	$\lambda; \gamma \vdash \mathbf{if} \ e \ \mathbf{then} \ c \ \mathbf{else} \ c' : \tau \ cmd$ $\lambda; \gamma \vdash e : \tau,$
(WHILE)	$\frac{\lambda; \gamma \vdash c : \tau,}{\lambda; \gamma \vdash c : \tau \ cmd}$
	$\lambda; \gamma \vdash \mathbf{while} \ e \ \mathbf{do} \ c: \tau \ cmd$
(LETVAR)	$\begin{array}{l} \lambda; \gamma \vdash e : \tau, \\ \lambda; \gamma[x : \tau \ var] \vdash c : \tau' \ cmd \end{array}$
	$\lambda; \gamma \vdash $ letvar $x := e $ in $c : \tau' $ cmd





 $\lambda; \gamma \vdash \mathbf{letvar} \ x := e \ \mathbf{in} \ c : au' \ cma$





(COMPOSE)

 $\lambda; \gamma \vdash c : \tau \ cmd, \\ \lambda; \gamma \vdash c' : \tau \ cmd \\ \overline{\lambda}; \gamma \vdash c; \ c' : \tau \ cmd \\ \lambda; \gamma \vdash c; \ c' : \tau \ cmd \\ \lambda; \gamma \vdash c; \ c' : \tau \ cmd \\ \lambda; \gamma \vdash c; \ c' : \tau \ cmd \\ \lambda; \gamma \vdash c; \ c' : \tau \ cmd \\ \lambda; \gamma \vdash c; \ c' : \tau \ cmd \\ \lambda; \gamma \vdash c; \ c' : \tau \ cmd \\ \lambda; \gamma \vdash c; \ c' : \tau \ cmd \\ \lambda; \gamma \vdash c; \ c' : \tau \ cmd \\ \lambda; \gamma \vdash c; \ c' : \tau \ cmd \\ \lambda; \gamma \vdash c; \ c' : \tau \ cmd \\ \lambda; \gamma \vdash c; \ c' : \tau \ cmd \\ \lambda; \gamma \vdash c; \ c' : \tau \ cmd \\ \lambda; \gamma \vdash c; \ c' : \tau \ cmd \\ \lambda; \gamma \vdash c; \ c' : \tau \ cmd \\ \lambda; \gamma \vdash c; \ c' : \tau \ cmd \\ \lambda; \gamma \vdash c; \ c' : \tau \ cmd \\ \lambda; \tau$

 $\begin{array}{ll} \lambda; \gamma \vdash e : \tau, \\ \lambda; \gamma \vdash c : \tau \ cmd, \\ \lambda; \gamma \vdash c' : \tau \ cmd \end{array}$

 $\lambda; \gamma \vdash \mathbf{if} \ e \ \mathbf{then} \ c \ \mathbf{else} \ c' : \tau \ cmd$

Suppose $\gamma(x) = L \ var, \ \gamma(y) = H \ var, \ \tau = L$, and $L \leq H$ so that $H \ cmd \subseteq L \ cmd$

$$\gamma \vdash \mathbf{if} \ x = 1 \ \mathbf{then} \ y := 1 \ \mathbf{else} \ y := 0$$

 $\lambda; \gamma \vdash e : \tau,$ $\lambda; \gamma \vdash c : \tau \ cmd$, (IF) $\lambda; \gamma \vdash c' : \tau \ cmd$ $\lambda; \gamma \vdash \mathbf{if} \ e \ \mathbf{then} \ c \ \mathbf{else} \ c' : \tau \ cmd$ Suppose $\gamma(x) = L \ var, \ \gamma(y) = H \ var, \ \tau = L$, and $L \leq H$ so that $H \ cmd \subseteq L \ cmd$ L $\gamma \vdash \mathbf{if} \ x = 1 \ \mathbf{then} \ y := 1 \ \mathbf{else} \ y := 0$

 $\lambda; \gamma \vdash \mathbf{letvar} \ x := e \ \mathbf{in} \ c : \tau' \ cmd$

 $\lambda; \gamma \vdash e : \tau,$ $\lambda; \gamma \vdash c : \tau \ cmd$, (IF) $\lambda; \gamma \vdash c' : \tau \ cmd$ $\lambda; \gamma \vdash \mathbf{if} \ e \ \mathbf{then} \ c \ \mathbf{else} \ c' : \tau \ cmd$ Suppose $\gamma(x) = L \ var, \ \gamma(y) = H \ var, \ \tau = L$, and $L \leq H$ so that $H \ cmd \subseteq L \ cmd$ $L \qquad H \ cmd \sim L \ cmd$ $\gamma \vdash \mathbf{if} \ x = 1 \ \mathbf{then} \ y := 1 \ \mathbf{else} \ y := 0$

 $\lambda; \gamma \vdash$ **letvar** x := e **in** $c : \tau'$ cmd

(IF) $\lambda; \gamma \vdash c : \tau \ cmd,$ $\lambda; \gamma \vdash c' : \tau \ cmd$ $\lambda; \gamma \vdash c; c' : \tau \ cmd$ $\lambda; \gamma \vdash e : \tau,$ $\lambda; \gamma \vdash c : \tau \ cmd,$ $\lambda; \gamma \vdash c' : \tau \ cmd$

 $\lambda; \gamma \vdash \mathbf{if} \ e \ \mathbf{then} \ c \ \mathbf{else} \ c' : \tau \ cmd$

Suppose $\gamma(x) = L \ var, \ \gamma(y) = H \ var, \ \tau = L$, and $L \leq H$ so that $H \ cmd \subseteq L \ cmd$

 $\gamma \vdash \mathbf{if} \ x = 1 \ \mathbf{then} \ y := 1 \ \mathbf{else} \ y := 0 : L \ cmd$

 $\lambda; \gamma \vdash$ **letvar** x := e **in** $c : \tau'$ cmd

(COMPOSE)	$\begin{aligned} \lambda; \gamma \vdash c : \tau \ cmd, \\ \lambda; \gamma \vdash c' : \tau \ cmd \\ \overline{\lambda; \gamma \vdash c; \ c' : \tau \ cmd} \end{aligned}$
(IF)	$\begin{aligned} \lambda; \gamma \vdash e : \tau, \\ \lambda; \gamma \vdash c : \tau \ cmd, \\ \lambda; \gamma \vdash c' : \tau \ cmd \end{aligned}$
	$\lambda; \gamma \vdash \mathbf{if} \ e \ \mathbf{then} \ c \ \mathbf{else} \ c' : \tau \ cmd$ $\lambda; \gamma \vdash e : \tau,$
(WHILE)	$\frac{\lambda; \gamma \vdash c : \tau,}{\lambda; \gamma \vdash c : \tau \ cmd}$
	$\lambda; \gamma \vdash \mathbf{while} \ e \ \mathbf{do} \ c: \tau \ cmd$
(LETVAR)	$\begin{array}{l} \lambda; \gamma \vdash e : \tau, \\ \lambda; \gamma[x : \tau \ var] \vdash c : \tau' \ cmd \end{array}$
	$\lambda; \gamma \vdash $ letvar $x := e $ in $c : \tau' $ cmd

Subtyping Rules

(BASE)	$\frac{\tau \leq \tau'}{\vdash \tau \subseteq \tau'}$
(REFLEX)	$\vdash \rho \subseteq \rho$
(TRANS)	$\frac{\vdash \rho \subseteq \rho', \ \vdash \rho' \subseteq \rho''}{\vdash \rho \subseteq \rho''}$
(CMD^{-})	$\frac{\vdash \tau \subseteq \tau'}{\vdash \tau' \ cmd \subseteq \tau \ cmd}$
(SUBTYPE)	$\begin{split} \lambda; \gamma \vdash p : \rho, \\ \vdash \rho \subseteq \rho' \\ \overline{\lambda; \gamma \vdash p : \rho'} \end{split}$

• Evaluation is performed relative to a memory $\mu: l \to n$

• Evaluation is performed relative to a memory $\mu: l \to n$

$$\mu \vdash e \Rightarrow n \qquad \mu \vdash c \Rightarrow \mu'$$

$(BASE) \qquad \mu \vdash n \Rightarrow n$	
(contents) $\mu \vdash l \Rightarrow \mu(l)$	if $l \in dom(\mu)$
(ADD) $\frac{\mu \vdash e \Rightarrow n, \ \mu}{\mu \vdash e + e' \Rightarrow r}$	
(UPDATE) $\frac{\mu \vdash e \Rightarrow n, \ l}{\mu \vdash l := e \Rightarrow \mu}$	
(SEQUENCE) $\frac{\mu \vdash c \Rightarrow \mu', \ \mu}{\mu \vdash c; \ c' \Rightarrow \mu''}$	·
(BRANCH) $\frac{\mu \vdash e \Rightarrow 1, \ \mu}{\mu \vdash \mathbf{if} \ e \ \mathbf{then}}$	$ \vdash c \Rightarrow \mu' $ $c \text{ else } c' \Rightarrow \mu' $
$\frac{\mu \vdash e \Rightarrow 0, \ \mu}{\mu \vdash \mathbf{if} \ e \ \mathbf{then}}$	$ \vdash c' \Rightarrow \mu' $ $c \text{ else } c' \Rightarrow \mu' $

(LOOP)
$$\frac{\mu \vdash e \Rightarrow 0}{\mu \vdash \text{while } e \text{ do } c \Rightarrow \mu} \\
\frac{\mu \vdash e \Rightarrow 1,}{\mu \vdash c \Rightarrow \mu',} \\
\frac{\mu \vdash c \Rightarrow \mu',}{\mu \vdash \text{while } e \text{ do } c \Rightarrow \mu''} \\
\frac{\mu \vdash \text{while } e \text{ do } c \Rightarrow \mu''}{\mu \vdash \text{while } e \text{ do } c \Rightarrow \mu''} \\
\mu \vdash e \Rightarrow n, \\
l \text{ is the first location not in } dom(\mu), \\
\frac{\mu[l := n] \vdash [l/x]c \Rightarrow \mu'}{\mu \vdash \text{letvar } x := e \text{ in } c \Rightarrow \mu' - l}$$

• Altering the initial values of locations of type τ cannot affect the initial values of any locations of type τ' , provided that $\tau \not\leq \tau'$

Simple Security

Lemma 6.3 If $\lambda \vdash e : \tau$, then for every l in $e, \lambda(l) \vdash \tau$

• Secrecy

Only locations at level τ or lower will have their contents read when e is evaluated (no read up)

• Confinement

If e has integrity level $\tau,$ then every location in e stores information at integrity level τ

Confinement

Lemma 6.4 If $\lambda; \gamma \vdash c : \tau \ cmd$, then for every l assigned to in $c, \lambda(l) \geq \tau$

• Secrecy

No location below level τ is updated in c (no write down)

Confinement

Every location assigned to in c can be updated by information at integrity level τ

Theorem 6.8 (Type Soundess) Suppose

(a) $\lambda \vdash c : \rho$,

c is well typed

Theorem 6.8 (Type Soundess) Suppose

 $\begin{array}{ll} (a) & \lambda \vdash c : \rho, \\ (b) & \mu \vdash c \Rightarrow \mu', \end{array}$

c is well typed execution one

Theorem 6.8 (Type Soundess) Suppose

$$\begin{array}{ll} (a) & \lambda \vdash c : \rho, \\ (b) & \mu \vdash c \Rightarrow \mu', \\ (c) & v \vdash c \Rightarrow v', \end{array}$$

c is well typed execution one execution two

Theorem 6.8 (Type Soundess) Suppose

(a) $\lambda \vdash c : \rho$, (b) $\mu \vdash c \Rightarrow \mu'$, (c) $v \vdash c \Rightarrow v'$, (d) $\operatorname{dom}(\mu) = \operatorname{dom}(v) = \operatorname{dom}(\lambda)$, and c is well typed execution one execution two

Theorem 6.8 (Type Soundess) Suppose

- (a) $\lambda \vdash c : \rho$,
- (b) $\mu \vdash c \Rightarrow \mu',$
- (c) $v \vdash c \Rightarrow v'$,
- (d) $\operatorname{\mathsf{dom}}(\mu) = \operatorname{\mathsf{dom}}(v) = \operatorname{\mathsf{dom}}(\lambda)$, and
- (e) $v(l) = \mu(l)$ for all l such that $\lambda(l) \le \tau$

c is well typed execution one execution two

the same low inputs

Theorem 6.8 (Type Soundess) Suppose

(a) $\lambda \vdash c : \rho$, c is well typed(b) $\mu \vdash c \Rightarrow \mu'$, execution one(c) $v \vdash c \Rightarrow v'$, execution two(d) $\operatorname{dom}(\mu) = \operatorname{dom}(v) = \operatorname{dom}(\lambda)$, and (e) $v(l) = \mu(l)$ for all l such that $\lambda(l) \leq \tau$ the same low inputs Then $v'(l) = \mu'(l)$ for all l such that $\lambda(l) \leq \tau$ the same low outputs