# Micro-Policies

## A Framework for Verified, Hardware-Assisted Security Monitors


Cătălin Hrițcu

INRIA Paris

# Current collaborators on this project

- **Formal verification side**
  - Arthur Azevedo de Amorim (UPenn & INRIA Paris)
  - Maxime Dénès (UPenn)
  - Nick Giannarakis (INRIA Paris & NTU Athens)
  - Cătălin Hrițcu (INRIA Paris)
  - Benjamin Pierce (UPenn)
  - Antal Spector-Zabusky (UPenn)
  - Andrew Tolmach (Portland State)
- **Architecture side**
  - André DeHon, Udit Dhawan, Nikos Vasilakis, ... (UPenn)

# Computer systems are insecure

# Computer systems are insecure

- Today’s CPUs are mindless bureaucrats
  - “write past the end of this buffer” ... *yes boss!*
  - “jump to this untrusted integer” ... *right boss!*
  - “return into the middle of this instruction” ... *sure boss!*

# Computer systems are insecure

- Today's CPUs are mindless bureaucrats
  - "write past the end of this buffer" ... *yes boss!*
  - "jump to this untrusted integer" ... *right boss!*
  - "return into the middle of this instruction" ... *sure boss!*
- Software bears most of the burden for security

  pervasive security enforcement impractical
  - security-performance tradeoff

  just write secure code ... all of it!

# Computer systems are insecure

- Today’s CPUs are mindless bureaucrats
  - “write past the end of this buffer” ... *yes boss!*
  - “jump to this untrusted integer” ... *right boss!*
  - “return into the middle of this instruction” ... *sure boss!*
- Software bears most of the burden for security

  pervasive security enforcement impractical
  - security-performance tradeoff

  just write secure code ... all of it!
- Consequence: vulnerabilities in every system
  - violations of known safety and security policies


# Micro-policies

- general dynamic enforcement mechanism for
  - critical invariants of **all** machine code
  - high-level abstractions and programming models

# Micro-policies

- general dynamic enforcement mechanism for
  - critical invariants of **all** machine code
  - high-level abstractions and programming models
- main idea: add **word-sized tag** to each machine word
  - “this word is an instruction, and this one is a pointer”
    “this word comes from the net, and this is private to A and B”

# Micro-policies

- general dynamic enforcement mechanism for
  - critical invariants of **all** machine code
  - high-level abstractions and programming models
- main idea: add **word-sized tag** to each machine word
  - “this word is an instruction, and this one is a pointer”
    “this word comes from the net, and this is private to A and B”
- tags efficiently propagated on each instruction
  - tags and rules **defined by software** (miss handler; verified)
    **accelerated by hardware** (rule cache, near-0 overhead hits)

# Micro-policies

- general dynamic enforcement mechanism for
  - critical invariants of **all** machine code
  - high-level abstractions and programming models
- main idea: add **word-sized tag** to each machine word
  - "this word is an instruction, and this one is a pointer"
    "this word comes from the net, and this is private to A and B"
- tags efficiently propagated on each instruction
  - tags and rules **defined by software** (miss handler; verified)
    **accelerated by hardware** (rule cache, near-0 overhead hits)

  low overhead: <10% runtime, <50% energy, <12% power

# Micro-policies for ...

- information flow control (IFC)

# Micro-policies for ...

- information flow control (IFC)
- monitor self-protection
- dynamic sealing
  compartmentalization
  memory safety
- control-flow integrity (CFI)
- hardware types (instr/ptr/...)
  taint tracking
- ...

# Micro-policies for ...

- information flow control (IFC) [POPL 2014]
- monitor self-protection
- dynamic sealing
  compartmentalization
  memory safety
- control-flow integrity (CFI)
- hardware types (instr/ptr/...)
  taint tracking
- ...

recent draft

Verified
(in Coq)

# Micro-policies for ...

- information flow control (IFC) [POPL 2014]
- monitor self-protection
- dynamic sealing

  compartmentalization

  memory safety
- control-flow integrity (CFI)
- hardware types (instr/ptr/...)

  taint tracking
- ...

recent draft

Verified (in Coq)

Evaluated (simulations)

# Micro-policies for ...

- information flow control (IFC) [POPL 2014]
- monitor self-protection
- dynamic sealing

  compartmentalization

  **memory safety**
- control-flow integrity (CFI)
- hardware types (instr/ptr/...)

  taint tracking
- ...

recent draft

**Verified**

(in Coq)

Evaluated

(simulations)

# Memory safety

- Prevent
  - **spatial violations**: reading/writing out of bounds
  - **temporal violations**: use after free, invalid free

# Memory safety

- Prevent
  - **spatial violations**: reading/writing out of bounds
  - **temporal violations**: use after free, invalid free
- Pointers become **unforgeable capabilities** 
  - can only obtain a valid pointer to a memory region

    by allocating that region or

    - by copying/offsetting an existing pointer to that region

# Memory safety micro-policy

| | |
|---|---|
| $T_v$ ::= $i$ \| $ptr(c)$ | tags on values |
| $T_m$ ::= $M(c,T_v)$ \| $F$ | tags on memory |

# Memory safety micro-policy

color of region

$T_v$ ::= i | ptr(c)    tags on values

$T_m$ ::= M(c,$T_v$) | F    tags on memory

# Memory safety micro-policy

color of region

$T_v$ ::= i | ptr(c)        tags on values

$T_m$ ::= M(c,$T_v$) | F     tags on memory

color of region

tag of content

# Memory safety micro-policy

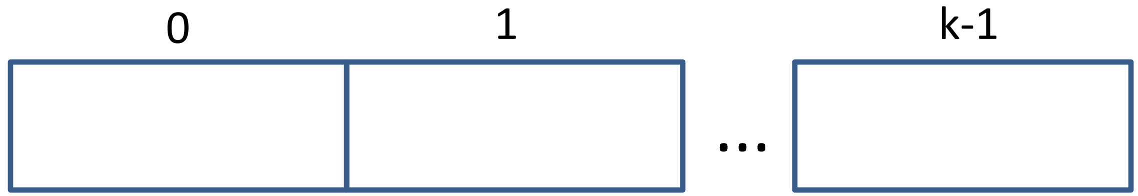
**p←malloc k**

$T_v$ ::= i | ptr(c)       tags on values

$T_m$ ::= M(c,$T_v$) | F     tags on memory

# Memory safety micro-policy

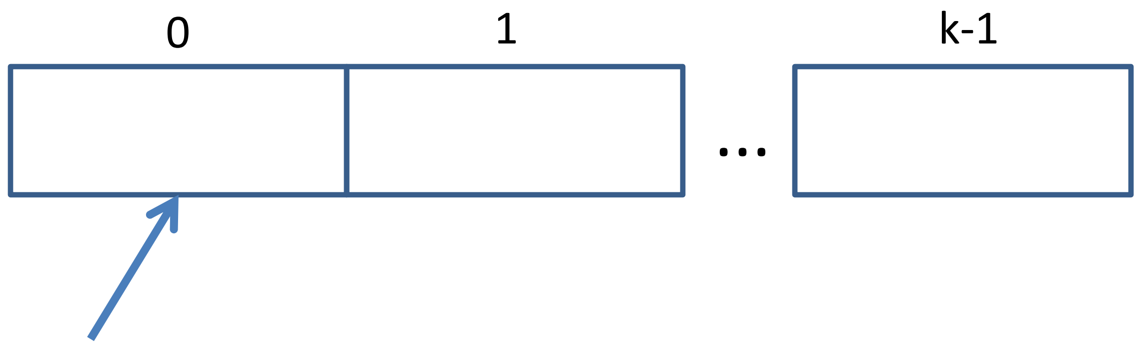
**p←malloc k**




| | | |
|---|---|---|
| $T_v$ | $::= i \mid ptr(c)$ | tags on values |
| $T_m$ | $::= M(c,T_v) \mid F$ | tags on memory |

# Memory safety micro-policy

**p←malloc k**



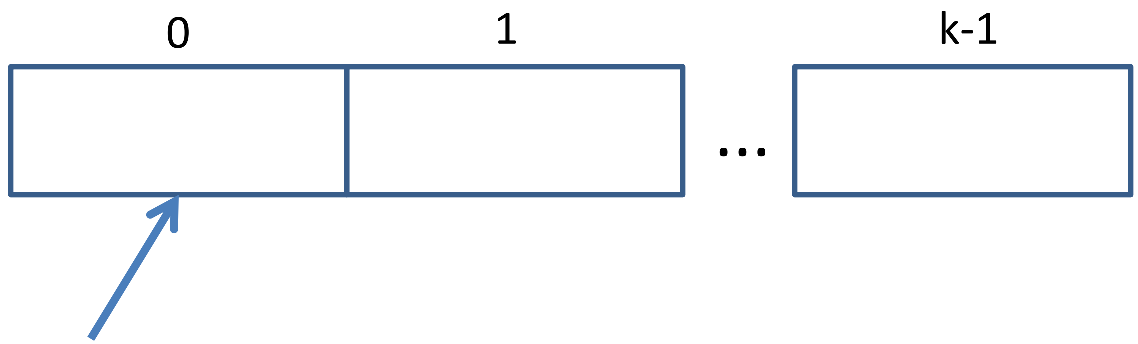

p = A8F0

| | |
|---|---|
| $T_v$ ::= i \| ptr(c) | tags on values |
| $T_m$ ::= M(c,$T_v$) \| F | tags on memory |

# Memory safety micro-policy

**p←malloc k**

fresh c




p = A8F0

$T_v$ ::= i | ptr(c)   tags on values

$T_m$ ::= M(c,$T_v$) | F   tags on memory

# Memory safety micro-policy

**p←malloc k**

fresh c

| 0 | 1 | | k-1 |
|---|---|---|---|
| 0@M(c,i) | 0@M(c,i) | … | 0@M(c,i) |

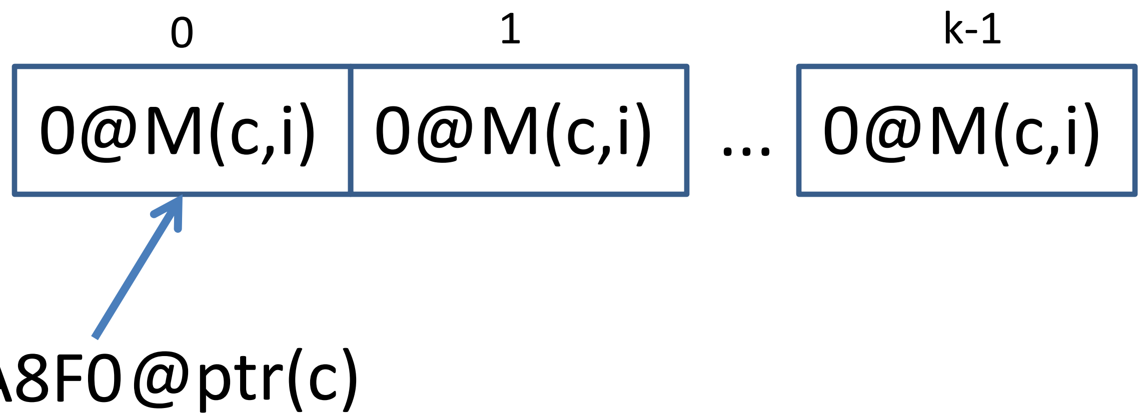
p = A8F0

| | |
|---|---|
| $T_v$ ::= i \| ptr(c) | tags on values |
| $T_m$ ::= M(c,$T_v$) \| F | tags on memory |

# Memory safety micro-policy

**p←malloc k**

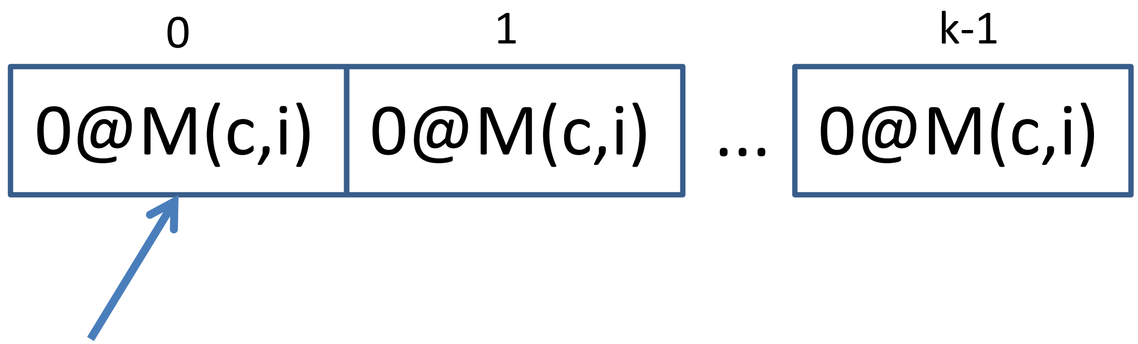
fresh c




$T_v$ ::= i | ptr(c)  tags on values

$T_m$ ::= M(c,$T_v$) | F  tags on memory

# Memory safety micro-policy

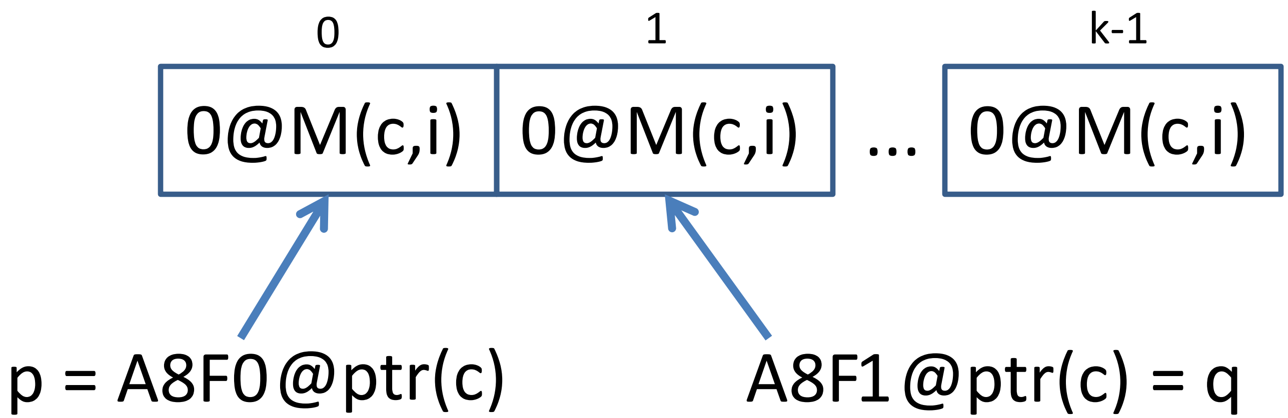



p = A8F0@ptr(c)

**q ← p + 1**

| | | |
|---|---|---|
| $T_v$ | ::= i \| ptr(c) | tags on values |
| $T_m$ | ::= M(c,$T_v$) \| F | tags on memory |

# Memory safety micro-policy

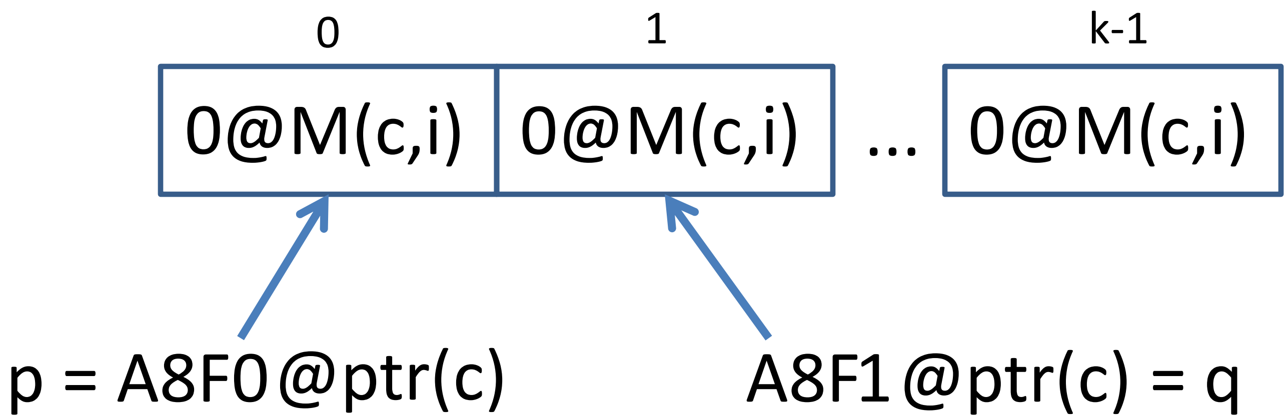



**q ← p + 1**

$T_v$ ::= i | ptr(c) tags on values

$T_m$ ::= M(c,$T_v$) | F tags on memory

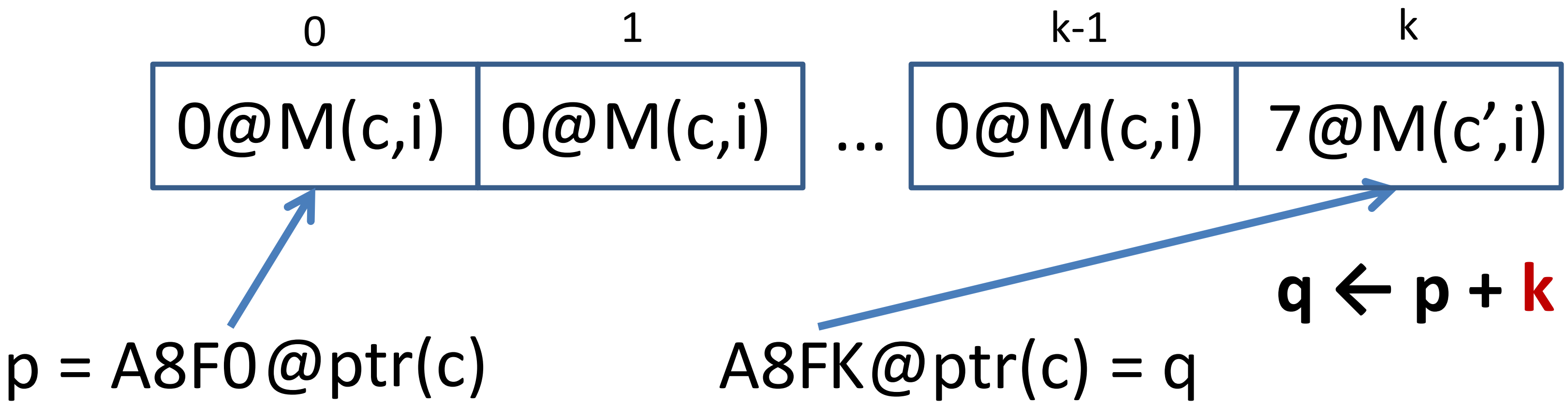
# Memory safety micro-policy




**q ← p + k**

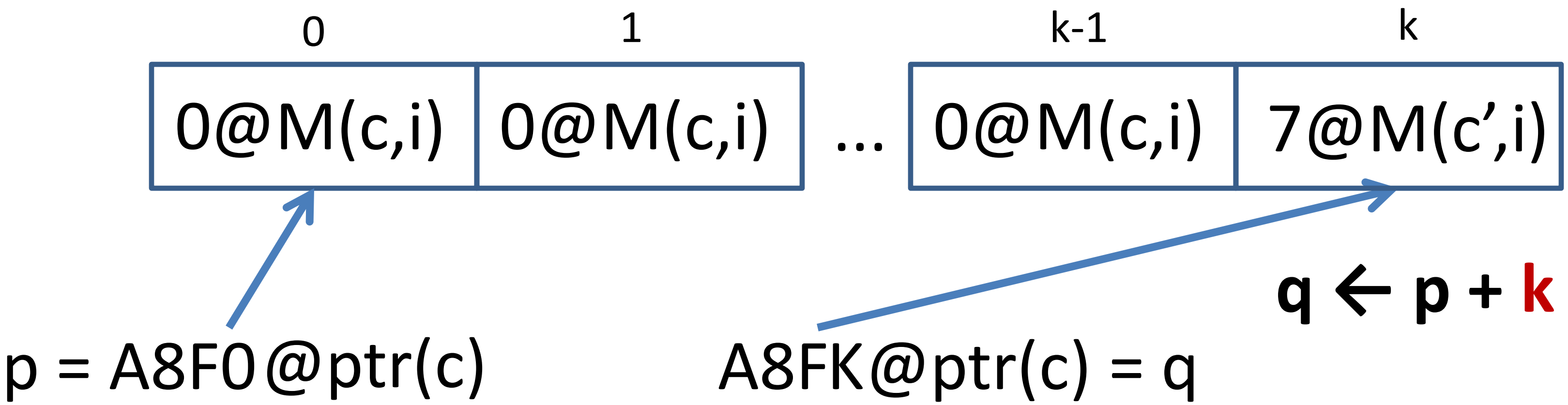
| | |
|---|---|
| $T_v$ ::= i \| ptr(c) | tags on values |
| $T_m$ ::= M(c,$T_v$) \| F | tags on memory |

# Memory safety micro-policy




| | | |
|---|---|---|
| $T_v$ | ::= i \| ptr(c) | tags on values |
| $T_m$ | ::= $M(c,T_v)$ \| F | tags on memory |

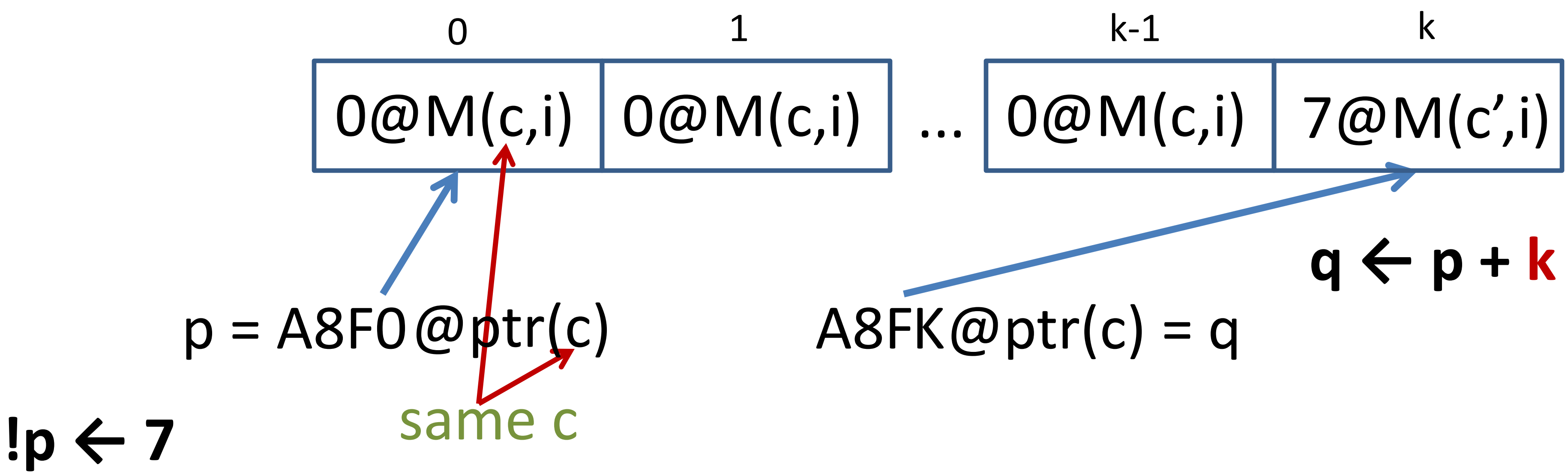
# Memory safety micro-policy




**!p ← 7**

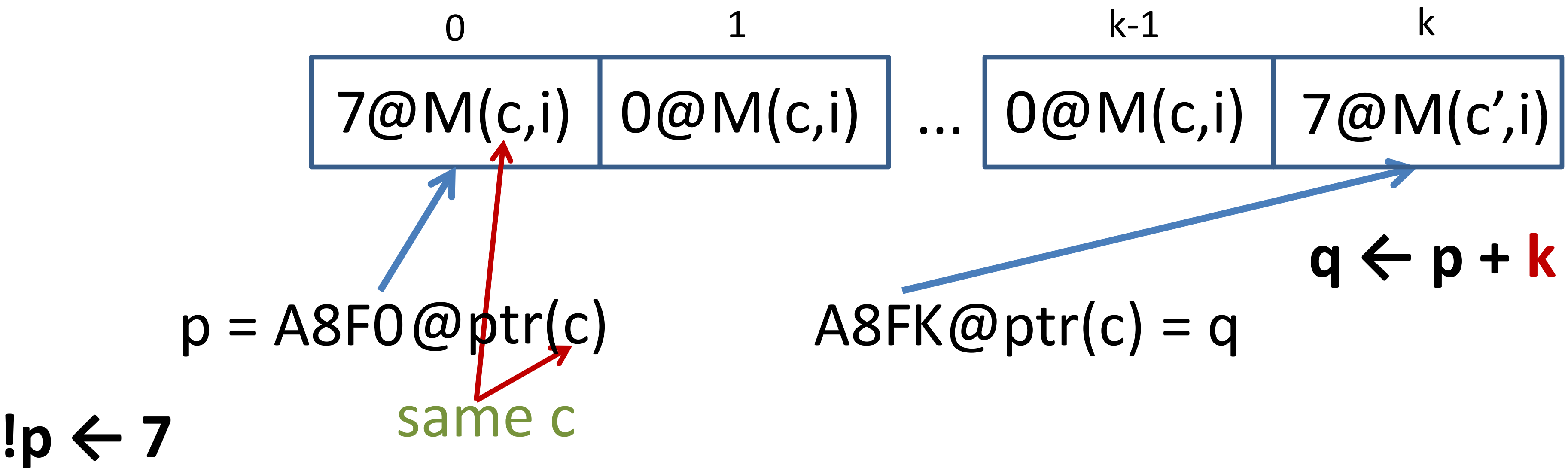
| | | |
|---|---|---|
| $T_v$ | ::= i \| ptr(c) | tags on values |
| $T_m$ | ::= M(c,$T_v$) \| F | tags on memory |

# Memory safety micro-policy




| | | |
|---|---|---|
| $T_v$ | ::= i \| ptr(c) | tags on values |
| $T_m$ | ::= M(c,$T_v$) \| F | tags on memory |

# Memory safety micro-policy




$T_v$ ::= i | ptr(c) — tags on values

$T_m$ ::= M(c,$T_v$) | F — tags on memory

# Memory safety micro-policy

| 0 | 1 | | k-1 | k |
|---|---|---|---|---|
| 7@M(c,i) | 0@M(c,i) | … | 0@M(c,i) | 7@M(c’,i) |

**q ← p + k**

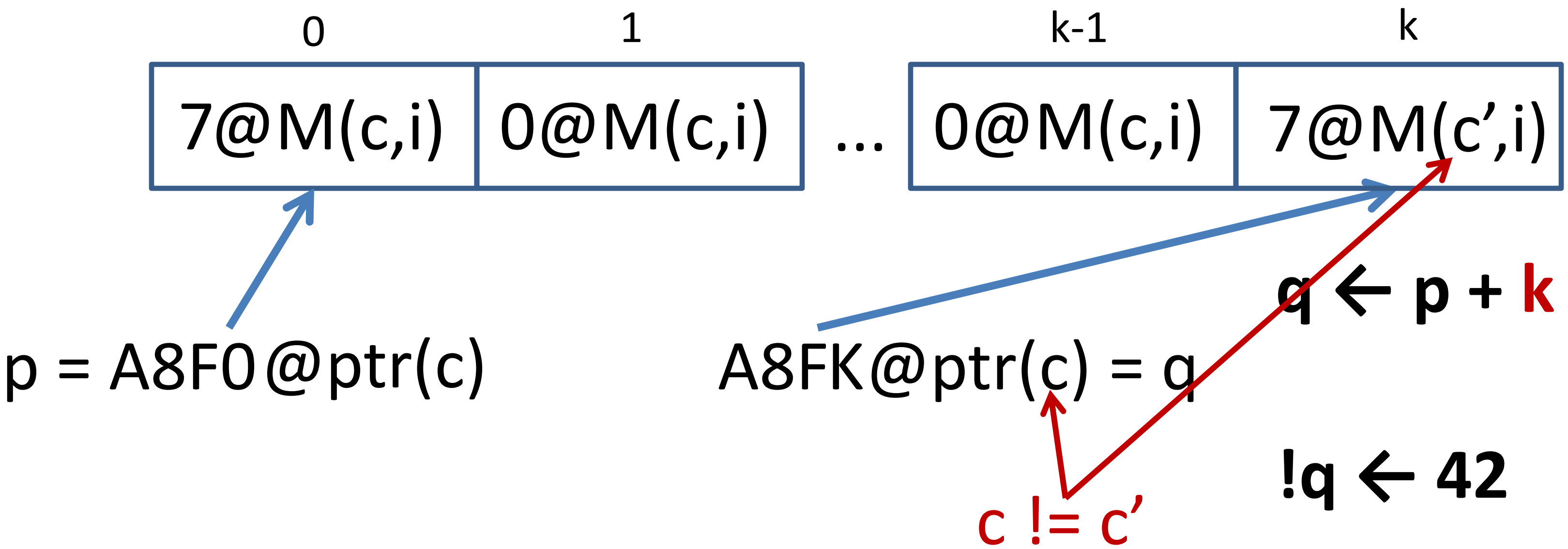
p = A8F0@ptr(c)

A8FK@ptr(c) = q

**!q ← 42**

| | | |
|---|---|---|
| $T_v$ | ::= i \| ptr(c) | tags on values |
| $T_m$ | ::= $M(c,T_v)$ \| F | tags on memory |

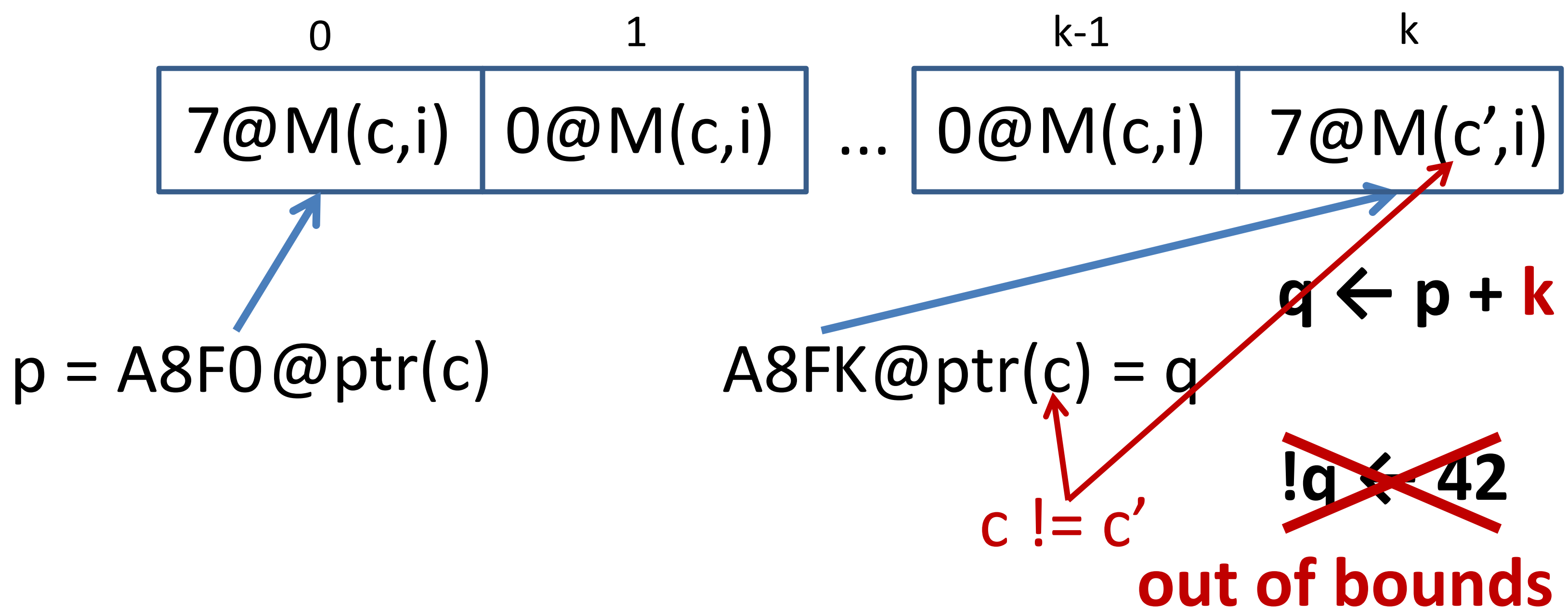
# Memory safety micro-policy




$T_v$ ::= i | ptr(c) tags on values

$T_m$ ::= M(c,$T_v$) | F tags on memory

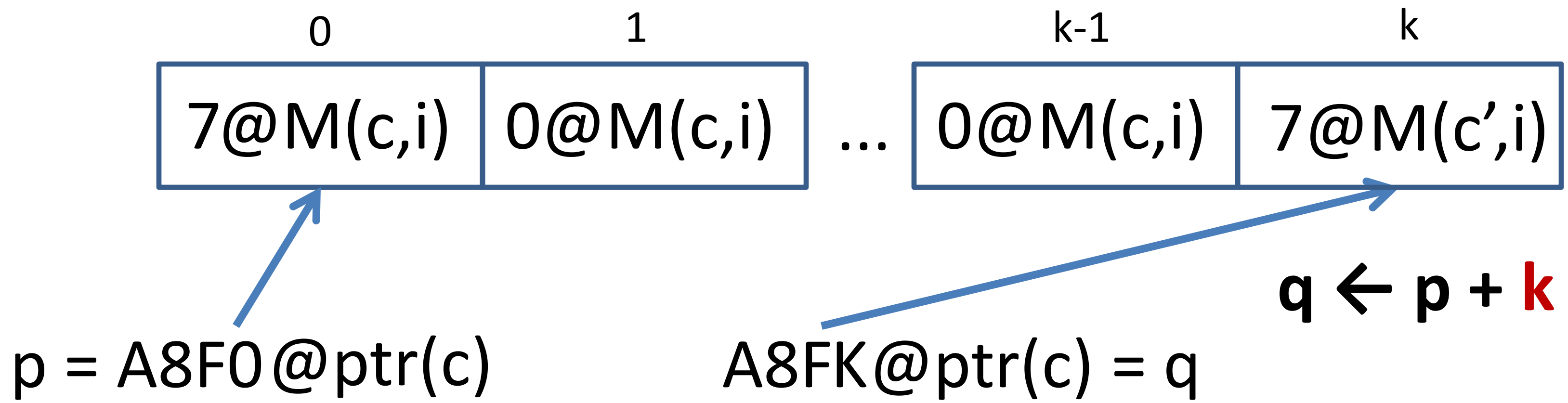
# Memory safety micro-policy




$T_v$ ::= i | ptr(c) tags on values

$T_m$ ::= M(c,$T_v$) | F tags on memory

# Memory safety micro-policy




**q ← p + k**

**~~!q ← 42~~**

**out of bounds**

**free p**

| | | |
|---|---|---|
| $T_v$ | ::= i \| ptr(c) | tags on values |
| $T_m$ | ::= $M(c,T_v)$ \| F | tags on memory |

# Memory safety micro-policy

| 0 | 1 | | k-1 | k |
|---|---|---|---|---|
| 7@F | 0@F | … | 0@F | 7@M(c',i) |

p = A8F0@ptr(c)

**q ← p + k**

A8FK@ptr(c) = q

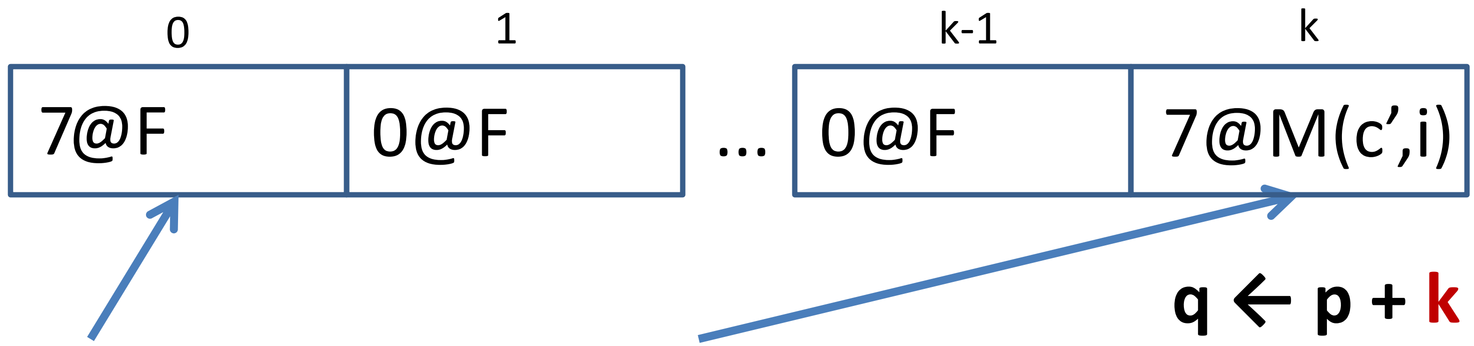
~~**!q ← 42**~~

**out of bounds**

**free p**

| | | |
|---|---|---|
| $T_v$ ::= i \| ptr(c) | | tags on values |
| $T_m$ ::= $M(c,T_v)$ \| F | | tags on memory |

# Memory safety micro-policy




**q ← p + k**

p = A8F0@ptr(c)

A8FK@ptr(c) = q

**~~!q ← 42~~**

**out of bounds**

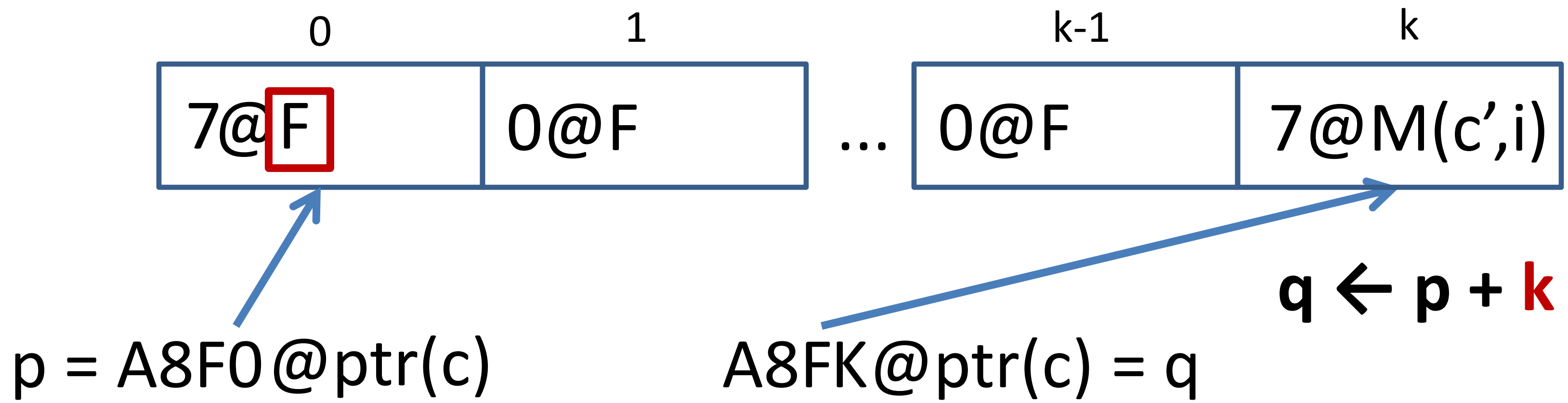
**free p**

**x ← !p**

$T_v$ ::= i | ptr(c)   tags on values

$T_m$ ::= M(c,$T_v$) | F   tags on memory

# Memory safety micro-policy




**q ← p + k**

~~**!q ← 42**~~

**out of bounds**

**free p**

**x ← !p**

| | |
|---|---|
| $T_v$ ::= i \| ptr(c) | tags on values |
| $T_m$ ::= $M(c,T_v)$ \| F | tags on memory |

# Memory safety micro-policy

| 0 | 1 | | k-1 | k |
|---|---|---|---|---|
| 7@$\boxed{\text{F}}$ | 0@F | … | 0@F | 7@M(c′,i) |

**q ← p + k**

p = A8F0@ptr(c)

A8FK@ptr(c) = q

~~**!q ← 42**~~

**out of bounds**

**free p**

~~**x ← !p**~~

**use after free**

| | | |
|---|---|---|
| $T_v$ ::= i \| ptr(c) | | tags on values |
| $T_m$ ::= M(c,$T_v$) \| F | | tags on memory |

# Memory safety micro-policy

# Memory safety micro-policy

**1. Sets of tags**

$T_v$ ::= i | ptr(c)

$T_m$ ::= M(c,$T_v$) | F

$T_{pc}$ ::= $T_v$

# Memory safety micro-policy

**1. Sets of tags**

$T_v$ ::= i | ptr(c)

$T_m$ ::= M(c,$T_v$) | F

$T_{pc}$ ::= $T_v$

**2. Transfer function**

Record IVec := { op:opcode ; $t_{pc}$:$T_{pc}$ ; $t_i$:$T_m$ ; ts: ... }

Record OVec (op:opcode) := { $t_{rpc}$ : $T_{pc}$ ; $t_r$ : ... }

transfer : (iv:IVec) -> option (OVec (op iv))

# Memory safety micro-policy

**1. Sets of tags**

$T_v$ ::= i | ptr(c)

$T_m$ ::= M(c,$T_v$) | F

$T_{pc}$ ::= $T_v$

**2. Transfer function**

Record IVec := { op:opcode ; $t_{pc}$:$T_{pc}$ ; $t_i$:$T_m$ ; ts: ... }

Record OVec (op:opcode) := { $t_{rpc}$ : $T_{pc}$ ; $t_r$ : ... }

transfer : (iv:IVec) -> option (OVec (op iv))

Definition transfer iv :=

match iv with

| {op=Load; $t_{pc}$=ptr($c_{pc}$); $t_i$=M($c_{pc}$,i); ts=[ptr(c); M(c,$T_v$)]}

=> $_{\{t}$rpc=pt$_{r(}$cpc)$_{;}$ $t_r$=Tv}

# Memory safety micro-policy

**1. Sets of tags**

$T_v$ ::= i | ptr(c)

$T_m$ ::= $M(c,T_v)$ | F

$T_{pc}$ ::= $T_v$

**2. Transfer function**

Record IVec := { op:opcode ; $t_{pc}$:$T_{pc}$ ; $t_i$:$T_m$ ; ts: ... }

Record OVec (op:opcode) := { $t_{rpc}$ : $T_{pc}$ ; $t_r$ : ... }

transfer : (iv:IVec) -> option (OVec (op iv))

Definition transfer iv :=

match iv with

| {op=Load; $t_{pc}$=ptr($c_{pc}$); $t_i$=M($c_{pc}$,i); ts=[ptr(c); M(c,$T_v$)]}

=> $\{t_{rpc}=ptr(c_{pc}); t_r=Tv\}$

| {op=Store; $t_{pc}$=ptr($c_{pc}$); $t_i$=M($c_{pc}$,i); ts=[ptr(c); $T_v$; M(c,$T_v$')]}

=> {$t_{rpc}$=ptr($c_{pc}$); $t_r$=M(c,$T_v$)}

...

# Memory safety micro-policy

**1. Sets of tags**

$T_v$ ::= i | ptr(c)

$T_m$ ::= M(c,$T_v$) | F

$T_{pc}$ ::= $T_v$

**2. Transfer function**

Record IVec := { op:opcode ; $t_{pc}$:$T_{pc}$ ; $t_i$:$T_m$ ; ts: ... }

Record OVec (op:opcode) := { $t_{rpc}$ : $T_{pc}$ ; $t_r$ : ... }

transfer : (iv:IVec) -> option (OVec (op iv))

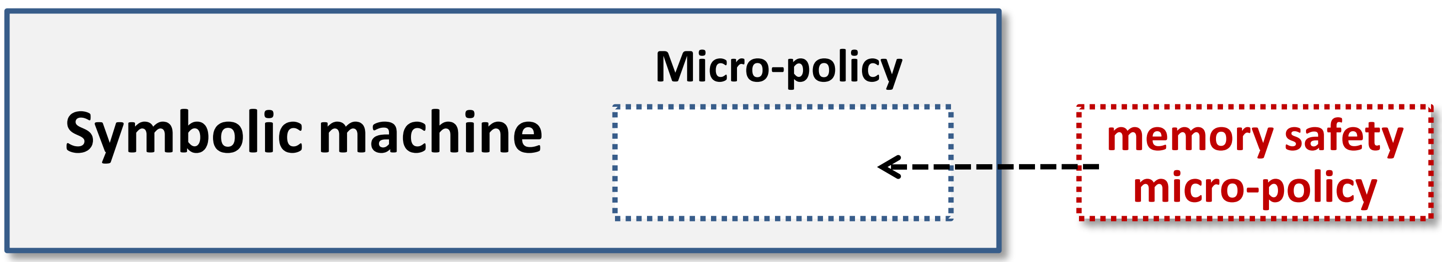
**3. Monitor services**

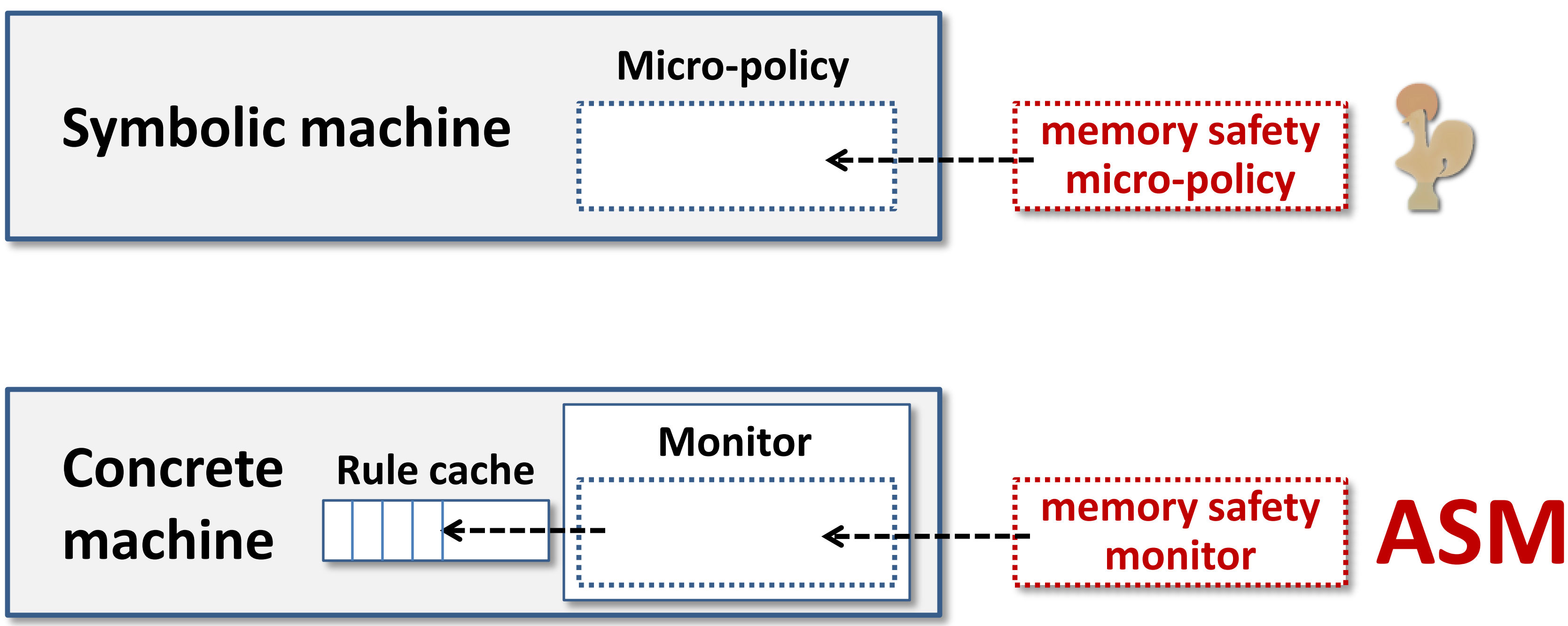
Record service := { addr : word; sem : state -> option state; ... }

Definition mem_safety_services : list service :=

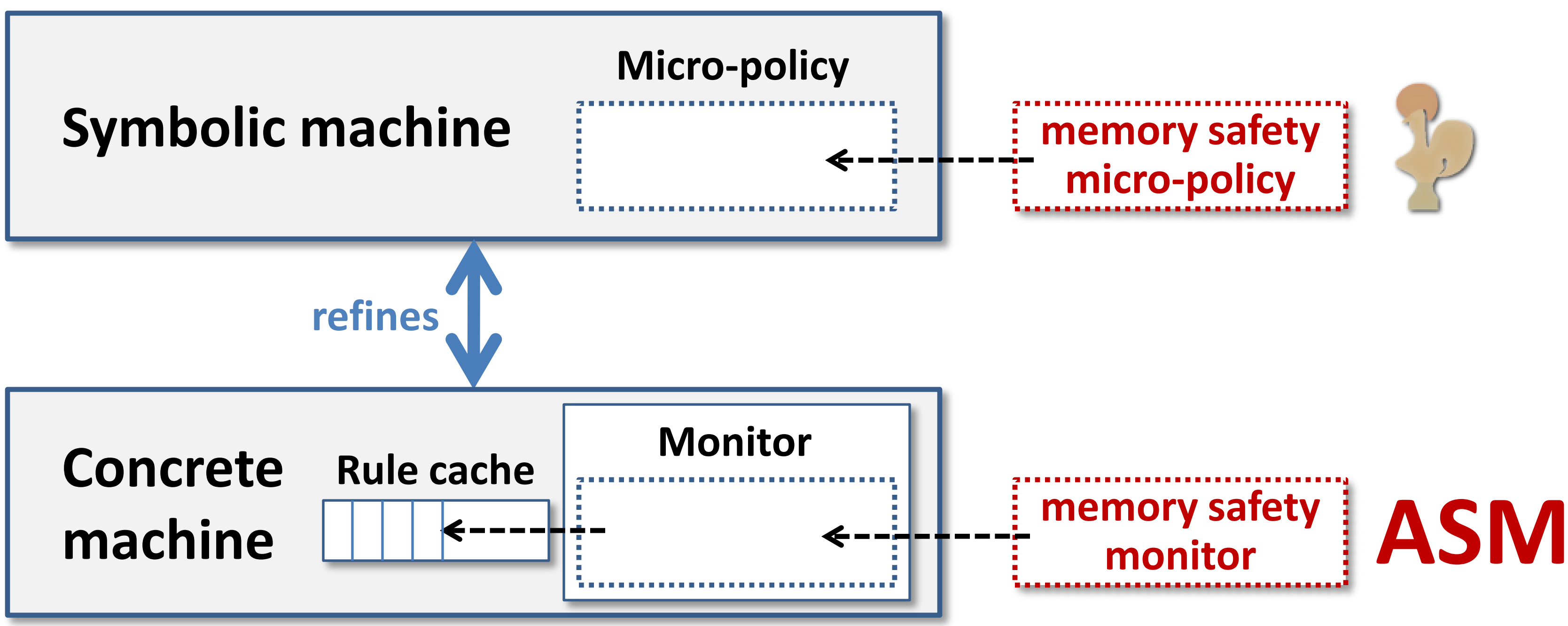
[malloc; free; size; base; eq].

**memory safety micro-policy**

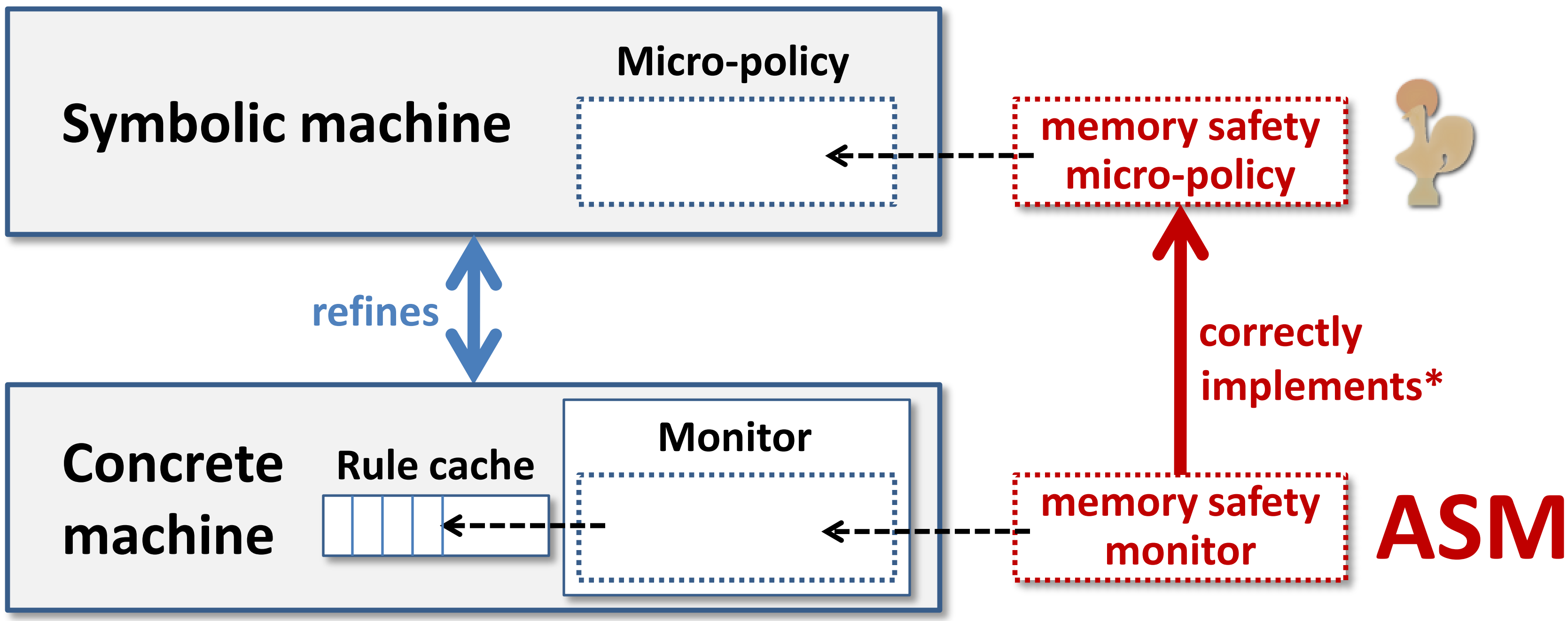

Symbolic machine
Micro-policy
memory safety
micro-policy


Symbolic machine
Micro-policy
memory safety
micro-policy
Concrete
machine
Rule cache
Monitor
memory safety
monitor
ASM

# Verification

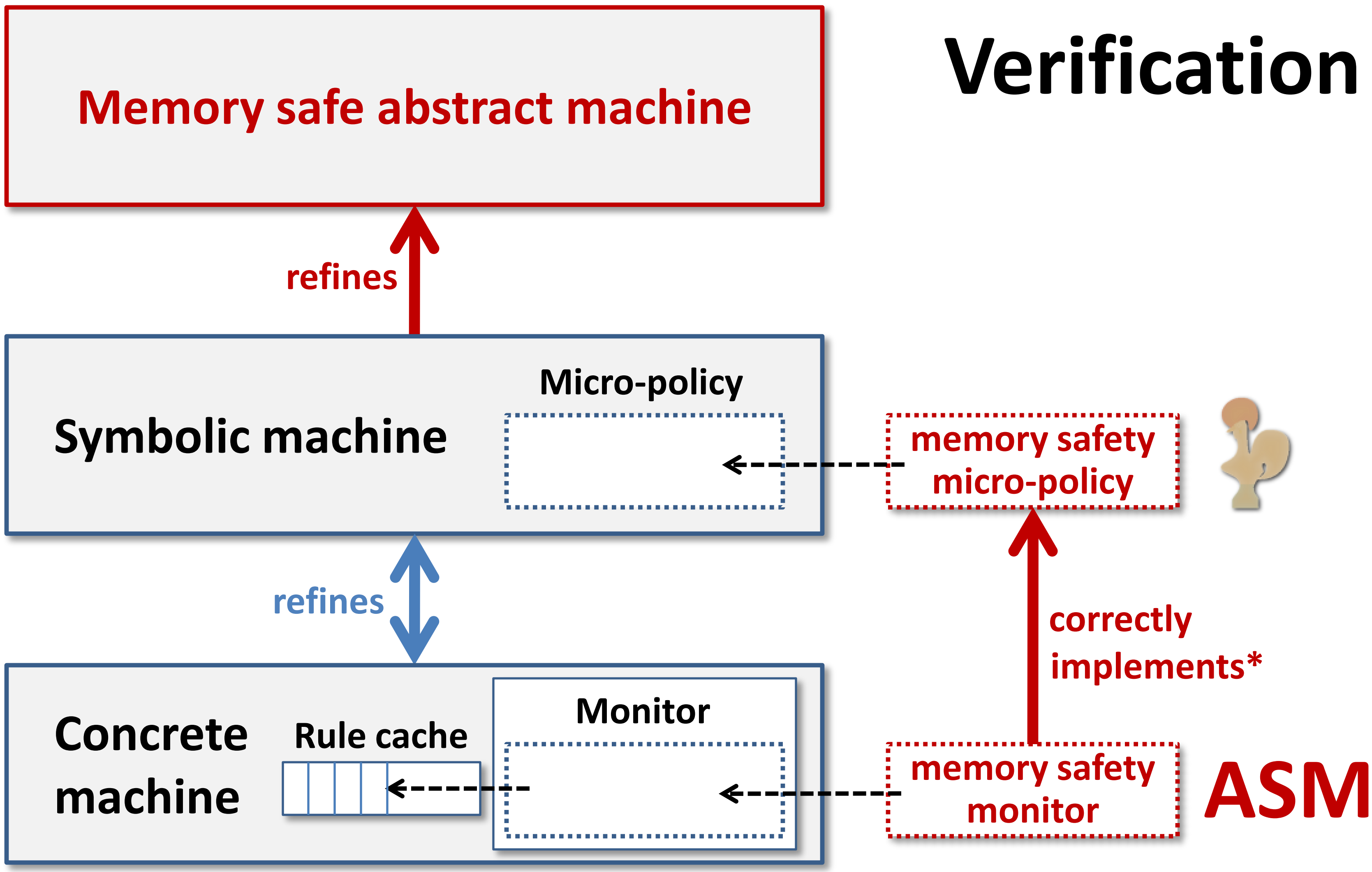


# Verification



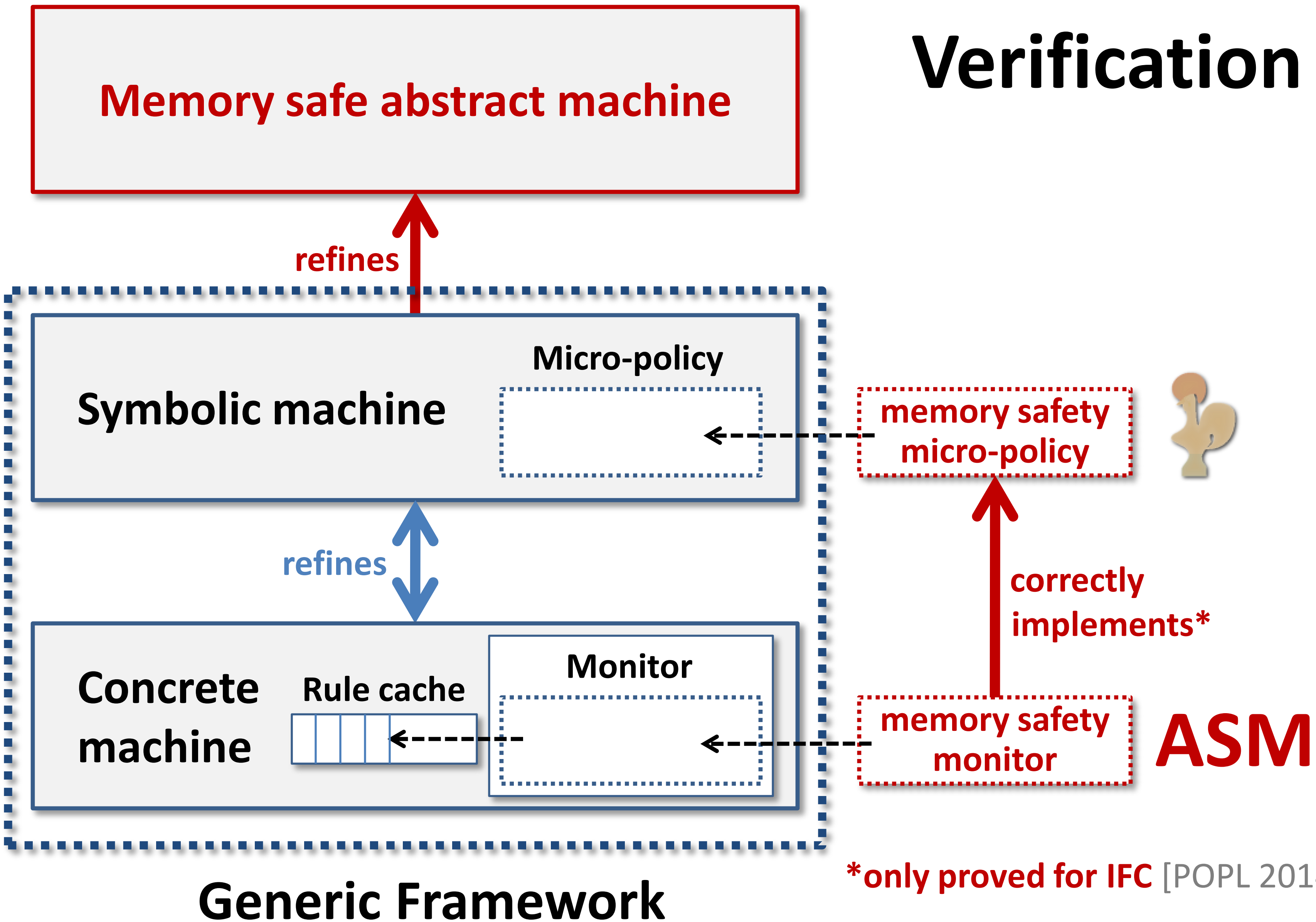

*only proved for IFC [POPL 2014]

# Verification



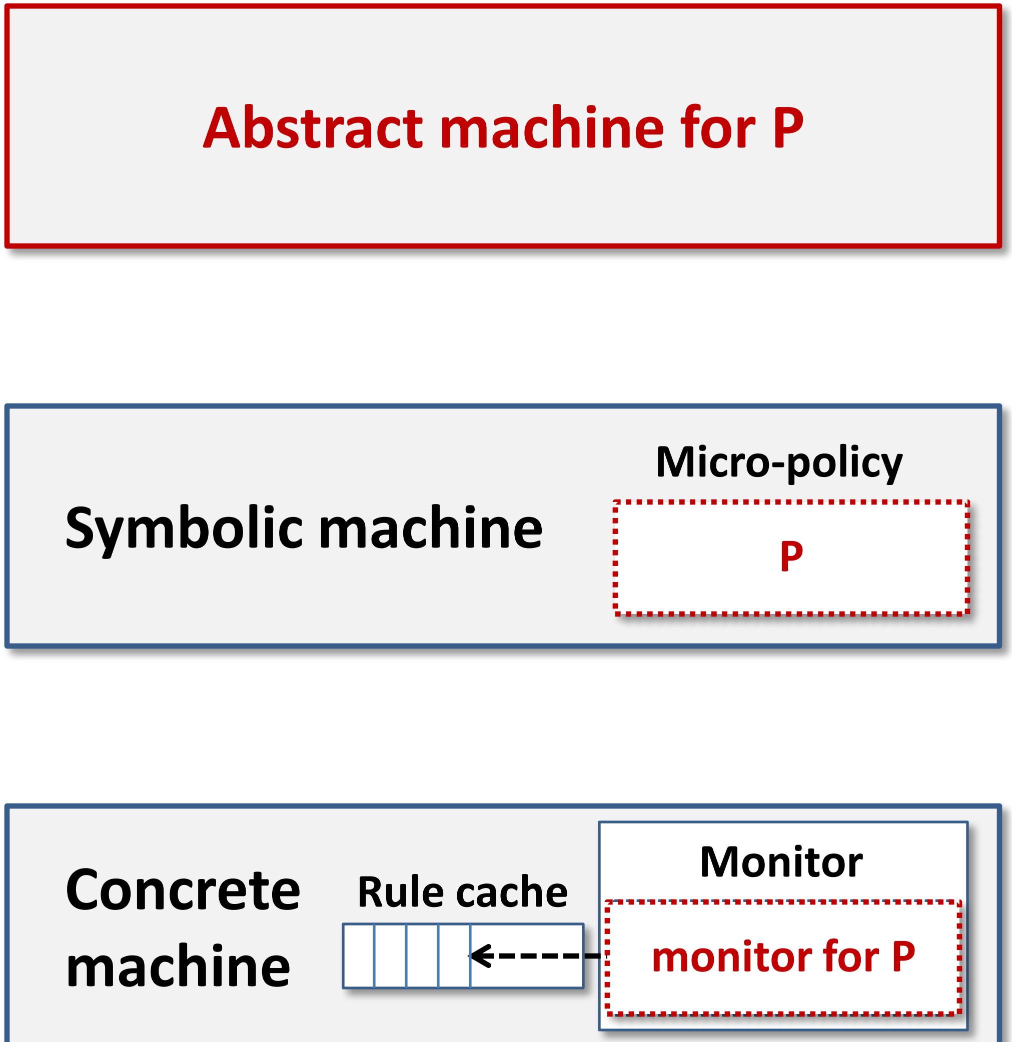

*only proved for IFC [POPL 2014]

# Verification

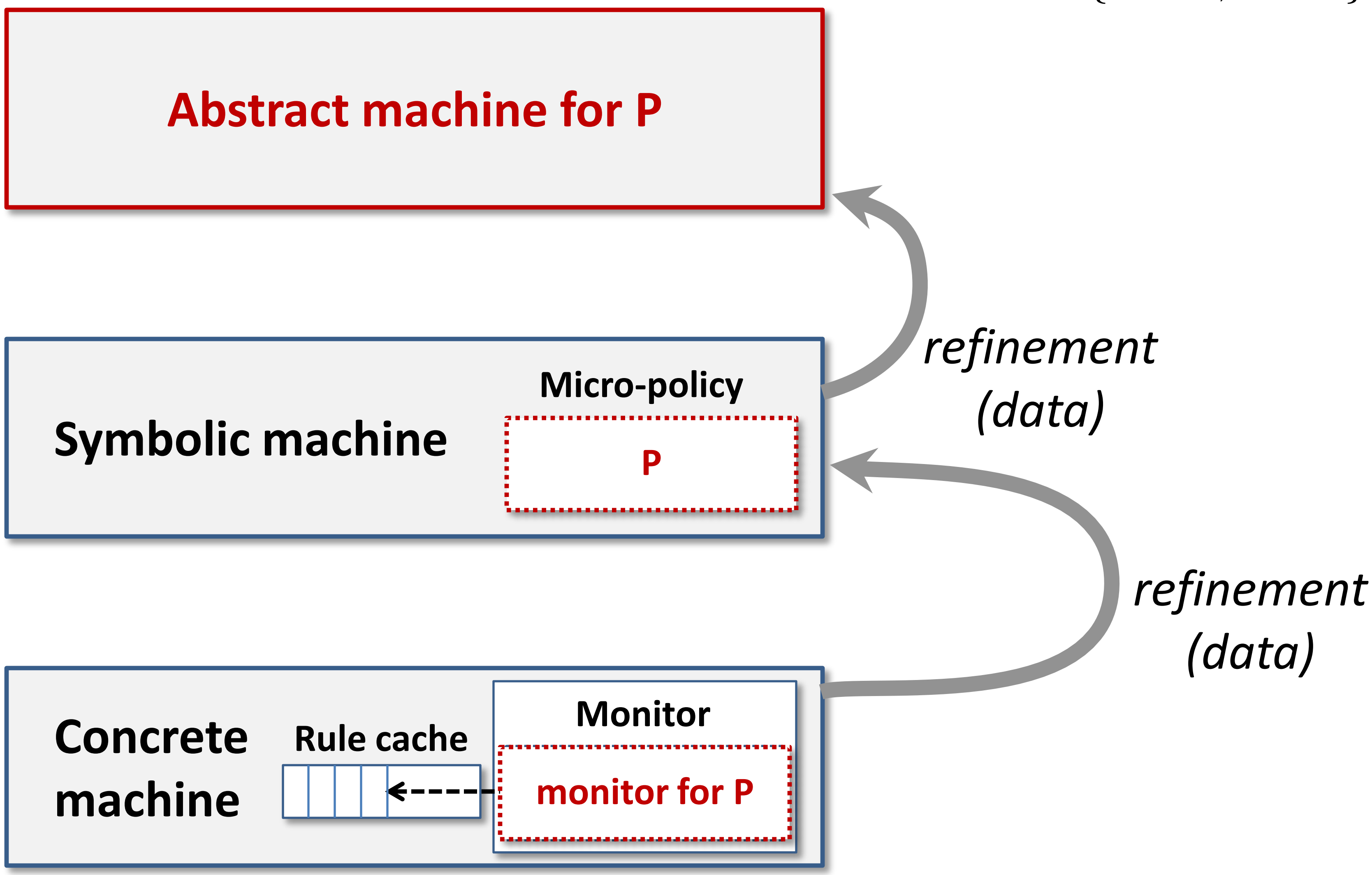



*only proved for IFC [POPL 2014]

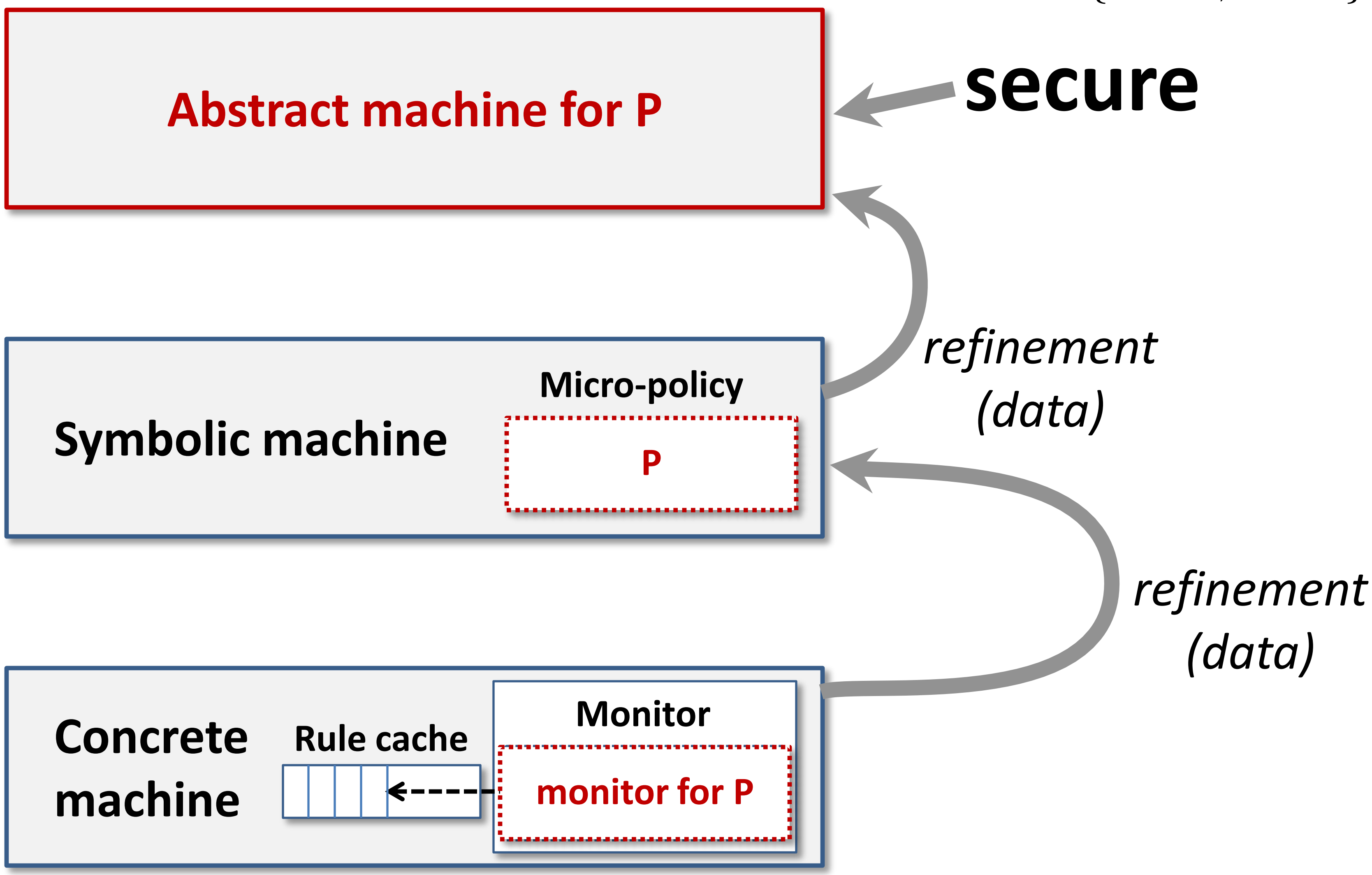
$\mathsf{P} \in \{IFC, CFI\}$

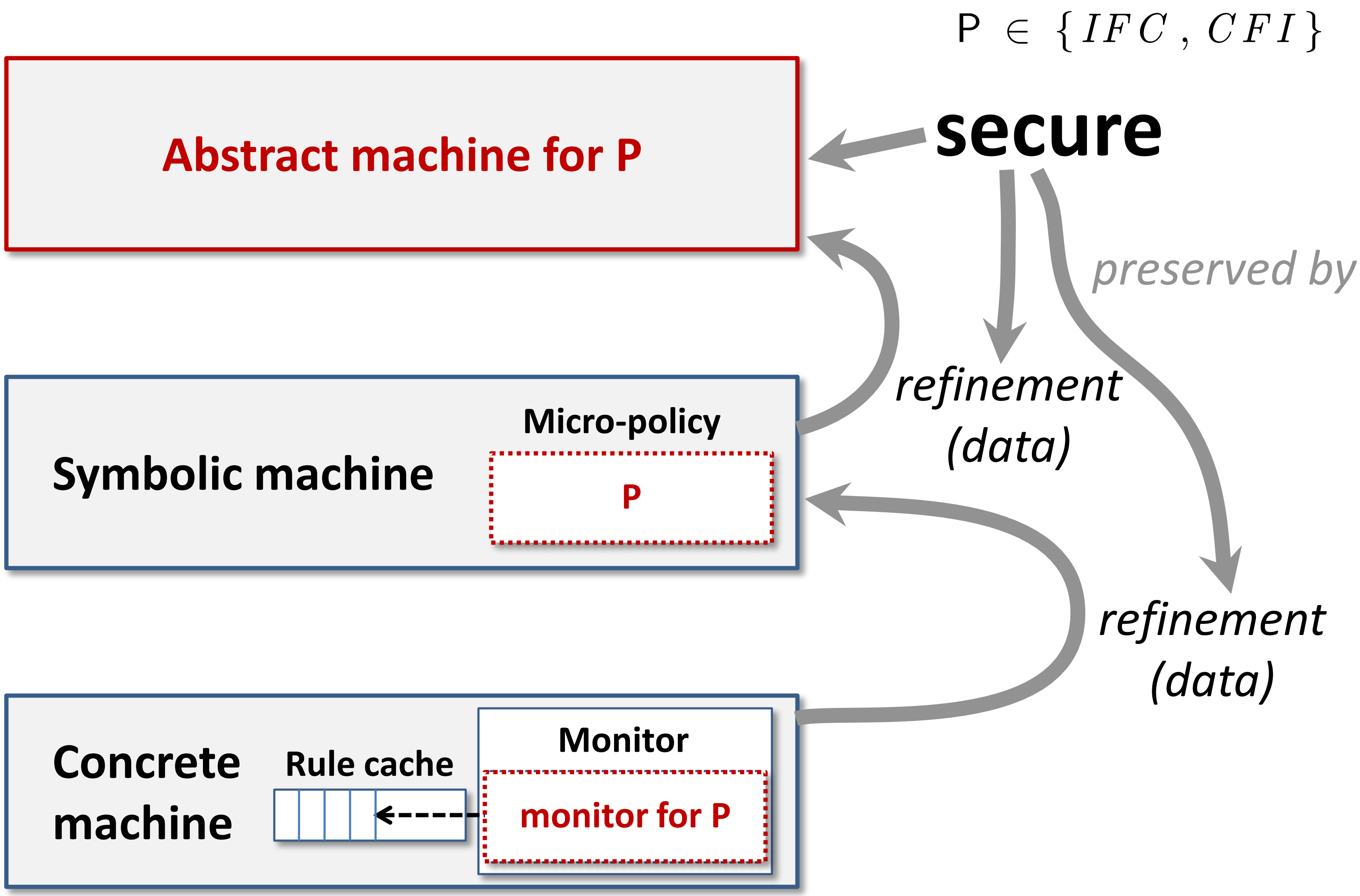


$\mathsf{P} \in \{IFC, CFI\}$

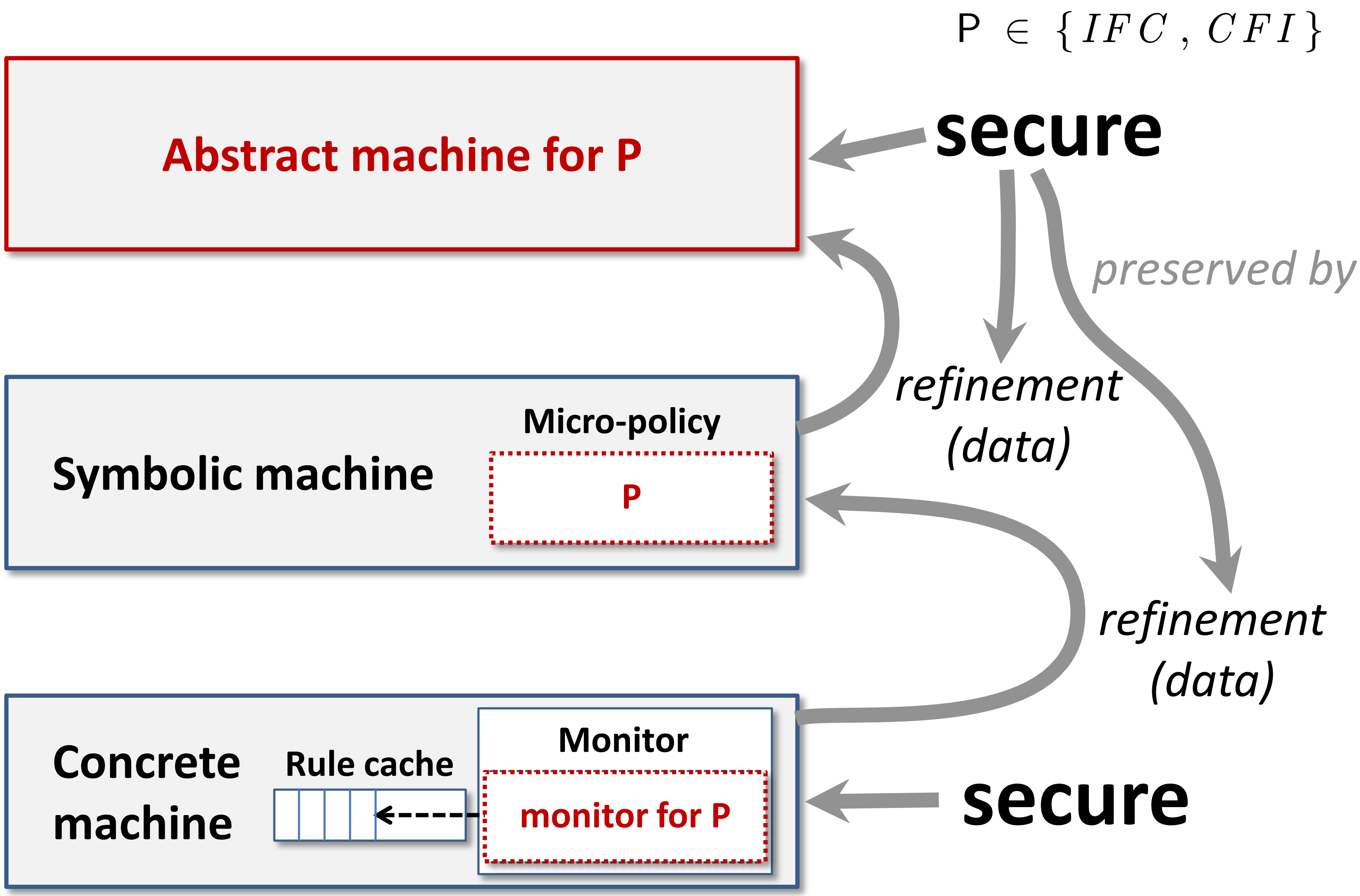


$\mathsf{P} \in \{IFC, CFI\}$





$P \in \{IFC, CFI\}$
secure
preserved by
Abstract machine for P
refinement
(data)
Symbolic machine
Micro-policy
P
refinement
(data)
Concrete machine
Rule cache
Monitor
monitor for P


P ∈ {IFC, CFI}
Abstract machine for P
secure
preserved by
refinement
(data)
Symbolic machine
Micro-policy
P
refinement
(data)
Concrete
machine
Rule cache
Monitor
monitor for P
secure

# Future verification challenges

1. Proofs for **real RISC architecture** (e.g. ARM)
2. Verify all monitors down to **machine-code level**
3. Formally study micro-policy **composition**
4. Devise **generic meta-language** for micro-policies
5. Study **more micro-policies** (e.g. stack protection, ...)
6. Formally study **expressive power** of micro-policies
7. Interaction with **loader** and **compiler** (static + dynamic)
8. ... and **operating system** (e.g. protect the OS itself)