A Verified Information-Flow Architecture

Arthur Azevedo de Amorim, Nathan Collins,
André DeHon, Delphine Demange, Cătălin Hriţcu,
David Pichardie, Benjamin C. Pierce,
Randy Pollack, Andrew Tolmach

Crash/
SAFE

Penn
University of Pennsylvania

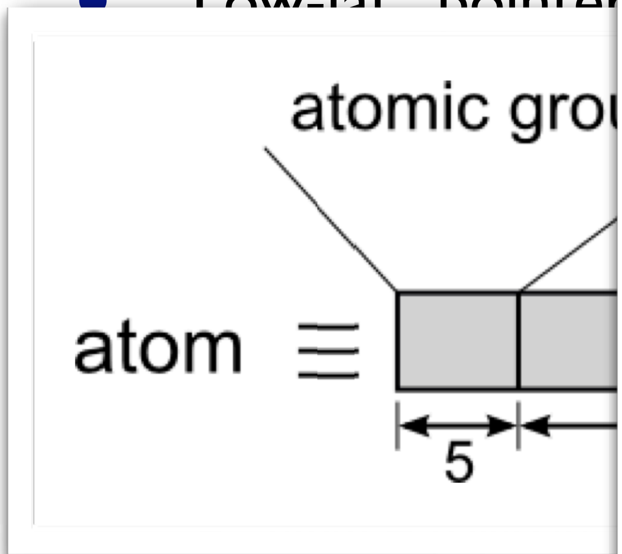
BAE SYSTEMS

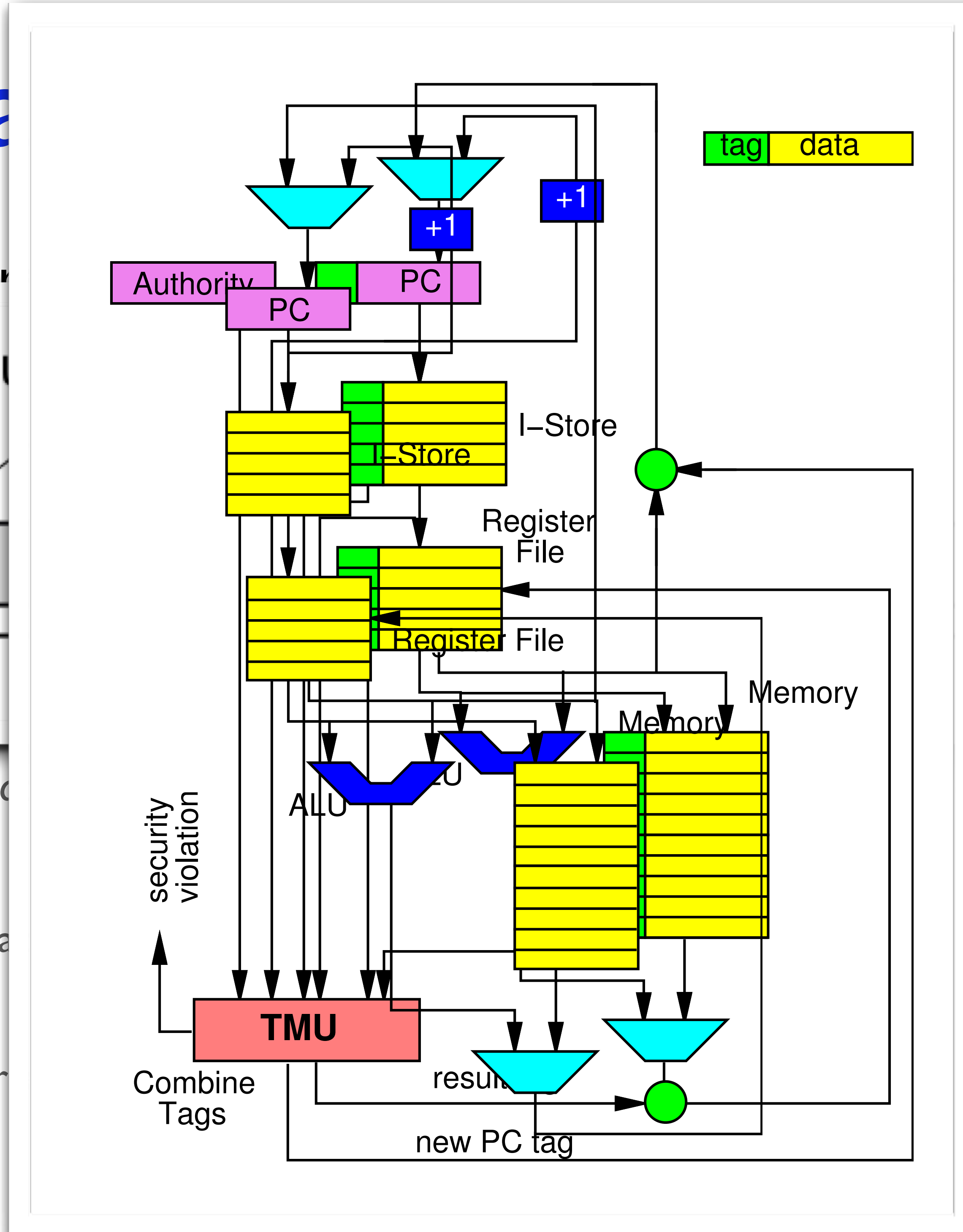
SAFE

- Clean-slate redesign of the entire system stack
  - Hardware
  - System software
  - Programming languages
- Support for critical security invariants at all levels
  - Memory safety
  - Strong dynamic typing
  - Information flow and access control
- Verification of key mechanisms deeply integrated into design process

SAFE: Ha

- “Low-fat” pointer





  - Hardware *rule c*

- And more...
  - Lightweight transa
  - Linear pointers
  - Hardware-suppor

Why new hardware?

- Explore how to spend hardware resources on security effectively
- Reconsider traditional sources of complexity and vulnerability
- Remove application compiler, libraries, etc. from TCB
    - Strong attack model

This work...

Formal Model of SAFE's Hardware Tagging and Low-Level Tag-Management Software

Proof of correctness

Goal for today...

Explain HW/SW architecture;
Sketch proof architecture

Simplifications

Major

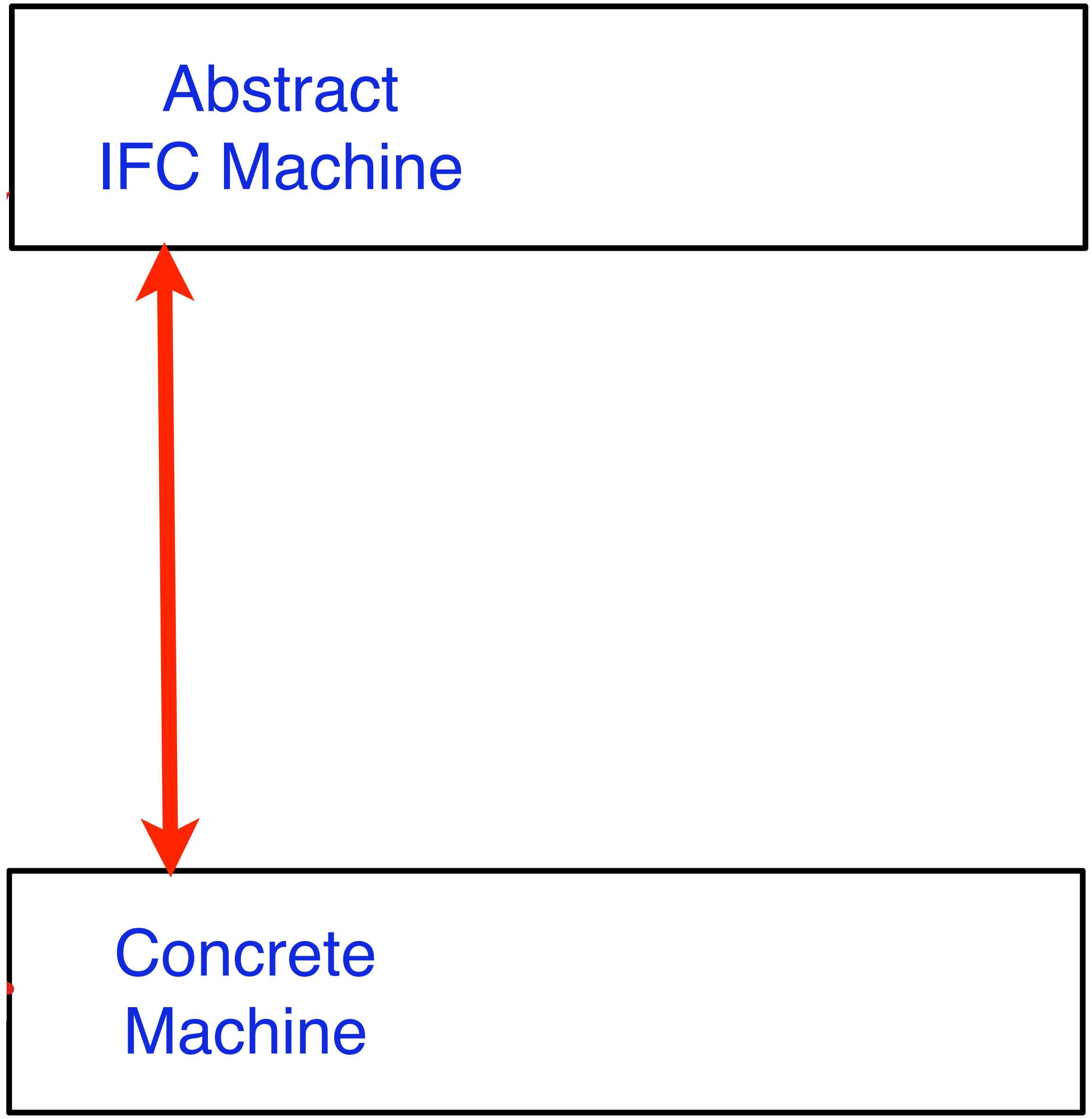
- Deterministic, single-threaded machine

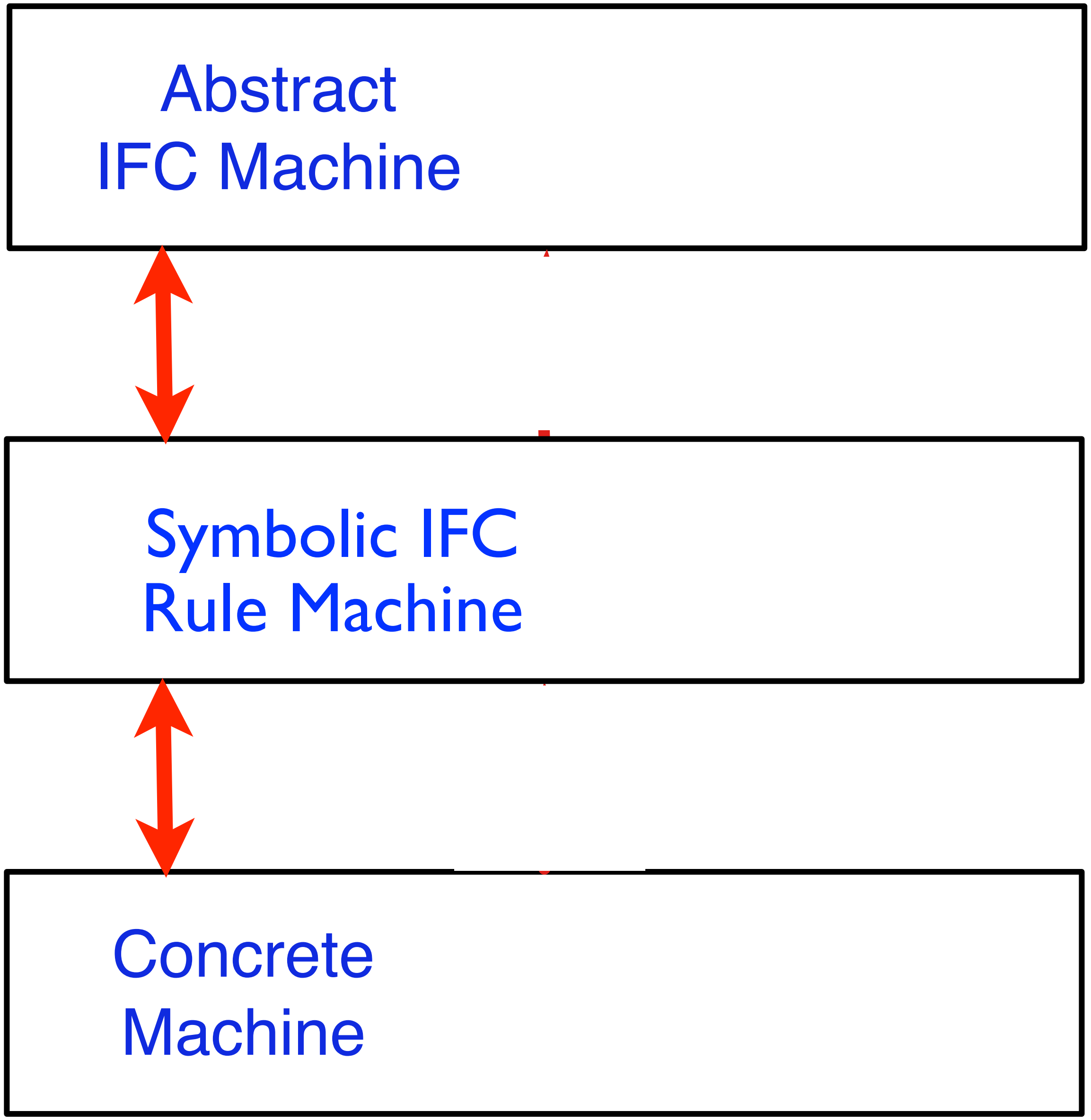
Minor

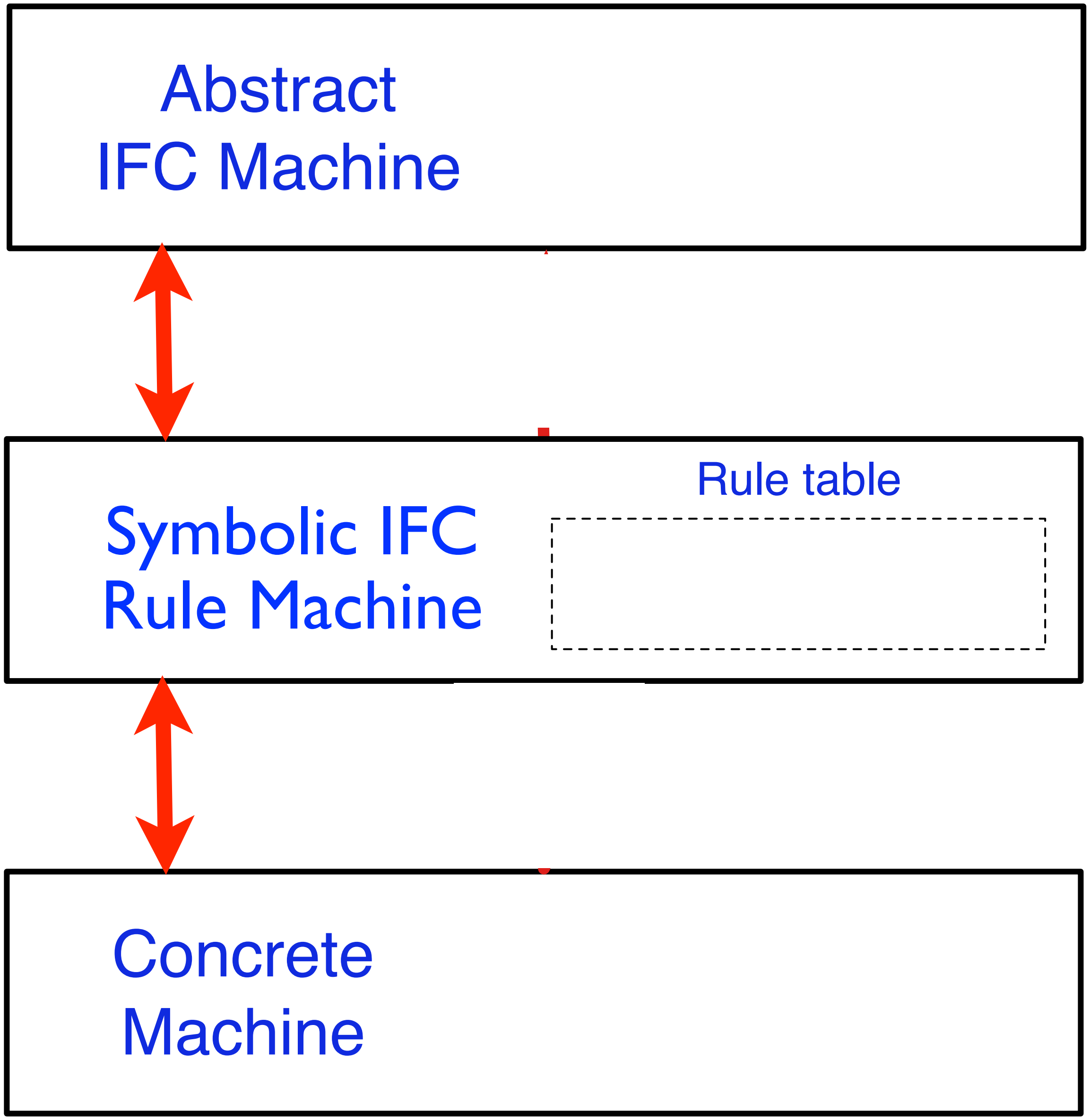
- Conventional memory model
  - pointers are just integers
  - single kernel protection domain
- Stack instead of registers
- No downgrading, public labels, dynamic principal generation, ...
- No exception handling
  - security violation halts the whole machine
- One-line rule cache

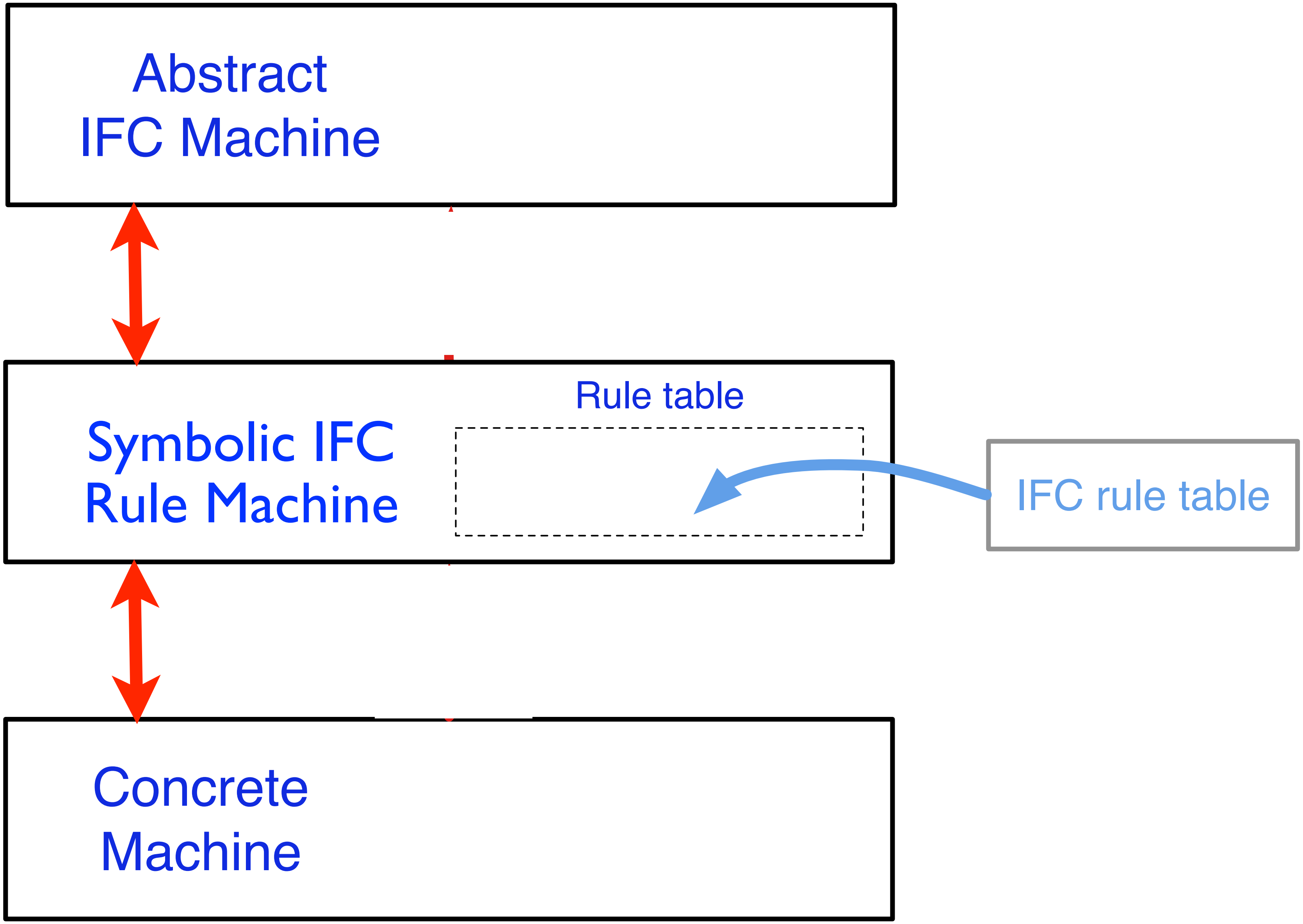
Outline

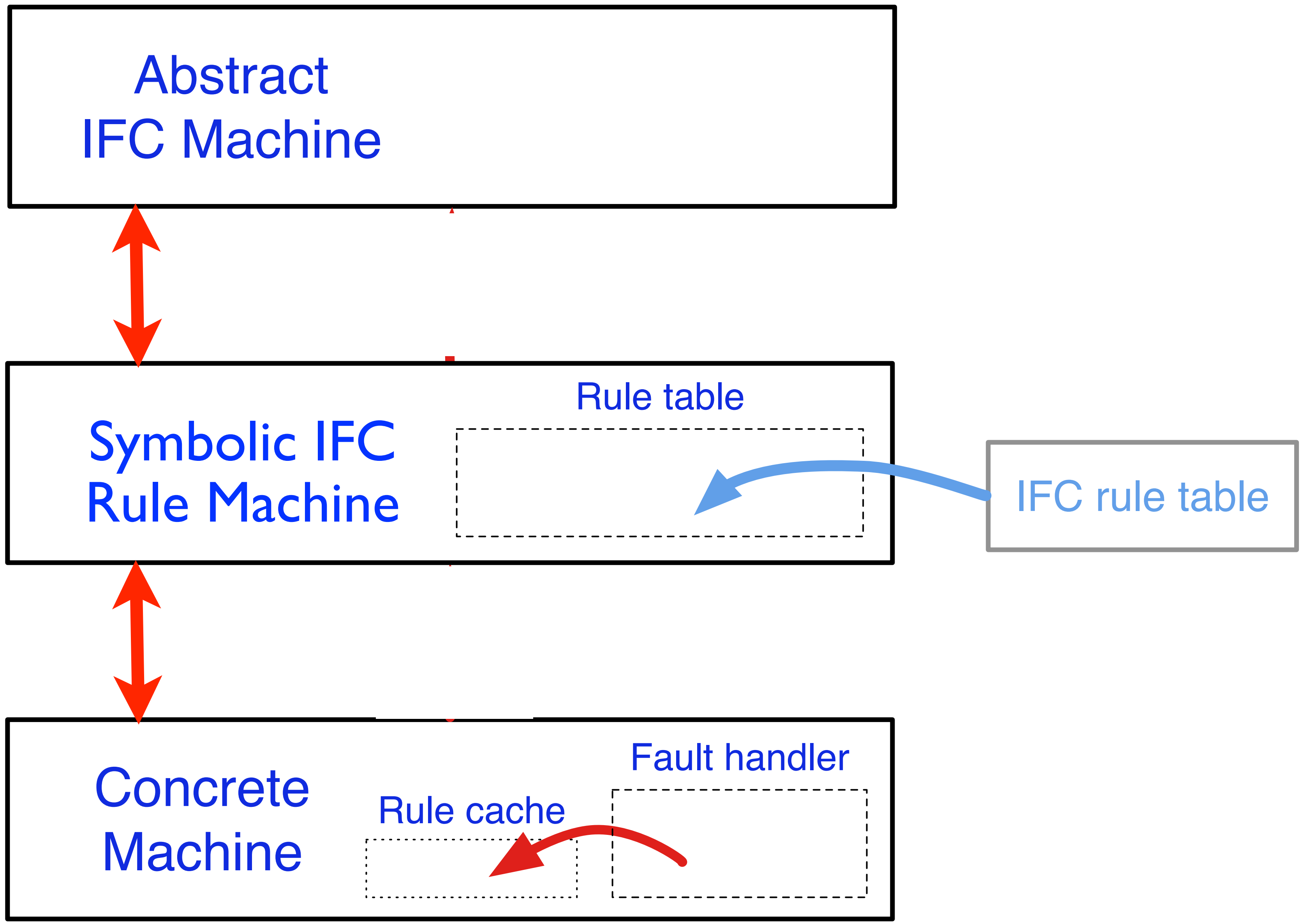
Concrete
Machine


Abstract
IFC Machine
Concrete
Machine

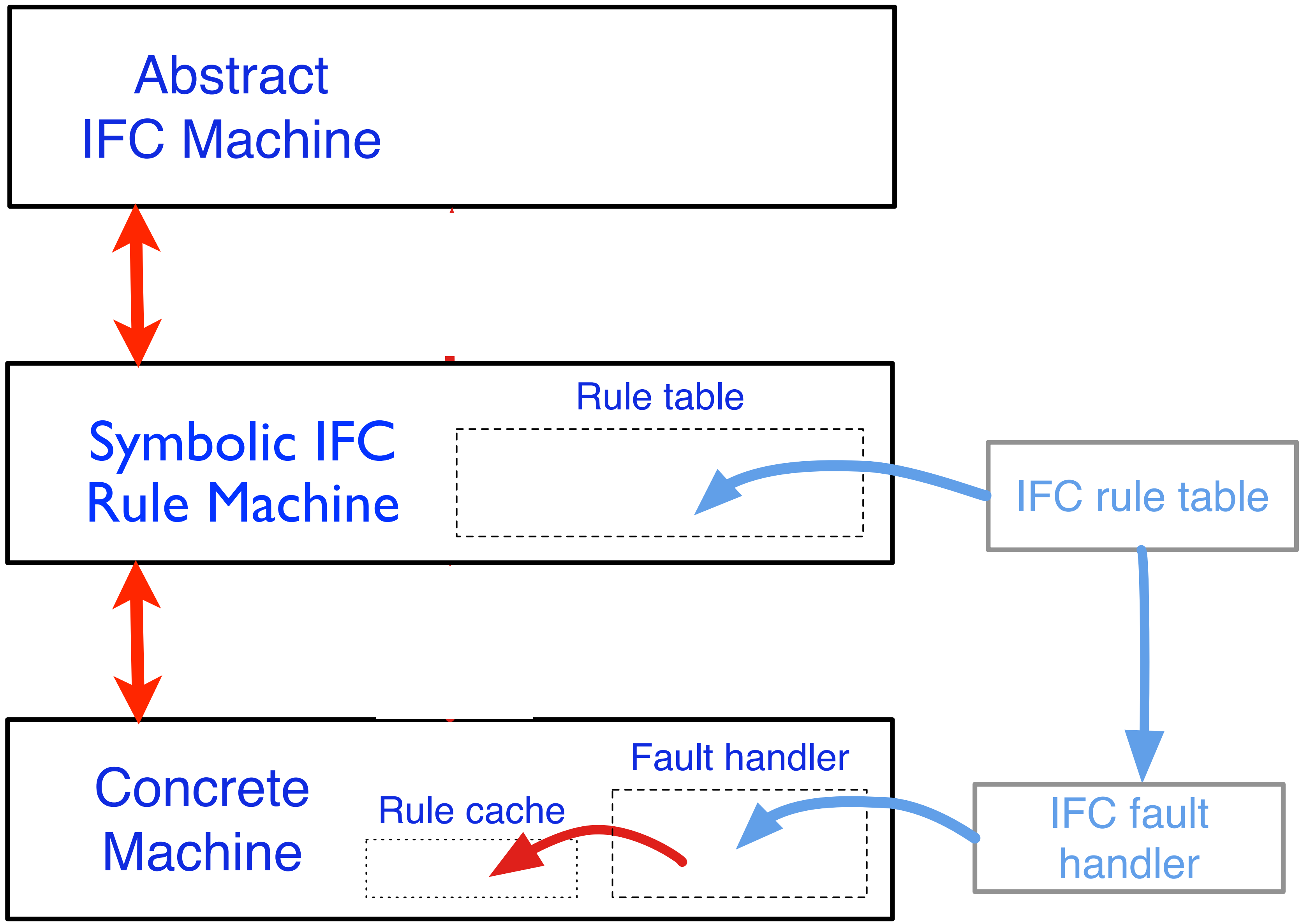

Abstract
IFC Machine
Symbolic IFC
Rule Machine
Concrete
Machine


Abstract
IFC Machine
Rule table
Symbolic IFC
Rule Machine
Concrete
Machine


Abstract
IFC Machine
Symbolic IFC
Rule Machine
Rule table
IFC rule table
Concrete
Machine



Abstract
IFC Machine
Symbolic IFC
Rule Machine
Rule table
IFC rule table
Concrete
Machine
Rule cache
Fault handler


Abstract
IFC Machine
Symbolic IFC
Rule Machine
Rule table
IFC rule table
Concrete
Machine
Rule cache
Fault handler
IFC fault
handler

Abstract Machine

Non-interference

- We design the abstract machine so that it is easy to prove a non-interference property
  - Strictly: termination-insensitive non-interference
  - Over arbitrary semi-lattice of labels, from point of view of arbitrary observer
- Roughly: "high" inputs cannot affect "low" outputs.
  - If two executions of a program start with the same "low" data, the "low" parts of their output traces will be the same

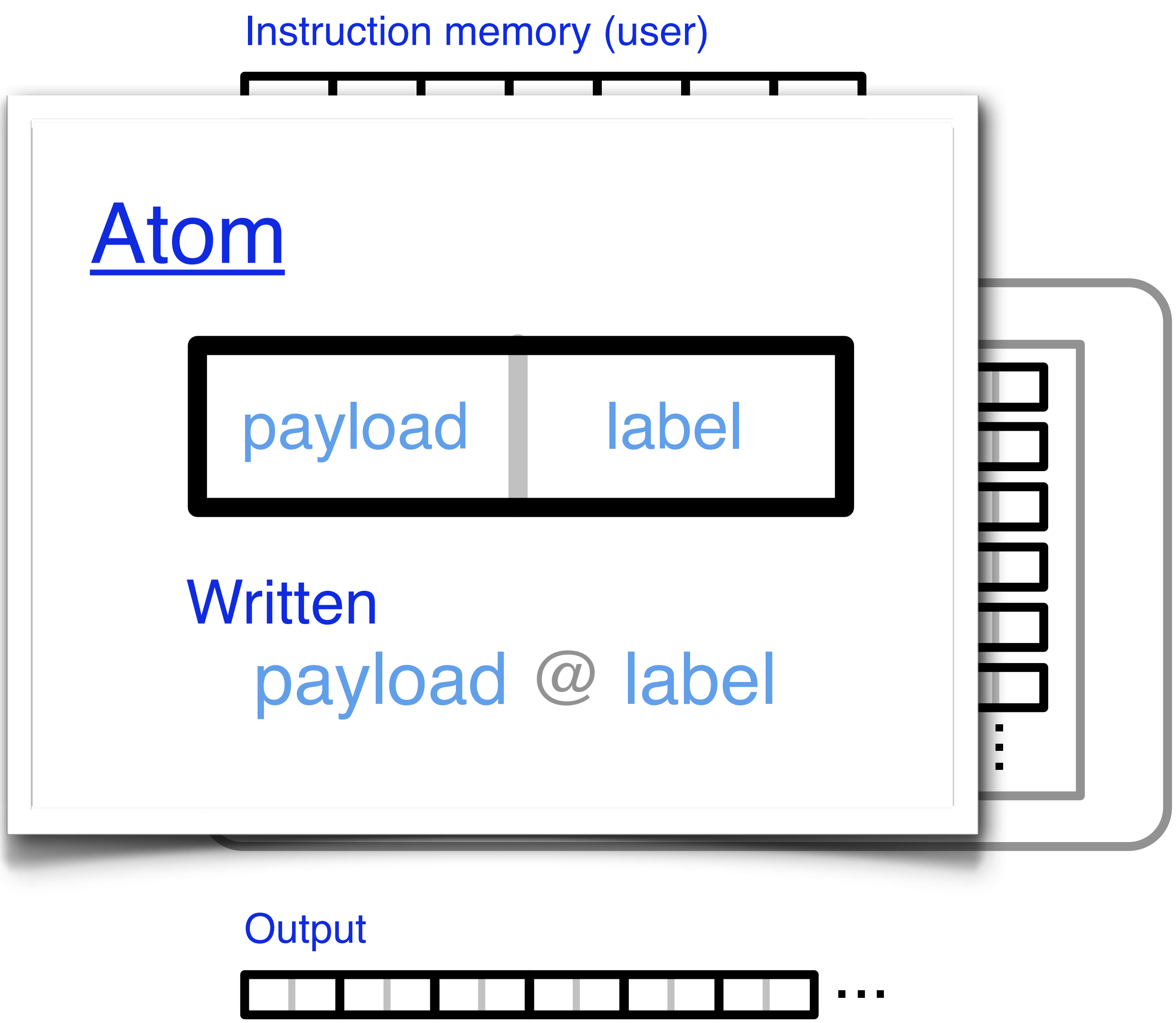
Abstract Machine

Instruction memory (user)

Machine state

instr	::=		Instructions
	|	Output	output top of stack
	|	Sub	subtract
	|	Push n	push constant integer
	|	Load	indirect load from data memory
	|	Store	indirect store to data memory
	|	Jump	unconditional indirect jump
	|	Bnz n	conditional relative jump
	|	Call	indirect call
	|	Ret	return

Abstract Machine



$$\frac{\iota(n) = \textsf{Sub}}{\mu\ [n_1@L_1, n_2@L_2, \sigma]\ n@L_{pc} \xrightarrow{\tau} \mu\ [(n_1 - n_2)@(L_1 \vee L_2), \sigma]\ n{+}1@L_{pc}}$$

output

$$\iota(n) = \textsf{Output}$$

$$\xrightarrow{m@L_1 \vee L_{pc}}$$

$$\mu_1\ [\sigma_1]\ pc_1 \xrightarrow{e} \mu_2\ [\sigma_2]\ pc_2$$

memory

stack

pc

next state

$$\frac{\iota(n) = \textsf{Store} \qquad \mu(p) = k@L_3 \qquad L_1 \vee L_{pc} \leq L_3 \qquad \mu(p) \leftarrow (m@L_1 \vee L_2 \vee L_{pc}) = \mu'}{\mu\ [p@L_1, m@L_2, \sigma]\ n@L_{pc} \xrightarrow{\tau} \mu'\ [\sigma]\ n{+}1@L_{pc}}$$

$$\frac{\iota(n) = \textsf{Jump}}{\mu\ [n'@L_1, \sigma]\ n@L_{pc} \xrightarrow{\tau} \mu\ [\sigma]\ n'@(L_1 \vee L_{pc})}$$

$$\frac{\iota(n) = \textsf{Bnz}\ k \qquad n' = n{+}(m = 0)?1 : k}{\mu\ [m@L_1, \sigma]\ n@L_{pc} \xrightarrow{\tau} \mu\ [\sigma]\ n'@(L_1 \vee L_{pc})}$$

$$\frac{\iota(n) = \textsf{Call}}{\mu\ [n'@L_1, a, \sigma]\ n@L_{pc} \xrightarrow{\tau} \mu\ [a, n{+}1@L_{pc}; \sigma]\ n'@(L_1 \vee L_{pc})}$$

$$\frac{\iota(n) = \textsf{Ret}}{\mu\ [n'@L_1; \sigma]\ n@L_{pc} \xrightarrow{\tau} \mu\ [\sigma]\ n'@L_1}$$

$$\frac{\iota(n) = \mathsf{Sub}}{\mu \quad [n_1@L_1, n_2@L_2, \sigma] \;\; n@L_{pc} \quad \xrightarrow{\tau}}$$

$$\frac{\iota(n) = \mathsf{Sub}}{\begin{array}{l}\mu \quad [n_1@L_1, n_2@L_2, \sigma] \;\; n@L_{pc} \quad \xrightarrow{\tau} \\ \mu \;\; [(n_1 - n_2)@(L_1 \vee L_2), \sigma] \;\; n{+}1@L_{pc}\end{array}}$$

$$\frac{\begin{array}{l}\iota(n) = \mathsf{Store} \qquad \mu(p) = k@L_3 \qquad L_1 \vee L_{pc} \leq L_3 \\ \mu(p) \leftarrow (m@L_1 \vee L_2 \vee L_{pc}) = \mu'\end{array}}{\mu \; [p@L_1, m@L_2, \sigma] \; n@L_{pc} \;\xrightarrow{\tau}\; \mu' \; [\sigma] \; n{+}1@L_{pc}}$$

$$\frac{\iota(n) = \mathsf{Jump}}{\mu \; [n'@L_1, \sigma] \; n@L_{pc} \;\xrightarrow{\tau}\; \mu \; [\sigma] \; n'@(L_1 \vee L_{pc})}$$

$$\frac{\iota(n) = \mathsf{Bnz}\; k \qquad n' = n{+}(m = 0)?1 : k}{\mu \; [m@L_1, \sigma] \; n@L_{pc} \;\xrightarrow{\tau}\; \mu \; [\sigma] \; n'@(L_1 \vee L_{pc})}$$

$$\frac{\iota(n) = \mathsf{Call}}{\mu \; [n'@L_1, a, \sigma] \; n@L_{pc} \;\xrightarrow{\tau}\; \mu \; [a, n{+}1@L_{pc}; \sigma] \; n'@(L_1 \vee L_{pc})}$$

$$\frac{\iota(n) = \mathsf{Ret}}{\mu \; [n'@L_1; \sigma] \; n@L_{pc} \;\xrightarrow{\tau}\; \mu \; [\sigma] \; n'@L_1}$$

Example

index n

Suppose:

$$\iota = [\ldots, \mathsf{Sub}, \ldots]$$

Then:

$$\mu \quad [7@\bot, 5@\top] \quad n@\bot \quad \longrightarrow$$

$$\mu \quad [2@\top] \quad (n+1)@\bot$$

$$\frac{\iota(n) = \mathsf{Sub}}{\begin{array}{l}\mu \quad [n_1@L_1, n_2@L_2, \sigma] \;\; n@L_{pc} \quad \xrightarrow{\tau} \\ \mu \;\; [(n_1 - n_2)@(L_1 \vee L_2), \sigma] \;\; n{+}1@L_{pc}\end{array}}$$

$$\iota(n) = \mathsf{Output}$$

$$\frac{\iota(n) = \mathsf{Bnz}\, k \qquad n' = n{+}((m = 0)?1 : k)}{\mu \; [m@L_1, \sigma] \; n@L_{pc} \;\; \xrightarrow{\tau} \;\; \mu \; [\sigma] \; n'@(L_1 \vee L_{pc})}$$

$$\frac{\begin{array}{l}\iota(n) = \mathsf{Store} \qquad \mu(p) = k@L_3 \qquad L_1 \vee L_{pc} \leq L_3 \\ \mu(p) \leftarrow (m@L_1 \vee L_2 \vee L_{pc}) = \mu'\end{array}}{\mu \; [p@L_1, m@L_2, \sigma] \; n@L_{pc} \;\; \xrightarrow{\tau} \;\; \mu' \; [\sigma] \; n{+}1@L_{pc}}$$

$$\frac{\iota(n) = \mathsf{Jump}}{\mu \; [n'@L_1, \sigma] \; n@L_{pc} \;\; \xrightarrow{\tau} \;\; \mu \; [\sigma] \; n'@(L_1 \vee L_{pc})}$$

$$\frac{\iota(n) = \mathsf{Bnz}\, k \qquad n' = n{+}(m = 0)?1 : k}{\mu \; [m@L_1, \sigma] \; n@L_{pc} \;\; \xrightarrow{\tau} \;\; \mu \; [\sigma] \; n'@(L_1 \vee L_{pc})}$$

$$\frac{\iota(n) = \mathsf{Call}}{\mu \; [n'@L_1, a, \sigma] \; n@L_{pc} \;\; \xrightarrow{\tau} \;\; \mu \; [a, n{+}1@L_{pc}; \sigma] \; n'@(L_1 \vee L_{pc})}$$

$$\frac{\iota(n) = \mathsf{Ret}}{\mu \; [n'@L_1; \sigma] \; n@L_{pc} \;\; \xrightarrow{\tau} \;\; \mu \; [\sigma] \; n'@L_1}$$

$$\frac{\iota(n) = \mathsf{Sub}}{\begin{array}{l}\mu \quad [n_1@L_1, n_2@L_2, \sigma] \;\; n@L_{pc} \quad \xrightarrow{\tau} \\ \mu \;\; [(n_1 - n_2)@(L_1 \vee L_2), \sigma] \;\; n{+}1@L_{pc}\end{array}}$$

$$\iota(n) = \mathsf{Output}$$

$$\frac{\iota(n) = \mathsf{Store} \qquad \mu(p) = k@L_3 \qquad L_1 \vee L_{pc} \leq L_3 \qquad \mu(p) \leftarrow (m@L_1 \vee L_2 \vee L_{pc}) = \mu'}{\mu\; [p@L_1, m@L_2, \sigma]\; n@L_{pc} \;\xrightarrow{\tau}\; \mu'\; [\sigma]\; (n{+}1)@L_{pc}}$$

$$\frac{\mu(p) \leftarrow (m@L_1 \vee L_2 \vee L_{pc}) = \mu'}{\mu\; [p@L_1, m@L_2, \sigma]\; n@L_{pc} \;\xrightarrow{\tau}\; \mu'\; [\sigma]\; n{+}1@L_{pc}}$$

$$\frac{\iota(n) = \mathsf{Jump}}{\mu\; [n'@L_1, \sigma]\; n@L_{pc} \;\xrightarrow{\tau}\; \mu\; [\sigma]\; n'@(L_1 \vee L_{pc})}$$

$$\frac{\iota(n) = \mathsf{Bnz}\; k \qquad n' = n{+}(m = 0)?1 : k}{\mu\; [m@L_1, \sigma]\; n@L_{pc} \;\xrightarrow{\tau}\; \mu\; [\sigma]\; n'@(L_1 \vee L_{pc})}$$

$$\frac{\iota(n) = \mathsf{Call}}{\mu\; [n'@L_1, a, \sigma]\; n@L_{pc} \;\xrightarrow{\tau}\; \mu\; [a, n{+}1@L_{pc}; \sigma]\; n'@(L_1 \vee L_{pc})}$$

$$\frac{\iota(n) = \mathsf{Ret}}{\mu\; [n'@L_1; \sigma]\; n@L_{pc} \;\xrightarrow{\tau}\; \mu\; [\sigma]\; n'@L_1}$$

$$\frac{\iota(n) = \mathsf{Sub}}{\begin{array}{l}\mu \quad [n_1@L_1, n_2@L_2, \sigma] \;\; n@L_{pc} \xrightarrow{\tau} \\ \mu \;\; [(n_1 - n_2)@(L_1 \vee L_2), \sigma] \;\; n{+}1@L_{pc}\end{array}}$$

$$\frac{\iota(n) = \mathsf{Output}}{\mu \; [m@L_1, \sigma] \; n@L_{pc} \xrightarrow{m@L_1 \vee L_{pc}} \mu \; [\sigma] \; n{+}1@L_{pc}}$$

$$\frac{\iota(n) = \mathsf{Push}\; m}{\mu \; [\sigma] \; n@L_{pc} \xrightarrow{\tau} \mu \; [m@\bot, \sigma] \; n{+}1@L_{pc}}$$

$$\frac{\iota(n) = \mathsf{Load} \qquad \mu(p) = m@L_2}{\mu \; [p@L_1, \sigma] \; n@L_{pc} \xrightarrow{\tau} \mu \; [m@L_1 \vee L_2, \sigma] \; n{+}1@L_{pc}}$$

$$\frac{\begin{array}{l}\iota(n) = \mathsf{Store} \qquad \mu(p) = k@L_3 \qquad L_1 \vee L_{pc} \leq L_3 \\ \mu(p) \leftarrow (m@L_1 \vee L_2 \vee L_{pc}) = \mu'\end{array}}{\mu \; [p@L_1, m@L_2, \sigma] \; n@L_{pc} \xrightarrow{\tau} \mu' \; [\sigma] \; n{+}1@L_{pc}}$$

$$\frac{\iota(n) = \mathsf{Call}}{\mu \; [n'@L_1, a, \sigma] \; n@L_{pc} \xrightarrow{\tau} \mu \; [a, n{+}1@L_{pc}; \sigma] \; n'@(L_1 \vee L_{pc})}$$

$$\frac{\iota(n) = \mathsf{Ret}}{\mu \; [n'@L_1; \sigma] \; n@L_{pc} \xrightarrow{\tau} \mu \; [\sigma] \; n'@L_1}$$

Symbolic IFC Rule Machine

Symbolic IFC Rule Machine

- Alternative presentation of abstract machine
  - Same machine states
  - Same step relation
- IFC side conditions factored out into a separate, explicit *rule table*

$$\frac{\iota(n) = \mathsf{Sub} \qquad \vdash_{\mathcal{R}} (L_{pc}, L_1, L_2, \_) \leadsto_{\mathsf{sub}} L_{rpc}, L_r}{\mu\ [n_1@L_1, n_1@L_2, \sigma]\ \ n@L_{pc} \xrightarrow{\tau} \mu\ [(n_1 - n_2)@L_r, \sigma]\ \ (n{+}1)@L_{rpc}}$$

$\iota(n) = \mathsf{Output}$ … $\mu\ [m@L_1,$ … L_{rpc}

$\iota(n$ … $(L_{pc}, \_, \_, \_) \leadsto_{\mathsf{push}} L_{rpc}, L_r$ … $n@L_r, \sigma]\ (n{+}1)@L_{rpc}$

… $) = \mathsf{Load} \quad \mu(n) = m@L_2$ …

to obtain result tags…

and opcode…

for tags…

consult rule table…

$$\frac{\iota(n) = \mathsf{Sub} \qquad \vdash_{\mathcal{R}} (L_{pc}, L_1, L_2, \_) \leadsto_{\mathsf{sub}} L_{rpc}, L_r}{\mu\ [n_1@L_1, n_1@L_2, \sigma]\ \ n@L_{pc} \xrightarrow{\tau} \mu\ [(n_1 - n_2)@L_r, \sigma]\ \ (n{+}1)@L_{rpc}}$$

$$\frac{\iota(n) = \mathsf{Bnz}\ k \qquad n' = n{+}(m = 0)\,?\,1 : k \qquad \vdash_{\mathcal{R}} (L_{pc}, L_1, \_, \_) \leadsto_{\mathsf{bnz}} L_{rpc}, \_}{\mu\ [m@L_1, \sigma]\ n@L_{pc} \xrightarrow{\tau} \mu\ [\sigma]\ n'@L_{rpc}}$$

$$\frac{\iota(n) = \mathsf{Call} \qquad \vdash_{\mathcal{R}} (L_{pc}, L_1, \_, \_) \leadsto_{\mathsf{call}} L_{rpc}, L_r}{\mu\ [n'@L_1, a, \sigma]\ n@L_{pc} \xrightarrow{\tau} \mu\ [a, (n{+}1)@L_r; \sigma]\ n'@L_{rpc}}$$

$$\frac{\iota(n) = \mathsf{Ret} \qquad \vdash_{\mathcal{R}} (L_{pc}, L_1, \_, \_) \leadsto_{\mathsf{ret}} L_{rpc}, \_}{\mu\ [n'@L_1; \sigma]\ n@L_{pc} \xrightarrow{\tau} \mu\ [\sigma]\ n'@L_{rpc}}$$

IFC Rule Table

is this operation allowed?

new pc label

label for result

$\mathcal{R}$ =

$opcode$	$allow$	e_{rpc}	e_r
sub	$\mathtt{TRUE}$	$\mathtt{LAB}_{pc}$	$\mathtt{LAB}_1 \sqcup \mathtt{LAB}_2$
output	$\mathtt{TRUE}$	$\mathtt{LAB}_{pc}$	$\mathtt{LAB}_1 \sqcup \mathtt{LAB}_{pc}$
push	$\mathtt{TRUE}$	$\mathtt{LAB}_{pc}$	$\mathtt{BOT}$
load	$\mathtt{TRUE}$	$\mathtt{LAB}_{pc}$	$\mathtt{LAB}_1 \sqcup \mathtt{LAB}_2$
store	$\mathtt{LAB}_1 \sqcup \mathtt{LAB}_{pc} \sqsubseteq \mathtt{LAB}_3$	$\mathtt{LAB}_{pc}$	$\mathtt{LAB}_1 \sqcup \mathtt{LAB}_2 \sqcup \mathtt{LAB}_{pc}$
jump	$\mathtt{TRUE}$	$\mathtt{LAB}_1 \sqcup \mathtt{LAB}_{pc}$	--
bnz	$\mathtt{TRUE}$	$\mathtt{LAB}_1 \sqcup \mathtt{LAB}_{pc}$	--
call	$\mathtt{TRUE}$	$\mathtt{LAB}_1 \sqcup \mathtt{LAB}_{pc}$	$\mathtt{LAB}_{pc}$
ret	$\mathtt{TRUE}$	$\mathtt{LAB}_1$	--

IFC Rule Table

subtraction is always allowed

pc label is unchanged

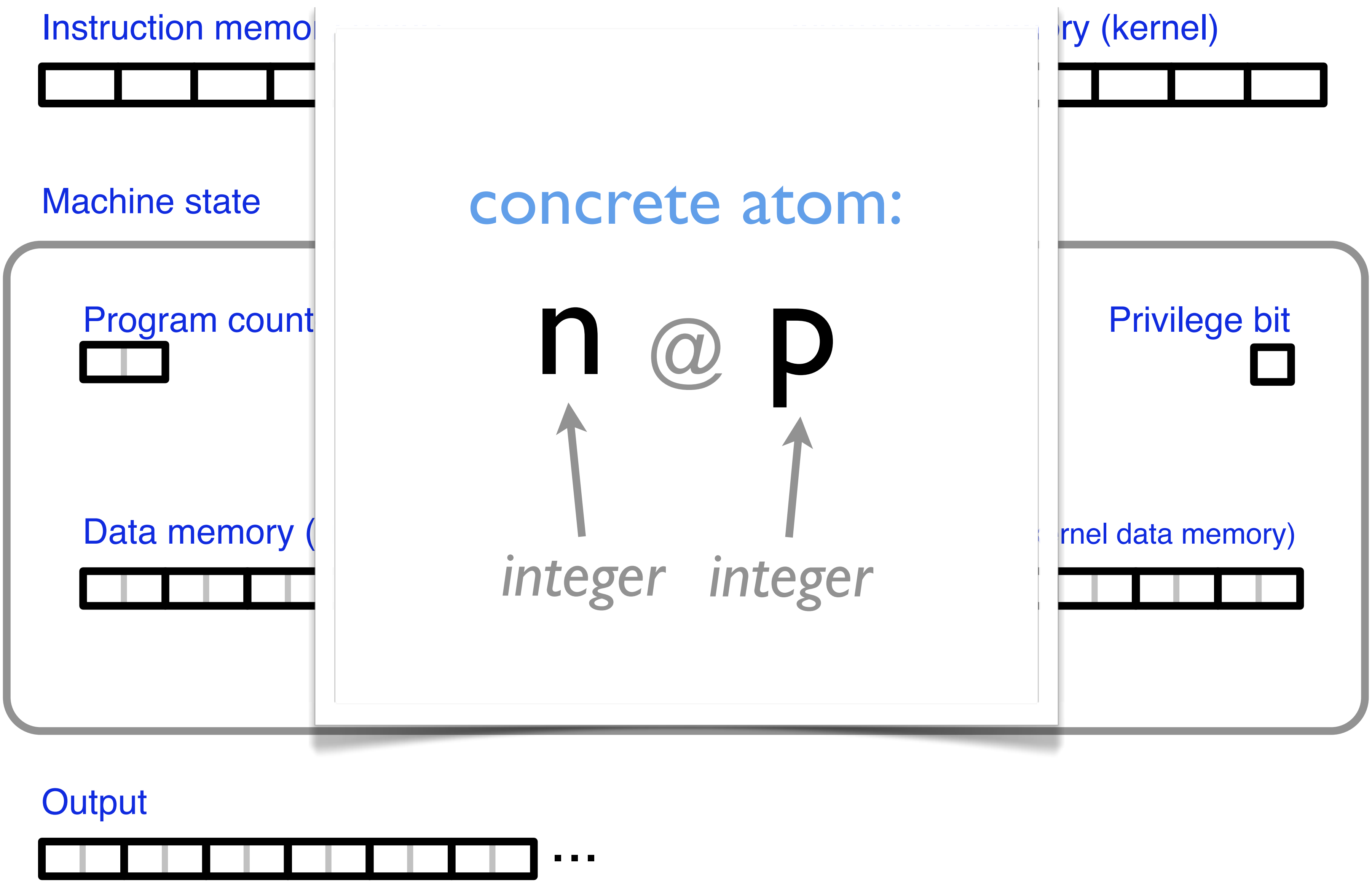
result label is join of arg labels

$\mathcal{R}$ =

opcode	*allow*	e_{rpc}	e_r
sub	TRUE	$\mathtt{LAB}_{pc}$	$\mathtt{LAB}_1 \sqcup \mathtt{LAB}_2$
output	TRUE	$\mathtt{LAB}_{pc}$	$\mathtt{LAB}_1 \sqcup \mathtt{LAB}_{pc}$
push	TRUE	$\mathtt{LAB}_{pc}$	BOT
load	TRUE	$\mathtt{LAB}_{pc}$	$\mathtt{LAB}_1 \sqcup \mathtt{LAB}_2$
store	$\mathtt{LAB}_1 \sqcup \mathtt{LAB}_{pc} \sqsubseteq \mathtt{LAB}_3$	$\mathtt{LAB}_{pc}$	$\mathtt{LAB}_1 \sqcup \mathtt{LAB}_2 \sqcup \mathtt{LAB}_{pc}$
jump	TRUE	$\mathtt{LAB}_1 \sqcup \mathtt{LAB}_{pc}$	--
bnz	TRUE	$\mathtt{LAB}_1 \sqcup \mathtt{LAB}_{pc}$	--
call	TRUE	$\mathtt{LAB}_1 \sqcup \mathtt{LAB}_{pc}$	$\mathtt{LAB}_{pc}$
ret	TRUE	$\mathtt{LAB}_1$	--

Concrete Machine

Concrete machine



Concrete machine

Instruction memory (user)

Instruction memory (kernel)

(single-line)

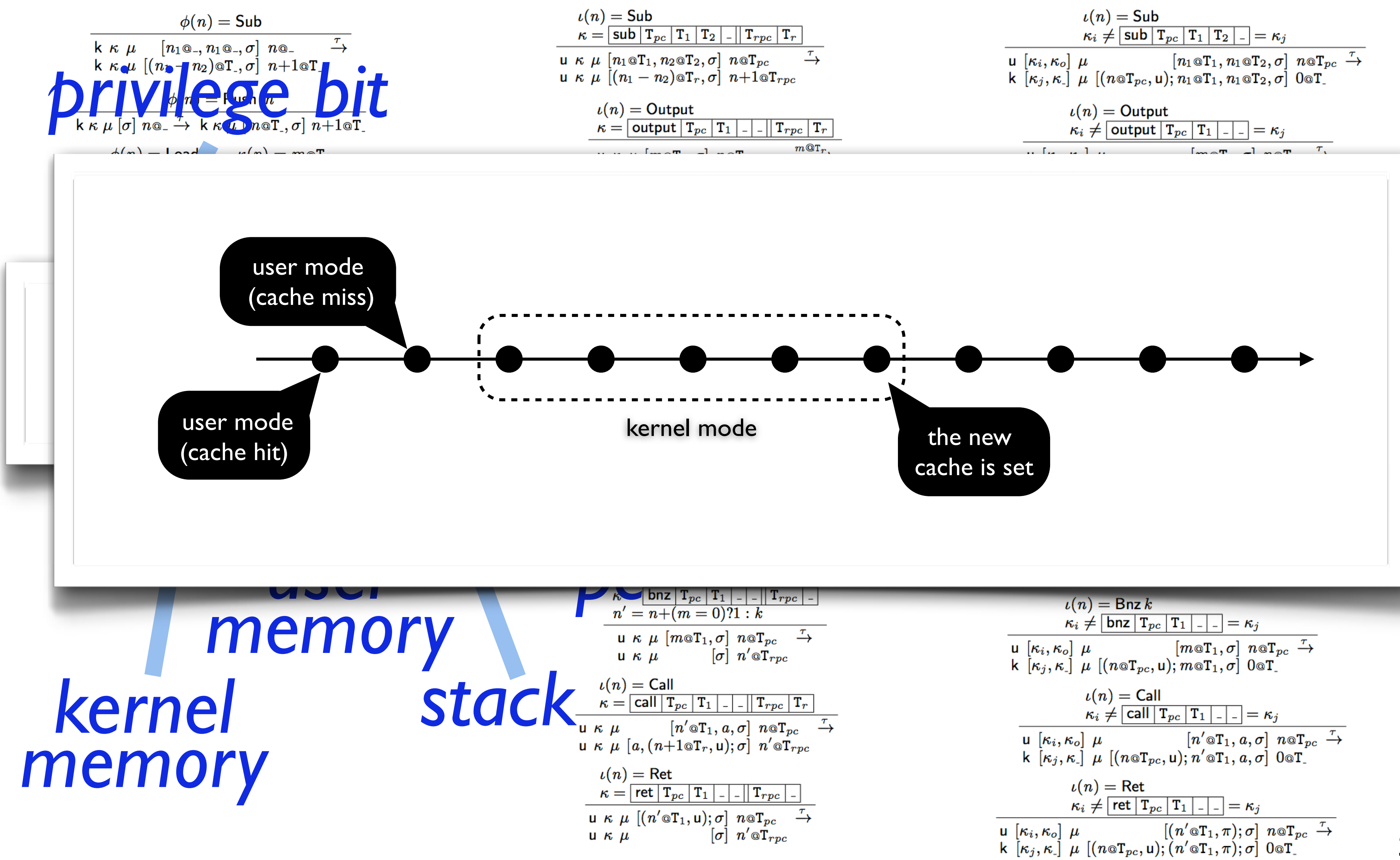
Rule cache:

Output

Kernel mode

User mode (cache hit)

User-to-kernel mode (cache miss)



User mode (cache hit)

User-to-kernel mode (cache miss)

Kernel mode

User mode (cache hit):

$$\frac{\iota(n) = \mathsf{Sub} \qquad \kappa = [\mathsf{sub} \mid \mathrm{T}_{pc} \mid \mathrm{T}_1 \mid \mathrm{T}_2 \mid \_ \,\|\, \mathrm{T}_{rpc} \mid \mathrm{T}_r]}{\mathsf{u}\ \kappa\ \mu\ [n_1@\mathrm{T}_1, n_2@\mathrm{T}_2, \sigma]\ n@\mathrm{T}_{pc} \xrightarrow{\tau} \mathsf{u}\ \kappa\ \mu\ [(n_1 - n_2)@\mathrm{T}_r, \sigma]\ n{+}1@\mathrm{T}_{rpc}}$$

$$\frac{\iota(n) = \mathsf{Output} \qquad \kappa = [\mathsf{output} \mid \mathrm{T}_{pc} \mid \mathrm{T}_1 \mid \_ \mid \_ \mid \mathrm{T}_{rpc} \mid \mathrm{T}_r]}{\mathsf{u}\ \kappa\ \mu\ [m@\mathrm{T}_1, \sigma]\ n@\mathrm{T}_{pc} \xrightarrow{m@\mathrm{T}_r} \mathsf{u}\ \kappa\ \mu\ [\sigma]\ n{+}1@\mathrm{T}_{rpc}}$$

… $n' = n{+}(m = 0)?1 : k$

$$\mathsf{u}\ \kappa\ \mu\ [m@\mathrm{T}_1, \sigma]\ n@\mathrm{T}_{pc} \xrightarrow{\tau} \mathsf{u}\ \kappa\ \mu\ [\sigma]\ n'@\mathrm{T}_{rpc}$$

$$\frac{\iota(n) = \mathsf{Call} \qquad \kappa = [\mathsf{call} \mid \mathrm{T}_{pc} \mid \mathrm{T}_1 \mid \_ \mid \_ \mid \mathrm{T}_{rpc} \mid \mathrm{T}_r]}{\mathsf{u}\ \kappa\ \mu\ [n'@\mathrm{T}_1, a, \sigma]\ n@\mathrm{T}_{pc} \xrightarrow{\tau} \mathsf{u}\ \kappa\ \mu\ [a, (n{+}1@\mathrm{T}_r, \mathsf{u}); \sigma]\ n'@\mathrm{T}_{rpc}}$$

$$\frac{\iota(n) = \mathsf{Ret} \qquad \kappa = [\mathsf{ret} \mid \mathrm{T}_{pc} \mid \mathrm{T}_1 \mid \_ \mid \_ \mid \mathrm{T}_{rpc} \mid \_]}{\mathsf{u}\ \kappa\ \mu\ [(n'@\mathrm{T}_1, \mathsf{u}); \sigma]\ n@\mathrm{T}_{pc} \xrightarrow{\tau} \mathsf{u}\ \kappa\ \mu\ [\sigma]\ n'@\mathrm{T}_{rpc}}$$

User-to-kernel mode (cache miss):

$$\frac{\iota(n) = \mathsf{Sub} \qquad \kappa_i \neq [\mathsf{sub} \mid \mathrm{T}_{pc} \mid \mathrm{T}_1 \mid \mathrm{T}_2 \mid \_] = \kappa_j}{\mathsf{u}\ [\kappa_i, \kappa_o]\ \mu\ [n_1@\mathrm{T}_1, n_1@\mathrm{T}_2, \sigma]\ n@\mathrm{T}_{pc} \xrightarrow{\tau} \mathsf{k}\ [\kappa_j, \kappa_\_]\ \mu\ [(n@\mathrm{T}_{pc}, \mathsf{u}); n_1@\mathrm{T}_1, n_1@\mathrm{T}_2, \sigma]\ 0@\mathrm{T}_\_}$$

$$\frac{\iota(n) = \mathsf{Output} \qquad \kappa_i \neq [\mathsf{output} \mid \mathrm{T}_{pc} \mid \mathrm{T}_1 \mid \_ \mid \_] = \kappa_j}{\mathsf{u}\ [\kappa_i, \kappa_o]\ \mu\ [m@\mathrm{T}_1, \sigma]\ n@\mathrm{T}_{pc} \xrightarrow{\tau} \mathsf{k}\ [\kappa_i, \kappa_\_]\ \mu\ [(n@\mathrm{T}_{pc}, \mathsf{u}); m@\mathrm{T}_1, \sigma]\ 0@\mathrm{T} \ldots}$$

$$\frac{\iota(n) = \mathsf{Bnz}\ k \qquad \kappa_i \neq [\mathsf{bnz} \mid \mathrm{T}_{pc} \mid \mathrm{T}_1 \mid \_ \mid \_] = \kappa_j}{\mathsf{u}\ [\kappa_i, \kappa_o]\ \mu\ [m@\mathrm{T}_1, \sigma]\ n@\mathrm{T}_{pc} \xrightarrow{\tau} \mathsf{k}\ [\kappa_j, \kappa_\_]\ \mu\ [(n@\mathrm{T}_{pc}, \mathsf{u}); m@\mathrm{T}_1, \sigma]\ 0@\mathrm{T}_\_}$$

$$\frac{\iota(n) = \mathsf{Call} \qquad \kappa_i \neq [\mathsf{call} \mid \mathrm{T}_{pc} \mid \mathrm{T}_1 \mid \_ \mid \_] = \kappa_j}{\mathsf{u}\ [\kappa_i, \kappa_o]\ \mu\ [n'@\mathrm{T}_1, a, \sigma]\ n@\mathrm{T}_{pc} \xrightarrow{\tau} \mathsf{k}\ [\kappa_j, \kappa_\_]\ \mu\ [(n@\mathrm{T}_{pc}, \mathsf{u}); n'@\mathrm{T}_1, a, \sigma]\ 0@\mathrm{T}_\_}$$

$$\frac{\iota(n) = \mathsf{Ret} \qquad \kappa_i \neq [\mathsf{ret} \mid \mathrm{T}_{pc} \mid \mathrm{T}_1 \mid \_ \mid \_] = \kappa_j}{\mathsf{u}\ [\kappa_i, \kappa_o]\ \mu\ [(n'@\mathrm{T}_1, \pi); \sigma]\ n@\mathrm{T}_{pc} \xrightarrow{\tau} \mathsf{k}\ [\kappa_j, \kappa_\_]\ \mu\ [(n@\mathrm{T}_{pc}, \mathsf{u}); (n'@\mathrm{T}_1, \pi); \sigma]\ 0@\mathrm{T}_\_}$$

Kernel mode:

$$\frac{\phi(n) = \mathsf{Sub}}{\mathsf{k}\ \kappa\ \mu\ [n_1@\_, n_1@\_, \sigma]\ n@\_ \xrightarrow{\tau} \mathsf{k}\ \kappa\ \mu\ [(n_1 - n_2)@\mathrm{T}_\_, \sigma]\ n{+}1@\mathrm{T}_\_}$$

$$\frac{\phi(n) = \mathsf{Push}\ m}{\mathsf{k}\ \kappa\ \mu\ [\sigma]\ n@\_ \xrightarrow{\tau} \mathsf{k}\ \kappa\ \mu\ [m@\mathrm{T}_\_, \sigma]\ n{+}1@\mathrm{T}_\_}$$

$\phi(n) = \mathsf{Load} \qquad \kappa(p) = m@\mathrm{T}_1$ …

User mode (cache hit)

$$\frac{\iota(n) = \textsf{Sub} \quad \kappa = \boxed{\textsf{sub} \mid T_{pc} \mid T_1 \mid T_2 \mid \_ \mid T_{rpc} \mid T_r}}{\textsf{u}\ \kappa\ \mu\ [n_1@T_1, n_2@T_2, \sigma]\ n@T_{pc} \xrightarrow{\tau} \textsf{u}\ \kappa\ \mu\ [(n_1 - n_2)@T_r, \sigma]\ n{+}1@T_{rpc}}$$

$$\frac{\iota(n) = \textsf{Output} \quad \kappa = \boxed{\textsf{output} \mid T_{pc} \mid T_1 \mid \_ \mid \_ \mid T_{rpc} \mid T_r}}{\textsf{u}\ \kappa\ \mu\ [m@T_1, \sigma]\ n@T_{pc} \xrightarrow{m@T_r} \ldots}$$

$$\frac{\ldots}{\textsf{u}\ \kappa\ \mu\ [n'@T_1, \sigma]\ n@T_{pc} \xrightarrow{\tau} \textsf{u}\ \kappa\ \mu\ [\sigma]\ n'@T_{rpc}}$$

$$\frac{\iota(n) = \textsf{Bnz}\ k \quad \kappa = \boxed{\textsf{bnz} \mid T_{pc} \mid T_1 \mid \_ \mid \_ \mid T_{rpc} \mid \_} \quad n' = n{+}(m = 0)?1 : k}{\textsf{u}\ \kappa\ \mu\ [m@T_1, \sigma]\ n@T_{pc} \xrightarrow{\tau} \textsf{u}\ \kappa\ \mu\ [\sigma]\ n'@T_{rpc}}$$

$$\frac{\iota(n) = \textsf{Call} \quad \kappa = \boxed{\textsf{call} \mid T_{pc} \mid T_1 \mid \_ \mid \_ \mid T_{rpc} \mid T_r}}{\textsf{u}\ \kappa\ \mu\ [n'@T_1, a, \sigma]\ n@T_{pc} \xrightarrow{\tau} \textsf{u}\ \kappa\ \mu\ [a, (n{+}1@T_r, \textsf{u}); \sigma]\ n'@T_{rpc}}$$

$$\frac{\iota(n) = \textsf{Ret} \quad \kappa = \boxed{\textsf{ret} \mid T_{pc} \mid T_1 \mid \_ \mid \_ \mid T_{rpc} \mid \_}}{\textsf{u}\ \kappa\ \mu\ [(n'@T_1, \textsf{u}); \sigma]\ n@T_{pc} \xrightarrow{\tau} \textsf{u}\ \kappa\ \mu\ [\sigma]\ n'@T_{rpc}}$$

User-to-kernel mode (cache miss)

$$\frac{\iota(n) = \textsf{Sub} \quad \kappa_i \neq \boxed{\textsf{sub} \mid T_{pc} \mid T_1 \mid T_2 \mid \_} = \kappa_j}{\textsf{u}\ [\kappa_i, \kappa_o]\ \mu\ [n_1@T_1, n_2@T_2, \sigma]\ n@T_{pc} \xrightarrow{\tau} \textsf{k}\ [\kappa_j, \kappa_\_]\ \mu\ [(n@T_{pc}, \textsf{u}); n_1@T_1, n_2@T_2, \sigma]\ 0@T_\_}$$

$$\frac{\iota(n) = \textsf{Output} \quad \kappa_i \neq \boxed{\textsf{output} \mid T_{pc} \mid T_1 \mid \_ \mid \_} = \kappa_j}{\textsf{u}\ [\kappa_i, \kappa_o]\ \mu\ [m@T_1, \sigma]\ n@T_{pc} \xrightarrow{\tau} \ldots}$$

$$\frac{\kappa_i \neq \boxed{\textsf{jump} \mid T_{pc} \mid T_1 \mid \_ \mid \_} = \kappa_j}{\textsf{u}\ [\kappa_i, \kappa_o]\ \mu\ [n'@T_1, \sigma]\ n@T_{pc} \xrightarrow{\tau} \textsf{k}\ [\kappa_j, \kappa_\_]\ \mu\ [(n@T_{pc}, \textsf{u}); n'@T_1, \sigma]\ 0@T_\_}$$

$$\frac{\iota(n) = \textsf{Bnz}\ k \quad \kappa_i \neq \boxed{\textsf{bnz} \mid T_{pc} \mid T_1 \mid \_ \mid \_} = \kappa_j}{\textsf{u}\ [\kappa_i, \kappa_o]\ \mu\ [m@T_1, \sigma]\ n@T_{pc} \xrightarrow{\tau} \textsf{k}\ [\kappa_j, \kappa_\_]\ \mu\ [(n@T_{pc}, \textsf{u}); m@T_1, \sigma]\ 0@T_\_}$$

$$\frac{\iota(n) = \textsf{Call} \quad \kappa_i \neq \boxed{\textsf{call} \mid T_{pc} \mid T_1 \mid \_ \mid \_} = \kappa_j}{\textsf{u}\ [\kappa_i, \kappa_o]\ \mu\ [n'@T_1, a, \sigma]\ n@T_{pc} \xrightarrow{\tau} \textsf{k}\ [\kappa_j, \kappa_\_]\ \mu\ [(n@T_{pc}, \textsf{u}); n'@T_1, a, \sigma]\ 0@T_\_}$$

$$\frac{\iota(n) = \textsf{Ret} \quad \kappa_i \neq \boxed{\textsf{ret} \mid T_{pc} \mid T_1 \mid \_ \mid \_} = \kappa_j}{\textsf{u}\ [\kappa_i, \kappa_o]\ \mu\ [(n'@T_1, \pi); \sigma]\ n@T_{pc} \xrightarrow{\tau} \textsf{k}\ [\kappa_j, \kappa_\_]\ \mu\ [(n@T_{pc}, \textsf{u}); (n'@T_1, \pi); \sigma]\ 0@T_\_}$$

Kernel mode

$$\frac{\phi(n) = \textsf{Sub}}{\textsf{k}\ \kappa\ \mu\ [n_1@\_, n_2@\_, \sigma]\ n@\_ \xrightarrow{\tau} \textsf{k}\ \kappa\ \mu\ [(n_1 - n_2)@T_\_, \sigma]\ n{+}1@T_\_}$$

$$\frac{\phi(n) = \textsf{Push}\ m}{\textsf{k}\ \kappa\ \mu\ [\sigma]\ n@\_ \xrightarrow{\tau} \textsf{k}\ \kappa\ \mu\ [m@T_\_, \sigma]\ n{+}1@T_\_}$$

$$\phi(n) = \textsf{Load} \qquad \kappa(p) = m@T_1 \quad \ldots$$

$$\frac{\iota(n) = \textsf{Sub} \quad \kappa_i \neq \boxed{\textsf{sub} \mid T_{pc} \mid T_1 \mid T_2 \mid \_} = \kappa_j}{\begin{array}{l}\textsf{u}\ [\kappa_i, \kappa_o]\ \mu\ [n_1@T_1, n_2@T_2, \sigma]\ n@T_{pc} \xrightarrow{\tau} \\ \textsf{k}\ [\kappa_j, \kappa_\_]\ \mu\ [(n@T_{pc}, \textsf{u}); n_1@T_1, n_2@T_2, \sigma]\ 0@T_\_\end{array}}$$

User mode (cache hit)

User-to-kernel mode (cache miss)

Kernel mode

$$\frac{\iota(n) = \mathsf{Sub} \qquad \kappa = \boxed{\mathsf{sub} \mid T_{pc} \mid T_1 \mid T_2 \mid \_ \mid T_{rpc} \mid T_r}}{\mathsf{u}\ \kappa\ \mu\ [n_1@T_1, n_2@T_2, \sigma]\ n@T_{pc} \xrightarrow{\tau} \mathsf{u}\ \kappa\ \mu\ [(n_1 - n_2)@T_r, \sigma]\ n{+}1@T_{rpc}}$$

$$\frac{\iota(n) = \mathsf{Output} \qquad \kappa = \boxed{\mathsf{output} \mid T_{pc} \mid T_1 \mid \_ \mid \_ \mid T_{rpc} \mid T_r}}{\mathsf{u}\ \kappa\ \mu\ [m@T_1, \sigma]\ n@T_{pc} \xrightarrow{m@T_r} \mathsf{u}\ \kappa\ \mu\ [\sigma]\ n{+}1@T_{rpc}}$$

$$\frac{\iota(n) = \mathsf{Sub} \qquad \kappa_i \neq \boxed{\mathsf{sub} \mid T_{pc} \mid T_1 \mid T_2 \mid \_} = \kappa_j}{\mathsf{u}\ [\kappa_i, \kappa_o]\ \mu\ [n_1@T_1, n_1@T_2, \sigma]\ n@T_{pc} \xrightarrow{\tau} \mathsf{k}\ [\kappa_j, \kappa_\_]\ \mu\ [(n@T_{pc}, \mathsf{u}); n_1@T_1, n_1@T_2, \sigma]\ 0@T_\_}$$

$$\frac{\iota(n) = \mathsf{Output} \qquad \kappa_i \neq \boxed{\mathsf{output} \mid T_{pc} \mid T_1 \mid \_ \mid \_} = \kappa_j}{\mathsf{u}\ [\kappa_i, \kappa_o]\ \mu\ [m@T_1, \sigma]\ n@T_{pc} \xrightarrow{\tau} \mathsf{k}\ [\kappa_j, \kappa_\_]\ \mu\ [(n@T_{pc}, \mathsf{u}); m@T_1, \sigma]\ 0@T_\_}$$

$$\frac{\phi(n) = \mathsf{Sub}}{\mathsf{k}\ \kappa\ \mu\ [n_1@\_, n_1@\_, \sigma]\ n@\_ \xrightarrow{\tau} \mathsf{k}\ \kappa\ \mu\ [(n_1 - n_2)@T_\_, \sigma]\ n{+}1@T_\_}$$

$$\frac{\phi(n) = \mathsf{Push}\ m}{\mathsf{k}\ \kappa\ \mu\ [\sigma]\ n@\_ \xrightarrow{\tau} \mathsf{k}\ \kappa\ \mu\ [m@T_\_, \sigma]\ n{+}1@T_\_}$$

$$\phi(n) = \mathsf{Load} \qquad \kappa(p) = m@T_1$$

$$\frac{\phi(n) = \mathsf{Sub}}{\begin{array}{l}\mathsf{k}\ \ \kappa\ \ \mu\ \ [n_1@\_, n_2@\_, \sigma]\ \ n@\_ \xrightarrow{\tau} \\ \mathsf{k}\ \ \kappa\ \ \mu\ \ [(n_1 - n_2)@T_\_, \sigma]\ \ n{+}1@T_\_\end{array}}$$

$$\frac{\iota(n) = \mathsf{Bnz}\ k \qquad \kappa = \boxed{\mathsf{bnz} \mid T_{pc} \mid T_1 \mid \_ \mid \_ \mid T_{rpc} \mid \_} \qquad n' = n{+}(m = 0)?1 : k}{\mathsf{u}\ \kappa\ \mu\ [m@T_1, \sigma]\ n@T_{pc} \xrightarrow{\tau} \mathsf{u}\ \kappa\ \mu\ [\sigma]\ n'@T_{rpc}}$$

$$\frac{\iota(n) = \mathsf{Call} \qquad \kappa = \boxed{\mathsf{call} \mid T_{pc} \mid T_1 \mid \_ \mid \_ \mid T_{rpc} \mid T_r}}{\mathsf{u}\ \kappa\ \mu\ [n'@T_1, a, \sigma]\ n@T_{pc} \xrightarrow{\tau} \mathsf{u}\ \kappa\ \mu\ [a, (n{+}1@T_r, \mathsf{u}); \sigma]\ n'@T_{rpc}}$$

$$\frac{\iota(n) = \mathsf{Ret} \qquad \kappa = \boxed{\mathsf{ret} \mid T_{pc} \mid T_1 \mid \_ \mid \_ \mid T_{rpc} \mid \_}}{\mathsf{u}\ \kappa\ \mu\ [(n'@T_1, \mathsf{u}); \sigma]\ n@T_{pc} \xrightarrow{\tau} \mathsf{u}\ \kappa\ \mu\ [\sigma]\ n'@T_{rpc}}$$

$$\frac{\iota(n) = \mathsf{Bnz}\ k \qquad \kappa_i \neq \boxed{\mathsf{bnz} \mid T_{pc} \mid T_1 \mid \_ \mid \_} = \kappa_j}{\mathsf{u}\ [\kappa_i, \kappa_o]\ \mu\ [m@T_1, \sigma]\ n@T_{pc} \xrightarrow{\tau} \mathsf{k}\ [\kappa_j, \kappa_\_]\ \mu\ [(n@T_{pc}, \mathsf{u}); m@T_1, \sigma]\ 0@T_\_}$$

$$\frac{\iota(n) = \mathsf{Call} \qquad \kappa_i \neq \boxed{\mathsf{call} \mid T_{pc} \mid T_1 \mid \_ \mid \_} = \kappa_j}{\mathsf{u}\ [\kappa_i, \kappa_o]\ \mu\ [n'@T_1, a, \sigma]\ n@T_{pc} \xrightarrow{\tau} \mathsf{k}\ [\kappa_j, \kappa_\_]\ \mu\ [(n@T_{pc}, \mathsf{u}); n'@T_1, a, \sigma]\ 0@T_\_}$$

$$\frac{\iota(n) = \mathsf{Ret} \qquad \kappa_i \neq \boxed{\mathsf{ret} \mid T_{pc} \mid T_1 \mid \_ \mid \_} = \kappa_j}{\mathsf{u}\ [\kappa_i, \kappa_o]\ \mu\ [(n'@T_1, \pi); \sigma]\ n@T_{pc} \xrightarrow{\tau} \mathsf{k}\ [\kappa_j, \kappa_\_]\ \mu\ [(n@T_{pc}, \mathsf{u}); (n'@T_1, \pi); \sigma]\ 0@T_\_}$$

Example (cache hit case)

Suppose

tag 0 represents label $\bot$

tag 1 represents label $\top$

cache inputs match
current machine state!

u	sub	0	0	1	_	0	1	μ	$[7@0, 5@1]$	$n@0$	$\longrightarrow$
u	sub	0	0	1	_	0	1	μ	$[2@1]$	$(n+1)@0$	

Example (cache miss case)

mismatch!

u	sub	0	0	1	_	0	1	μ	$[7@1, 5@1]$	$n@0$	$\longrightarrow$
k	sub	0	1	1	_	_	_	μ	$[(n@0, \mathsf{u}); 7@1, 5@1]$	$0@\_$	$\longrightarrow$
	... fault handler runs ...										
k	sub	0	1	1	_	0	1	μ	$[(n@0, \mathsf{u}); 7@1, 5@1]$	$k@\_$	$\longrightarrow$
u	sub	0	1	1	_	0	1	μ	$[7@1, 5@1]$	$n@0$	$\longrightarrow$
u	sub	0	0	1	_	0	1	μ	$[2@1]$	$(n+1)@0$	

Fault Handler

IFC RULE TABLE

opcode	*allow*	e_{rpc}	e_r
sub	TRUE	$\mathtt{LAB}_{pc}$	$\mathtt{LAB}_1 \sqcup \mathtt{LAB}_2$
output	TRUE	$\mathtt{LAB}_{pc}$	$\mathtt{LAB}_1 \sqcup \mathtt{LAB}_{pc}$
push	TRUE	$\mathtt{LAB}_{pc}$	BOT
load	TRUE	$\mathtt{LAB}_{pc}$	$\mathtt{LAB}_1 \sqcup \mathtt{LAB}_2$
store	$\mathtt{LAB}_1 \sqcup \mathtt{LAB}_{pc} \sqsubseteq \mathtt{LAB}_3$	$\mathtt{LAB}_{pc}$	$\mathtt{LAB}_1 \sqcup \mathtt{LAB}_2 \sqcup \mathtt{LAB}_{pc}$
jump	TRUE	$\mathtt{LAB}_1 \sqcup \mathtt{LAB}_{pc}$	--
bnz	TRUE	$\mathtt{LAB}_1 \sqcup \mathtt{LAB}_{pc}$	--
call	TRUE	$\mathtt{LAB}_1 \sqcup \mathtt{LAB}_{pc}$	$\mathtt{LAB}_{pc}$
ret	TRUE	$\mathtt{LAB}_1$	--

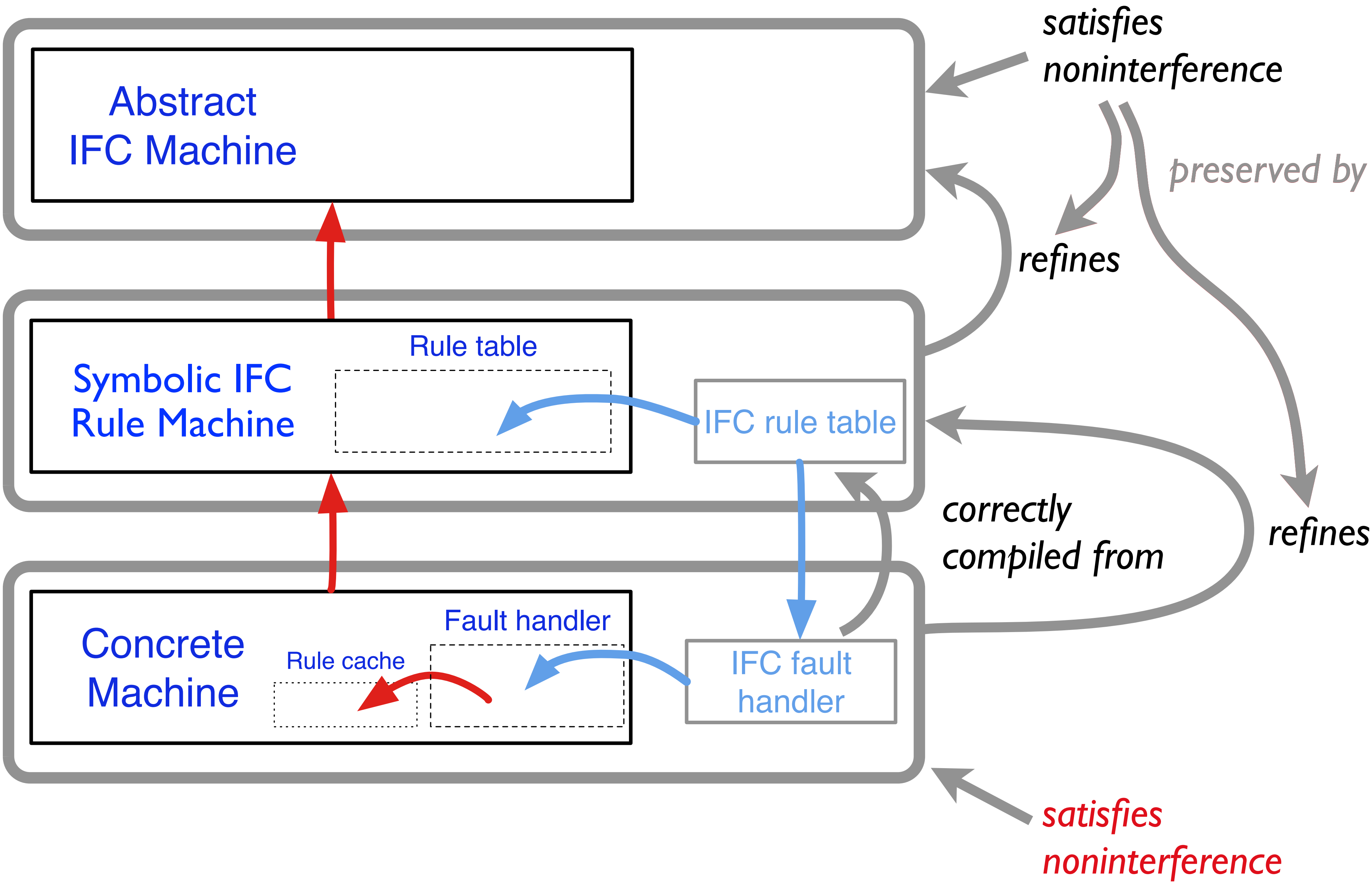
FAULT HANDLER

Handler Generation

- IFC rule table entries form a small DSL for computing labels and booleans
  - parameterized over lattice $\bot$, $\sqcup$ and $\sqsubseteq$
- The handler is constructed by compiling the DSL into concrete machine instructions
  - Table-driven interpreter would be an alternative
- We use structured code generators to simplify verification

Non-interference for concrete machine

- Running this <u>particular</u> fault handler
- Together with <u>arbitrary</u> user code


Abstract
IFC Machine
Symbolic IFC
Rule Machine
Rule table
IFC rule table
Concrete
Machine
Fault handler
Rule cache
IFC fault
handler
satisfies
noninterference
refines
preserved by
refines
correctly
compiled from
satisfies
noninterference

Points to note

- Refinement framework *very* useful for reasoning
  - start with concrete object
  - propose abstracted version
    - incorporate convenient structure and annotations
  - prove refinement
  - prove interesting property of abstract object
  - automatically follows for concrete object
- Need a *generic* notion of noninterference that makes sense for all machines
  - Includes a notion of abstracting concrete tags (and associated memory states) into labels

Some Verification Challenges...

More uses for tags

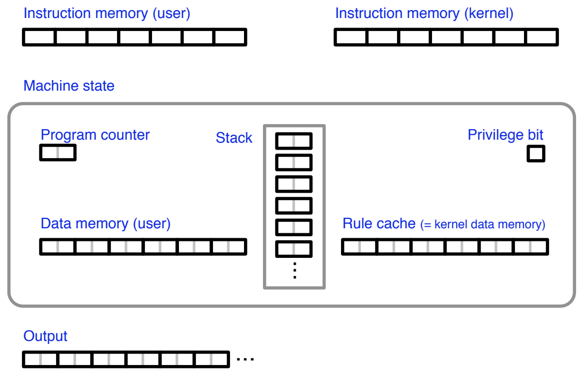
- SAFE architecture is quite generic
  - Can be used to implement a range of IFC label models just by varying the rule table [Montagu CSF '13]
- Other potential uses
  - access control (clearance)
  - memory protection
  - linearity
  - dynamic typing

More Security Issues

- Downgrading
- “Least Privilege”
- Concurrency

Real SAFE Machine

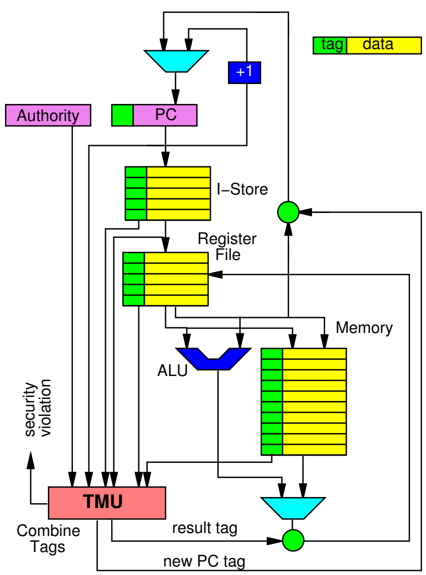
- Scaling methodology to full SAFE hardware and ConcreteWare
  - random testing vs. verification
- Breeze compiler correctness
  - for defense in depth



opcode	*allow*	e_{rpc}	e_r
sub	TRUE	$\mathtt{LAB}_{pc}$	$\mathtt{LAB}_1 \sqcup \mathtt{LAB}_2$
output	TRUE	$\mathtt{LAB}_{pc}$	$\mathtt{LAB}_1 \sqcup \mathtt{LAB}_{pc}$
push	TRUE	$\mathtt{LAB}_{pc}$	BOT
load	TRUE	$\mathtt{LAB}_{pc}$	$\mathtt{LAB}_1 \sqcup \mathtt{LAB}_2$
store	$\mathtt{LAB}_1 \sqcup \mathtt{LAB}_{pc} \sqsubseteq \mathtt{LAB}_3$	$\mathtt{LAB}_{pc}$	$\mathtt{LAB}_1 \sqcup \mathtt{LAB}_2 \sqcup \mathtt{LAB}_{pc}$
jump	TRUE	$\mathtt{LAB}_1 \sqcup \mathtt{LAB}_{pc}$	--
bnz	TRUE	$\mathtt{LAB}_1 \sqcup \mathtt{LAB}_{pc}$	--
call	TRUE	$\mathtt{LAB}_1 \sqcup \mathtt{LAB}_{pc}$	$\mathtt{LAB}_{pc}$
ret	TRUE	$\mathtt{LAB}_1$	--

Thank you!

Questions??



$$\frac{\iota(n) = \mathsf{Sub}}{\mu\ [n_1@L_1, n_2@L_2, \sigma]\ n@L_{pc} \xrightarrow{\tau} \mu\ [(n_1 - n_2)@(L_1 \vee L_2), \sigma]\ n{+}1@L_{pc}}$$

$$\frac{\iota(n) = \mathsf{Output}}{\mu\ [m@L_1, \sigma]\ n@L_{pc} \xrightarrow{m@L_1 \vee L_{pc}} \mu\ [\sigma]\ n{+}1@L_{pc}}$$

$$\frac{\iota(n) = \mathsf{Push}\ m}{\mu\ [\sigma]\ n@L_{pc} \xrightarrow{\tau} \mu\ [m@\bot, \sigma]\ n{+}1@L_{pc}}$$

$$\frac{\iota(n) = \mathsf{Load} \qquad \mu(p) = m@L_2}{\mu\ [p@L_1, \sigma]\ n@L_{pc} \xrightarrow{\tau} \mu\ [m@L_1 \vee L_2, \sigma]\ n{+}1@L_{pc}}$$

$$\frac{\iota(n) = \mathsf{Store} \qquad \mu(p) = k@L_3 \qquad L_1 \vee L_{pc} \le L_3 \quad \mu(p) \leftarrow (m@L_1 \vee L_2 \vee L_{pc}) = \mu'}{\mu\ [p@L_1, m@L_2, \sigma]\ n@L_{pc} \xrightarrow{\tau} \mu'\ [\sigma]\ n{+}1@L_{pc}}$$

$$\frac{\iota(n) = \mathsf{Jump}}{\mu\ [n'@L_1, \sigma]\ n@L_{pc} \xrightarrow{\tau} \mu\ [\sigma]\ n'@(L_1 \vee L_{pc})}$$