SNIP: Speculative Execution and Non-Interference Preservation for Compiler Transformations

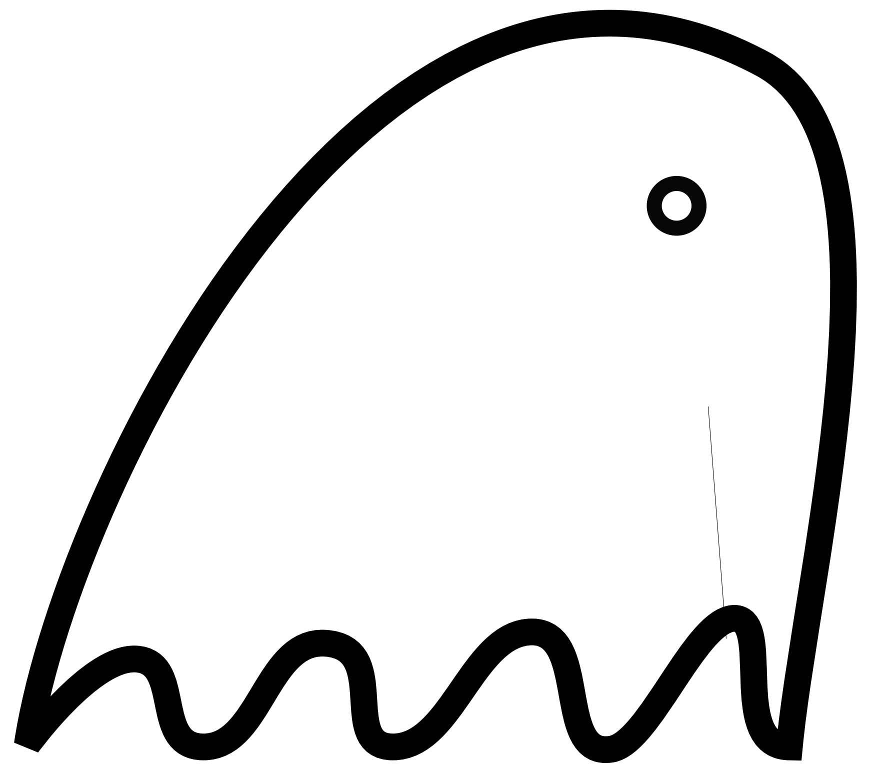
Your binaries are haunted by your compiler!

Sören van der Wall, Roland Meyer, PriSC 2025, Denver

$P :: [P]$☠

Speculative Execution

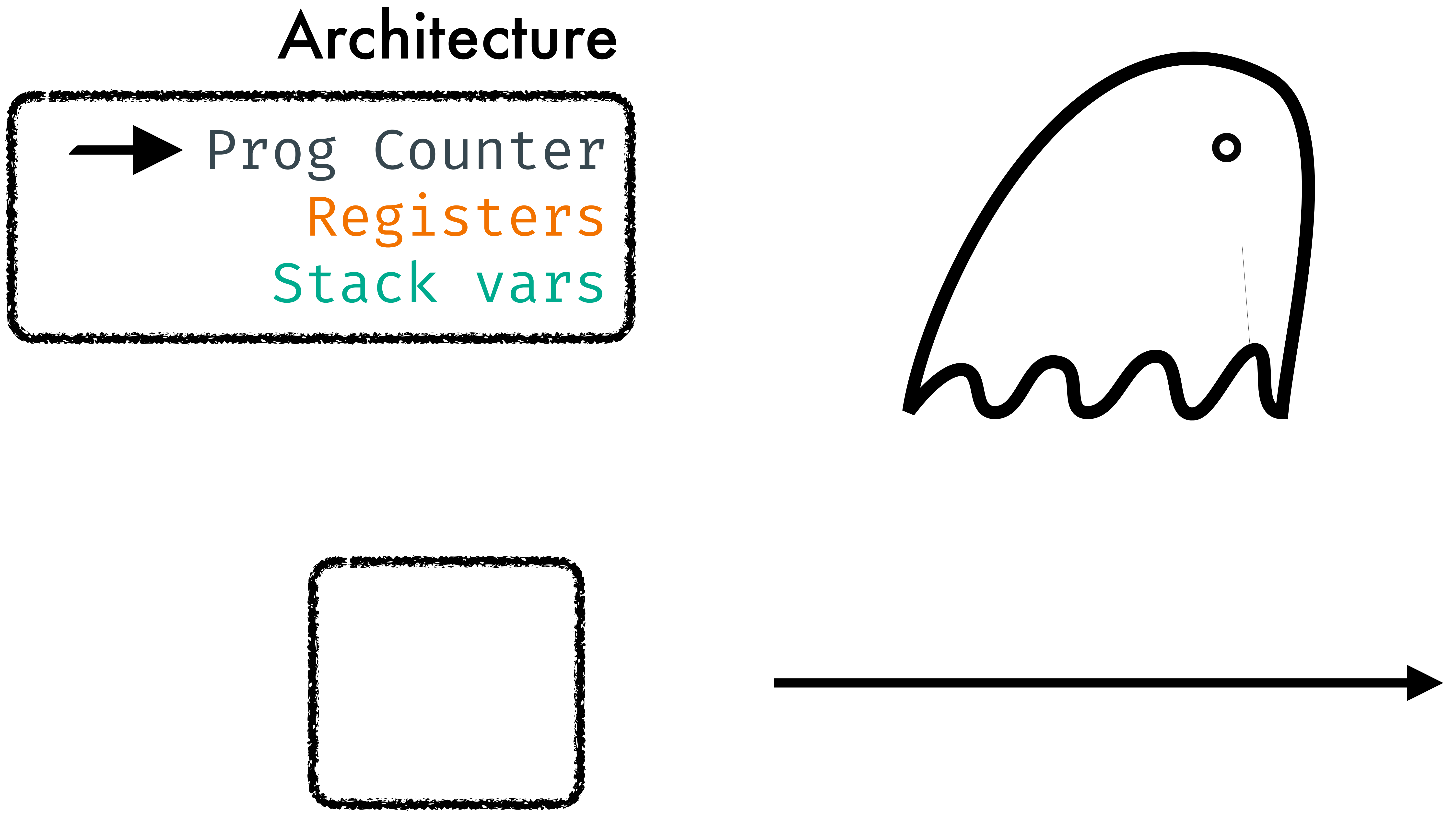
SpectreV1



Speculative Execution

SpectreV1

Architecture

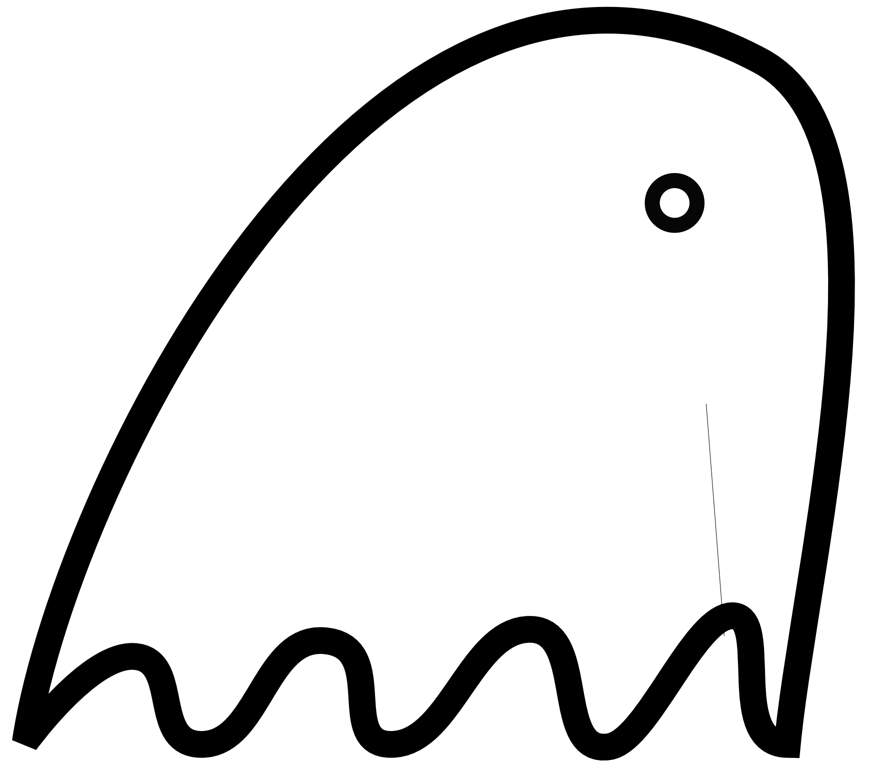


Speculative Execution

SpectreV1

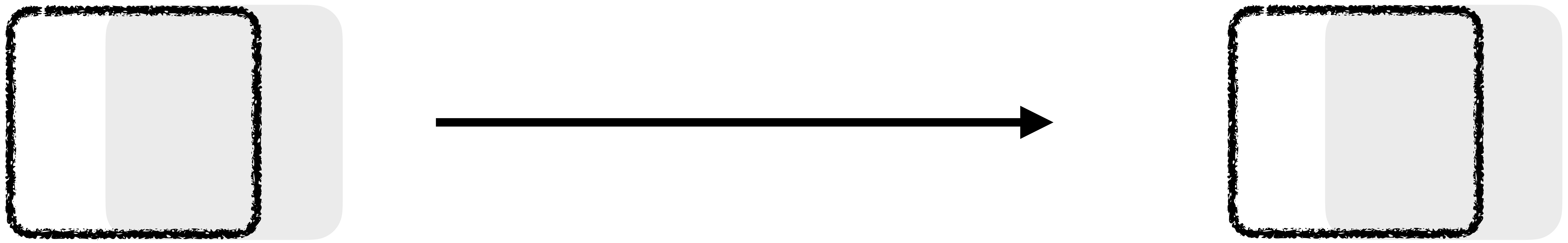
Architecture

→ Prog Counter
Registers
Stack vars



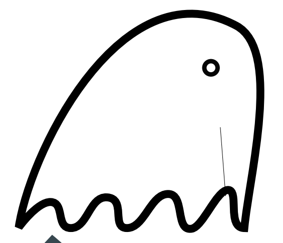
μ-Architecture

Caches
Reorder Buffer
Branch predictor



Speculative Execution

SpectreV1

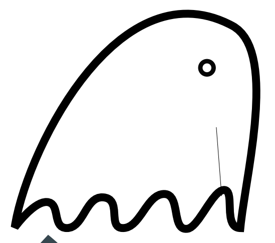


```
if (i < size) {
 a = buf[i];
 _ = buf2[a];
}
```

Registers
Stack Variables

Speculative Execution

SpectreV1



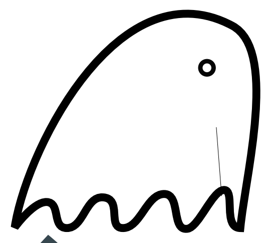
```
→ if (i < size) {
    a = buf[i];
    _ = buf2[a];
  }
```

```
i    : 20      size:  8
               sec : 42
```

Registers
Stack Variables

Speculative Execution

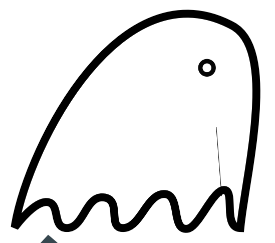
SpectreV1



```
         20   8
→ if (i < size) {
    a = buf[i];
    _ = buf2[a];
  }
```

```
i    : 20      size:  8
               sec : 42
```

Registers
Stack Variables

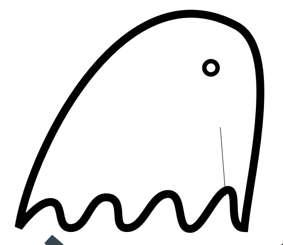


```
if (i < size) {
  a = buf[i];
  _ = buf2[a];
}
```

```
i    : 20      size:  8
               sec : 42
```

Registers
Stack Variables

Speculative Execution

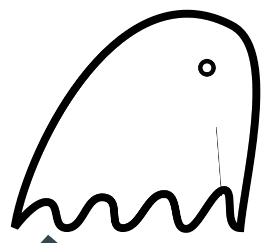


```
if (i < size) {
 a = buf[i];
 _ = buf2[a];
}
```

```
i    : 20      size:  8
               sec  : 42
```

Registers
Stack Variables

Speculative Execution



```
if (i < size) {
  a = buf[i];
  _ = buf2[a];
}
```

&buf + 20 = &sec

```
i    : 20      size:  8
               sec :  42
```

Registers
Stack Variables

Speculative Execution

```
if (i < size) {
 a = buf[i];
 _ = buf2[a];
}
```

&buf + 20 = &sec

```
i    : 20      size:  8
a    : 42      sec  : 42
```

Registers
Stack Variables

Speculative Execution

```
if (i < size) {
  a = buf[i];
  _ = buf2[a];
}
```

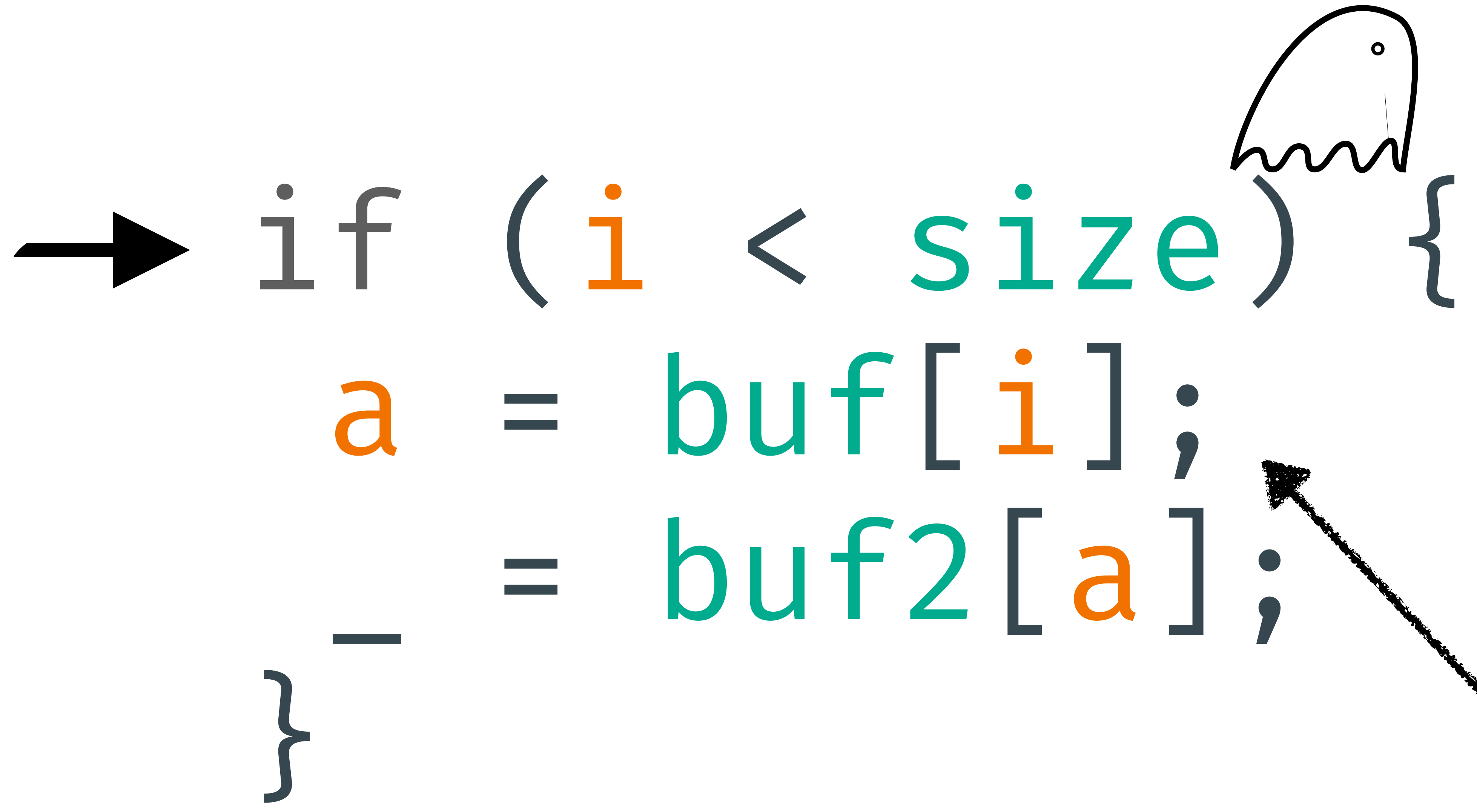
&buf + 20 = &sec

```
i    : 20      size:  8
               sec : 42
```

Registers
Stack Variables

Speculative Execution

SpectreV1



```
if (i < size) {
 a = buf[i];
 _ = buf2[a];
}
```

Side-Channel
Leakage

&buf + 20 = &sec

```
i    : 20      size:  8
               sec : 42
```

Registers
Stack Variables

Speculative Execution

SpectreV1

```
if (i < size) {
  a = buf[i];
  _ = buf2[a];
}
```

Side-Channel Leakage

BR false

&buf + 20 = &sec

```
i    : 20       size:   8
                sec  : 42
```

Registers
Stack Variables

Speculative Execution

```
→ if (i < size) {
    a = buf[i];
    _ = buf2[a];
  }
```

Side-Channel Leakage

```
BR false

LD 20
```

```
&buf + 20 = &sec
```

```
i    : 20      size:  8
               sec : 42
```

Registers
Stack Variables

Speculative Execution

SpectreV1

```
if (i < size) {
  a = buf[i];
  _ = buf2[a];
}
```

Side-Channel Leakage

```
BR false

LD 20

LD 42
```

```
&buf + 20 = &sec
```

```
i    : 20       size:  8
                sec : 42
```

Registers
Stack Variables

Branch-Prediction: Non-Det!

```
→ if (i < size) {
    a = buf[i];
    _ = buf2[a];
  }
```

```
i    : 20      size:  8
               sec : 42
```

&buf + 20 = &sec

Side-Channel Leakage

```
BR false
LD 20
LD 42
```

Registers
Stack Variables

Speculative Execution

SpectreV1

Micro-Arch Directive

Branch-Prediction: Non-Det!

```
if (i < size) {
  a = buf[i];
  _ = buf2[a];
}
```

Side-Channel Leakage

```
BR false
LD 20
LD 42
```

&buf + 20 = &sec

```
i    : 20      size:  8
               sec : 42
```

Registers
Stack Variables

Speculative Execution

SpectreV1

Micro-Arch Directive

miss

Branch-Prediction: Non-Det!

```
→ if (i < size) {
    a = buf[i];
    _ = buf2[a];
  }
```

Side-Channel Leakage

```
BR false
LD 20
LD 42
```

&buf + 20 = &sec

```
i    : 20       size:  8
                sec : 42
```

Registers
Stack Variables

Speculative Execution

SpectreV1

Micro-Arch Directive

```
miss

oob sec
```

Branch-Prediction: Non-Det!

```
if (i < size) {
  a = buf[i];
  _ = buf2[a];
}
```

Side-Channel Leakage

```
BR false

LD 20

LD 42
```

```
&buf + 20 = &sec
```

```
i    : 20      size:  8
               sec :  42
```

Registers
Stack Variables

Speculative Execution

Micro-Arch Directive

```
miss

oob sec

step
```

Branch-Prediction: Non-Det!

```
if (i < size) {
  a = buf[i];
  _ = buf2[a];
}
```

Side-Channel Leakage

```
BR false

LD 20

LD 42
```

```
&buf + 20 = &sec
```

```
i    : 20       size:  8
                sec : 42
```

Registers
Stack Variables

Micro-Arch Directive

```
miss

oob sec

step
```

Removes Non-Det!

Branch-Prediction: Non-Det!

```
if (i < size) {
  a = buf[i];
  _ = buf2[a];
}
```

Side-Channel Leakage

```
BR false

LD 20

LD 42
```

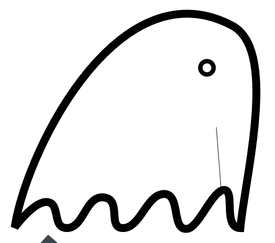
```
&buf + 20 = &sec
```

```
i    : 20      size:  8
               sec  : 42
```

Registers
Stack Variables

Speculative Execution

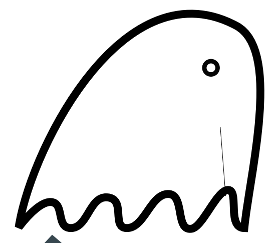
Directive		Leakage
miss	`if (i < size)`	BR false
oob sec	`a = buf[i];`	LD 20
step	`_ = buf2[a];`	LD 42



$\boxed{S}$

Speculative Execution

Directive:Leakage $\longrightarrow$



Directive		Leakage
miss	`if (i < size)`	BR false
oob sec	`a = buf[i];`	LD 20
step	`_ = buf2[a];`	LD 42

$\boxed{S}$ Directive

Speculative Execution

$$\xrightarrow{\texttt{Directive:Leakage}}$$

Directive		Leakage
`miss`	`if (i < size)`	`BR false`
`oob sec`	`a = buf[i];`	`LD 20`
`step`	`_ = buf2[a];`	`LD 42`

$$\begin{tikzcd} \boxed{S} \arrow[r, "\texttt{Directive}"] & \boxed{U} \end{tikzcd}$$

Speculative Execution

$\xrightarrow{\texttt{Directive:Leakage}}$

Directive		Leakage
miss	`if (i < size)`	BR false
oob sec	`a = buf[i];`	LD 20
step	`_ = buf2[a];`	LD 42

$$\begin{tikzcd} \boxed{S} \arrow[r, "\texttt{Directive:Leakage}"] & \boxed{U} \end{tikzcd}$$

Speculative Execution

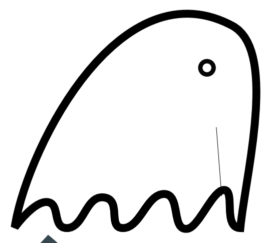
$\xrightarrow{\texttt{Directive:Leakage}}$

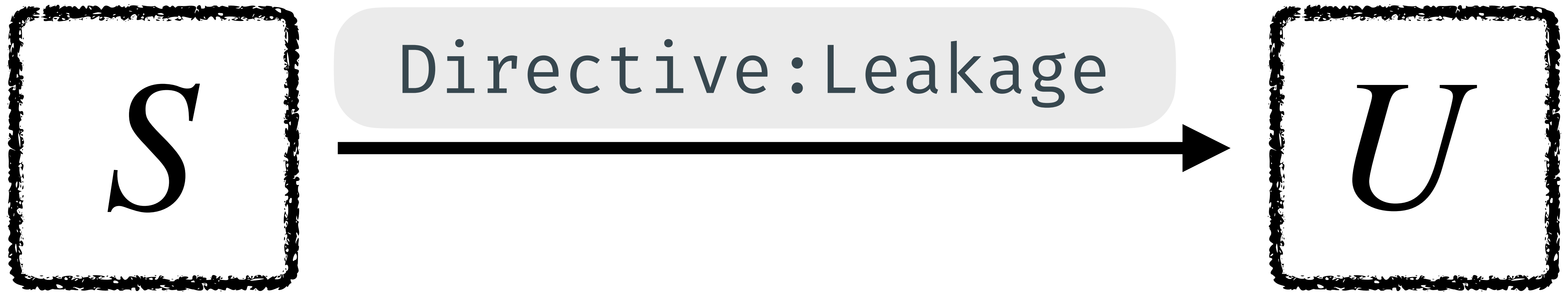
Directive		Leakage
`miss`	`if (i < size)`	`BR false`
`oob sec`	`a = buf[i];`	`LD 20`
`step`	`_ = buf2[a];`	`LD 42`

$$\begin{tikzcd} \boxed{S} \arrow[r, "\texttt{Directive:Leakage}"] & \boxed{U} \end{tikzcd}$$

Speculative Execution

Directive		Leakage
miss	`if (i < size)`	BR false
oob sec	`a = buf[i];`	LD 20
step	`_ = buf2[a];`	LD 42



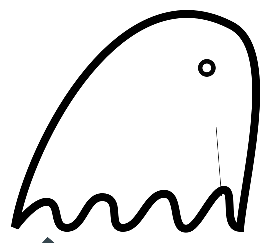


Directive

miss

oob sec

```
if (i < size)
 a = buf[i];

 _ = buf2[a];
```



Leakage

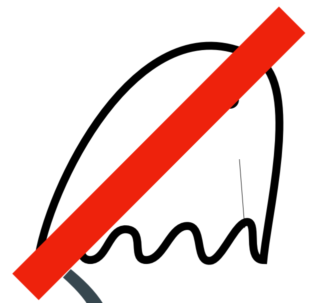
BR false

LD 20

Directive

miss

oob sec



```
if (i < size)
 a = buf[i];

 _ = buf2[a];
```

Leakage

BR false

LD 20

Speculative Execution

Directive

miss

oob sec

```
if (i < size)
 a = buf[i];
 SFENCE;
 _ = buf2[a];
```

Leakage

BR false

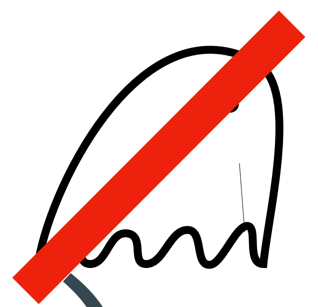
LD 20

Directive

miss

oob sec

```
if (i < size)
 a = buf[i];
 a = slh(a);
 _ = buf2[a];
```



Leakage

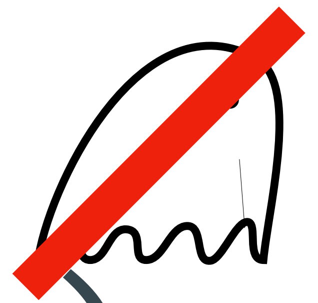
BR false

LD 20

Directive

miss

oob sec



```
if (i < size)
 a = buf[i];
 a = slh(a);
 _ = buf2[a];
```

Leakage

BR false

LD 20

```
i    : 20      size:  8
a    : 42      sec : 42
```

Directive

```
miss

oob sec

step
```

```
if (i < size)
 a = buf[i];
 a = slh(a);
 _ = buf2[a];
```

Leakage

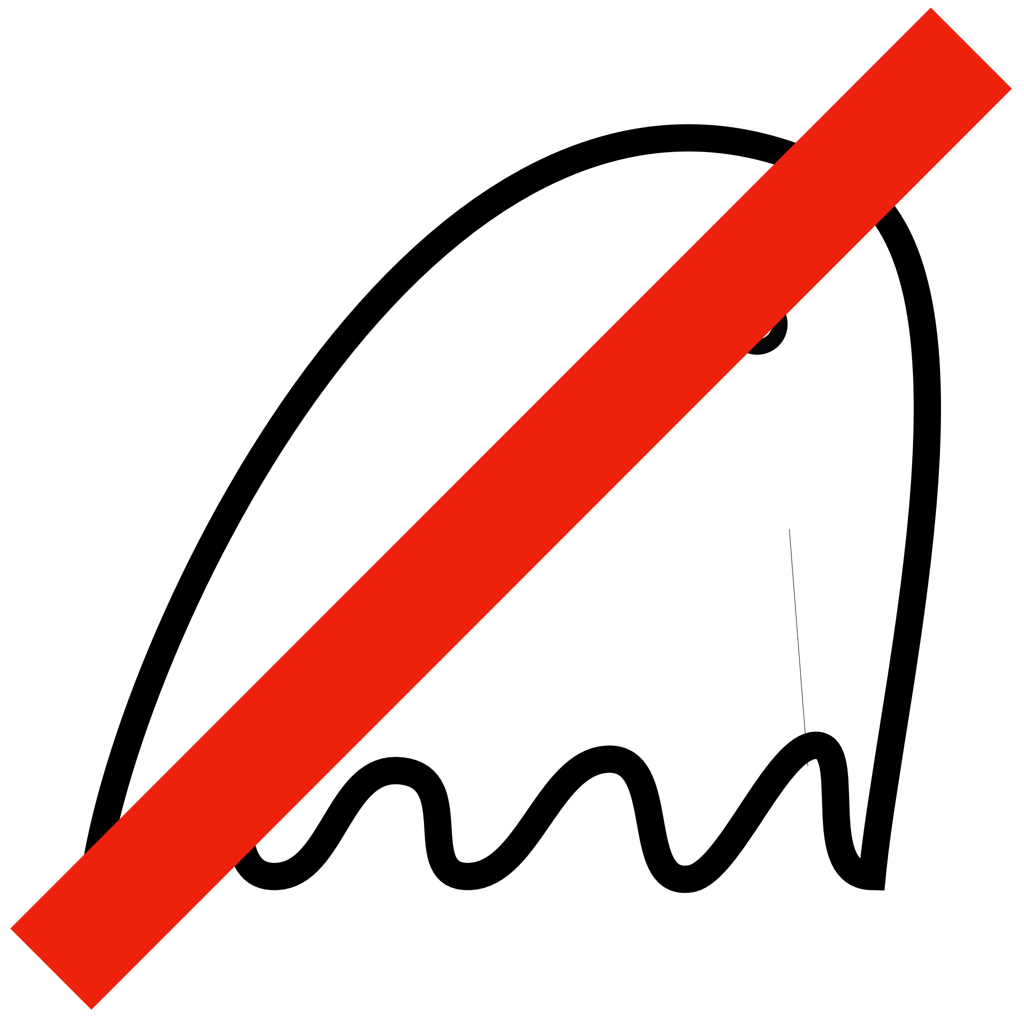
```
BR false

LD 20
```

```
i    : 20      size:  8
a    :  0      sec : 42
```

Directive		Leakage
`miss`	`if (i < size)`	`BR false`
`oob sec`	`a = buf[i];`	`LD 20`
`step`	`a = slh(a);`	
`step`	`_ = buf2[a];`	`LD 0`

```
i    : 20      size:   8
a    :  0      sec : 42
```

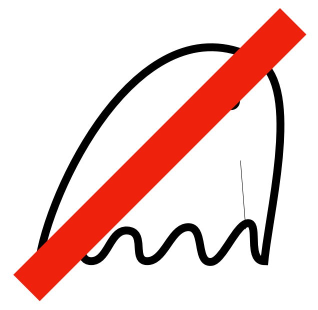
S Non-Interference

Proving 

S Non-Interference

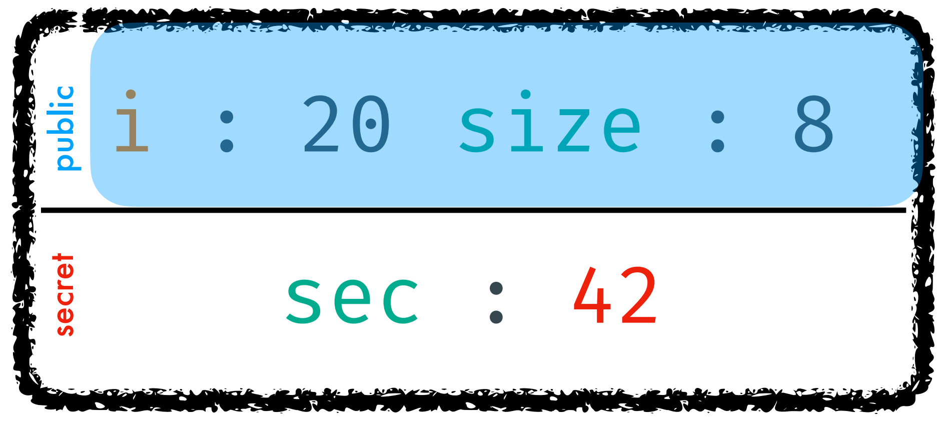
```
i : 20 size : 8

    sec : 42
```

S Non-Interference



public

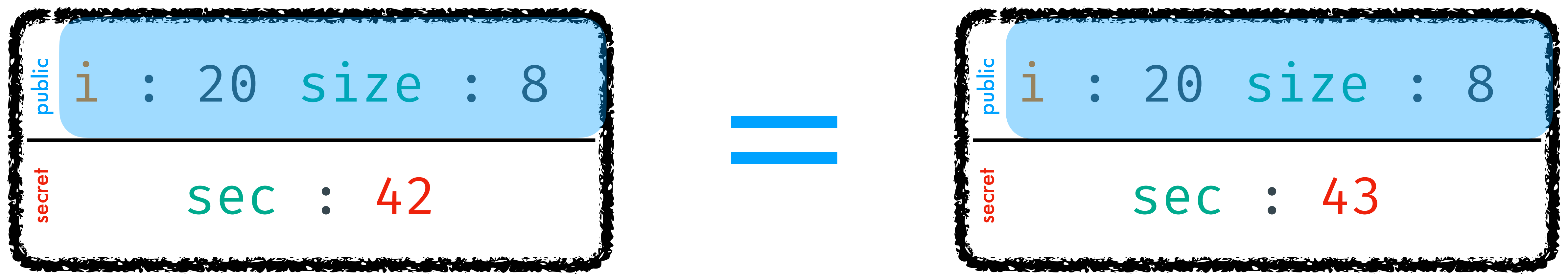
secret



S Non-Interference

public

secret



S Non-Interference

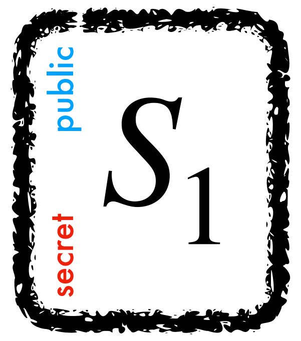
$P \models$ **SNI**

IF

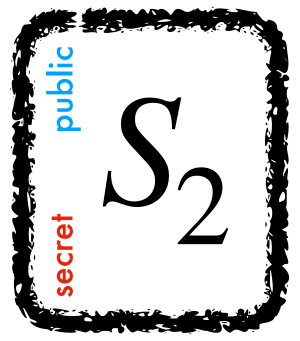
S Non-Interference

$P \models \mathbf{SNI}$

IF $\forall$



$=$

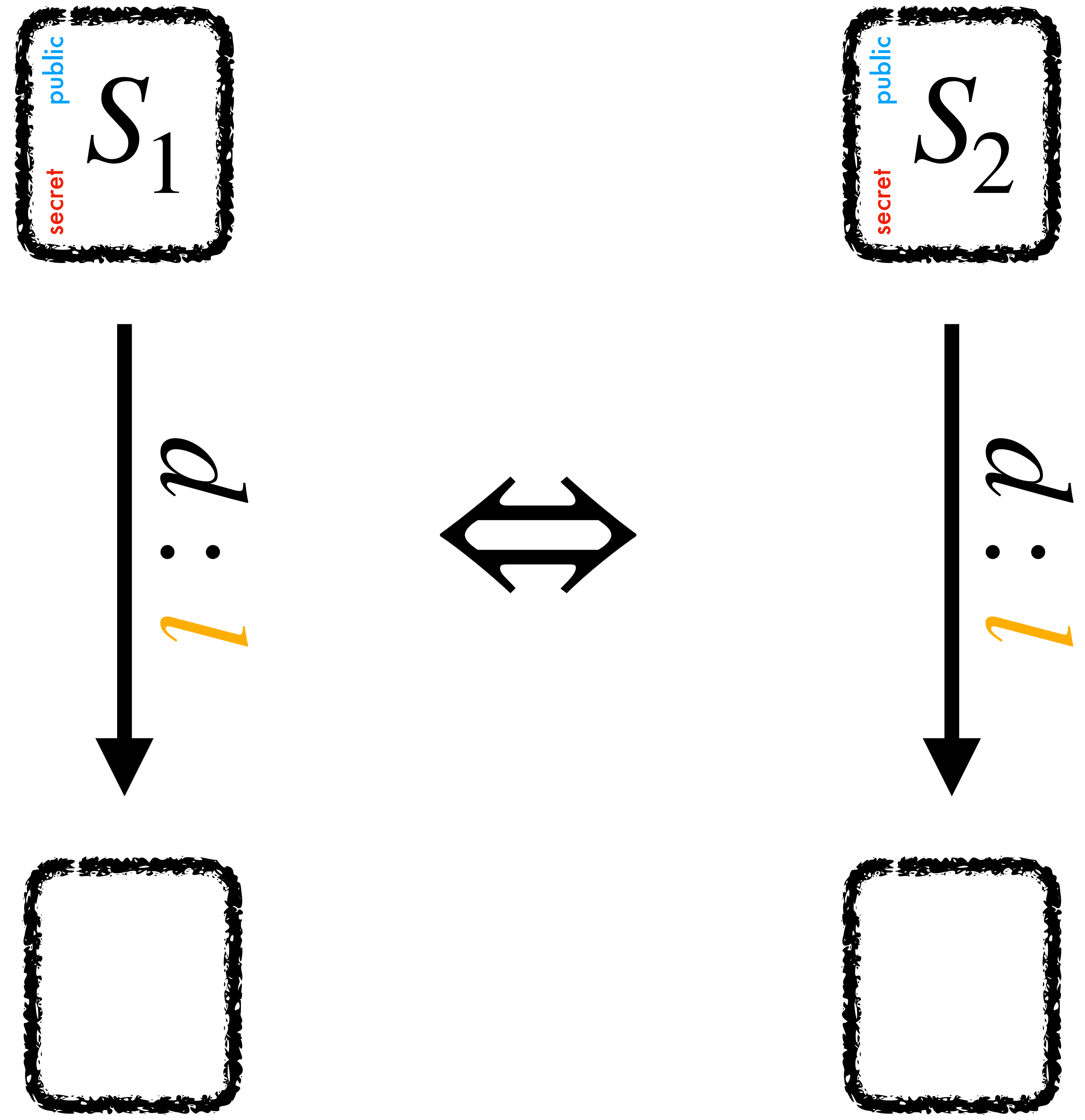


$d \in \mathtt{Dir}^*$:

S Non-Interference

$P \models$ **SNI**

IF $\forall \boxed{S_1} = \boxed{S_2} :$ $d \in \mathtt{Dir}^*$



SNI Preservation

[.] : Compiler Pass

SNI Preservation

$$P \models \textbf{SNI}_{\cancel{\text{👻}}}$$

$[\,.\,]$: Compiler Pass

SNI Preservation

SNIP

$$P \models \textbf{SNI} \implies [P] \models \textbf{SNI}$$

$[\,.\,]$: Compiler Pass

SNI Preservation

$[\,.\,] \models$ **SNIP**

IF $\forall P \quad P \models \textbf{SNI} \Longrightarrow [P] \models \textbf{SNI}$

$[\,.\,]$: Compiler Pass

SNI Preservation

LLVM RegAlloc

$$P \models \textbf{SNI} \implies [P] \models \textbf{SNI}$$

SNI Preservation

$$P \models \textbf{SNI} \overset{?}{\Longrightarrow} [P]_{\mathsf{ra}} \models \textbf{SNI}$$

SNI Preservation

$$P \models \textbf{SNI} \quad \overset{?}{\Longrightarrow} \quad [P]_{\text{ra}} \models \textbf{SNI}$$

```
fn( public ind, secret sec, public ...)

 if (b < size)
  buf[b] = sec;

 _ = buf[ind]
```

SNI Preservation

$$P \models \textbf{SNI} \quad \overset{?}{\Longrightarrow} \quad [P]_{\text{ra}} \models \textbf{SNI}$$

```
fn( public ind, secret sec, public ...)

  if (b < size)
   buf[b] = sec;

  _ = buf[ind]
```

SNI Preservation

LLVM RegAlloc

$P \vDash \textbf{SNI}$ $\overset{?}{\Longrightarrow}$ $[P]_{ra} \vDash \textbf{SNI}$

```
fn( public ind, secret sec, public ...)

 if (b < size)
  buf[b] = sec;

 _ = buf[ind]
```

```
fn( public ind, secret sec, public ...)
 stk = ind;
 if (b < size)
  buf[b] = sec;
 ind = stk;
 _ = buf2[ind]
```

SNI Preservation

LLVM **RegAlloc**

$$P \models \mathrm{SNI} \quad \overset{?}{\not\Longrightarrow} \quad [P]_{ra} \models \mathrm{SNI}$$

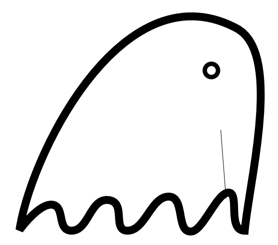
```
fn( public ind, secret sec, public …)

 if (b < size)
  buf[b] = sec;

 _ = buf[ind]
```

```
fn( public ind, secret sec, public …)
 stk = ind;
 if (b < size)
  buf[b] = sec;
 ind = stk;
 _ = buf2[ind]
```

SNI Preservation

LLVM RegAlloc

$P \vDash \mathbf{SNI}$ $\cancel{\xRightarrow{?}}$ $[P]_{ra} \vDash \mathbf{SNI}$



```
fn( public ind, secret sec, public ...)

 if (b < size)
  buf[b] = sec;

_ = buf[ind]
```

```
                fn( public ind, secret sec, public ...)
step            stk = ind;
miss            if (b < size)
oob stk          buf[b] = sec;
step            ind = stk;
step            _ = buf2[ind]
```

SNI Preservation

LLVM **RegAlloc**

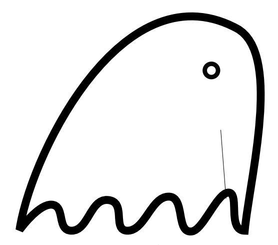
$P \models \mathrm{SNI}$ $\overset{?}{\cancel{\Longrightarrow}}$ $[P]_{ra} \models \mathrm{SNI}$

```
fn( public ind, secret sec, public …)

 if (b < size)
  buf[b] = sec;

 _ = buf[ind]
```

```
                     fn( public ind, secret sec, public …)
                                             42
→ step         stk = ind;
  miss         if (b < size)
  oob stk       buf[b] = sec;
  step         ind = stk;
  step         _ = buf2[ind]
```

```
ind :  0    b    : 20    size:  8
sec : 42
```

$P \vDash \text{SNI}$ $\quad \overset{?}{\not\Longrightarrow} \quad [P]_{ra} \vDash \text{SNI}$



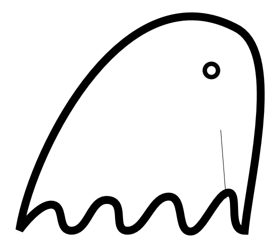
```
fn( public ind, secret sec, public ...)        fn( public ind, secret sec, public ...)
                                                                    42
                                          step       stk = ind;
 if (b < size)                        →   miss       if (b < size)
  buf[b] = sec;                           oob stk     buf[b] = sec;
                                          step       ind = stk;
 _ = buf[ind]                             step       _ = buf2[ind]
```

```
ind :  0    b   : 20    size:  8
sec : 42                stk :  0
```

SNI Preservation

LLVM RegAlloc

$P \vDash \mathbf{SNI}$ $\overset{?}{\not\Longrightarrow}$ $[P]_{ra} \vDash \mathbf{SNI}$



```
fn( ind, sec, ...)          fn( ind, sec, ...)
    public secret public        public secret public
                                          42
                      step      stk = ind;
 if (b < size)        miss      if (b < size)
  buf[b] = sec;  ──►  oob stk    buf[b] =42 sec;
                      step      ind = stk;
 _ = buf[ind]         step      _ = buf2[ind]
```

```
ind : 0    b : 20    size: 8
sec : 42             stk : 42
```

$P \models \text{SNI} \overset{?}{\cancel{\Longrightarrow}} [P]_{ra} \models \text{SNI}$

```
fn( public ind, secret sec, public ...)

 if (b < size)
  buf[b] = sec;

 _ = buf[ind]
```

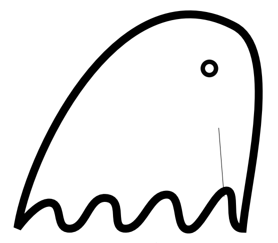
```
              fn( public ind, secret sec, public ...)
step          stk = ind;
miss          if (b < size)
oob stk        buf[b] =42 sec;
step          ind =42 stk;
step          _ = buf2[ind]
```

```
ind :  0    b    : 20    size:  8
sec : 42                 stk : 42
```

SNI Preservation

LLVM RegAlloc

$P \vDash \mathbf{SNI}$ $\overset{?}{\cancel{\Longrightarrow}}$ $[P]_{ra} \vDash \mathbf{SNI}$



```
fn( public ind, secret sec, public ...)

 if (b < size)
  buf[b] = sec;

 _ = buf[ind]
```

```
                 fn( public ind, secret sec, public ...)
                                        42
step         stk = ind;
miss         if (b < size)
oob stk       buf[b] =42sec;
step         ind =42stk;
step         _ = buf2[ind]        LD 42
                      42
```

```
ind : 42    b    : 20    size:  8
sec : 42                 stk : 42
```

$\mathit{LLVM} \nvDash \textbf{SNIP}$

$$\text{LLVM} \nvDash \textbf{SNIP}$$

Each of LLVM's 4 allocators!

$$\mathit{LLVM} \nvDash \textbf{SNIP}$$

Each of LLVM’s 4 allocators!

Slightly modified `libsodium` code!

Goals

Goals

How do we prove

$$[\,.\,] \vDash \textbf{SNIP}\ ?$$

Goals

How do we prove

$$[\,.\,] \vDash \textbf{SNIP}\ ?$$

Can we fix Register Allocation so that

$$[\,.\,]_{ra} \vDash \textbf{SNIP}\ ?$$

Goals

How do we prove

$[\,.\,] \models$ **SNIP** ?

POPL

Can we fix Register Allocation so that

$[\,.\,]_{ra} \models$ **SNIP** ?

Goals

How do we prove

$[\,\cdot\,] \models$ **SNIP** ¿

POPL

Can we fix Register Allocation so that

$[\,\cdot\,]_{ra} \models$ **SNIP** ¿

Goals

How do we prove

$[\,\cdot\,] \models$ **SNIP** ?

POPL

Can we fix Register Allocation so that

$[\,\cdot\,]_{ra} \models$ **SNIP** ?

(And in a better way than just inserting Mitigations everywhere?)

Make $[\ .\ ]_{ra} \vDash$ SNIP again!

$\left(\text{How do we prove } [\ .\ ] \vDash \textbf{SNIP} \text{ ?} \right)$

Make $[\,.\,]_{ra} \vDash$ SNIP again!

$\Big($ How do we prove $[\,.\,] \vDash$ SNIP ? $\Big)$

P

$[P]$

Make $[\,.\,]_{ra} \models$ SNIP again!

(How do we prove $[\,.\,] \models$ SNIP ?)

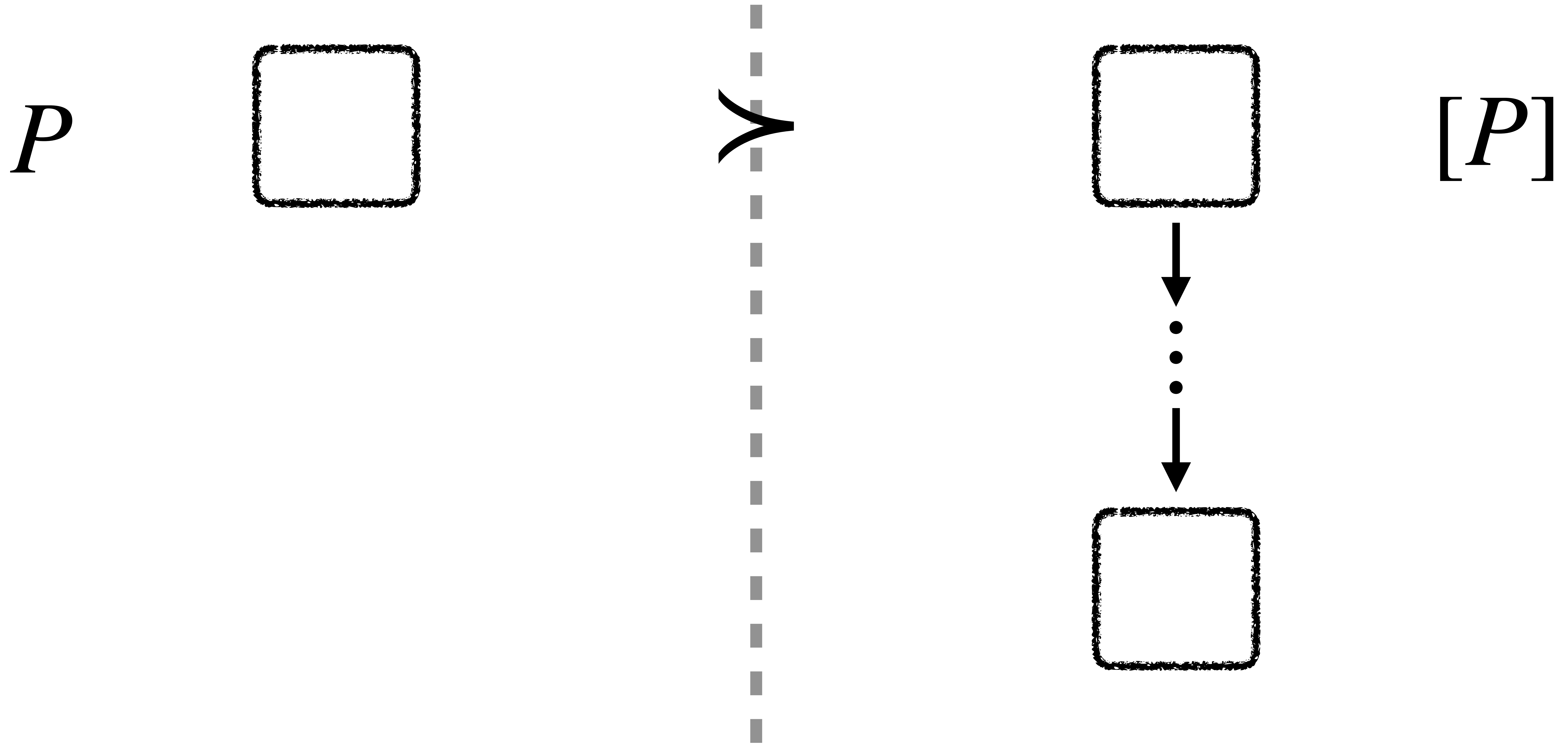
Define $\succ$

$$P \qquad \square \qquad \succ \qquad \square \qquad [P]$$

Make $[\,.\,]_{ra} \models$ SNIP again!

(How do we prove $[\,.\,] \models$ SNIP ?)

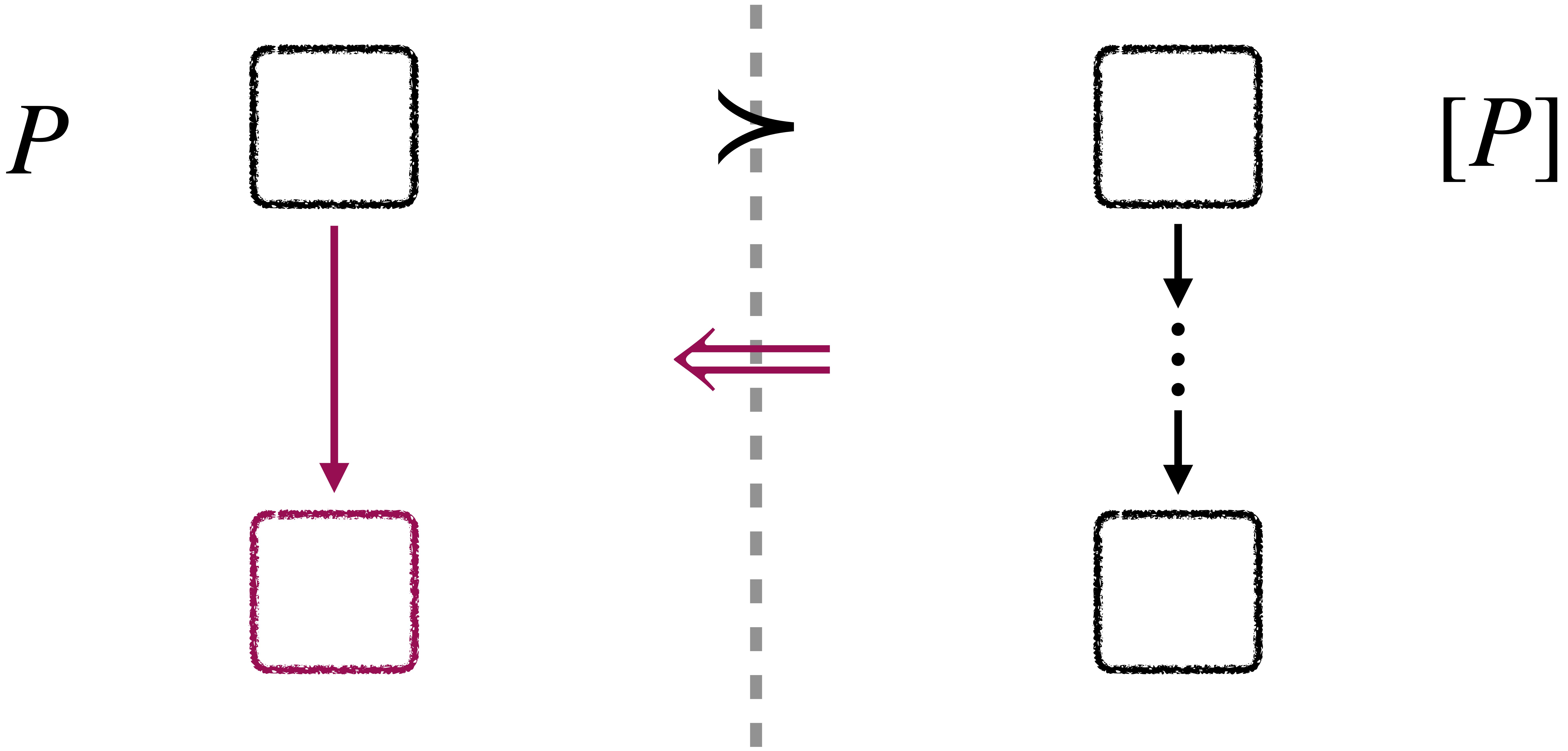
Define $\succ$



(How do we prove $[\,.\,] \vDash$ **SNIP** ?)

Make $[\,.\,]_{\mathrm{ra}} \vDash$ SNIP again!

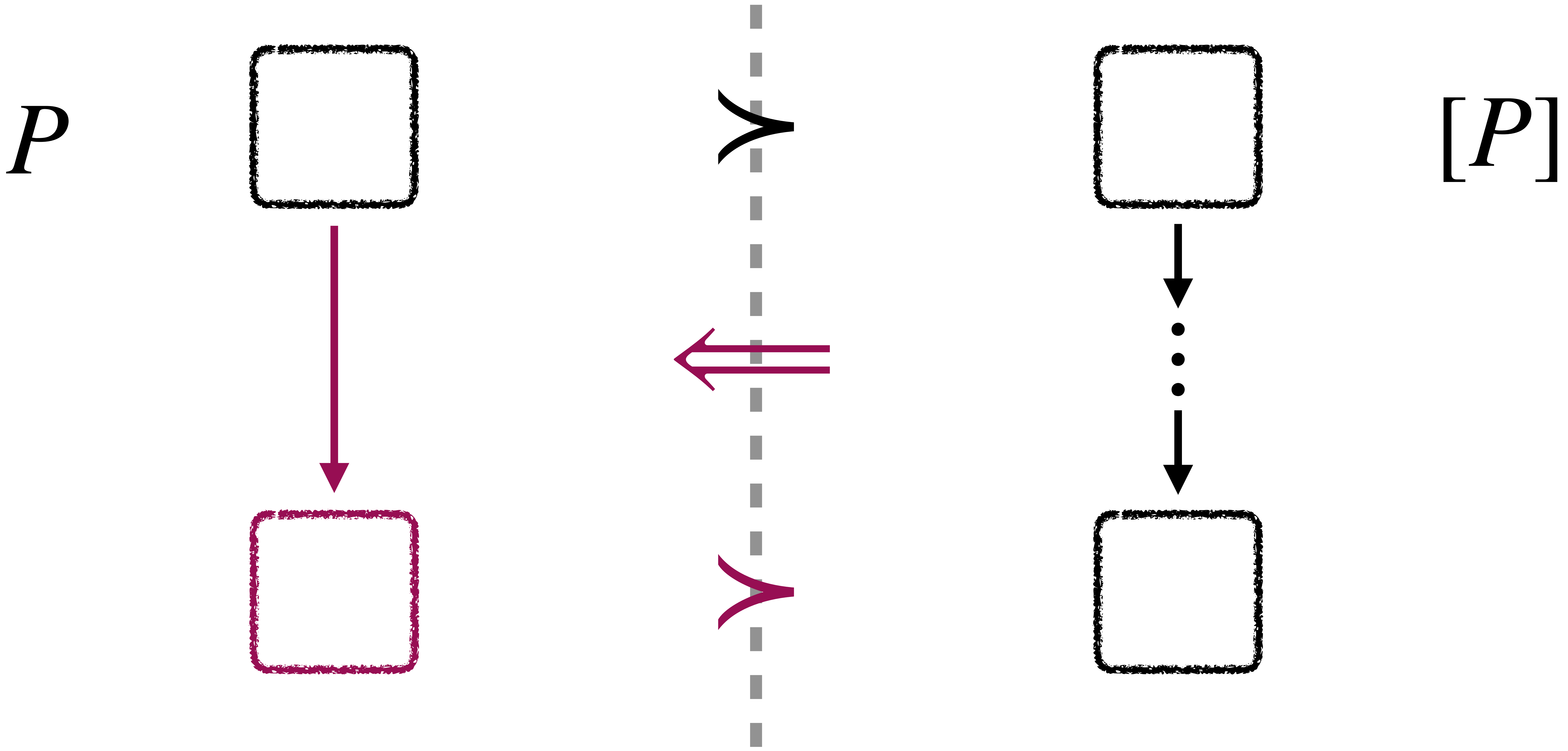
Define $\succ$



Make $[\,.\,]_{ra} \vDash$ SNIP again!

Define $\succ$

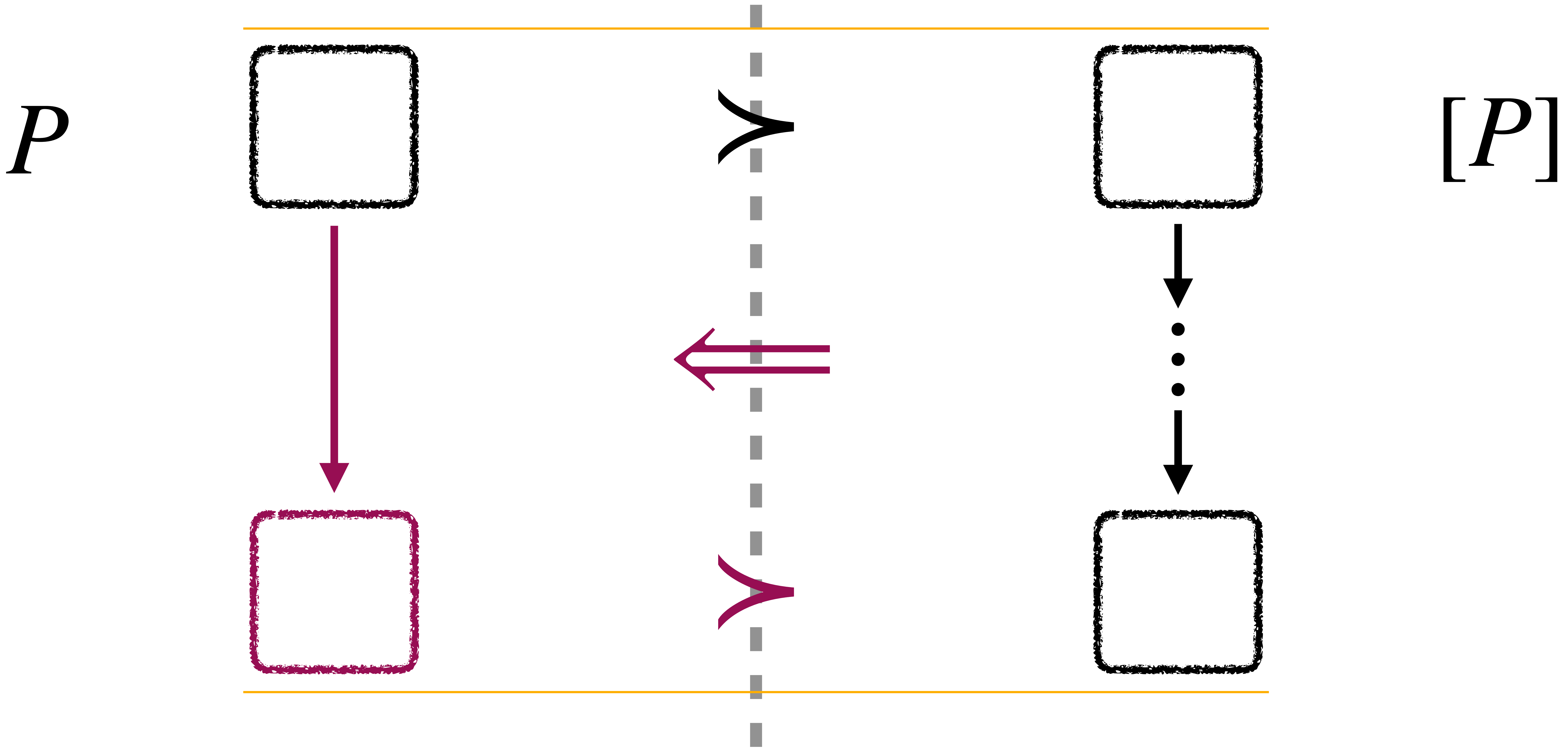
(*How do we prove* $[\,.\,] \vDash$ SNIP ?)



Make $[\,.\,]_{ra} \vDash$ SNIP again!

Define $\succ$

(How do we prove $[\,.\,] \vDash$ SNIP ?)



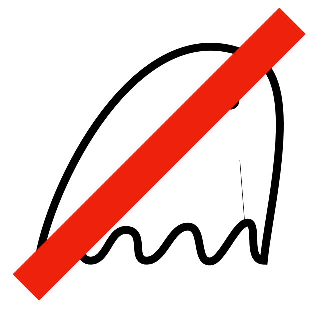
Make $[\,.\,]_{ra} \models$ SNIP again!

Define $\succ$

Make $[\,.\,]_{ra} \models$ SNIP again!

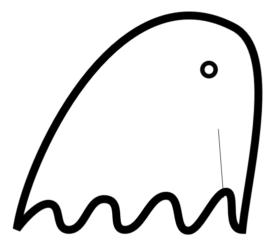
Define ≻

P



```
fn( public ind, secret sec, public ...)

 if (b < size)
  buf[b] = sec;

 _ = buf[ind]
```

$[P]_{ra}$

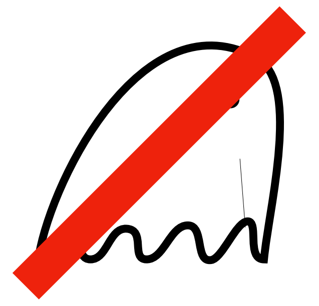


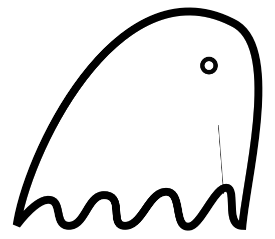
```
fn( public ind, secret sec, public ...)
 stk = ind;
 if (b < size)
  buf[b] = sec;
 ind = stk;
 _ = buf[ind]
```

Make $[\,.\,]_{ra} \models$ SNIP again!

Define ≻

P $[P]_{ra}$





```
fn( ind, sec, …)              fn( ind, sec, …)
                               stk = ind;
  if (b < size)                if (b < size)
   buf[b] = sec;                buf[b] = sec;
                               ind = stk;
  _ = buf[ind]                 _ = buf[ind]
```

public public secret secret public public

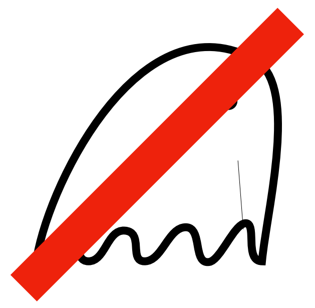
← same instruction →
+ spill code

Make $[\,.\,]_{ra} \models$ SNIP again!

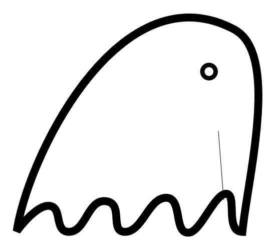
Define >

RegAlloc

P



$[P]_{ra}$



```
fn( ind, sec, ...)                               fn( ind, sec, ...)
                                         step     stk = ind;
 if (b < size)   miss                    miss     if (b < size)
  buf[b] = sec;                                    buf[b] = sec;
                                                  ind = stk;
 _ = buf[ind]                                     _ = buf[ind]
```

public secret public

← **same instruction** →

+ spill code

Make $[\,.\,]_{ra} \models$ SNIP again!

Define >

P $\qquad\qquad\qquad\qquad$ $[P]_{ra}$

```
fn( public ind, secret sec, public …)                fn( public ind, secret sec, public …)
                                          step       stk = ind;
 if (b < size)   miss                     miss       if (b < size)
  buf[b] = sec;                                       buf[b] = sec;   ◀
                                                     ind = stk;
 _ = buf[ind]     ←  same instruction  →             _ = buf[ind]
                       + spill code
```

```
sec   :  42
b     :  20
-----------
stk   :   4
size  :   8
buf[]:1…8
```

Make $[\,.\,]_{ra} \models$ SNIP again!

Define >

P

```
fn( ind, sec, ...)

 if (b < size)   miss
  buf[b] = sec;

 _ = buf[ind]
```

public public secret public

$[P]_{ra}$

```
fn( ind, sec, ...)
step     stk = ind;
miss     if (b < size)
          buf[b] = sec;
         ind = stk;
          _ = buf[ind]
```

public secret public

same instruction
+ spill code

sec	:	42
b	:	20
ind	:	**4**
size	:	8
buf[]	:	1…8

sec	:	42
b	:	20
stk	:	**4**
size	:	8
buf[]	:	1…8

Make $[\,.\,]_{ra} \vDash$ SNIP again!

Define ≻

P

```
fn( public ind, secret sec, public ... )

 if (b < size)   miss
  buf[b] = sec;

 _ = buf[ind]
```

$[P]_{ra}$

```
        fn( public ind, secret sec, public ... )
step    stk = ind;
miss    if (b < size)
         buf[b] = sec;
        ind = stk;
        _ = buf[ind]
```

← **same instruction** →
+ spill code

sec	:	42
b	:	20
ind	:	**4**
size	:	8
buf[]	:	1…8

$\succ$

sec	:	42
b	:	20
stk	:	**4**
size	:	8
buf[]	:	1…8

Make $[\,.\,]_{ra} \models$ SNIP again!

Define ≻

P

```
fn( public ind, secret sec, public …)

 if (b < size)   miss
  buf[b] = sec;

_ = buf[ind]
```

$[P]_{ra}$

```
      fn( public ind, secret sec, public …)
step  stk = ind;   ←  Relocate ind
miss  if (b < size)
       buf[b] = sec;
      ind = stk;
      _ = buf[ind]
```

← **same instruction** →
+ spill code

```
sec    :  42
b      :  20
ind    :   4
-------------
size   :   8
buf[]:1…8
```

$\succ$

```
sec    :  42
b      :  20
-------------
stk    :   4
size   :   8
buf[]:1…8
```

Make $[\,.\,]_{ra} \models$ SNIP again!

Define ≻

P

```
fn( ind, sec, …)
       public ind, secret sec, public …
if (b < size)   miss
 buf[b] = sec;

 _ = buf[ind]
```

$[P]_{ra}$

```
fn( ind, sec, …)
       public ind, secret sec, public …
step   stk = ind;   ← Relocate ind
miss   if (b < size)
        buf[b] = sec;
       ind = stk;
        _ = buf[ind]
```

← same instruction →
+ spill code

sec	:	42
b	:	20
ind	:	**4**
size	:	8
buf[]	:	1…8

$\succ$

sec	:	42
b	:	20
stk	:	**4**
size	:	8
buf[]	:	1…8

Make $[\,.\,]_{ra} \models$ SNIP again!

Define ≻

P $\qquad\qquad\qquad\qquad$ $[P]_{ra}$

```
fn( public ind, secret sec, public ...)          fn( public ind, secret sec, public ...)
                                        step      stk = ind;   ← Relocate ind
 if (b < size)  miss                    miss      if (b < size)
  buf[b] = sec;                                    buf[b] = sec;
                                                  ind = stk;
_ = buf[ind]                                      _ = buf[ind]
```

← same instruction →
+ spill code

sec	:	42
b	:	20
ind	:	**4**
size	:	8
buf[]	:	1…8

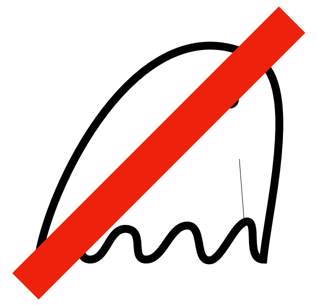
$\succ$

sec	:	42
b	:	20
stk	:	**4**
size	:	8
buf[]	:	1…8

Equal up to relocation

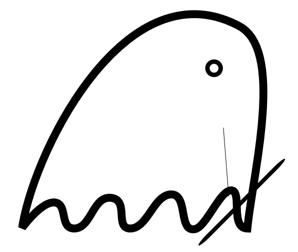
Make $[\,.\,]_{ra} \models$ SNIP again!

Poison



```
if (b < size)     miss
 buf[b] = sec;

_ = buf[ind]
```

```
step
miss
```



```
stk = ind;
if (b < size)
 buf[b] = sec;
ind = stk;
_ = buf[ind]
```

```
b     : 20
sec   : 42
ind   :  4
-----------
size  :  8
buf[]:1…8
```

$\succ$

```
sec   : 42
b     : 20
-----------
stk   :  4
size  :  8
buf[]:1…8
```

Make $[\,.\,]_{ra} \vDash$ SNIP again!

Poison

```
if (b < size)    miss
 buf[b] = sec;

_ = buf[ind]
```

```
step
miss
oob stk
```

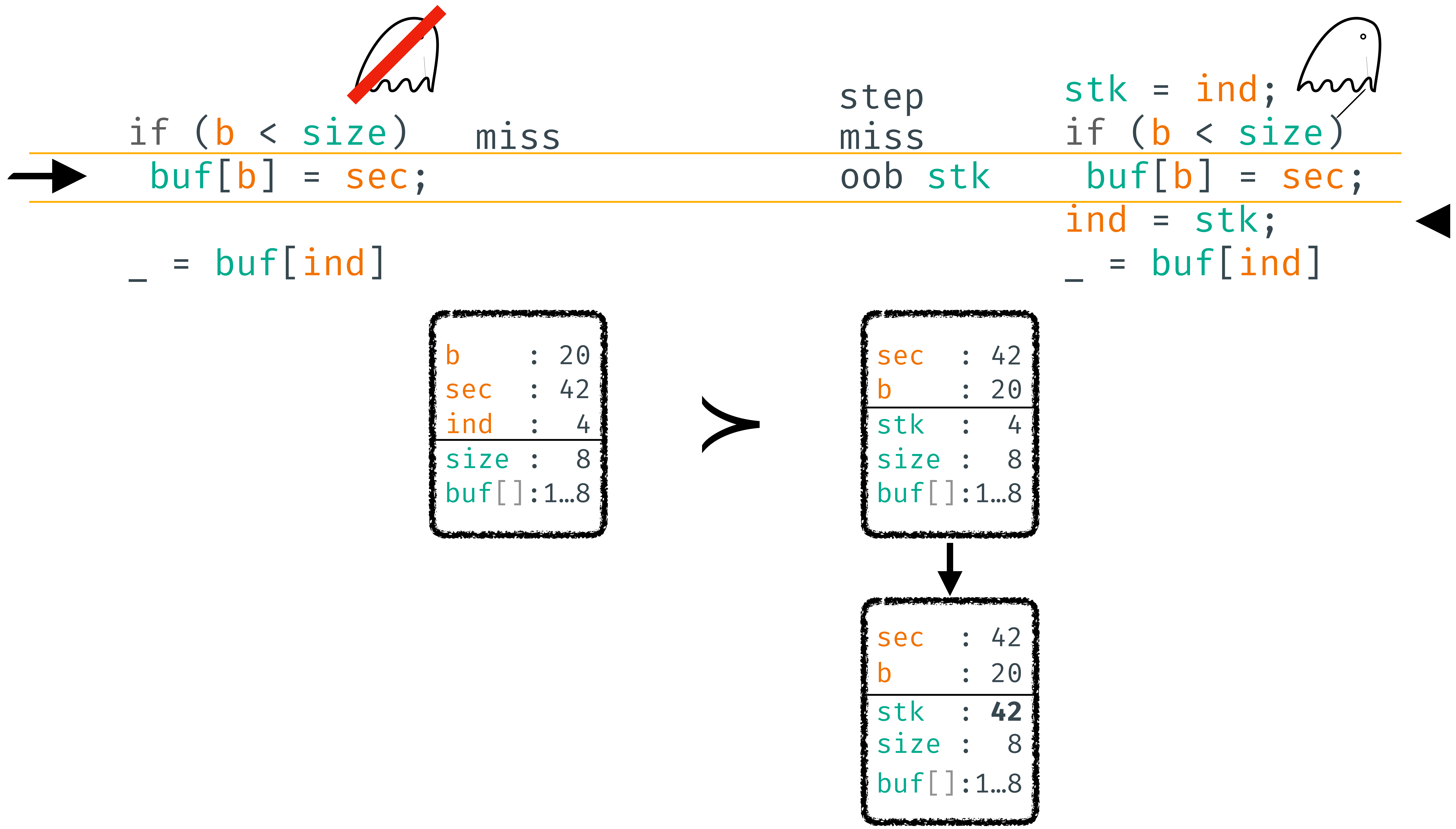
```
stk = ind;
if (b < size)
 buf[b] = sec;
ind = stk;
_ = buf[ind]
```

```
b     : 20
sec   : 42
ind   :  4
-----------
size  :  8
buf[]:1…8
```

$\succ$

```
sec   : 42
b     : 20
-----------
stk   :  4
size  :  8
buf[]:1…8
```

Make $[\,.\,]_{ra} \models$ SNIP again!



```
if (b < size)    miss
 buf[b] = sec;

_ = buf[ind]
```

```
step
miss
oob stk
```

```
stk = ind;
if (b < size)
 buf[b] = sec;
ind = stk;
_ = buf[ind]
```

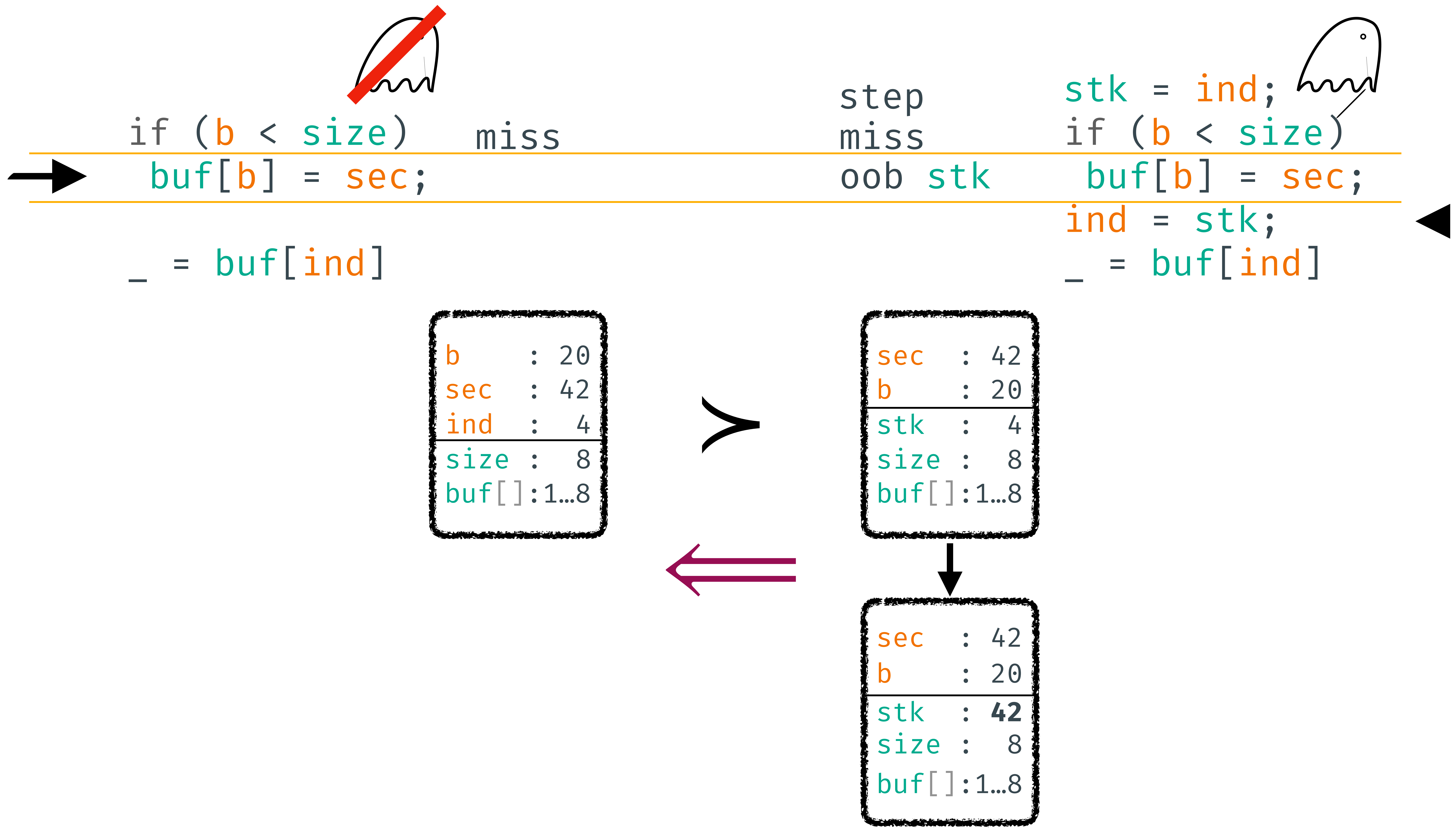
```
b     : 20
sec   : 42
ind   :  4
size  :  8
buf[]:1…8
```

$\succ$

```
sec   : 42
b     : 20
stk   :  4
size  :  8
buf[]:1…8
```

```
sec   : 42
b     : 20
stk   : 42
size  :  8
buf[]:1…8
```

Make $[\,.\,]_{ra} \models$ SNIP again!



```
if (b < size)    miss
 buf[b] = sec;

_ = buf[ind]
```

```
step
miss
oob stk
```

```
stk = ind;
if (b < size)
 buf[b] = sec;
ind = stk;
_ = buf[ind]
```

```
b     :  20
sec   :  42
ind   :   4
-----------
size  :   8
buf[]:1…8
```

$\succ$

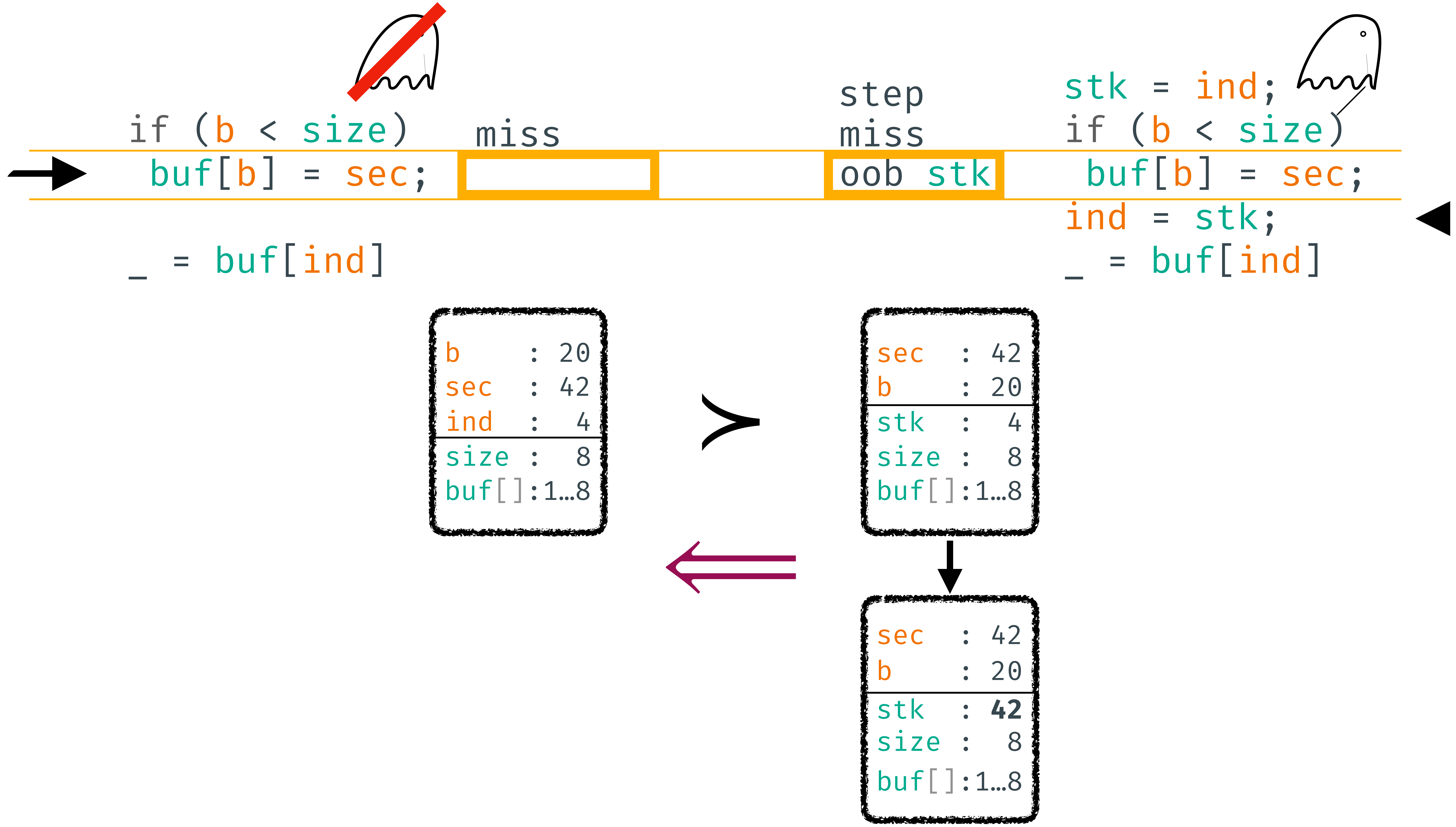
```
sec   :  42
b     :  20
-----------
stk   :   4
size  :   8
buf[]:1…8
```

$\Longleftarrow$

```
sec   :  42
b     :  20
-----------
stk   :  42
size  :   8
buf[]:1…8
```

Make $[\,.\,]_{ra} \models$ SNIP again!

Poison



```
if (b < size)
 buf[b] = sec;

_ = buf[ind]
```

```
stk = ind;
if (b < size)
 buf[b] = sec;
ind = stk;
_ = buf[ind]
```

```
b     : 20
sec   : 42
ind   :  4
----------
size  :  8
buf[]:1…8
```

$\succ$

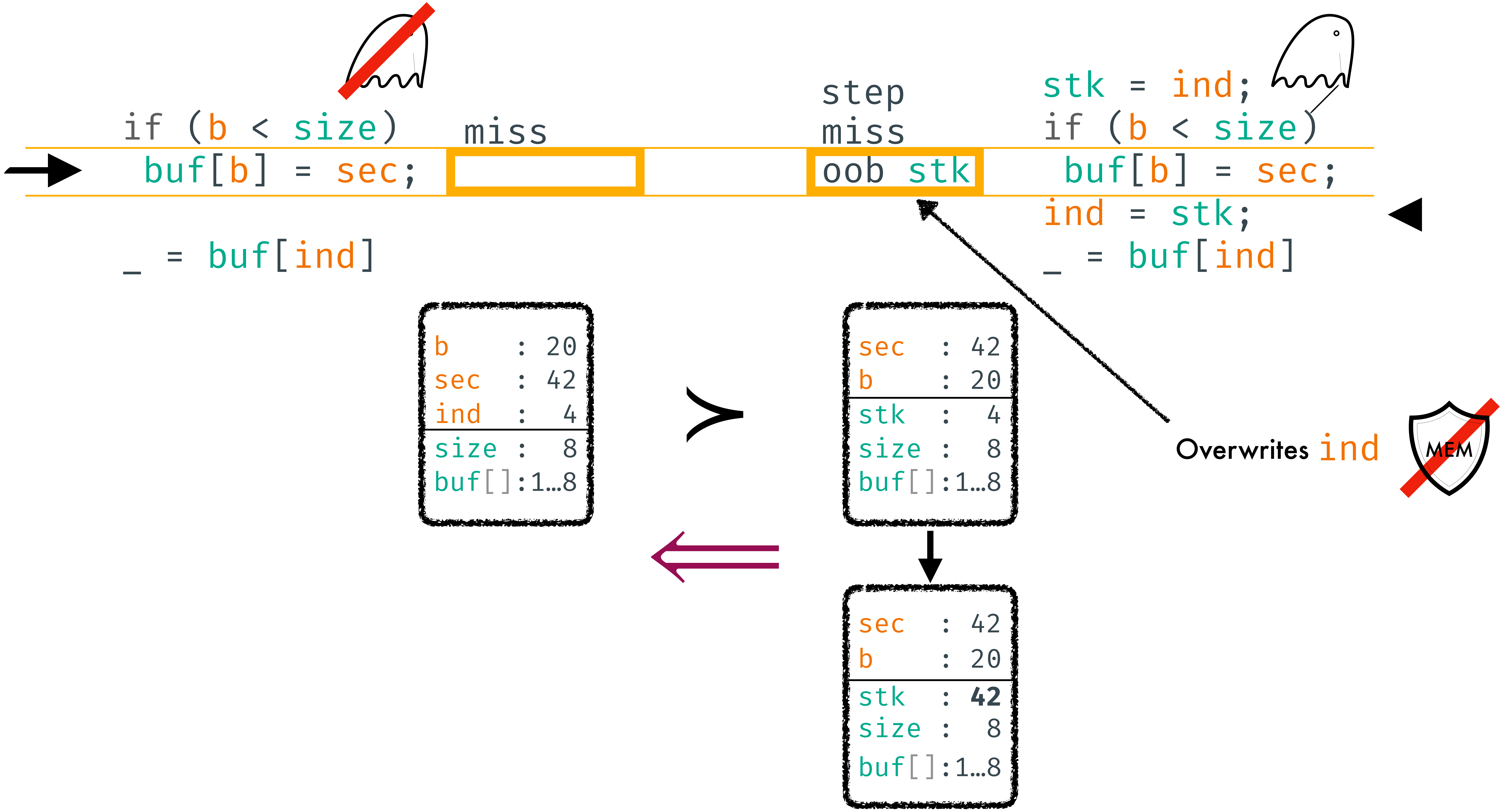
```
sec   : 42
b     : 20
----------
stk   :  4
size  :  8
buf[]:1…8
```

$\Longleftarrow$

```
sec   : 42
b     : 20
----------
stk   : 42
size  :  8
buf[]:1…8
```

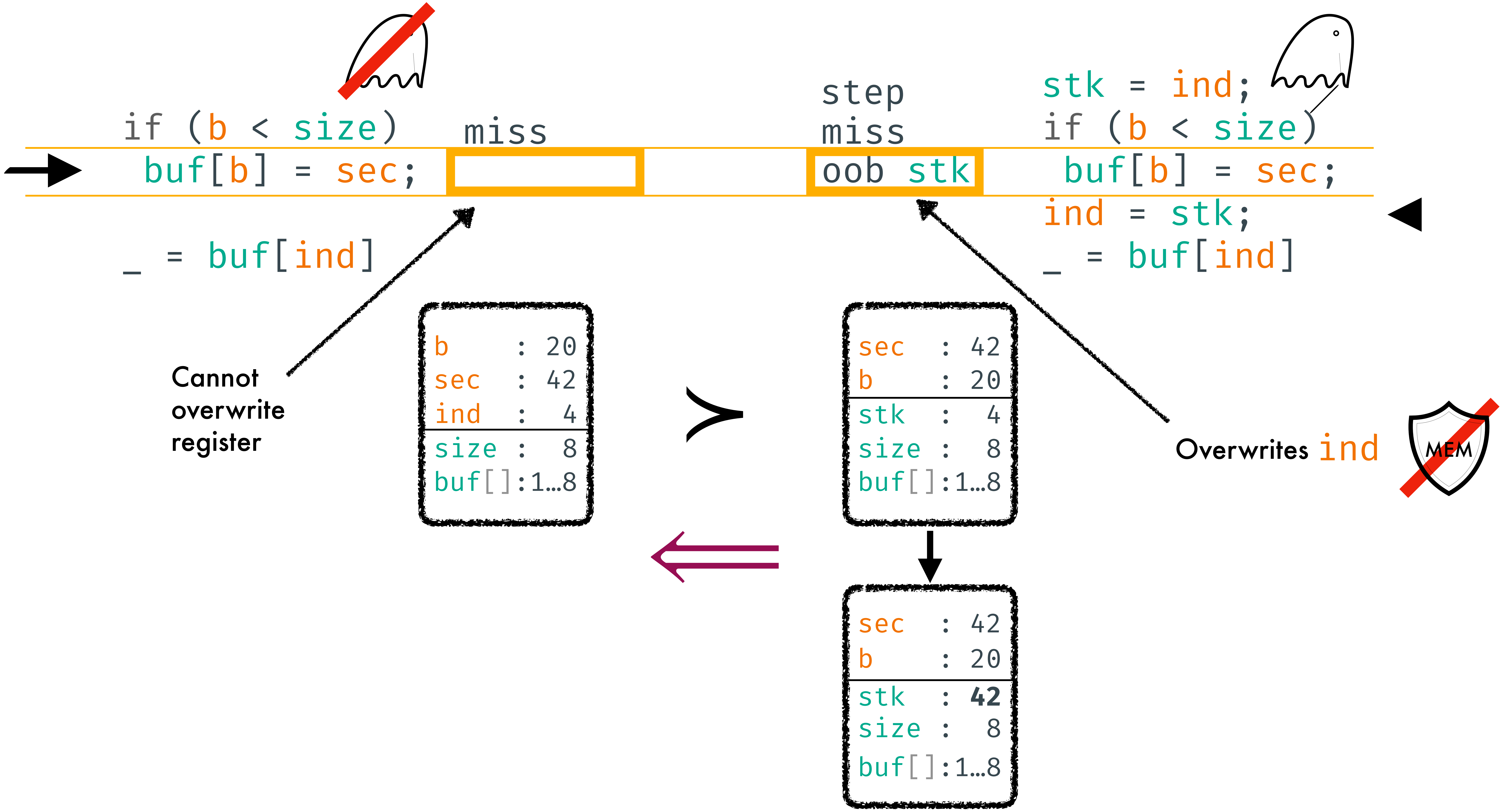
Make $[\,.\,]_{ra} \models$ SNIP again!

Poison



Make $[\,.\,]_{ra} \models$ SNIP again!

Poison



```
if (b < size)
 buf[b] = sec;

_ = buf[ind]
```

```
stk = ind;
if (b < size)
 buf[b] = sec;
ind = stk;
_ = buf[ind]
```

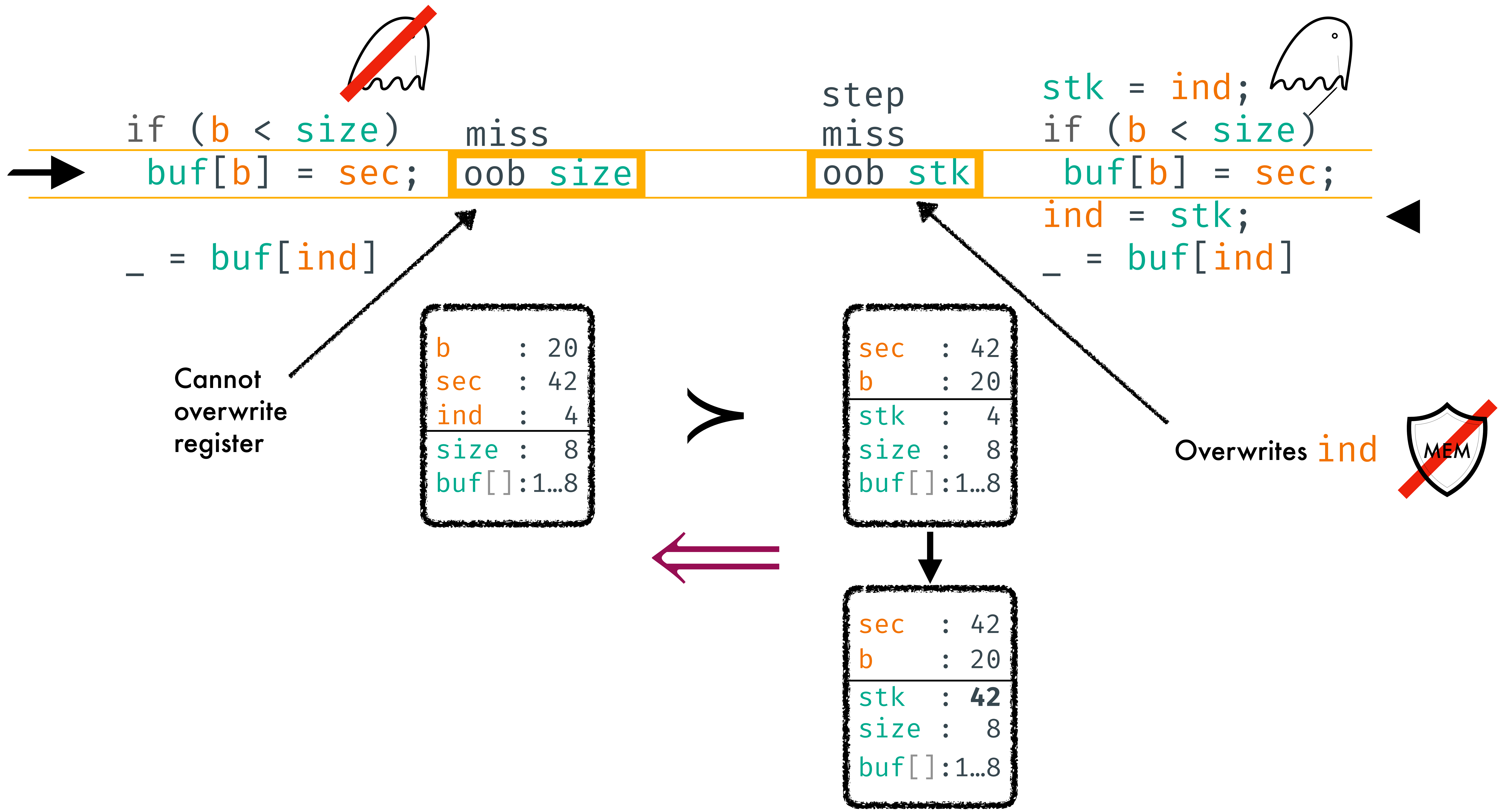
```
b     : 20
sec   : 42
ind   :  4
size  :  8
buf[]:1…8
```

```
sec   : 42
b     : 20
stk   :  4
size  :  8
buf[]:1…8
```

```
sec   : 42
b     : 20
stk   : 42
size  :  8
buf[]:1…8
```

Make $[\,.\,]_{ra} \models$ SNIP again!

Poison



```
if (b < size)
 buf[b] = sec;

_ = buf[ind]
```

```
miss
oob size
```

```
step
miss
oob stk
```

```
stk = ind;
if (b < size)
 buf[b] = sec;
ind = stk;
_ = buf[ind]
```

```
b      : 20
sec    : 42
ind    :  4
size   :  8
buf[]:1…8
```

$\succ$

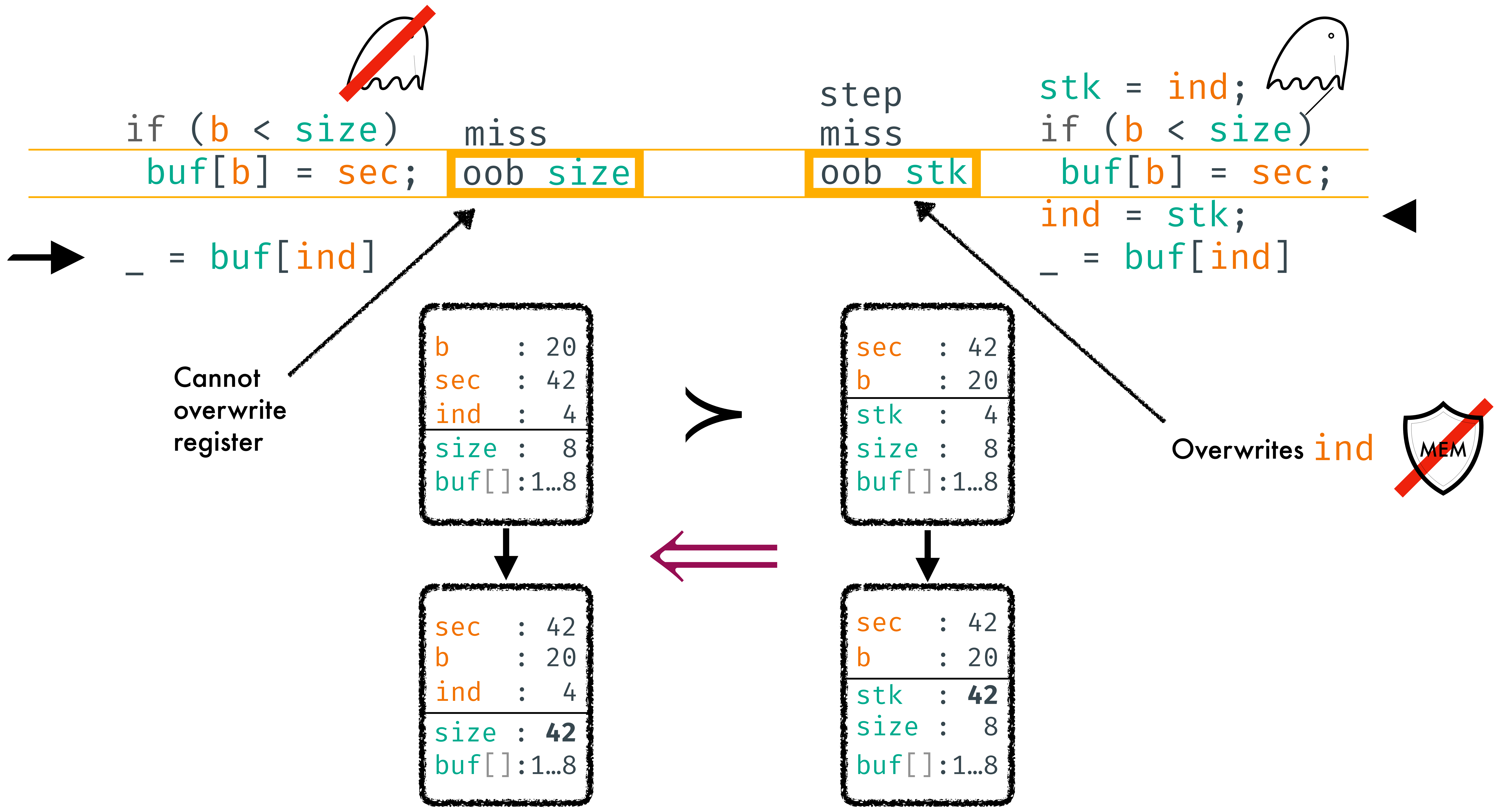
```
sec    : 42
b      : 20
stk    :  4
size   :  8
buf[]:1…8
```

```
sec    : 42
b      : 20
stk    : 42
size   :  8
buf[]:1…8
```

$\Longleftarrow$

Make $[\,.\,]_{ra} \vDash$ SNIP again!

Poison



```
if (b < size)
 buf[b] = sec;

_ = buf[ind]
```

```
miss
oob size
```

```
step
miss
oob stk
```

```
stk = ind;
if (b < size)
 buf[b] = sec;
ind = stk;
_ = buf[ind]
```

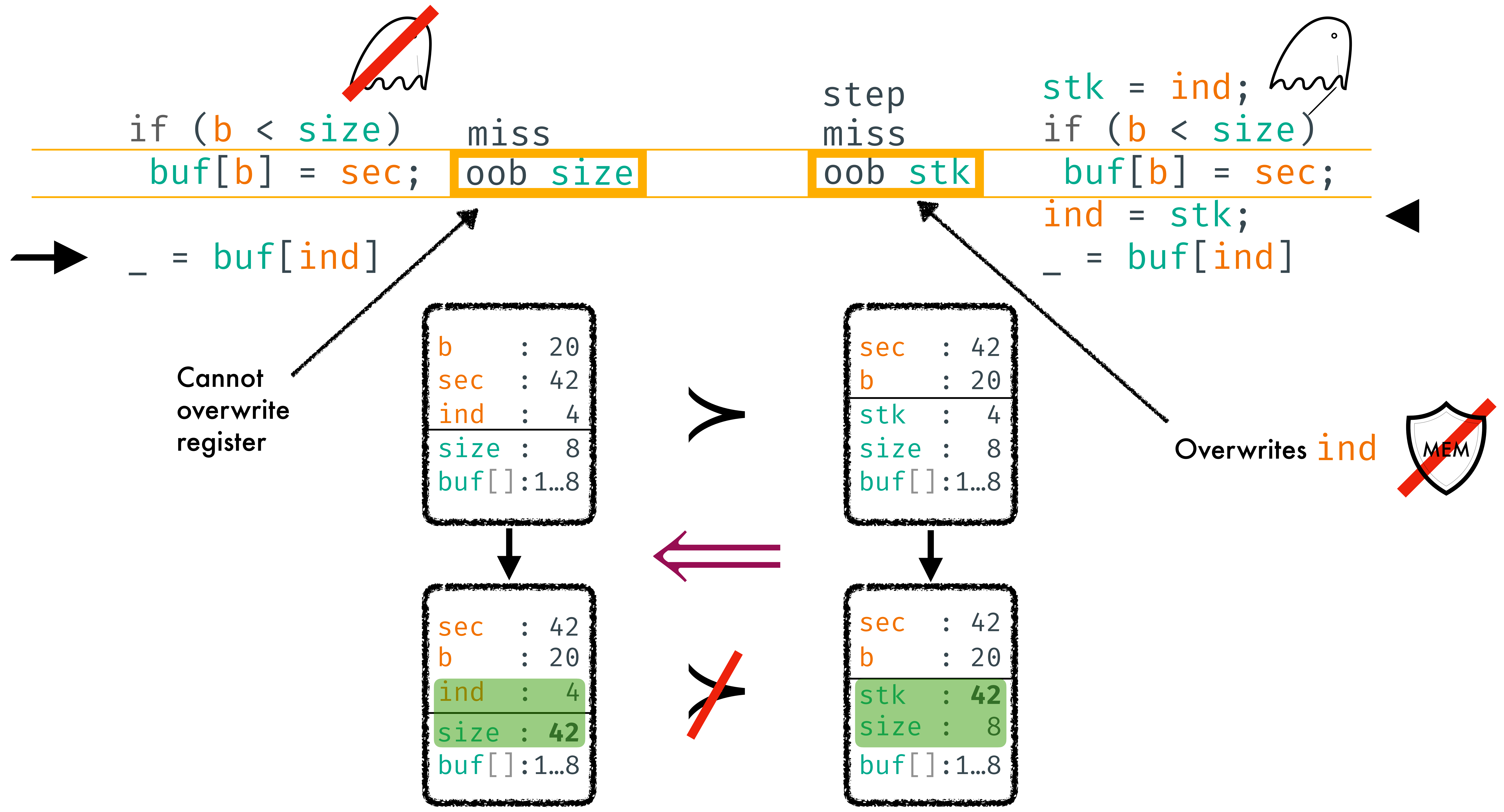
```
b      : 20
sec    : 42
ind    :  4
size   :  8
buf[]:1…8
```

```
sec    : 42
b      : 20
ind    :  4
size   : 42
buf[]:1…8
```

```
sec    : 42
b      : 20
stk    :  4
size   :  8
buf[]:1…8
```

```
sec    : 42
b      : 20
stk    : 42
size   :  8
buf[]:1…8
```

Make $[\,.\,]_{ra} \models$ SNIP again!



```
if (b < size)
 buf[b] = sec;

_ = buf[ind]
```

```
stk = ind;
if (b < size)
 buf[b] = sec;
ind = stk;
_ = buf[ind]
```

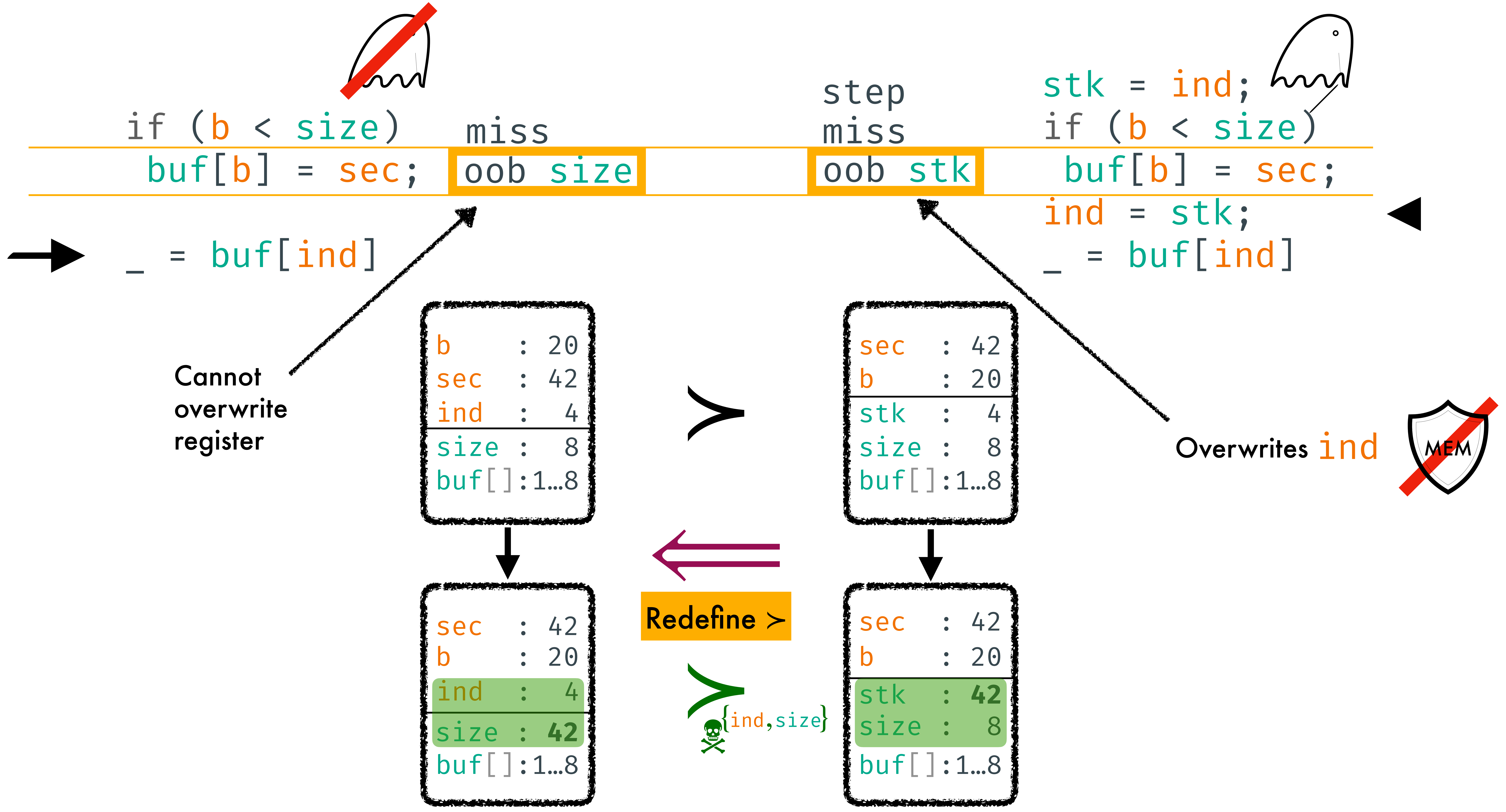
```
b      : 20
sec    : 42
ind    :  4
size   :  8
buf[]:1…8
```

```
sec    : 42
b      : 20
stk    :  4
size   :  8
buf[]:1…8
```

```
sec    : 42
b      : 20
ind    :  4
size   : 42
buf[]:1…8
```

```
sec    : 42
b      : 20
stk    : 42
size   :  8
buf[]:1…8
```

Make $[\,.\,]_{ra} \vDash$ SNIP again!



```
if (b < size)
 buf[b] = sec;

_ = buf[ind]
```

```
stk = ind;
if (b < size)
 buf[b] = sec;
ind = stk;
_ = buf[ind]
```

```
b     : 20
sec   : 42
ind   :  4
size  :  8
buf[]:1…8
```

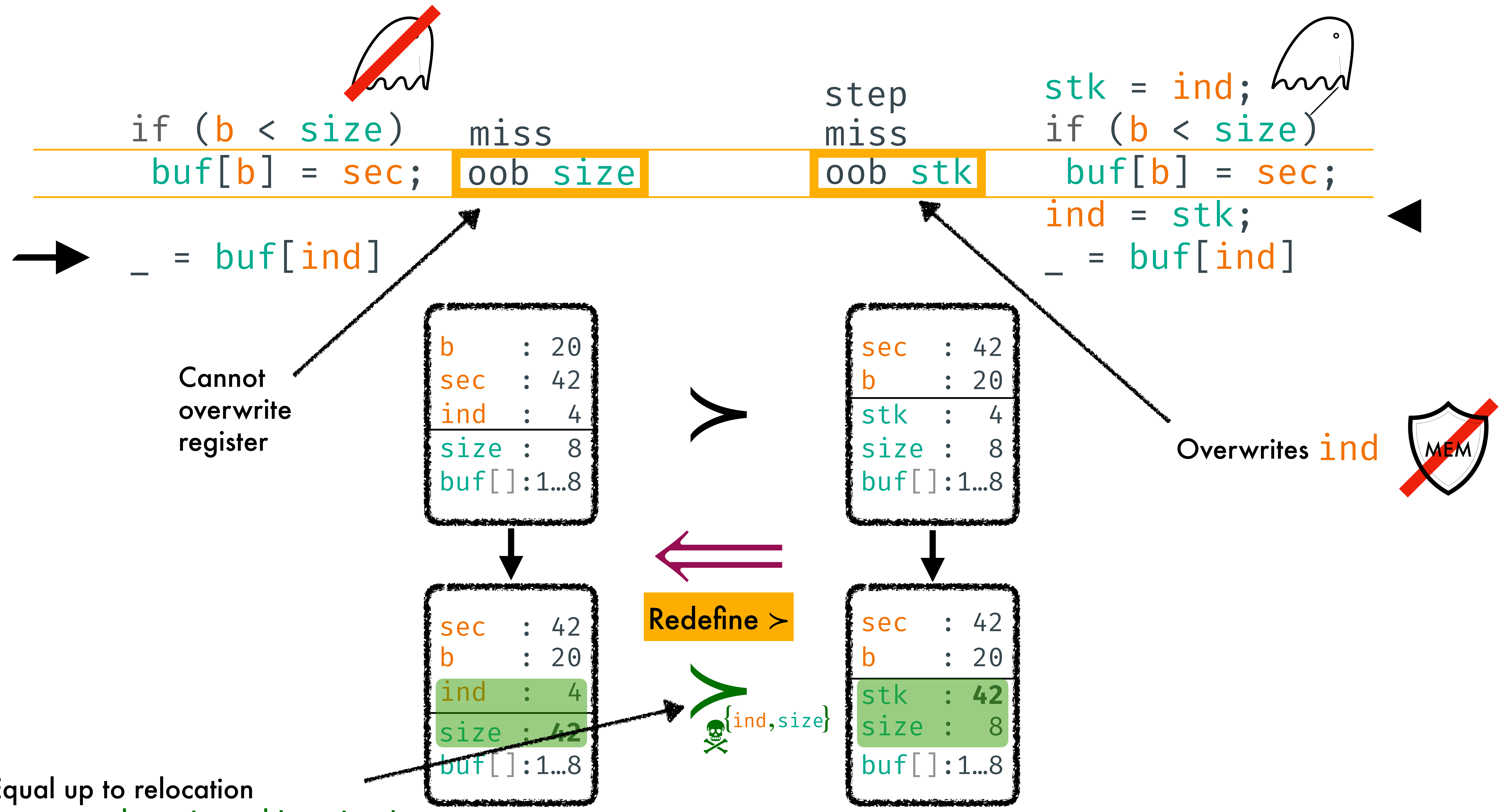
```
sec   : 42
b     : 20
stk   :  4
size  :  8
buf[]:1…8
```

```
sec   : 42
b     : 20
ind   :  4
size  : 42
buf[]:1…8
```

```
sec   : 42
b     : 20
stk   : 42
size  :  8
buf[]:1…8
```

Make $[\,.\,]_{ra} \models$ SNIP again!

Poison



Equal up to relocation
except on the poisoned Locations!

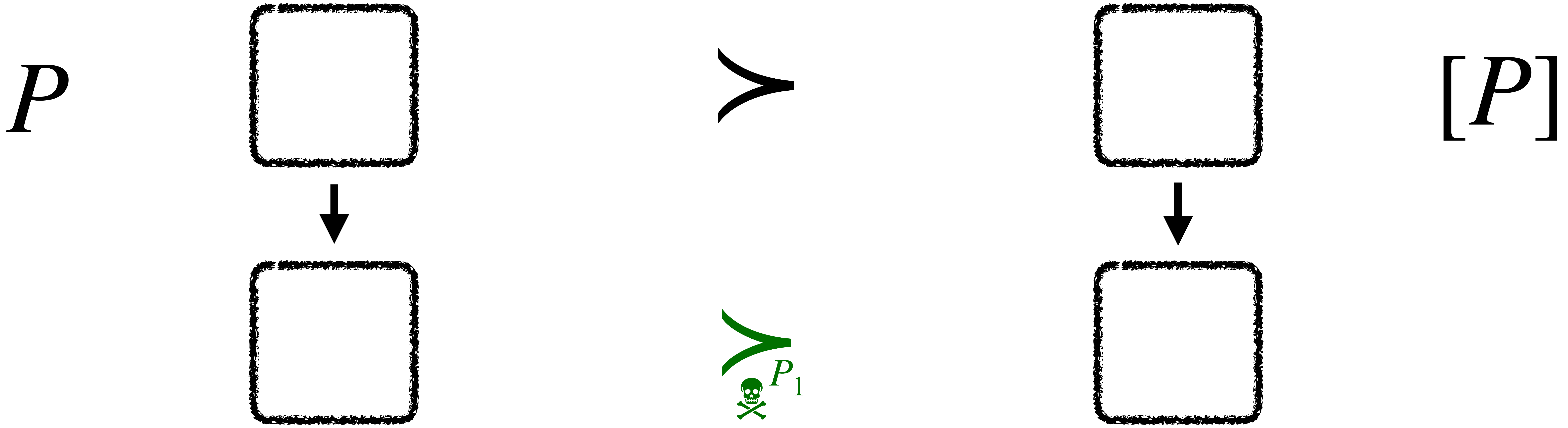
Make $[\,.\,]_{ra} \vDash$ SNIP again!

Define $\succ$

$$P \qquad \square \qquad \succ \qquad \square \qquad [P]$$

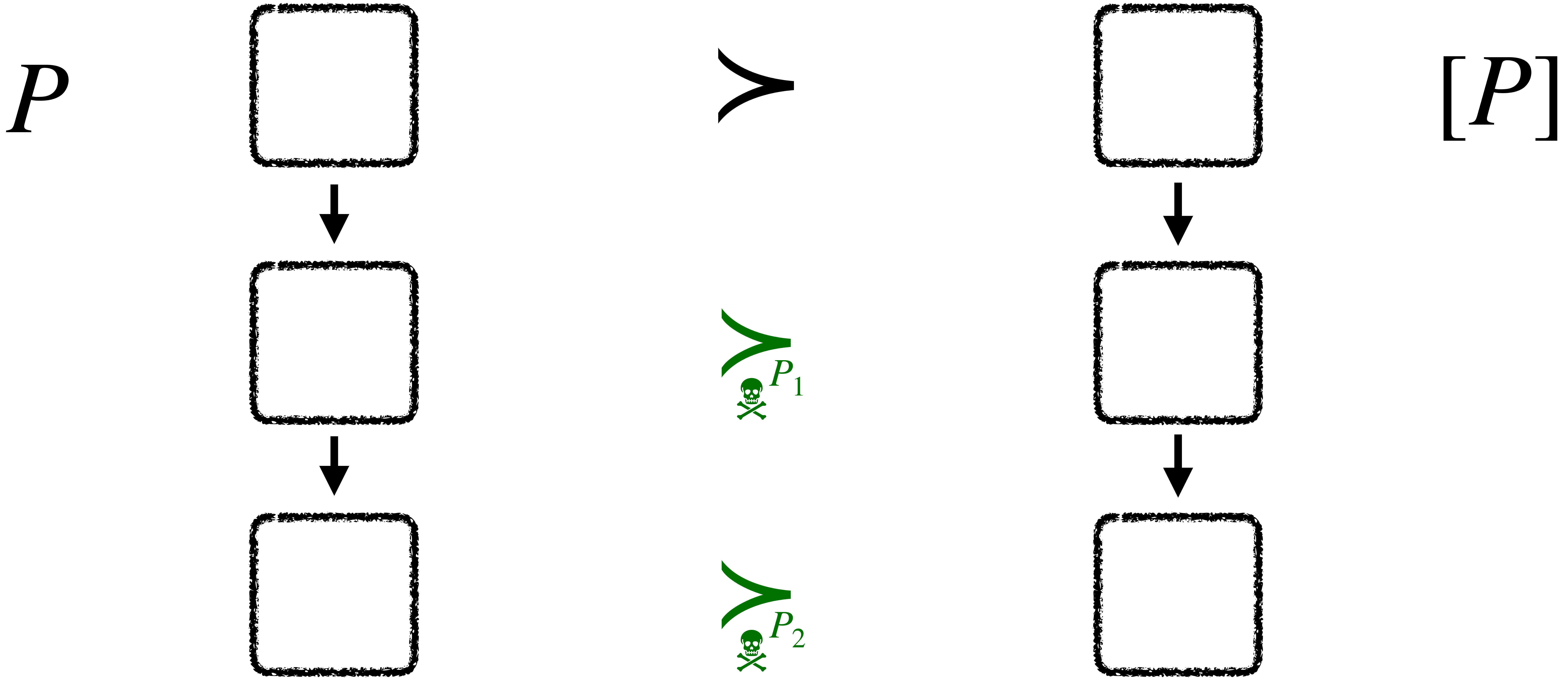
Make $[\,.\,]_{ra} \models$ SNIP again!

Define $\succ$



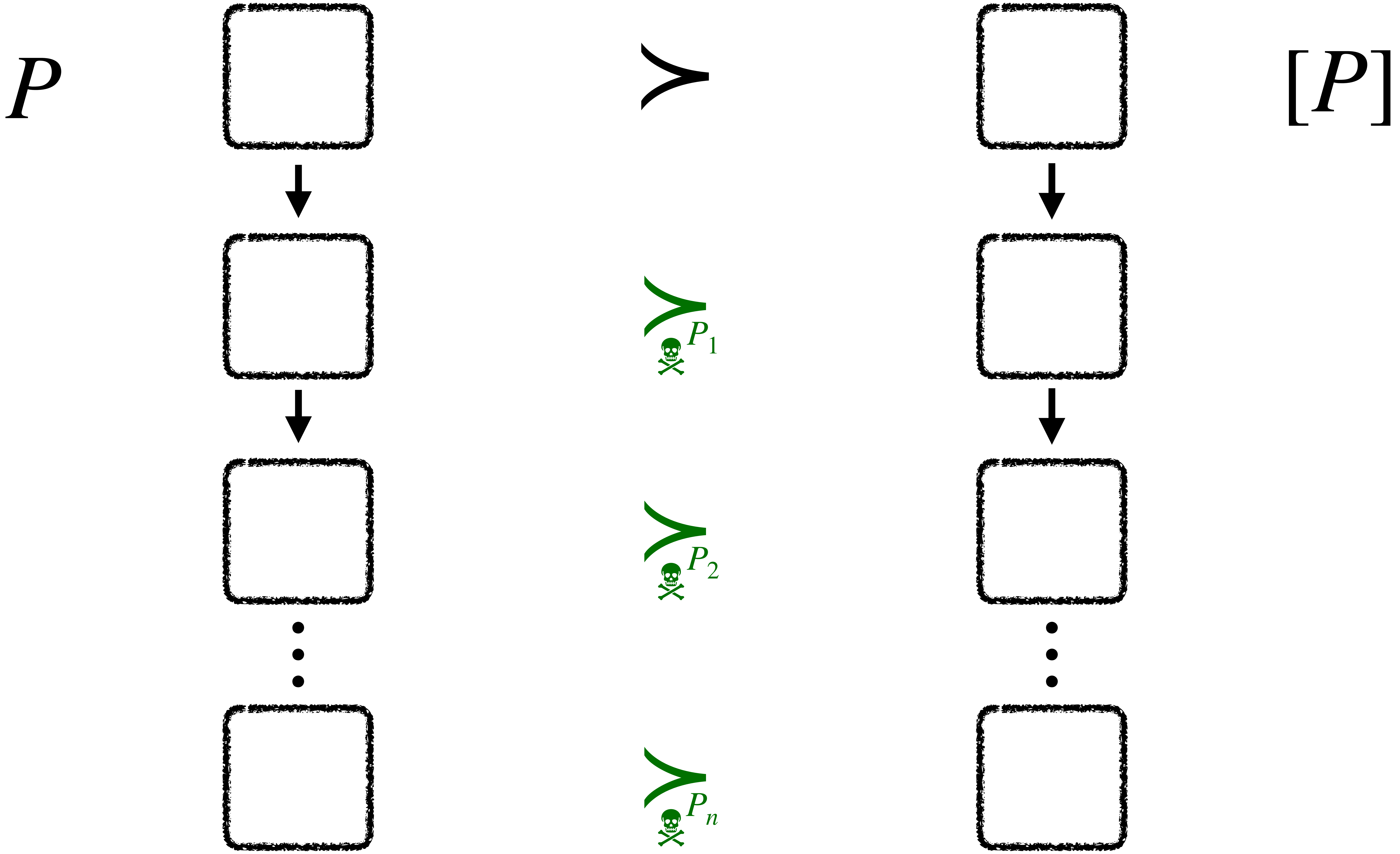
Make $[\,.\,]_{ra} \models$ SNIP again!

Define $\succ$



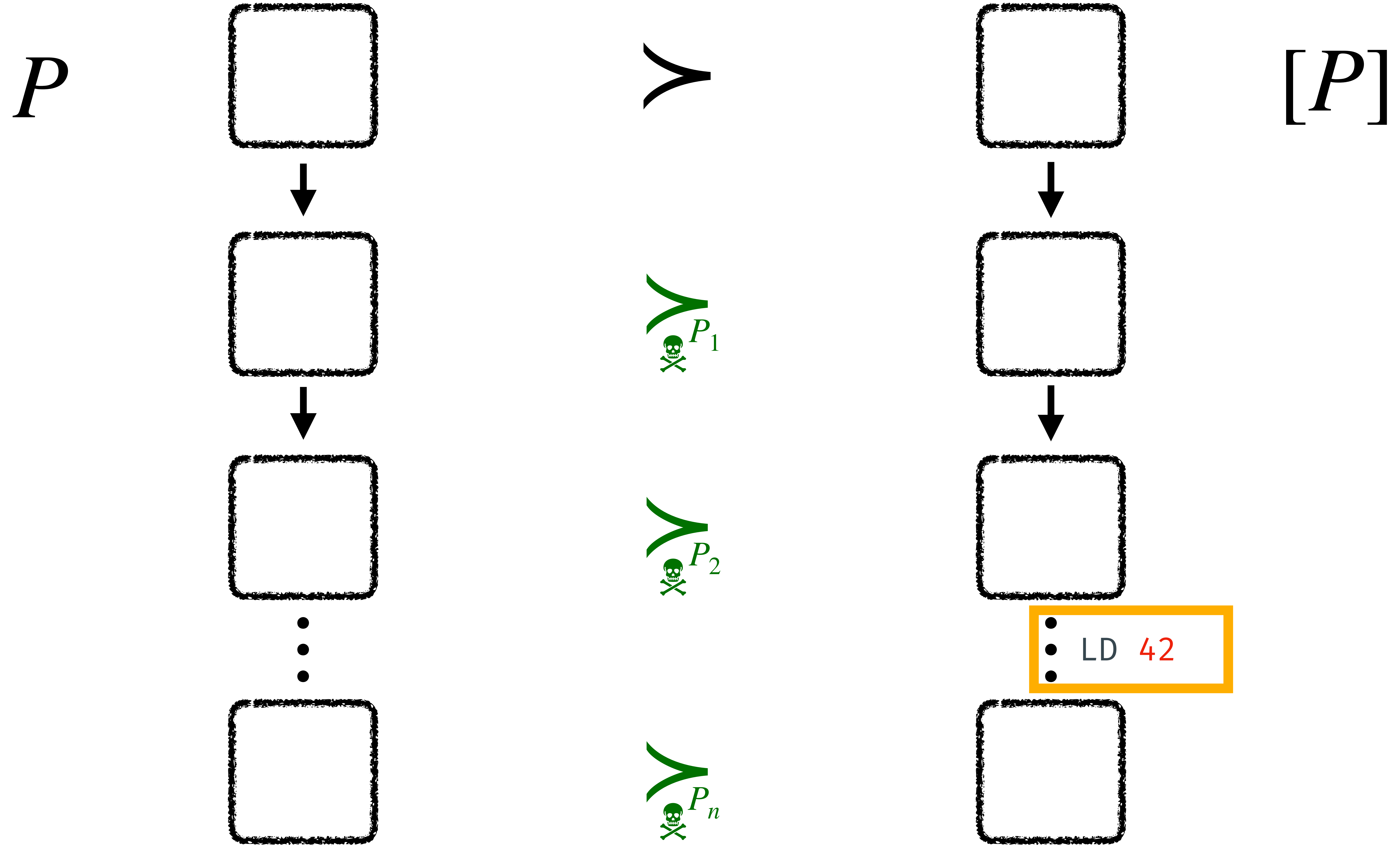
Make $[\,.\,]_{ra} \models$ SNIP again!

Define $\succ$



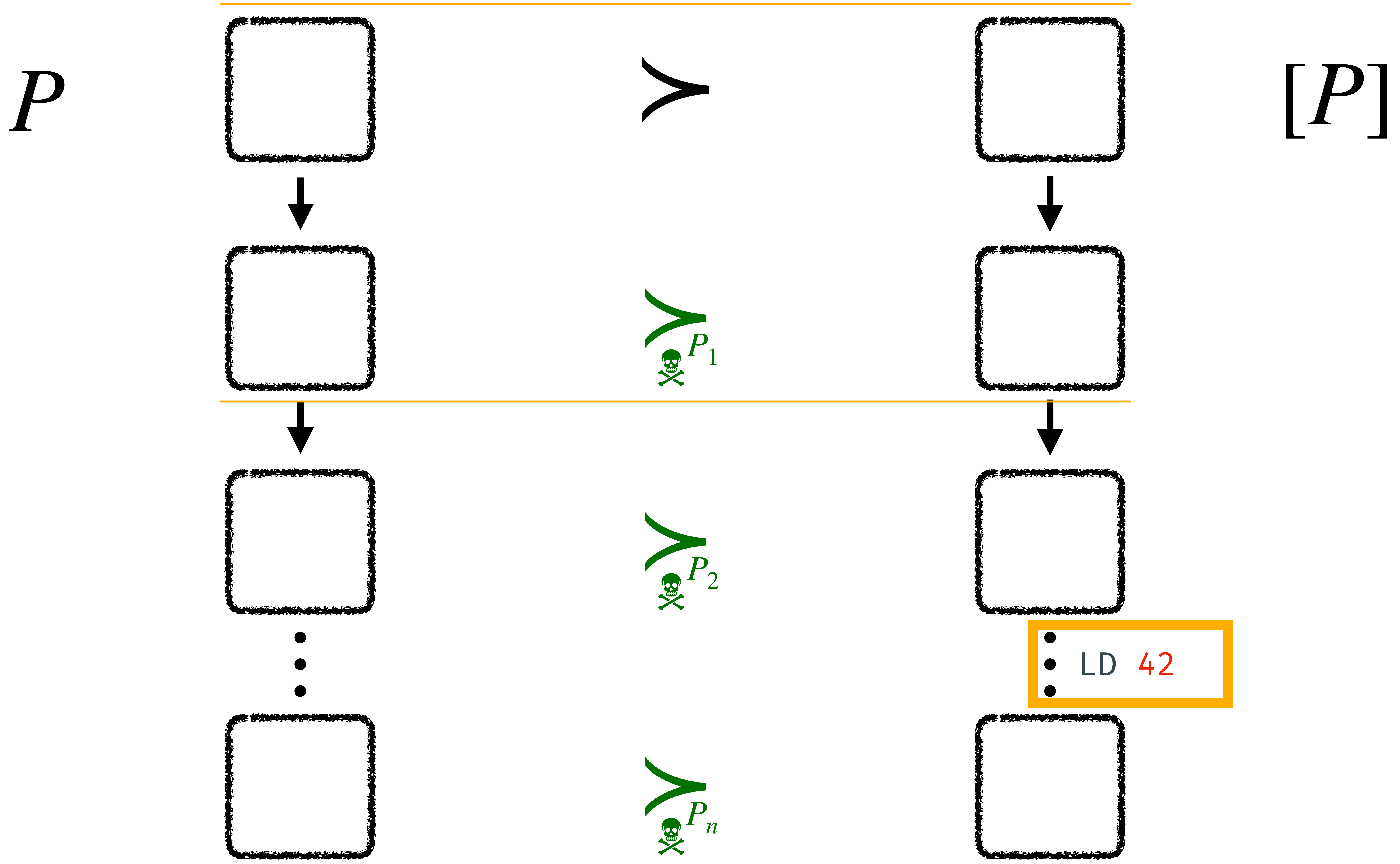
Make $[\,.\,]_{ra} \models$ SNIP again!

Define $\succ$



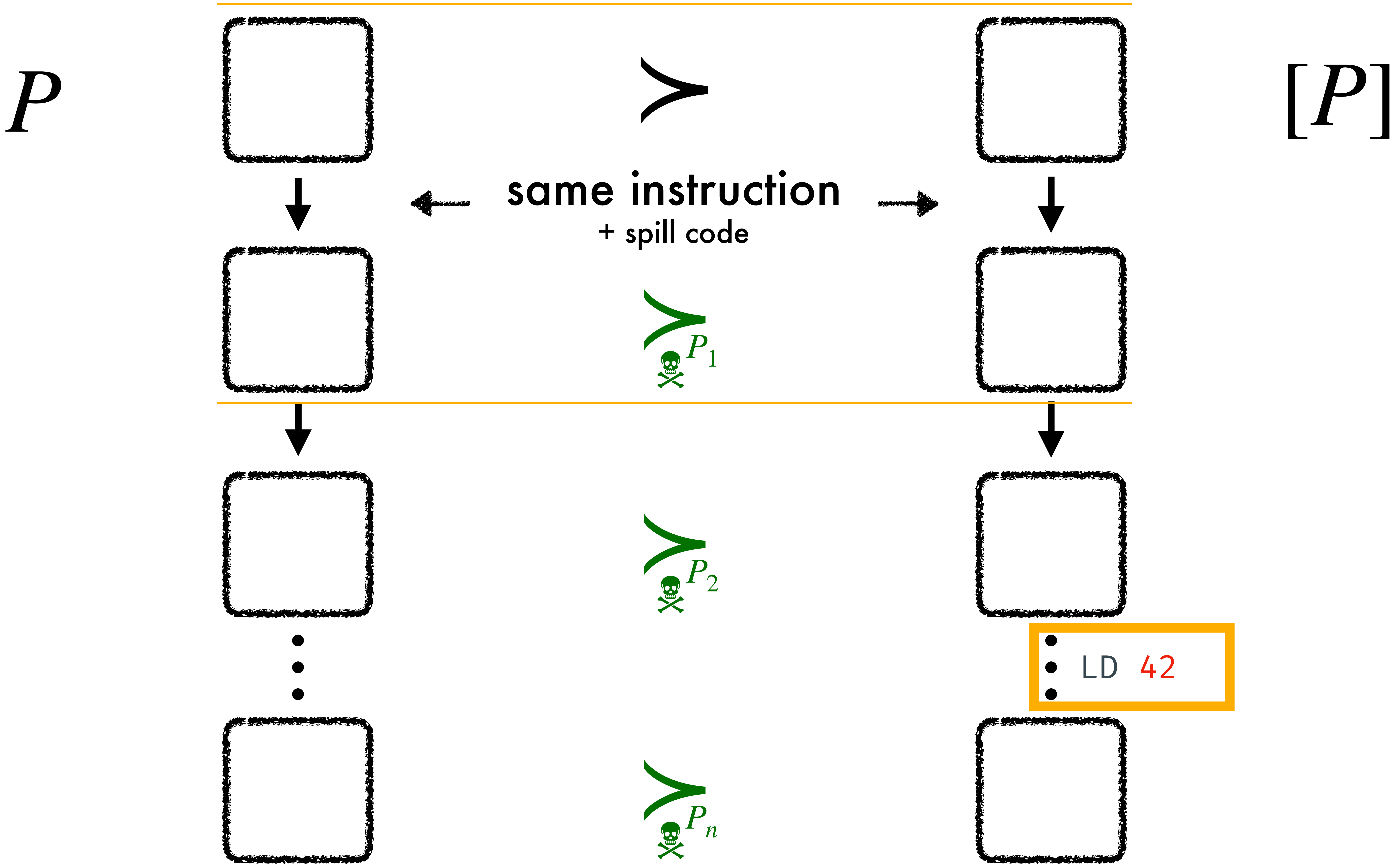
Make $[\,.\,]_{ra} \models$ SNIP again!

Define $\succ$



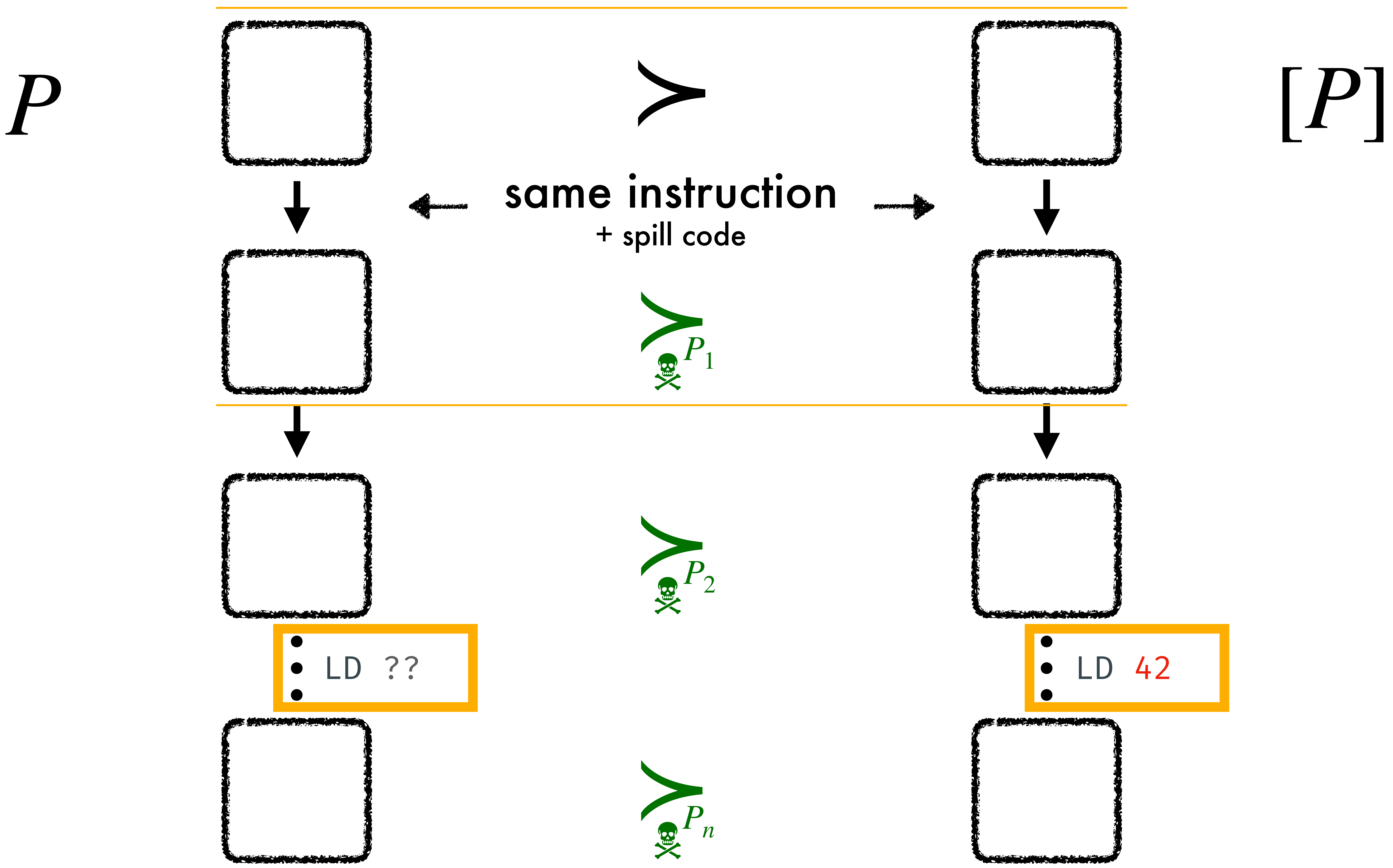
Make $[\,.\,]_{ra} \models$ SNIP again!

Define $\succ$



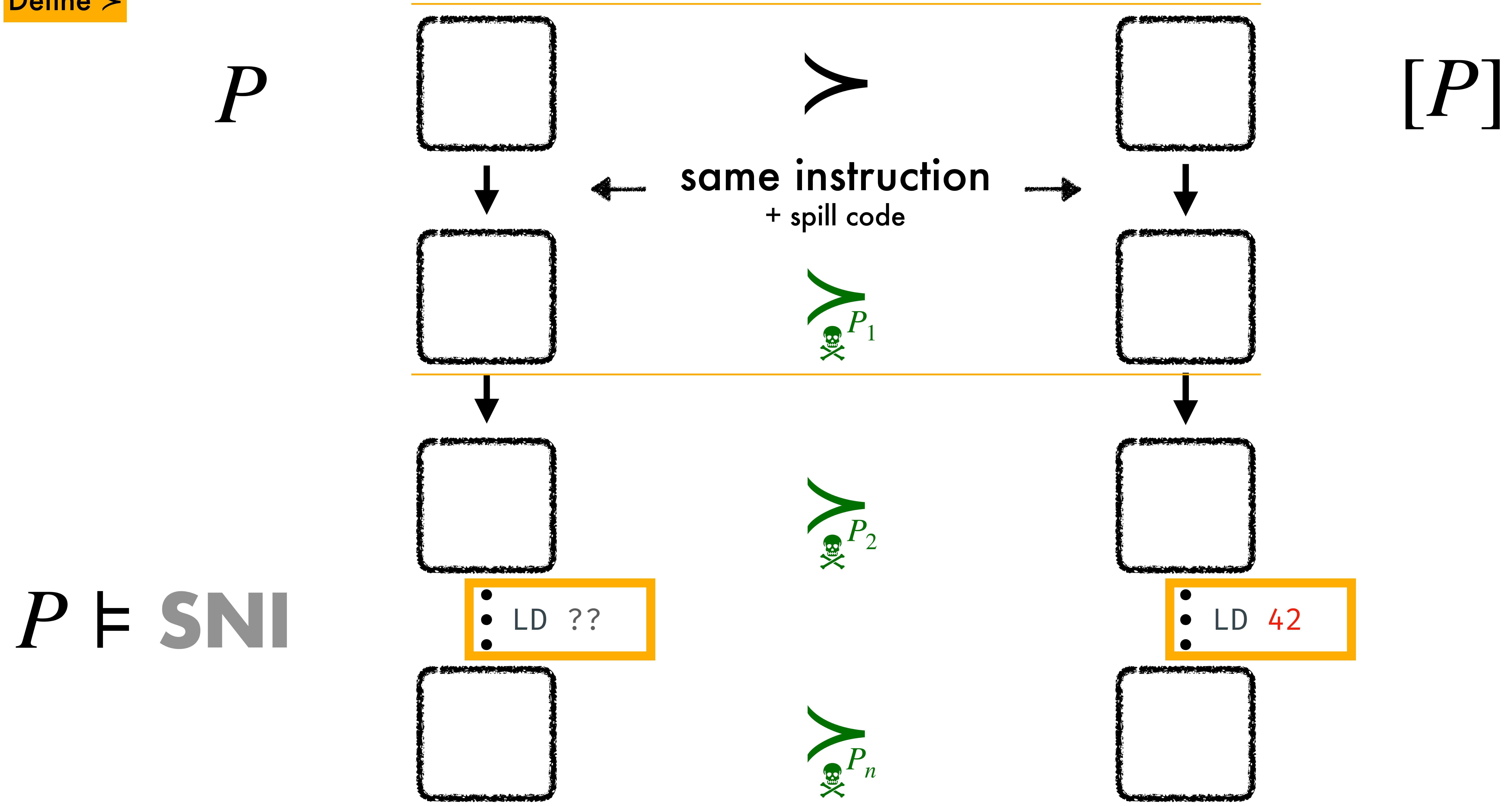
Make $[\,.\,]_{ra} \models$ SNIP again!

Define $\succ$



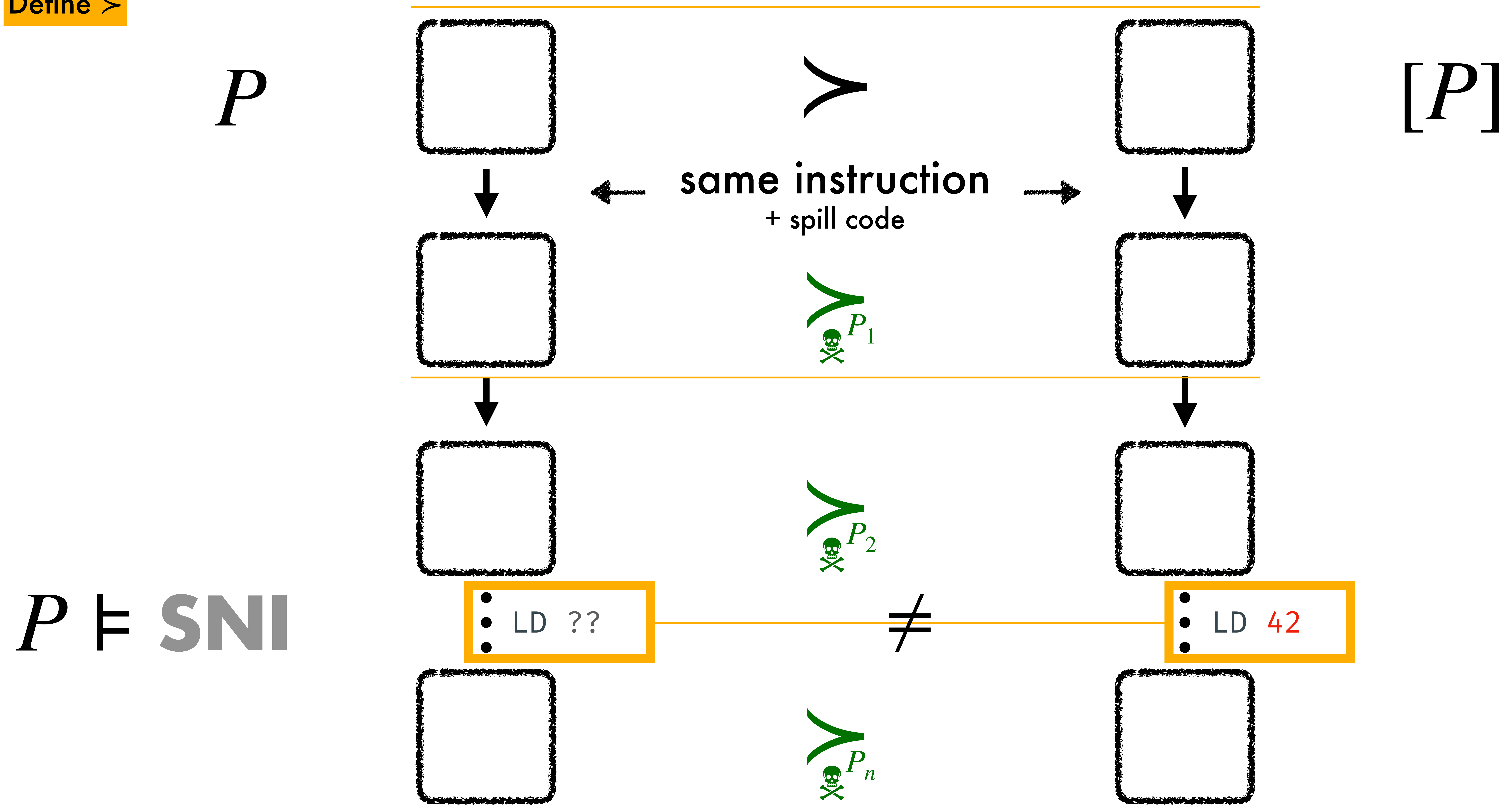
Make $[\,.\,]_{ra} \vDash$ SNIP again!

Define $\succ$



Make $[\,.\,]_{ra} \models$ SNIP again!

Define $\succ$

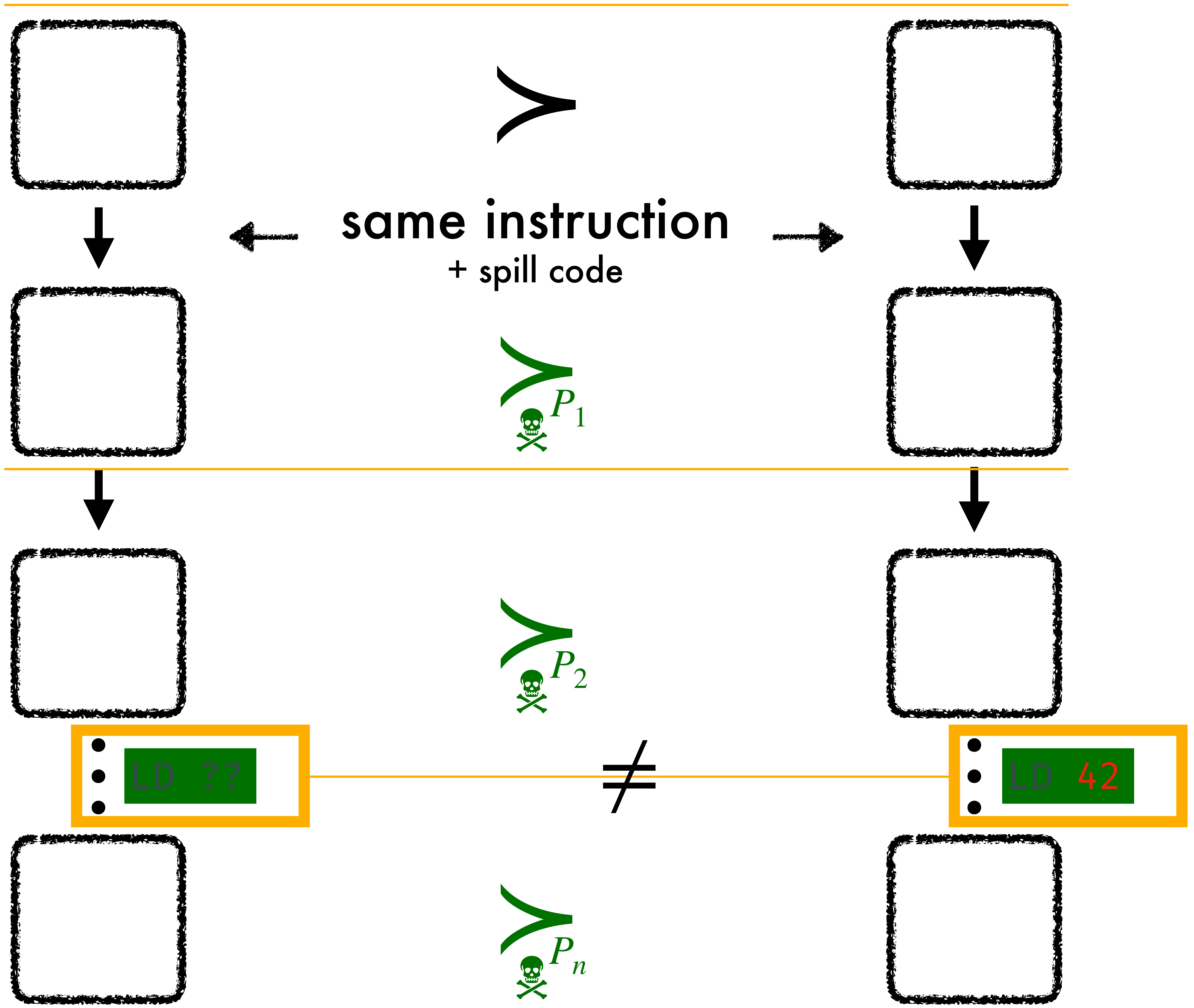


Make $[\,.\,]_{ra} \vDash$ SNIP again!

Define $\succ$

P

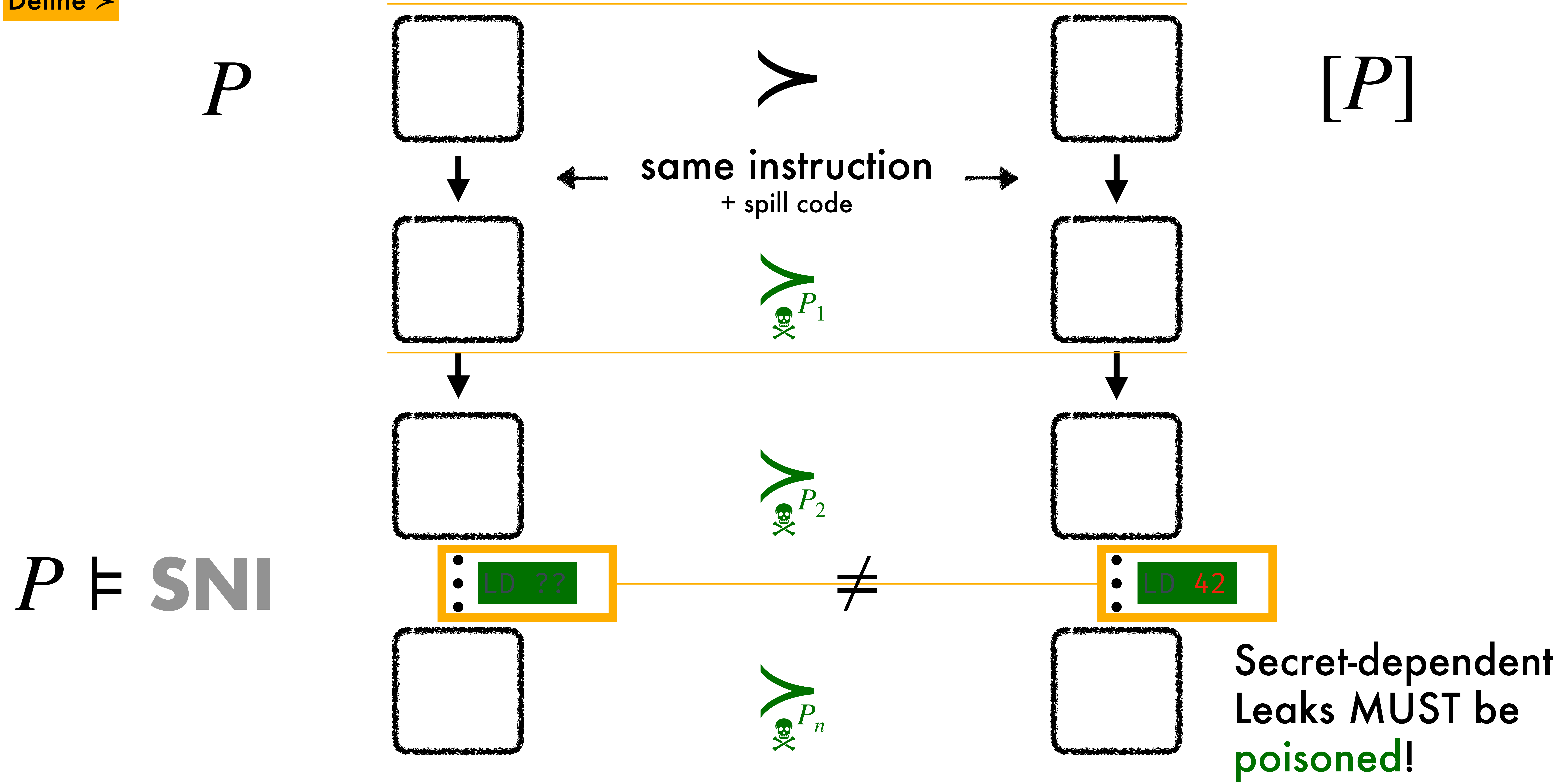
$[P]$



$P \vDash$ SNI

Make $[\,.\,]_{ra} \vDash$ SNIP again!

Define $\succ$



Secret-dependent Leaks MUST be poisoned!

Make $[\,.\,]_{ra} \vDash$ SNIP again!

Define $\succ$

Secret-dependent Leaks MUST be poisoned!

Make $[\,.\,]_{\text{ra}} \models$ SNIP again!

Define $\succ$

Secret-dependent Leaks MUST be poisoned!

No poisoned leaks in $[P]_{\text{ra}}$

Make $[\,.\,]_{\mathsf{ra}} \models$ SNIP again!

Define $\succ$

Secret-dependent Leaks MUST be poisoned!

$$\text{No poisoned leaks in } [P]_{\mathsf{ra}} \implies [P]_{\mathsf{ra}} \models \textbf{SNI}$$

Make $[\,.\,]_{ra} \vDash$ SNIP again!

Define >

Secret-dependent Leaks MUST be poisoned!

No poisoned leaks in $[P]_{ra}$ $\Longrightarrow$ $[P]_{ra} \vDash$ **SNI**

Can we fix Register Allocation so that

$[\,\cdot\,]_{ra} \vDash$ **SNIP** ?

(And in a better way than just inserting Mitigations everywhere?)

Make $[\,.\,]_{ra} \models$ **SNIP again!**

Define >

Secret-dependent Leaks MUST be poisoned!

No poisoned leaks in $[P]_{ra}$ $\Longrightarrow$ $[P]_{ra} \models$ **SNI**

Can we fix Register Allocation so that

$[\,\cdot\,]_{ra} \models$ **SNIP** ?

Yes! Stop leaking poisoned values!

(And in a better way than just inserting Mitigations everywhere?)

Stop Leaking Poisoned Values

Stop Leaking Poisoned Values

Does $[P]_{\mathsf{ra}}$ *leak poisoned values? And if so, where?*

Stop Leaking Poisoned Values

Does $[P]_{\text{ra}}$ *leak poisoned values? And if so, where?*

$\boxed{\text{Static Poison Analysis}}$

Stop Leaking Poisoned Values

Does $[P]_{\text{ra}}$ *leak poisoned values? And if so, where?*

$\boxed{\text{Static Poison Analysis}}$

How to stop the leak?

Stop Leaking Poisoned Values

Does $[P]_{\text{ra}}$ leak poisoned values? And if so, where?

Static Poison Analysis

How to stop the leak?

Insert Mitigations

```
a = slh(a);  SFENCE;
```

Fixing RegAlloc

$P \models$ **SNI**

Fixing RegAlloc

$$\begin{tikzcd}
P \models \text{SNI} \arrow[d] \\
[P]_{\text{ra}}
\end{tikzcd}$$

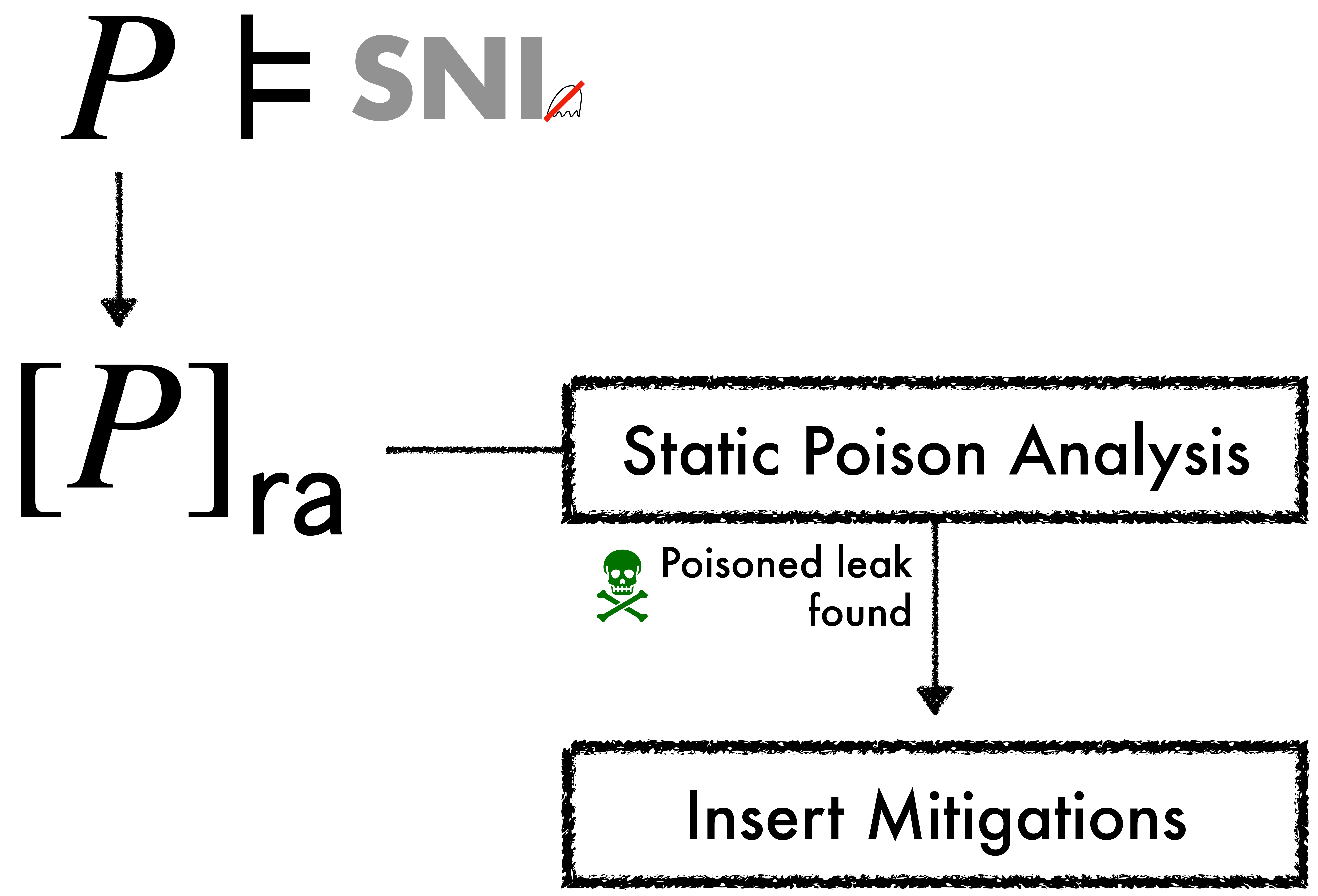
Fixing RegAlloc

$$\begin{tikzcd} P \models \textbf{SNI} \arrow[d] & \\ {[P]_{\mathsf{ra}}} \arrow[r, dash] & \boxed{\text{Static Poison Analysis}} \end{tikzcd}$$

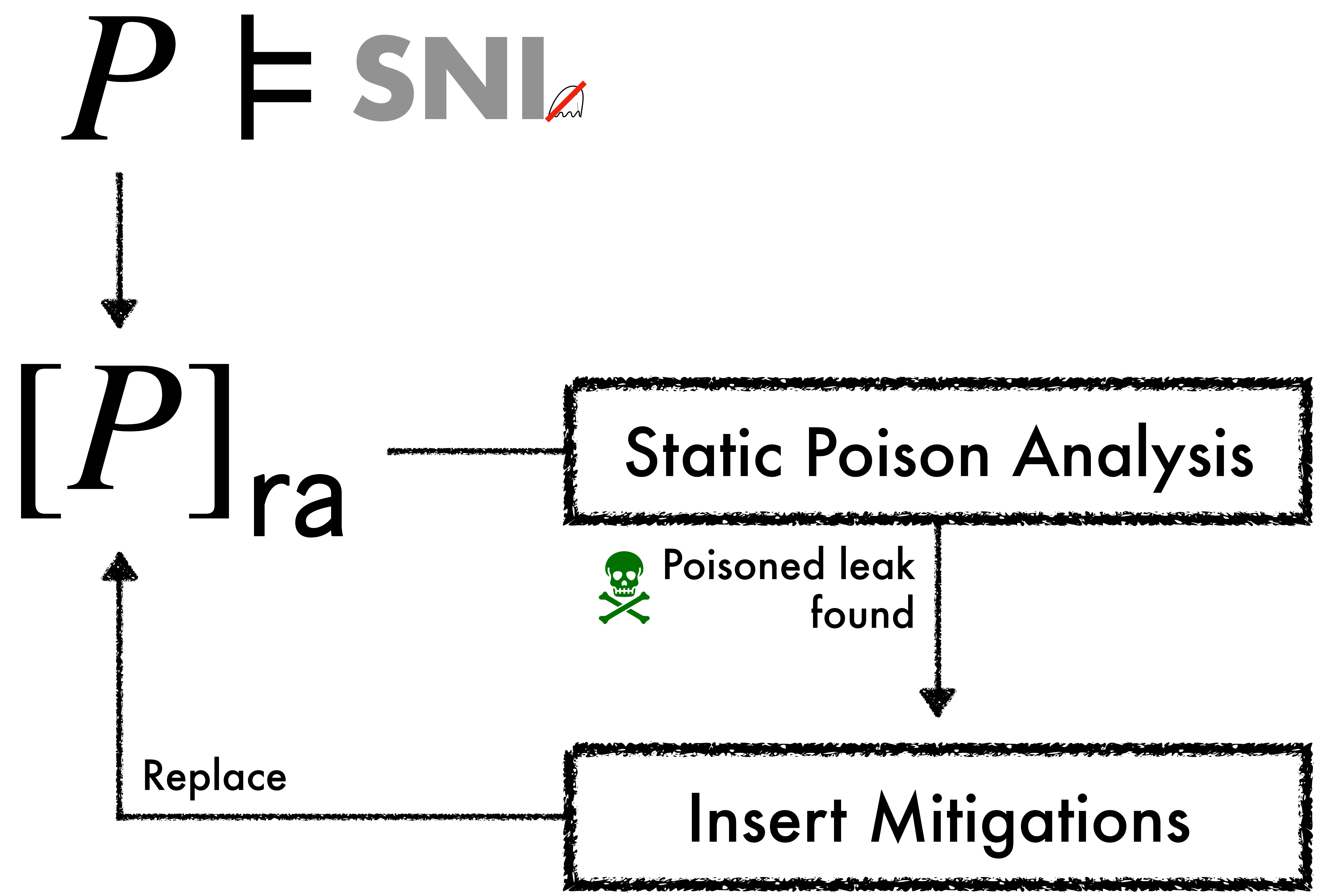
Fixing RegAlloc

$$\begin{tikzcd}
P \models \textbf{SNI} \arrow[d] & \\
{[P]_{\text{ra}}} \arrow[r, dash] & \boxed{\text{Static Poison Analysis}} \arrow[d, "\text{☠ Poisoned leak found}"'] \\
 & {}
\end{tikzcd}$$

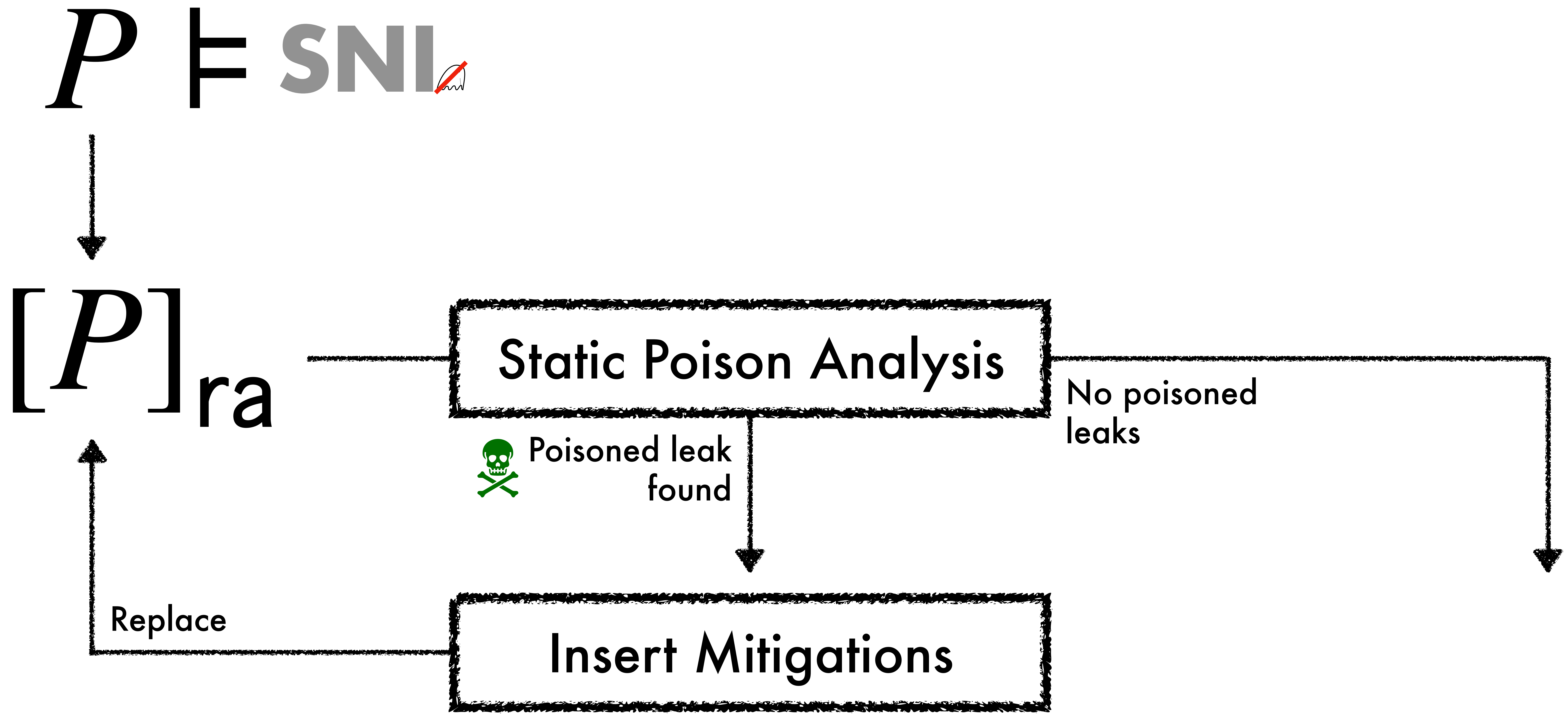
Fixing RegAlloc



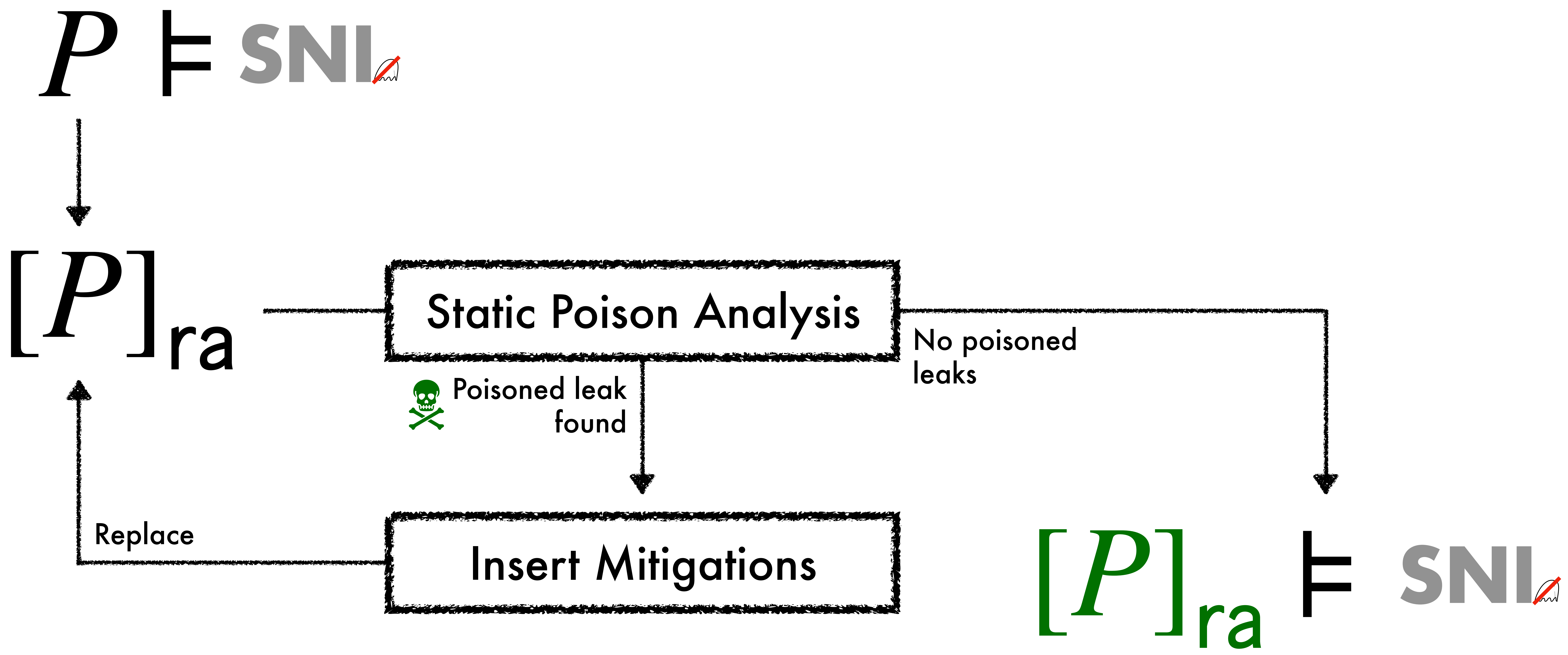
Fixing RegAlloc



Fixing RegAlloc



Fixing RegAlloc



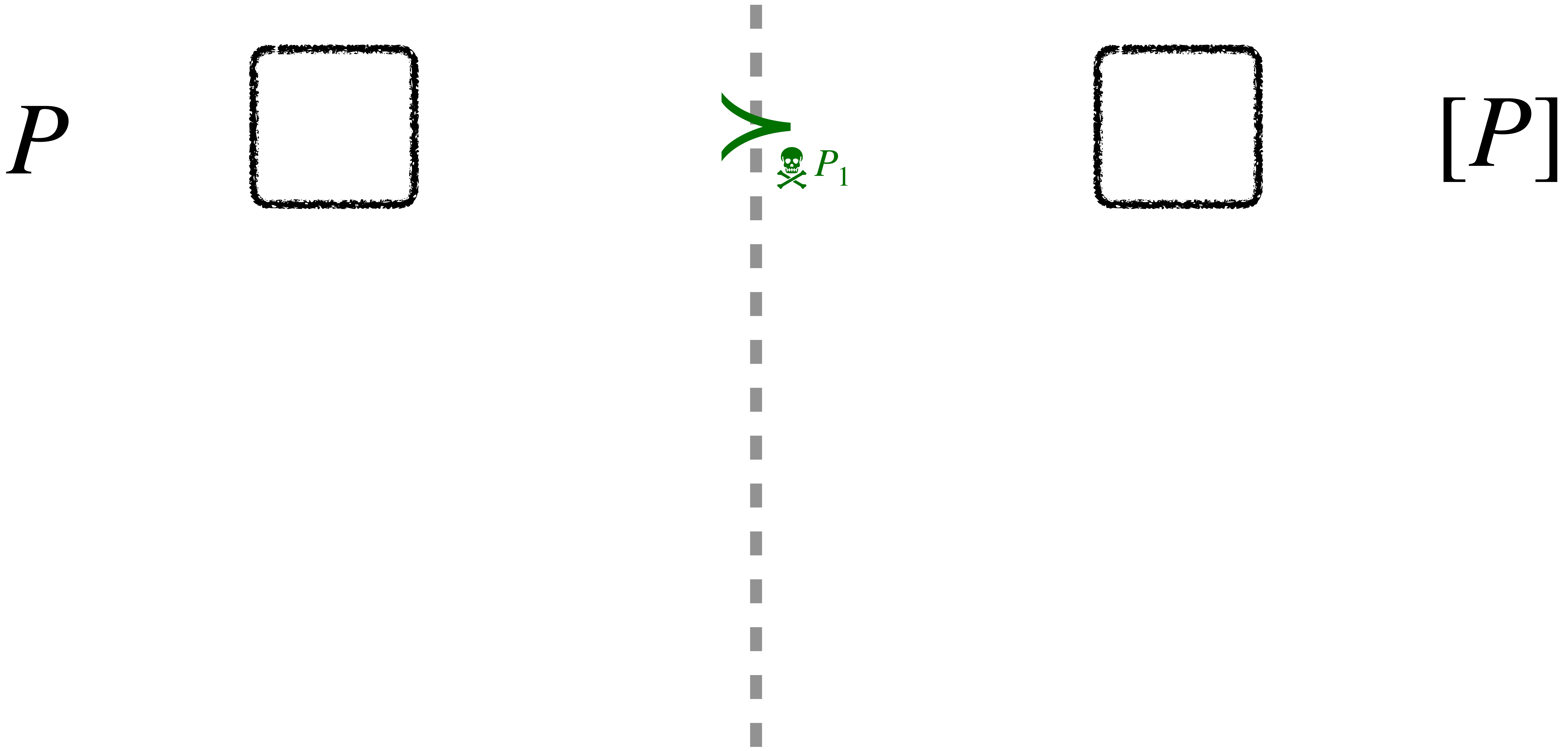
Static Poison Analysis

☠️ Defined between P and $[P]$

P

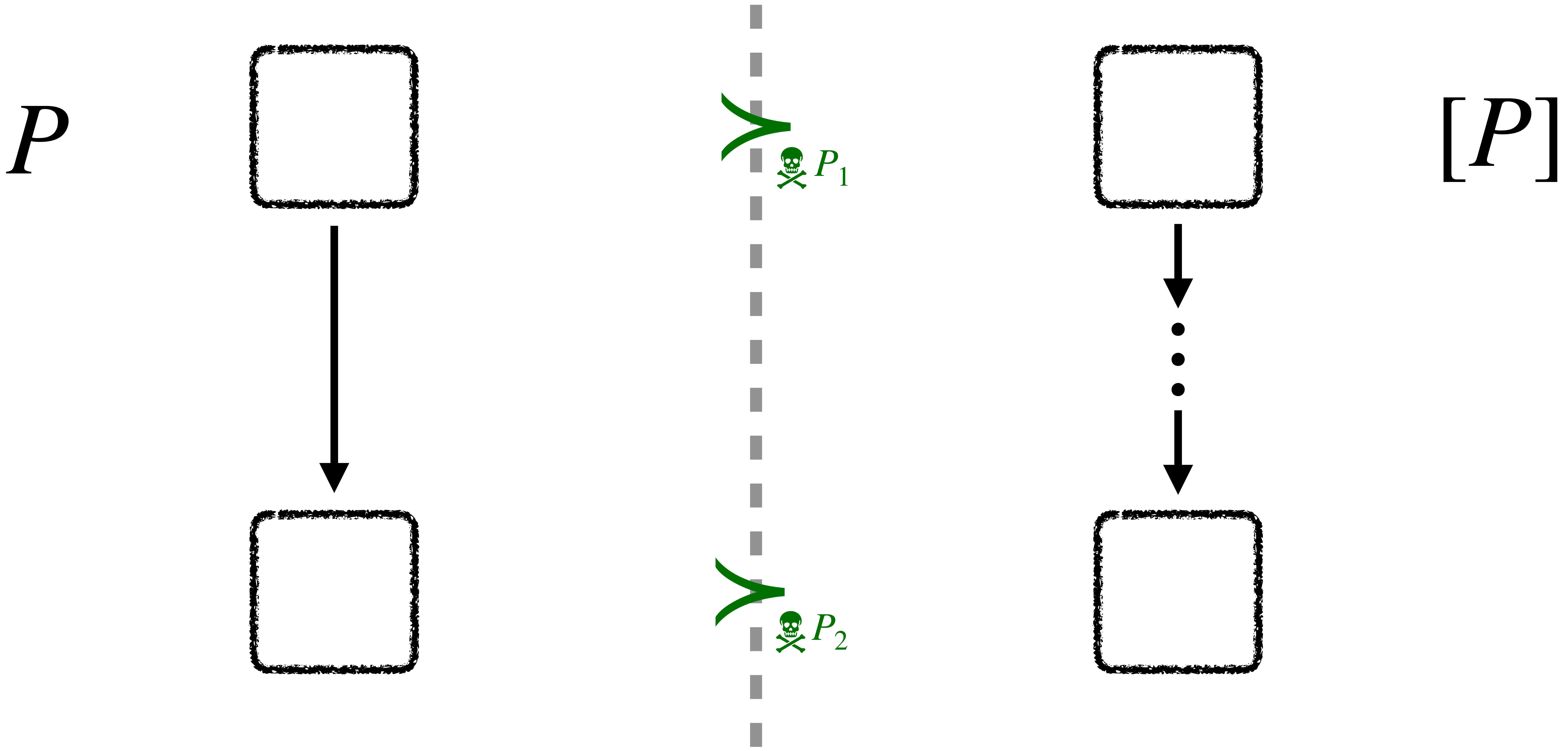
$[P]$

☠ Defined between P and $[P]$



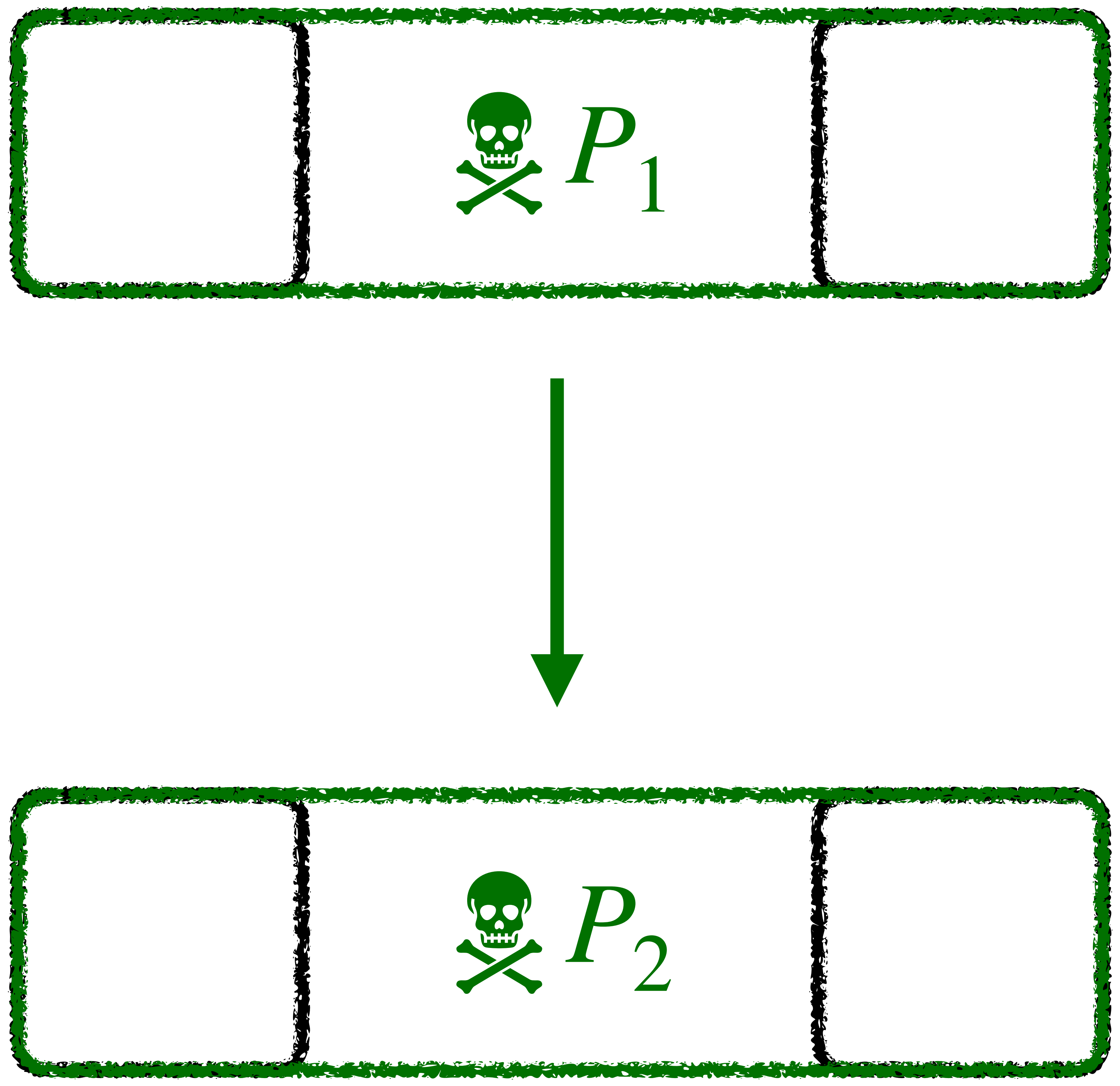
Static Poison Analysis

☠ Defined between P and $[P]$



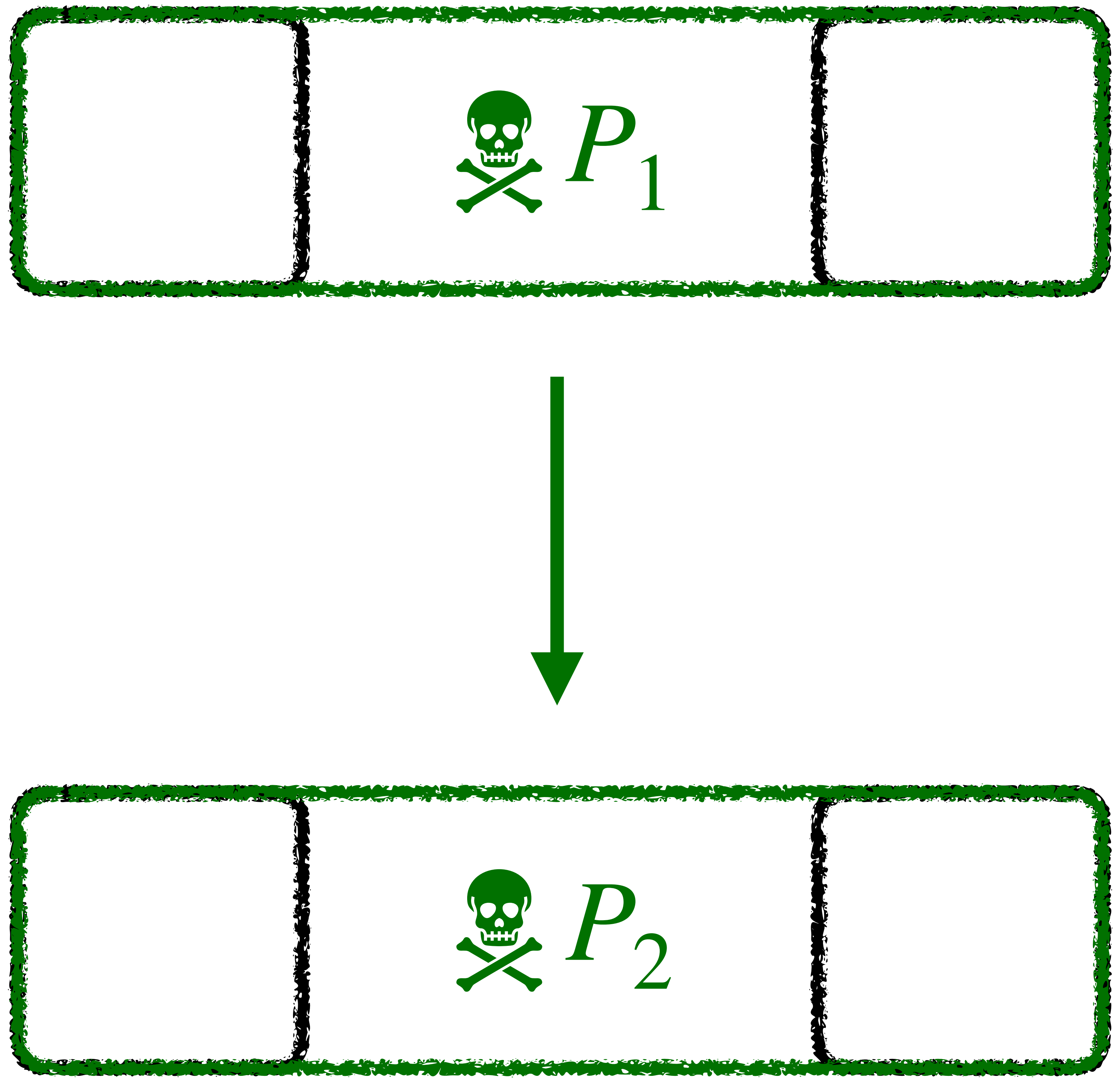
☠ Defined between P and $[P]$

$$P :: [P]$$



☠ Defined between P and $[P]$

$$P \; :: \; [P]$$



☠ P Statically approximate (ctrl-flow sensitive)

☠ P Statically approximate (ctrl-flow sensitive)

☠ P Statically approximate (ctrl-flow sensitive)

```
                          a stk = ind;
1 if (b < size)           b if (b < size)
2  buf[b] = sec;  • •     c  buf[b] = sec;
                  • •     d ind = stk;

3 _ = buf[ind]            f _ = buf[ind]
```

☠ P Statically approximate (ctrl-flow sensitive)

```
                                  a stk = ind;
1 if (b < size)                   b if (b < size)
2  buf[b] = sec;  • •             c  buf[b] = sec;
                  • •             d ind = stk;

3 _ = buf[ind]                    f _ = buf[ind]
```

☠ P Statically approximate (ctrl-flow sensitive)

```
                                  a stk = ind;
1 if (b < size)                   b if (b < size)
2  buf[b] = sec;  • •             c  buf[b] = sec;
                  • •             d ind = stk;

3 _ = buf[ind]                    f _ = buf[ind]
```


☠ P Statically approximate (ctrl-flow sensitive)

```
                            a stk = ind;
1 if (b < size)             b if (b < size)
2  buf[b] = sec;  • •       c  buf[b] = sec;
                  • •       d ind = stk;

3 _ = buf[ind]              f _ = buf[ind]
```

1,a	∅

☠ P Statically approximate (ctrl-flow sensitive)

```
                      a stk = ind;
→ 1 if (b < size)     b if (b < size)     ←
  2  buf[b] = sec; •• c  buf[b] = sec;
                   •• d ind = stk;

  3 _ = buf[ind]      f _ = buf[ind]
```

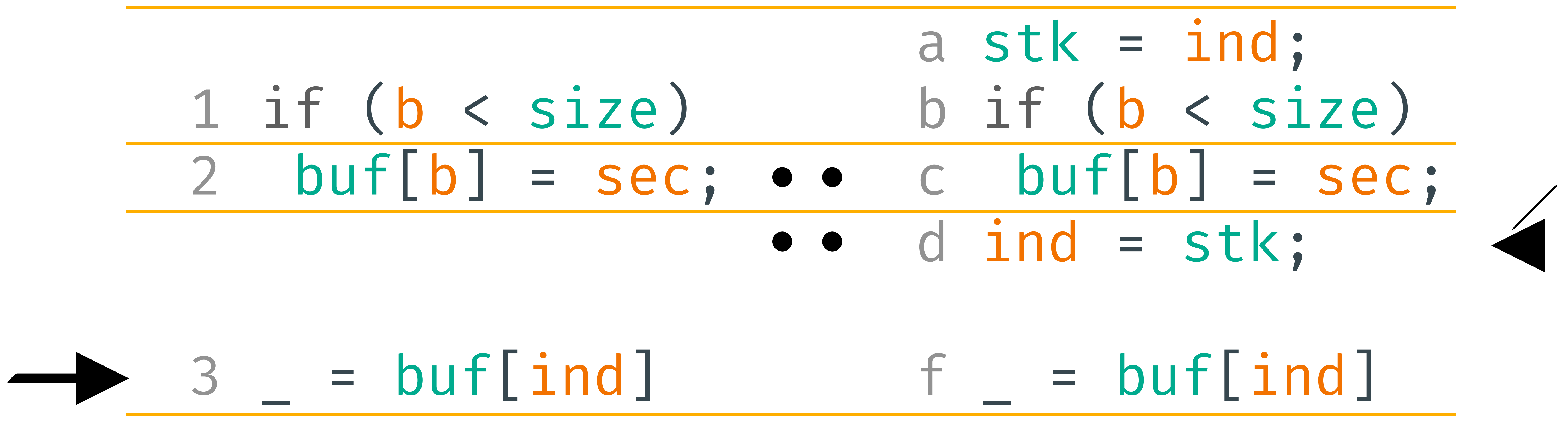
1,a	Ø
1,b	Ø

☠ P Statically approximate (ctrl-flow sensitive)

```
                              a stk = ind;
  1 if (b < size)             b if (b < size)
→ 2  buf[b] = sec;  • •       c  buf[b] = sec;   ←
                    • •       d ind = stk;

  3 _ = buf[ind]              f _ = buf[ind]
```

1,a	Ø
1,b	Ø
2,c	Ø

☠ P Statically approximate (ctrl-flow sensitive)



```
                                a stk = ind;
1 if (b < size)                 b if (b < size)
2  buf[b] = sec;     • •        c  buf[b] = sec;
                     • •        d ind = stk;

3 _ = buf[ind]                  f _ = buf[ind]
```

1,a	∅
1,b	∅
2,c	∅
3,d	??

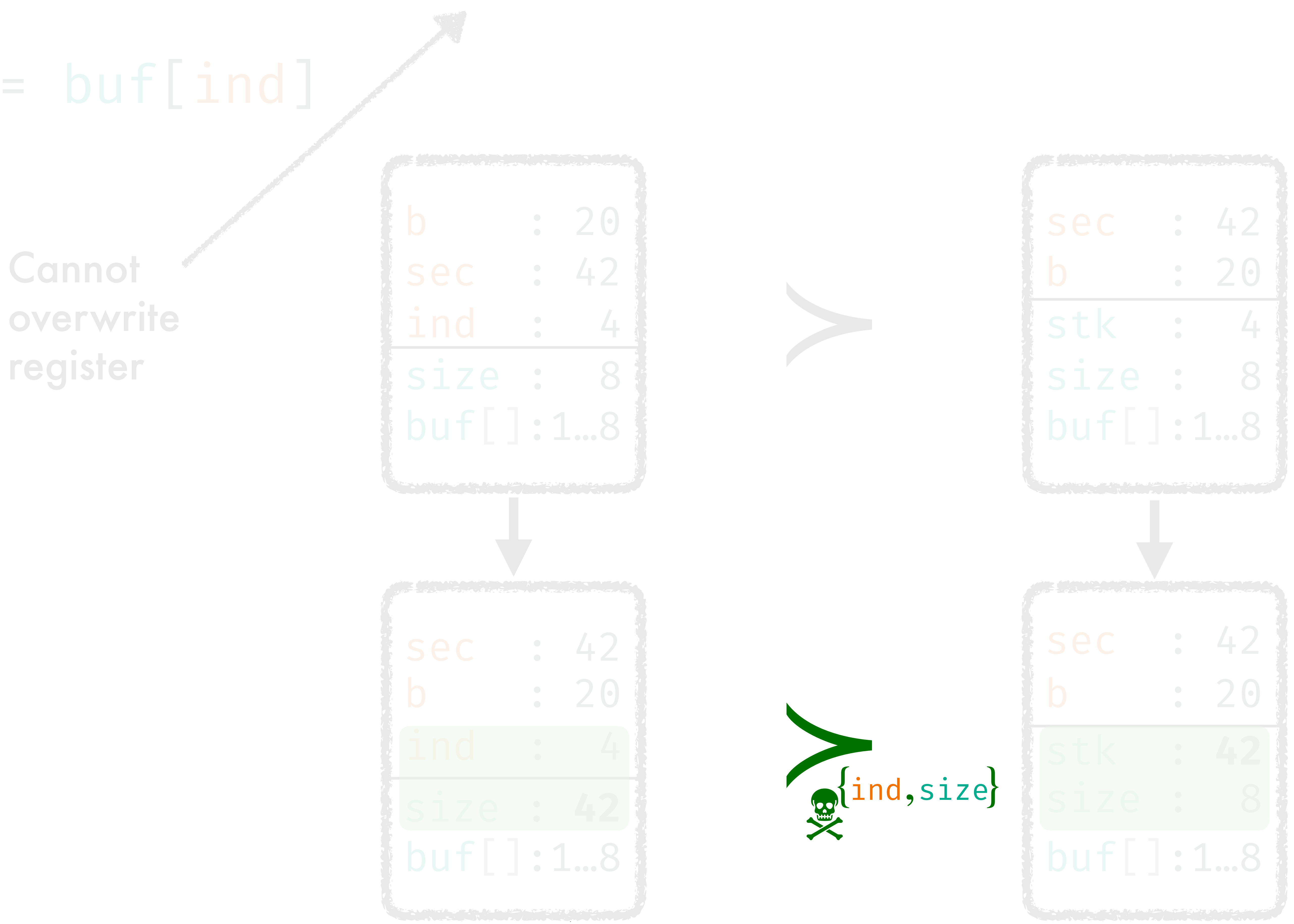
Earlier...

```
if (b < size)
 buf[b] = sec;

_ = buf[ind]
```

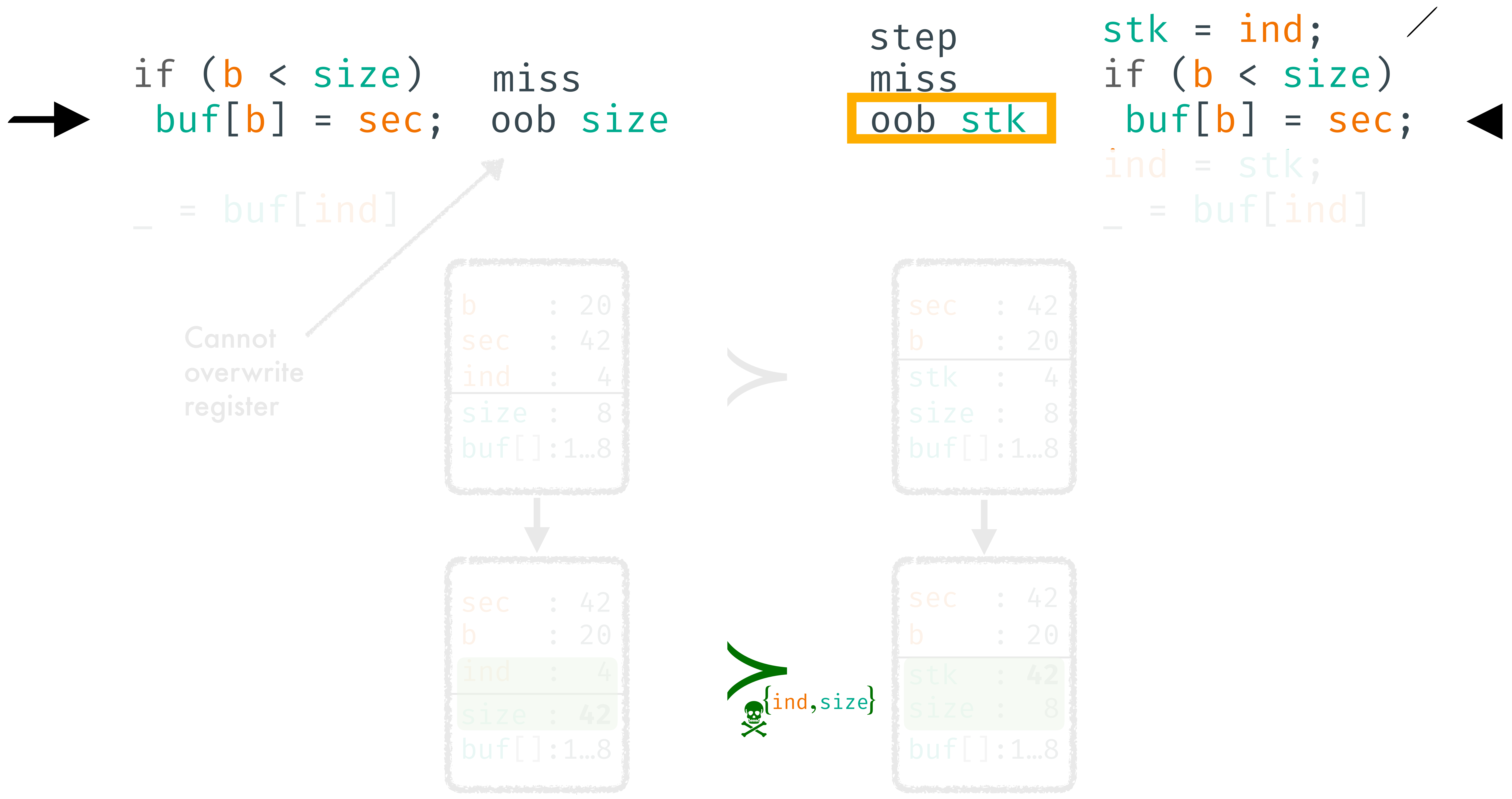
miss
oob size

step
miss
oob stk

```
stk = ind;
if (b < size)
 buf[b] = sec;
ind = stk;
_ = buf[ind]
```



Earlier...



```
if (b < size)     miss
 buf[b] = sec;    oob size
_ = buf[ind]
```

```
step              stk = ind;
miss              if (b < size)
oob stk            buf[b] = sec;
                  ind = stk;
                  _ = buf[ind]
```

Earlier...

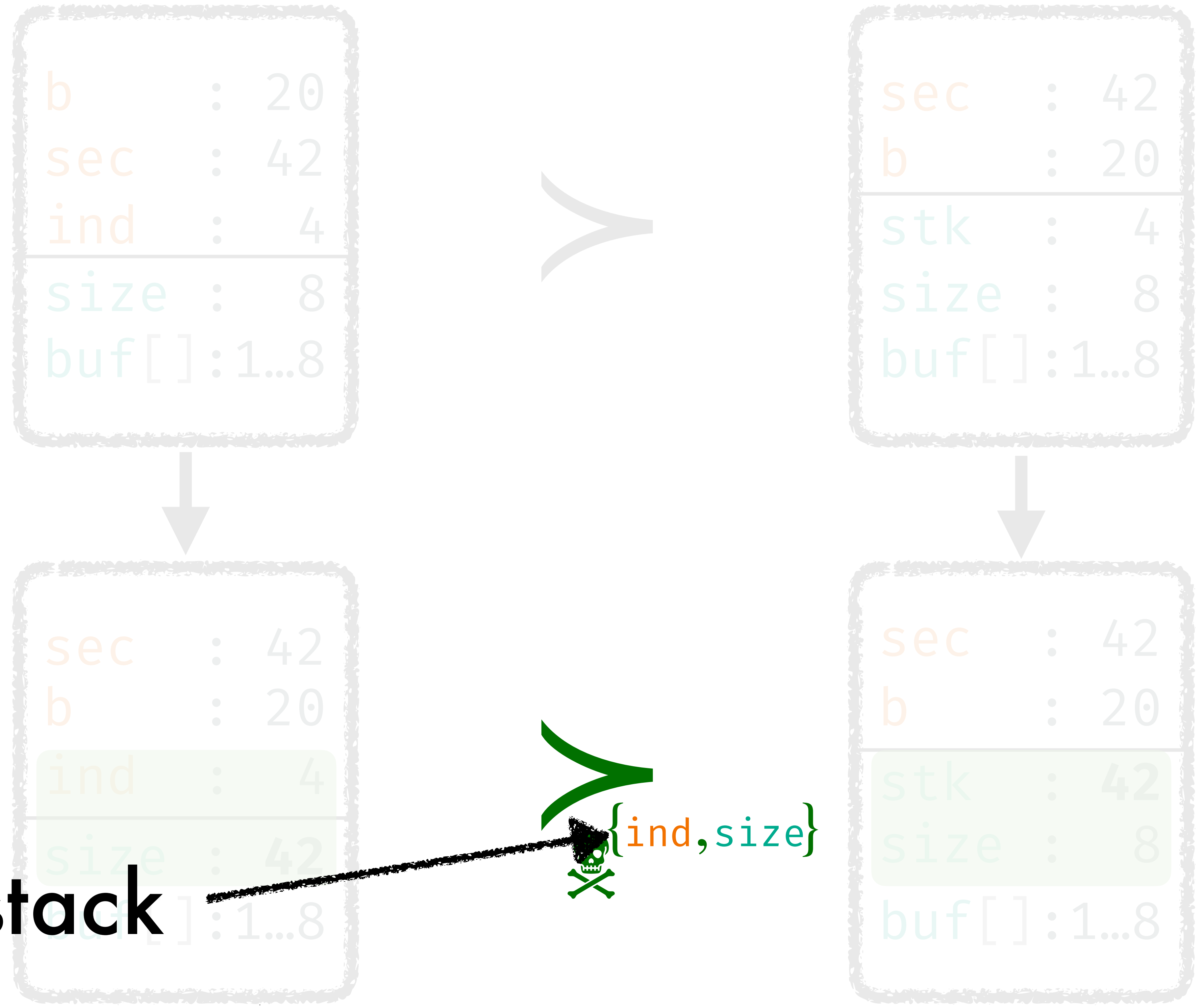
```
if (b < size)
 buf[b] = sec;

_ = buf[ind]
```

miss
oob size

step
miss
$\boxed{\text{oob stk}}$

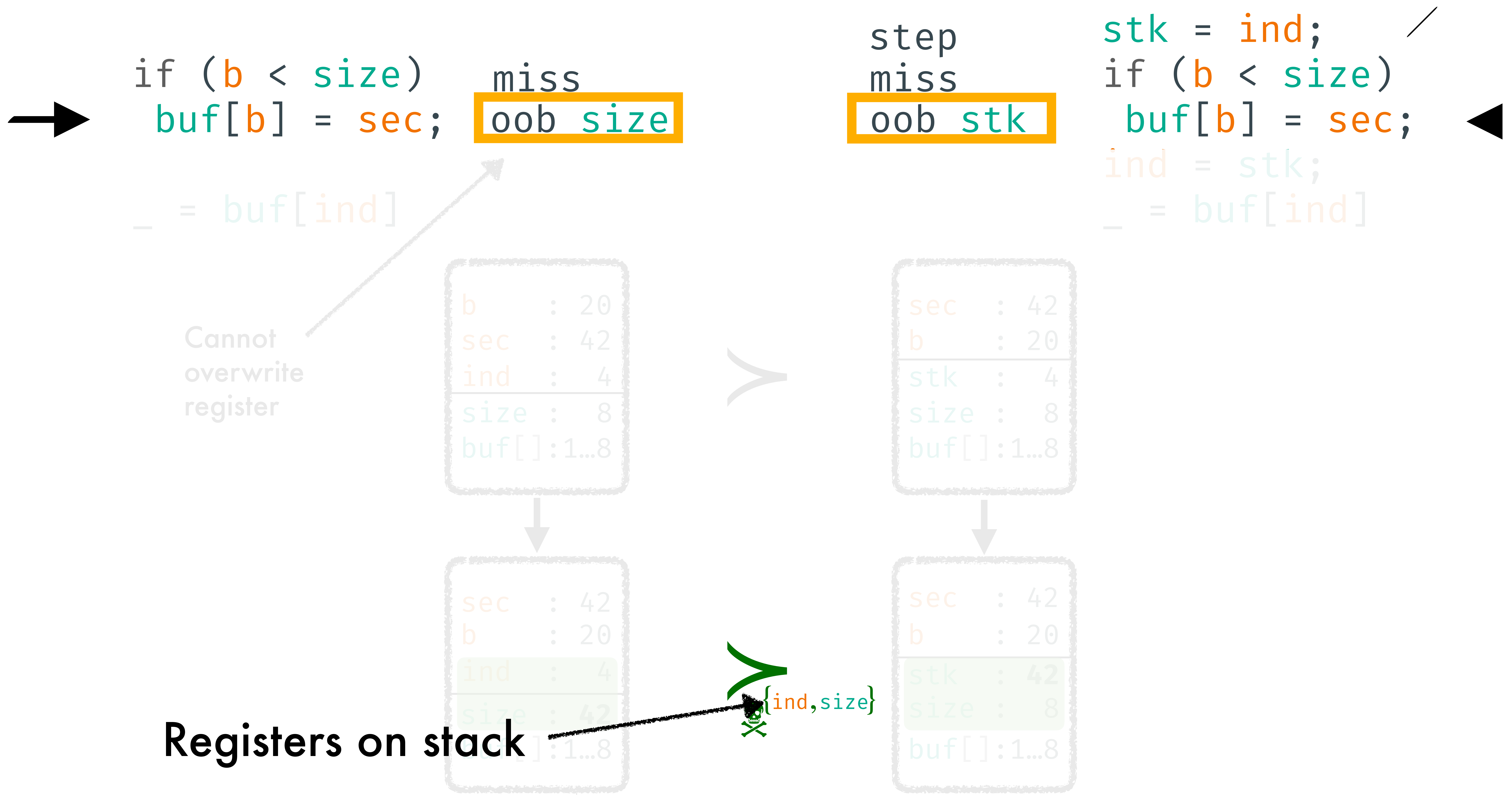
```
stk = ind;
if (b < size)
 buf[b] = sec;
ind = stk;
_ = buf[ind]
```

Cannot overwrite register



Registers on stack

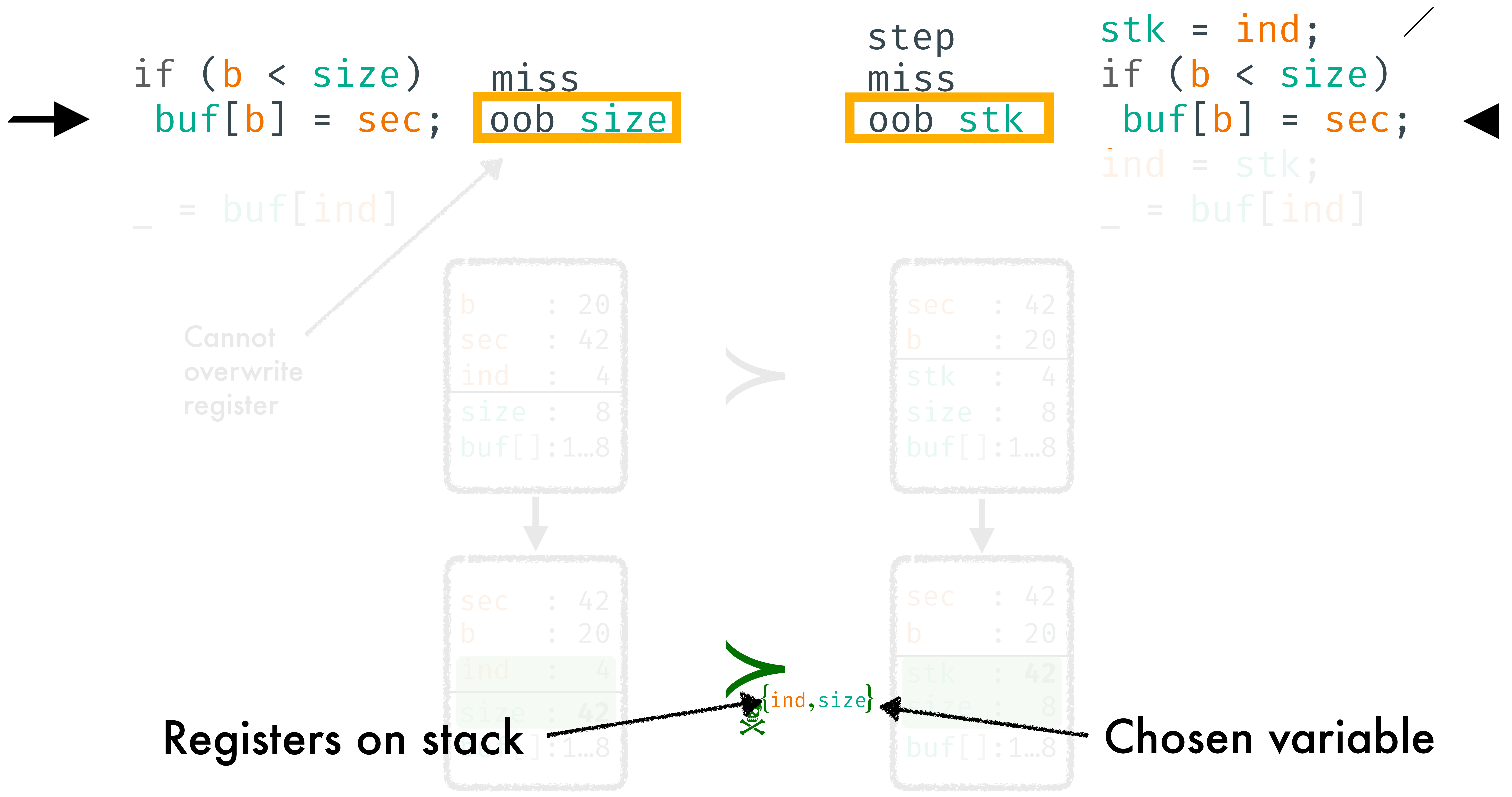
Earlier...



```
if (b < size)
 buf[b] = sec;

_ = buf[ind]
```

```
stk = ind;
if (b < size)
 buf[b] = sec;
ind = stk;
_ = buf[ind]
```

Earlier...



```
if (b < size)
 buf[b] = sec;

_ = buf[ind]
```

```
stk = ind;
if (b < size)
 buf[b] = sec;
ind = stk;
_ = buf[ind]
```

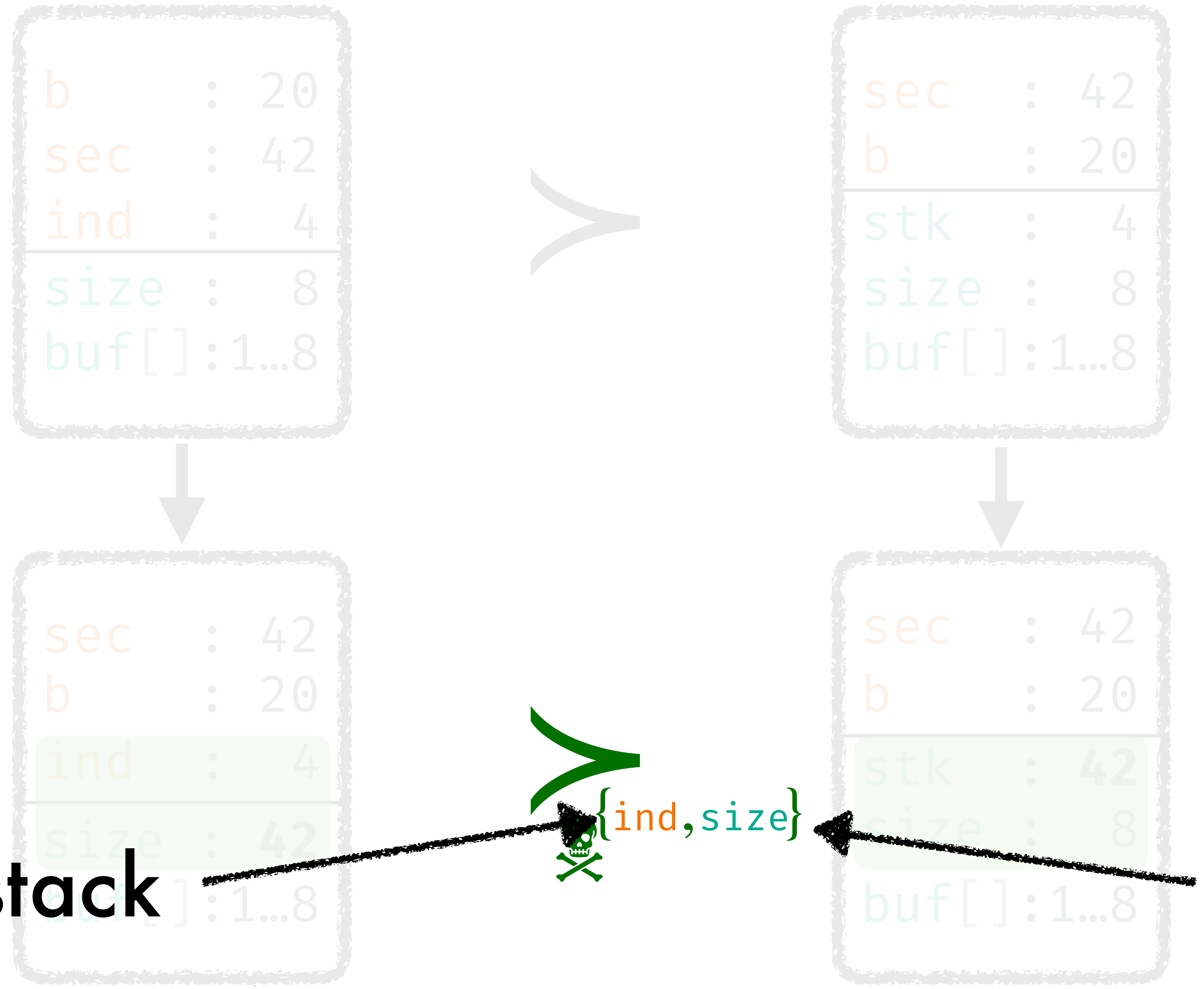
Earlier...

Could choose any!

```
if (b < size)      miss
 buf[b] = sec;     oob size

_ = buf[ind]
```

```
step               stk = ind;
miss               if (b < size)
oob stk             buf[b] = sec;
                   ind = stk;
                   _ = buf[ind]
```

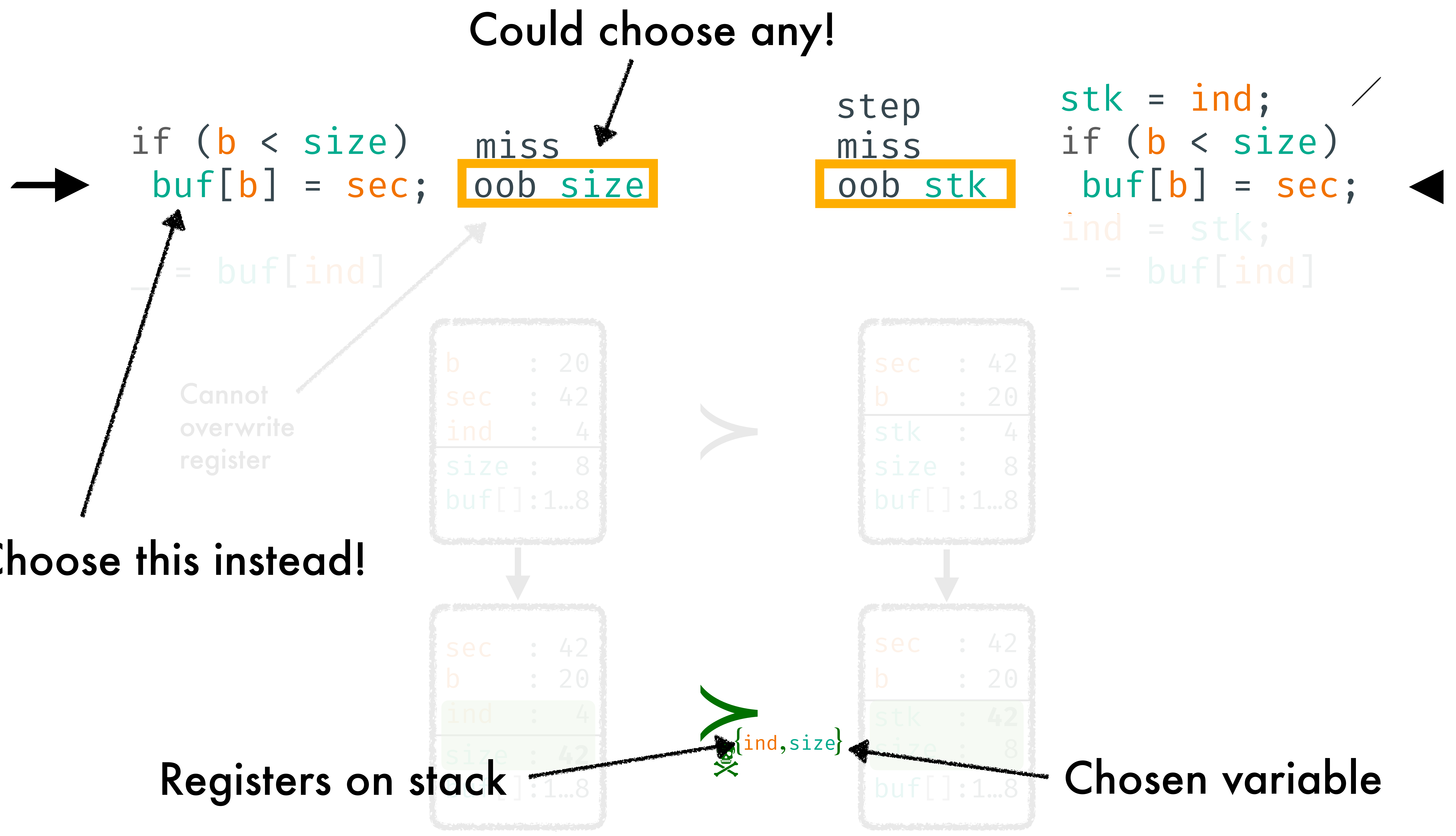
Cannot overwrite register



Registers on stack

Chosen variable

Earlier...



```
if (b < size)      miss
 buf[b] = sec;     oob size
_ = buf[ind]
```

```
step
miss
oob stk
```

```
stk = ind;
if (b < size)
 buf[b] = sec;
ind = stk;
_ = buf[ind]
```

Static Poison Analysis

```
1 if (b < size)
2  buf[b] = sec;

3 _ = buf[ind]
```

```
a stk = ind;
b if (b < size)
c  buf[b] = sec;
d ind = stk;

f _ = buf[ind]
```

1,a	Ø
1,b	Ø
2,c	Ø
3,d	??

Static Poison Analysis

```
1 if (b < size)
2  buf[b] = sec;


3 _ = buf[ind]
```

```
a stk = ind;
b if (b < size)
c  buf[b] = sec;
d ind = stk;

f _ = buf[ind]
```

1,a	∅
1,b	∅
2,c	∅
3,d	{ind, buf}

Static Poison Analysis

```
1 if (b < size)
2  buf[b] = sec;

3 _ = buf[ind]
```

```
a stk = ind;
b if (b < size)
c  buf[b] = sec;
d ind = stk;

f _ = buf[ind]
```

1,a	∅
1,b	∅
2,c	∅
3,d	{ind, buf}
3,f	{ind, buf}

Static Poison Analysis

```
                                  a stk = ind;
1 if (b < size)                   b if (b < size)
2  buf[b] = sec;  • •             c  buf[b] = sec;
                  • •             d ind = stk;

3 _ = buf[ind]                    f _ = buf[ind]
```

1,a	$\emptyset$
1,b	$\boxed{\emptyset}$
2,c	$\emptyset$
3,d	{ind, buf}
3,f	{ind, buf}

Static Poison Analysis

```
1 if (b < size)
2  buf[b] = sec;

3 _ = buf[ind]
```

```
a stk = ind;
b if (b < size)
c  buf[b] = sec;
d ind = stk;

f _ = buf[ind]
```

1,a	Ø
1,b	Ø
2,c	$\boxed{Ø}$
3,d	{ind, buf}
3,f	{ind, buf}

Static Poison Analysis

```
1 if (b < size)
2  buf[b] = sec;

3 _ = buf[ind]
```

```
a stk = ind;
b if (b < size)
c  buf[b] = sec;
d ind = stk;

f _ = buf[ind]
```

1,a	$\emptyset$
1,b	$\emptyset$
2,c	$\emptyset$
3,d	{ind, buf}
3,f	$\boxed{\{ind, buf\}}$

Static Poison Analysis

```
                                 a stk = ind;
  1 if (b < size)                b if (b < size)
  2  buf[b] = sec;      • •      c  buf[b] = sec;
                        • •      d ind = stk;

→ 3 _ = buf[ind]                 f _ = buf[ind]   ◀
```

1,a	∅
1,b	∅
2,c	∅
3,d	{ind, buf}
3,f	{ind, buf}

Static Poison Analysis

```
1 if (b < size)
2  buf[b] = sec;

3 _ = buf[ind]
```

```
a stk = ind;
b if (b < size)
c  buf[b] = sec;
d ind = stk;

f _ = buf[ind]
```

1,a	∅
1,b	∅
2,c	∅
3,d	{ind, buf}
3,f	{ind, buf}

Mitigate!

Static Poison Analysis

```
1 if (b < size)
2  buf[b] = sec;

3 _ = buf[ind]
```

• •
• •

```
a stk = ind;
b if (b < size)
c  buf[b] = sec;
d ind = stk;
e ind = slh(ind);
f _ = buf[ind]
```

1,a	∅
1,b	∅
2,c	∅
3,d	{ind, buf}
3,f	{ind, buf}

Static Poison Analysis

```
1 if (b < size)
2  buf[b] = sec;

3 _ = buf[ind]
```

```
a stk = ind;
b if (b < size)
c  buf[b] = sec;
d ind = stk;
e ind = slh(ind);
f _ = buf[ind]
```

1,a	$\emptyset$
1,b	$\emptyset$
2,c	$\emptyset$
3,d	{ind, buf}
3,e	{ind, buf}
3,f	{ind, buf}

Static Poison Analysis

```
                                  a stk = ind;
  1 if (b < size)                 b if (b < size)
  2  buf[b] = sec;     • •        c  buf[b] = sec;
                       • •        d ind = stk;
                                  e ind = slh(ind);
→ 3 _ = buf[ind]                  f _ = buf[ind]      ◀
```

1,a	∅
1,b	∅
2,c	∅
3,d	{ind, buf}
3,e	{ind, buf}
3,f	{ buf }

Static Poison Analysis

```
                               a stk = ind;
1 if (b < size)                b if (b < size)
2  buf[b] = sec;        • •    c  buf[b] = sec;
                        • •    d ind = stk;
                               e ind = slh(ind);
→ 3 _ = buf[ind]               f _ = buf[ind]      ◀
```

1,a	∅
1,b	∅
2,c	∅
3,d	{ind, buf}
3,e	{ind, buf}
3,f	{ buf }

Static Poison Analysis

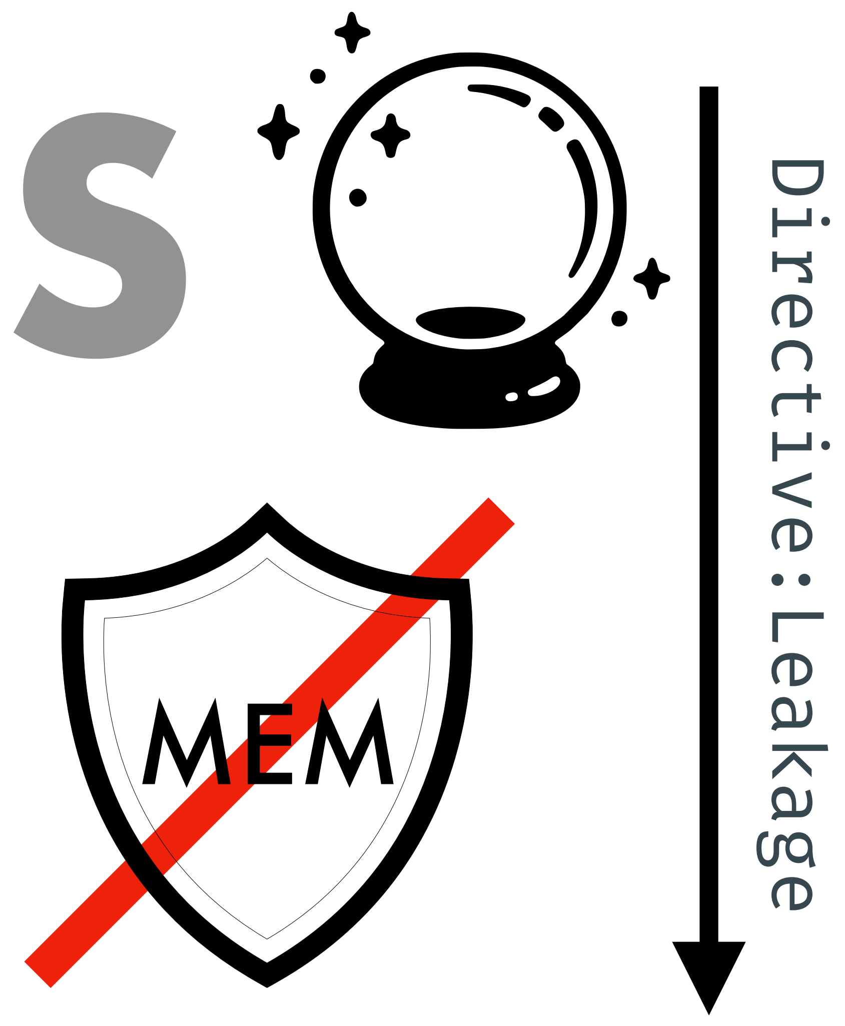
$$[P]_{ra} \vDash \mathbf{SNI}$$

```
   1 if (b < size)         a stk = ind;
   2  buf[b] = sec;  : :   b if (b < size)
                           c  buf[b] = sec;
                           d ind = stk;
                           e ind = slh(ind);
→  3 _ = buf[ind]          f _ = buf[ind]     ◀
```

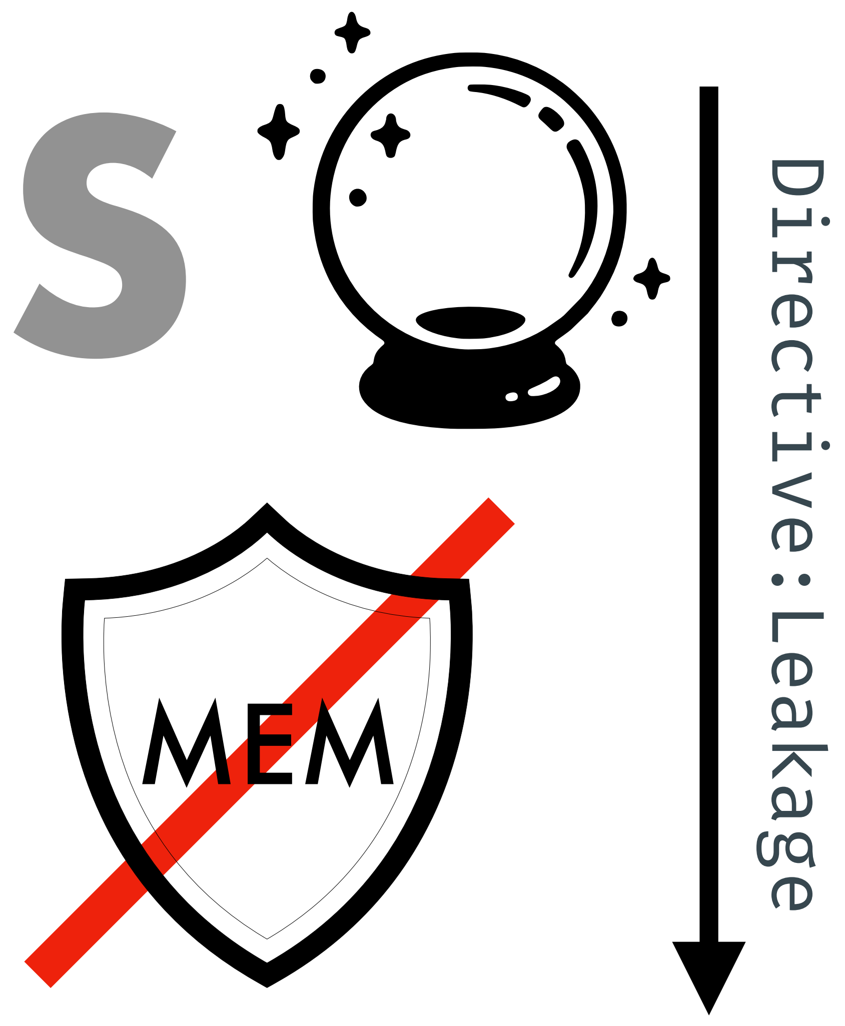
1,a	$\emptyset$
1,b	$\emptyset$
2,c	$\emptyset$
3,d	{ind, buf}
3,e	{ind, buf}
3,f	{ buf }

SNIP: Speculative Execution and Non-Interference Preservation for Compiler Transformations

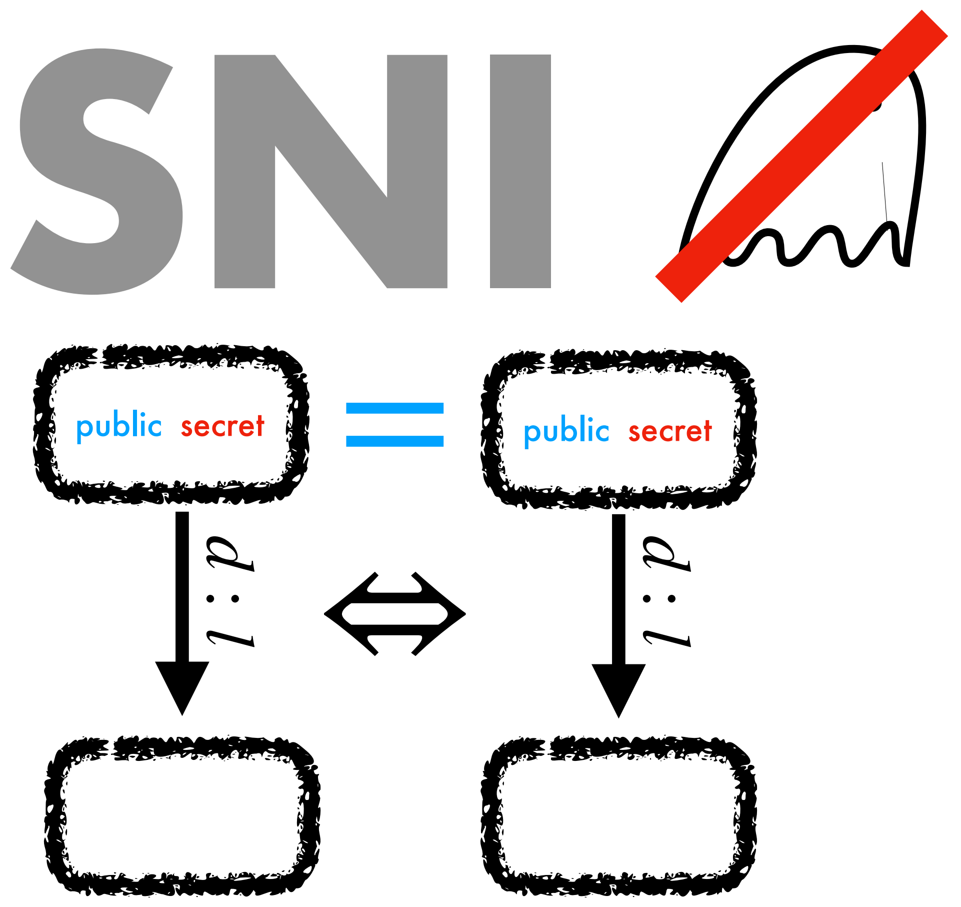
s.van-der-wall@tu-bs.de


S
MEM
Directive:Leakage

S



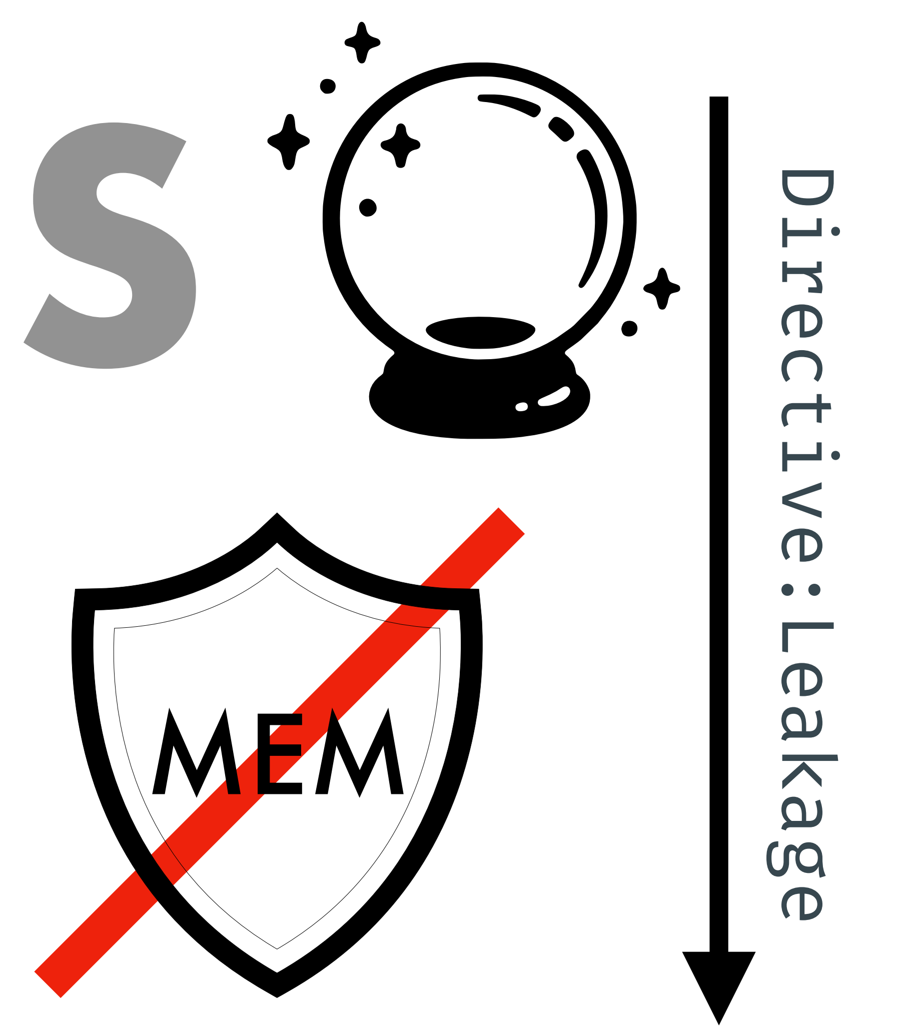
SNI

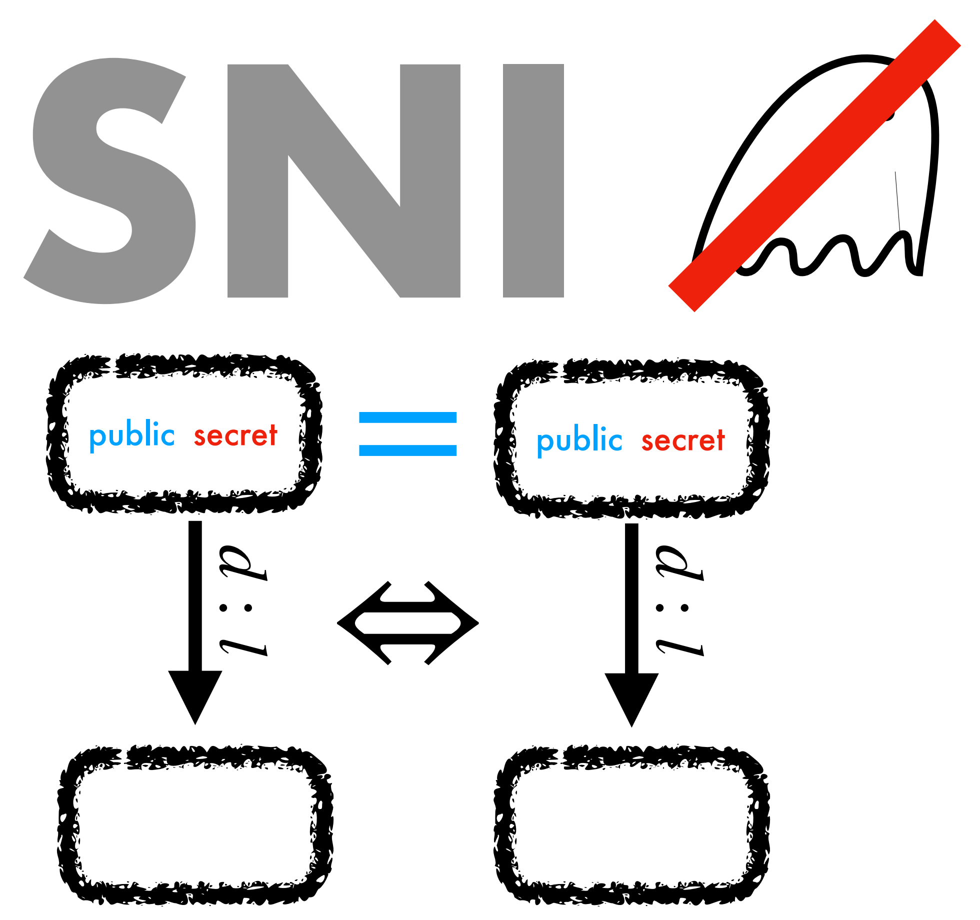


SNIP: Speculative Execution and Non-Interference Preservation for Compiler Transformations

s.van-der-wall@tu-bs.de

SNIP $P \models$ SNI $\implies [P] \models$ SNI





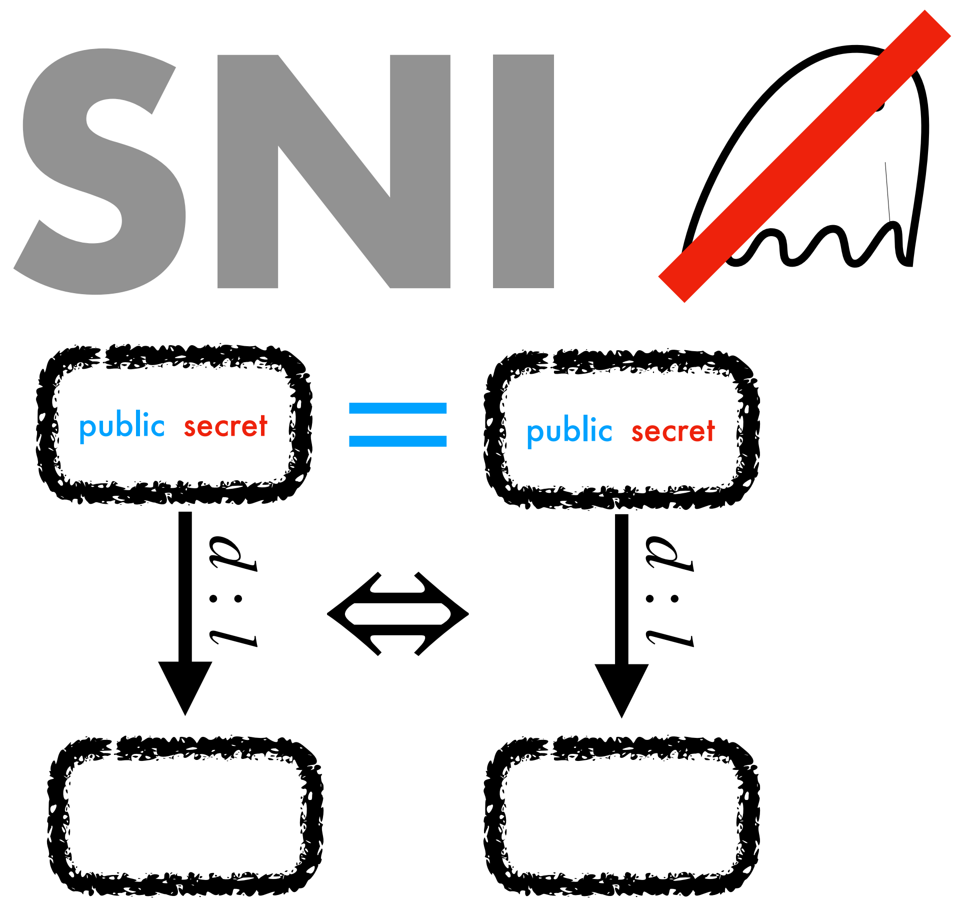
SNIP: Speculative Execution and Non-Interference Preservation for Compiler Transformations

s.van-der-wall@tu-bs.de

S
MEM
Directive:Leakage

SNIP $P \models$ SNI $\implies [P] \models$ SNI

LLVM $\not\models$ SNIP
(Register Allocation)

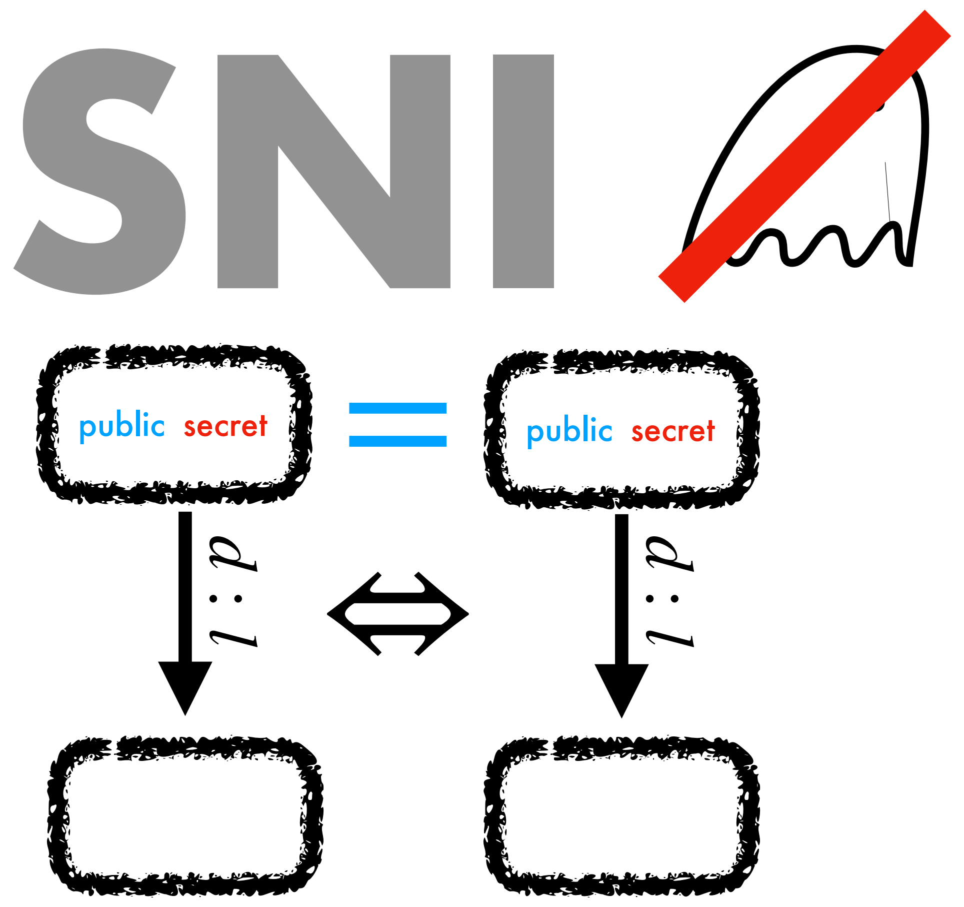


SNIP: Speculative Execution and Non-Interference Preservation for Compiler Transformations
s.van-der-wall@tu-bs.de

SNIP $P \models \mathrm{SNI} \implies [P] \models \mathrm{SNI}$

$\mathit{LLVM} \not\models \mathrm{SNIP}$

(Register Allocation)



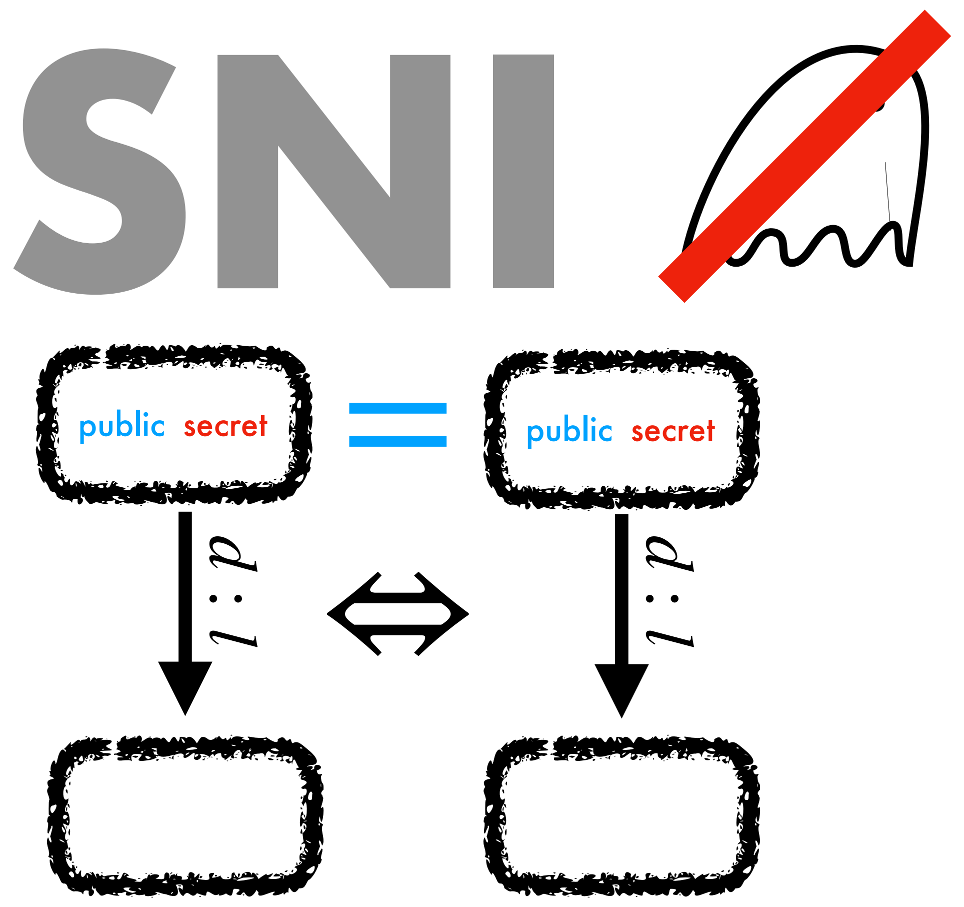
SNIP: Speculative Execution and Non-Interference Preservation for Compiler Transformations

s.van-der-wall@tu-bs.de

SNIP $P \models \text{SNI} \implies [P] \models \text{SNI}$

$\text{LLVM} \not\models \text{SNIP}$

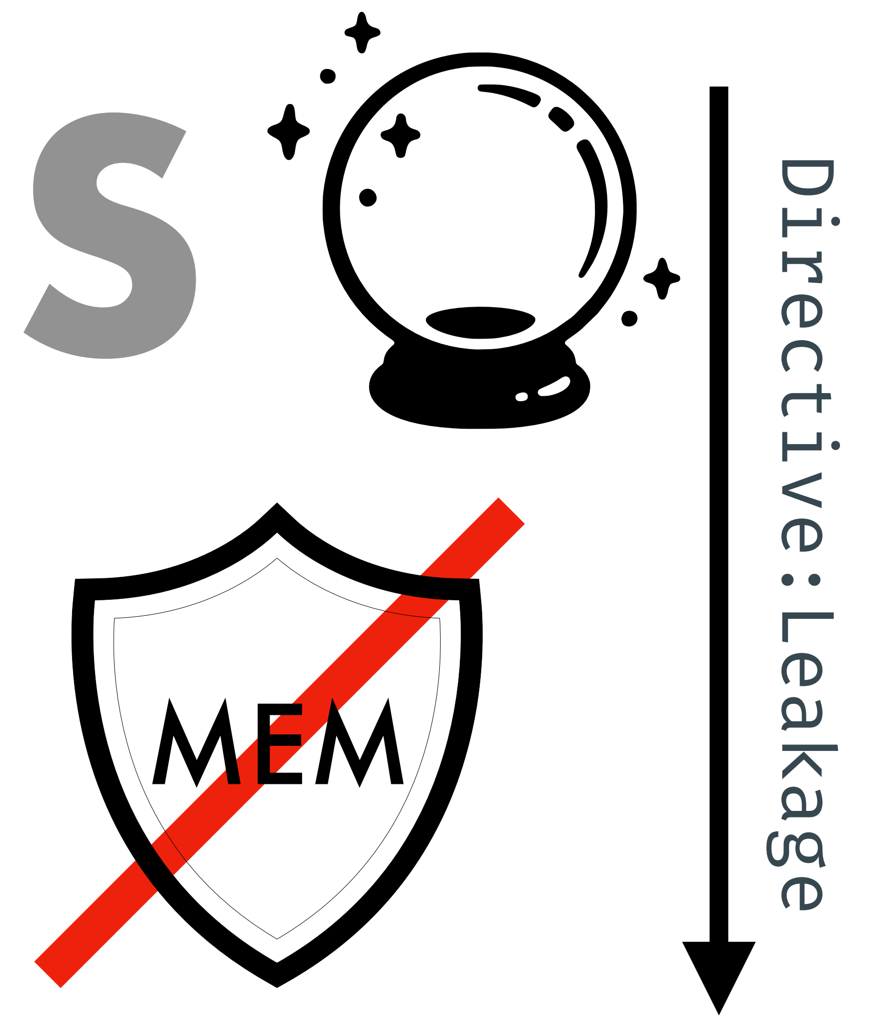
(Register Allocation)



LD 42 Sec-dep Leaks
are poisoned!

SNIP: Speculative Execution and Non-Interference Preservation for Compiler Transformations

s.van-der-wall@tu-bs.de



SNIP $P \vDash \text{SNI} \implies [P] \vDash \text{SNI}$

$\mathit{LLVM} \nvDash$ SNIP

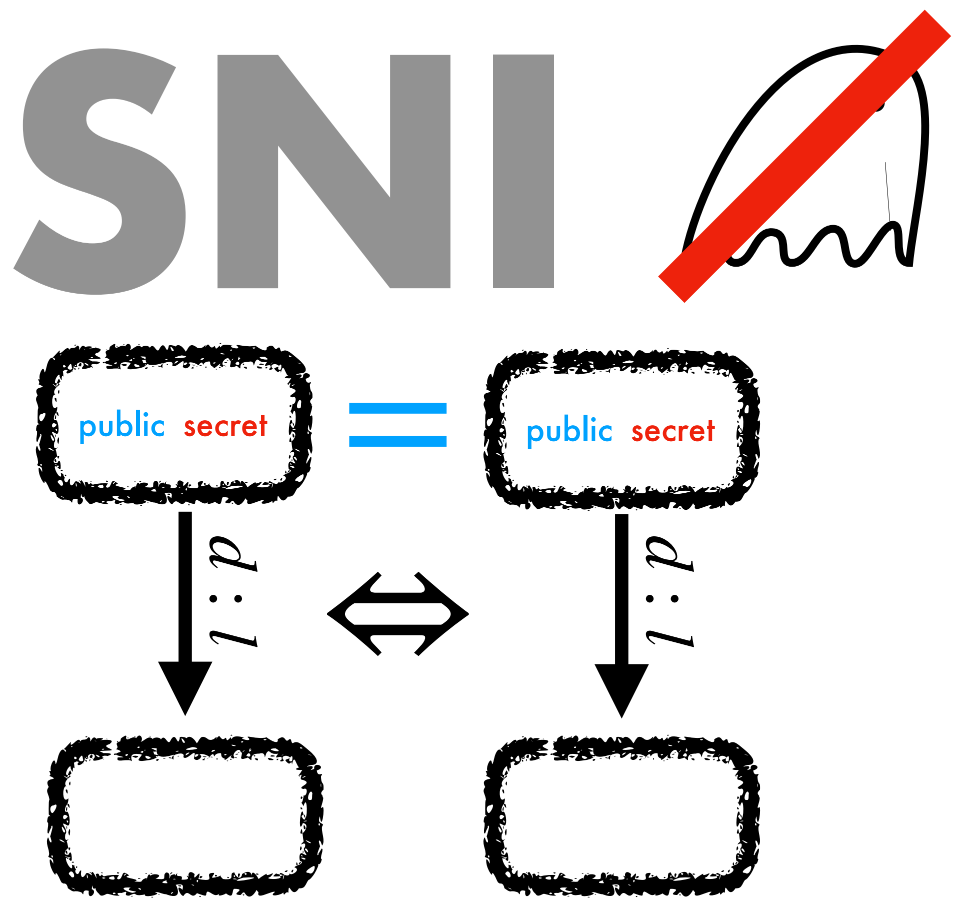
(Register Allocation)

$\succ_{\{\texttt{ind},\texttt{size}\}}$

$\boxed{\text{Static Poison Analysis}}$

$P :: [P]$

`LD 42` Sec-dep Leaks
are poisoned!



SNIP: Speculative Execution and Non-Interference Preservation for Compiler Transformations

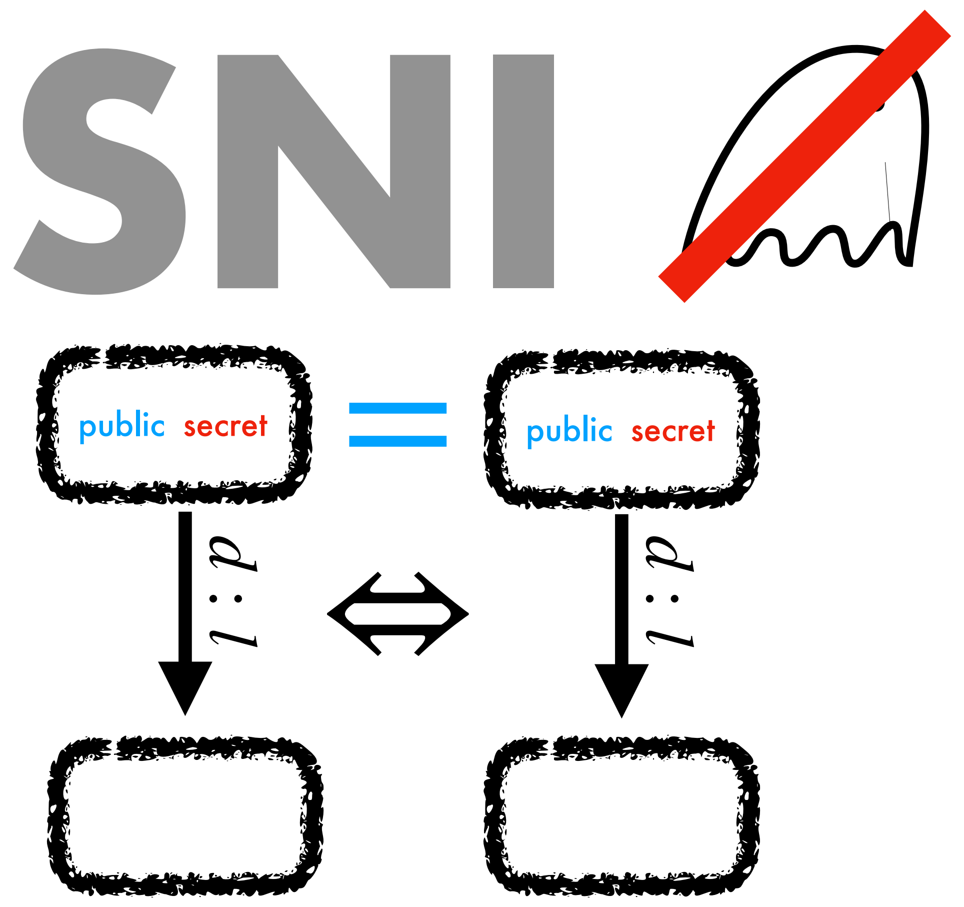
s.van-der-wall@tu-bs.de

SNIP $P \vDash \text{SNI} \implies [P] \vDash \text{SNI}$

$\text{LLVM} \nvDash \text{SNIP}$
(Register Allocation)

Static Poison Analysis

$P :: [P]$ ☠



LD 42 Sec-dep Leaks are poisoned!

$[\,\cdot\,]_{ra} \vDash \text{SNIP}$

SNIP: Speculative Execution and Non-Interference Preservation for Compiler Transformations
s.van-der-wall@tu-bs.de