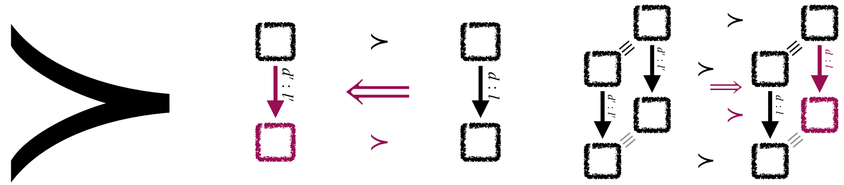
SNIP: Speculative Execution and Non-Interference Preservation for Compiler Transformations

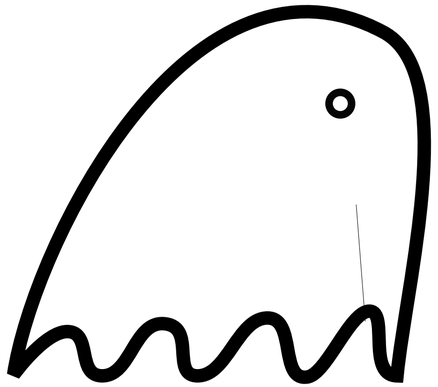
Your binaries are haunted by your compiler!



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

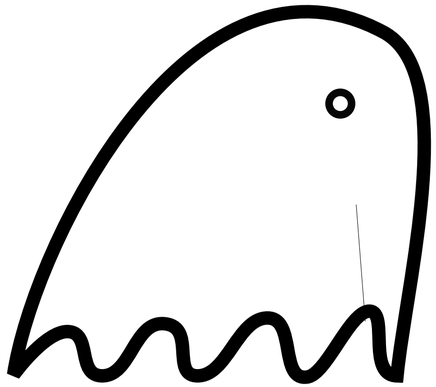
Sören van der Wall, Roland Meyer, POPL 2025, Denver

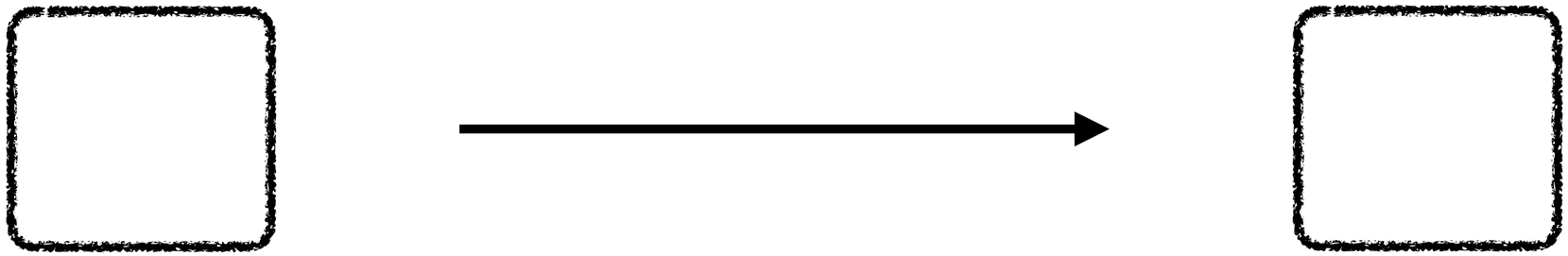
Speculative Execution



Speculative Execution

Architecture



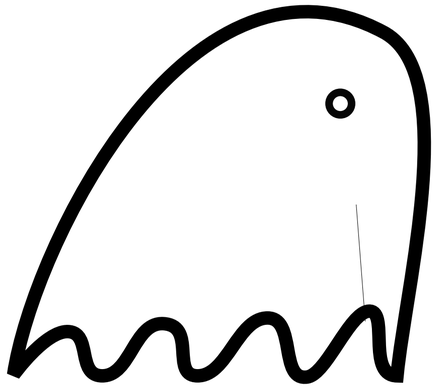


Speculative Execution

SpectreV1

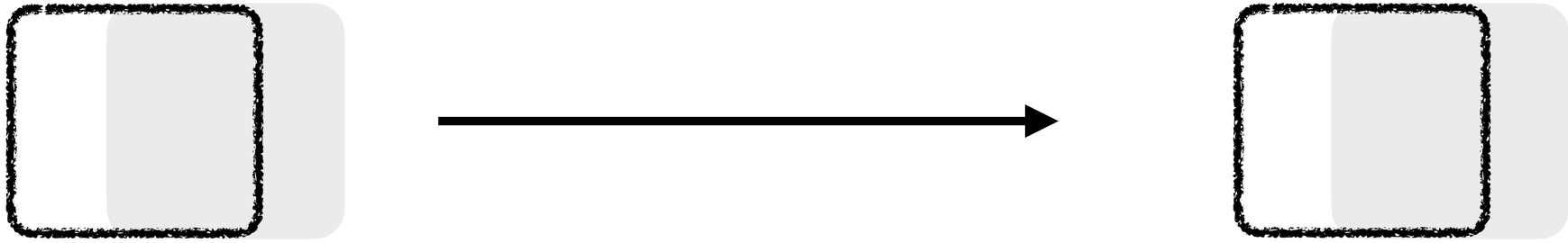
Architecture

```
→ Prog Counter
  Registers
  Stack vars
```



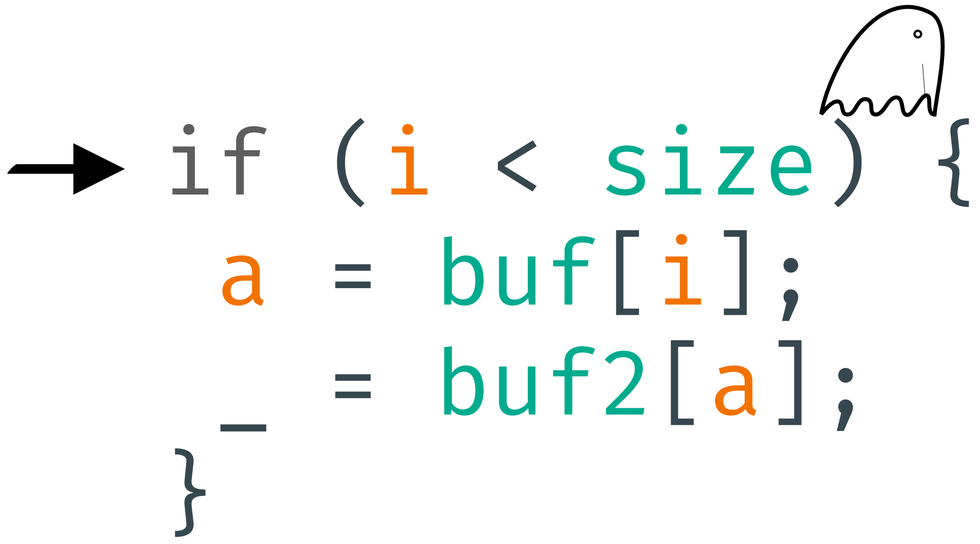
μ-Architecture

```
Caches
Reorder Buffer
Branch predictor
```

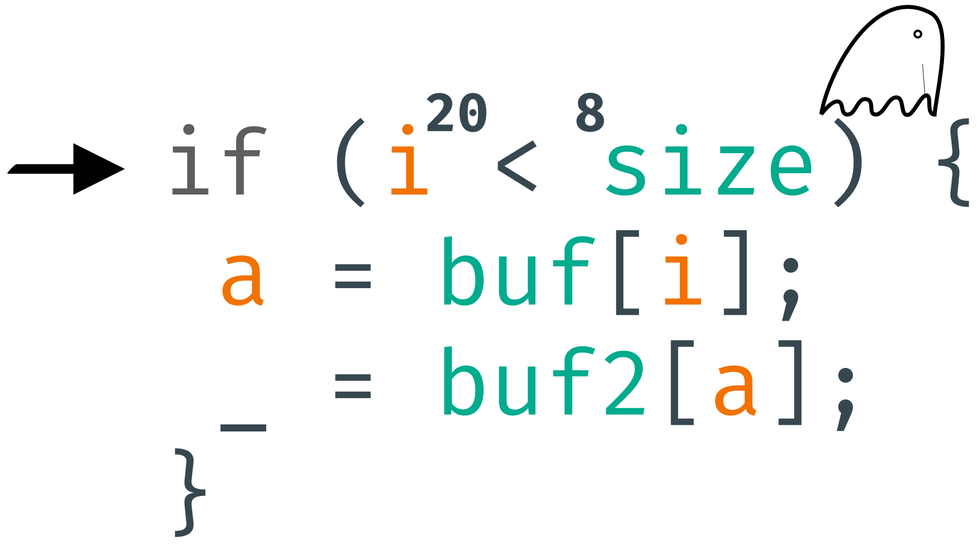


```
if (i < size) {
  a = buf[i];
  _ = buf2[a];
}
```

Registers
Stack Variables

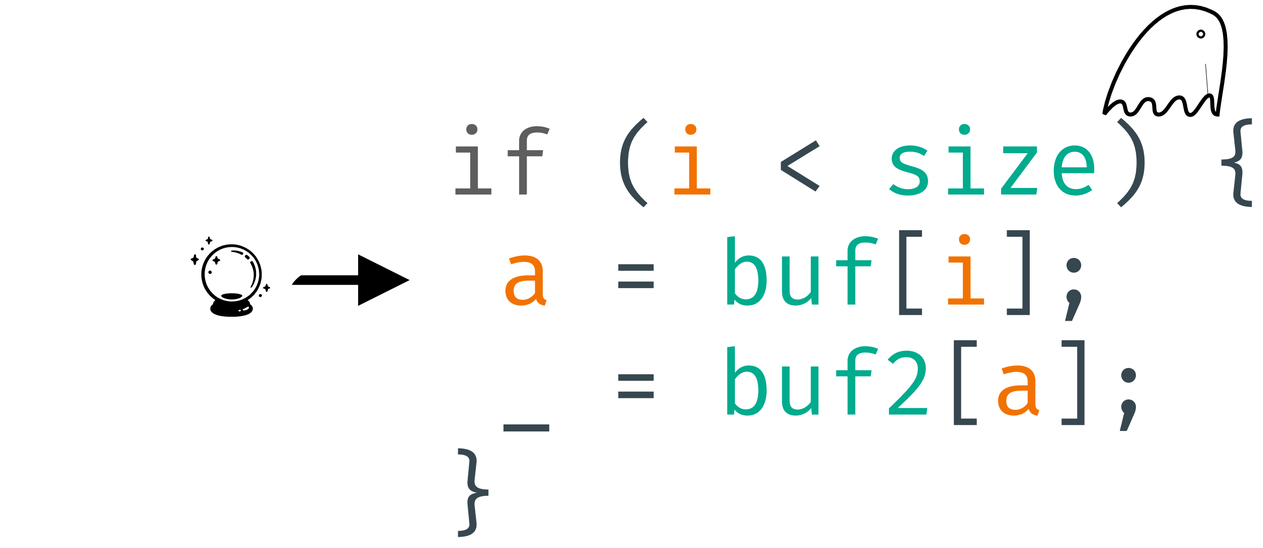


Registers
Stack Variables



i : 20	size: 8
	sec : 42

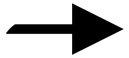
Registers
Stack Variables



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

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

&buf + 20 = &sec

i : 20	size: 8
	sec : 42

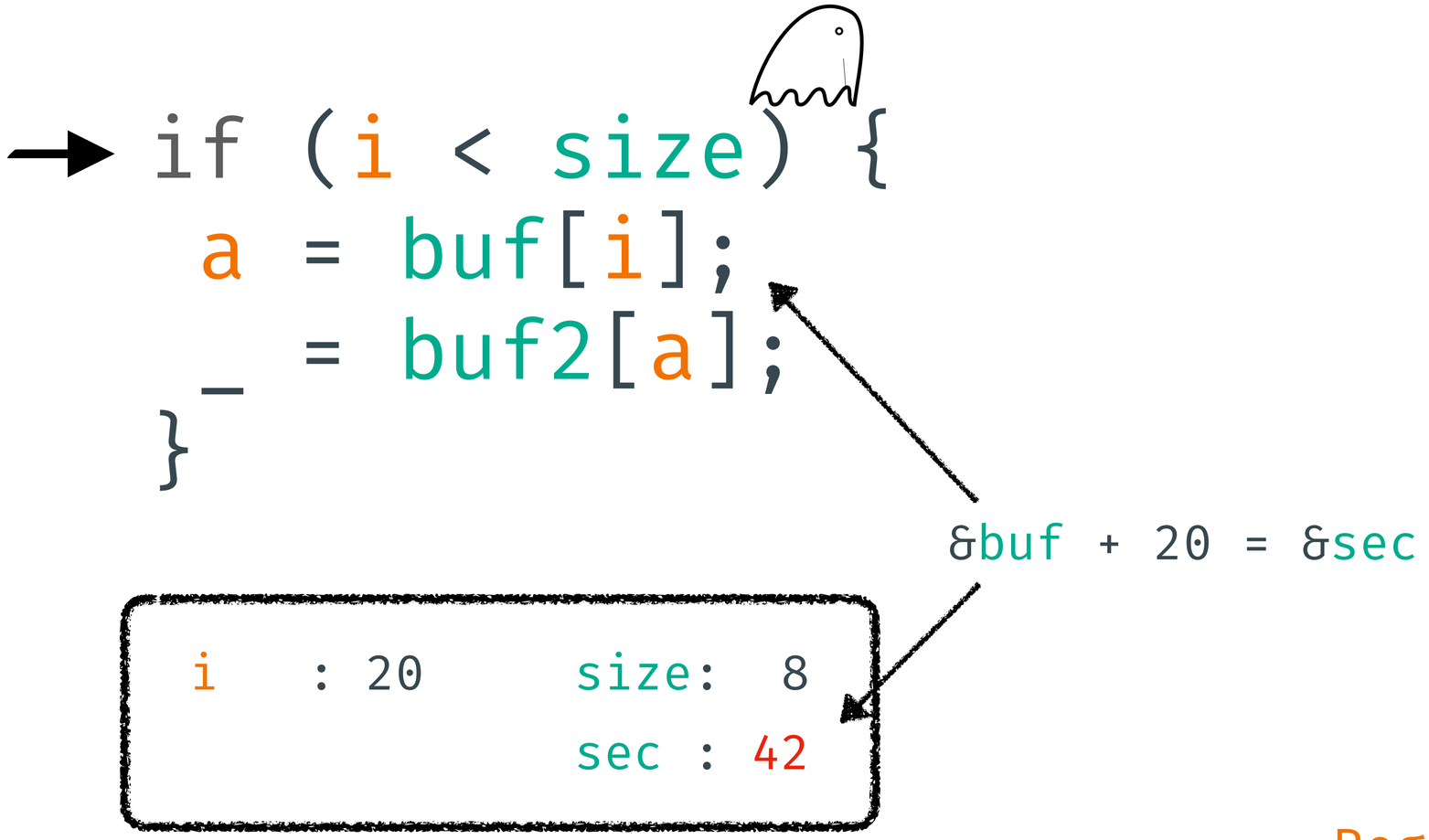
Registers
Stack Variables

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

&buf + 20 = &sec

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

Registers
Stack Variables



Registers
Stack Variables

Side-Channel Leakage

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

&buf + 20 = &sec

i : 20	size: 8
	sec : 42

Registers
Stack Variables


```
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

```
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

```
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];
}
```

Side-Channel Leakage

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

&buf + 20 = &sec

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

Registers
Stack Variables

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

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

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

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
```

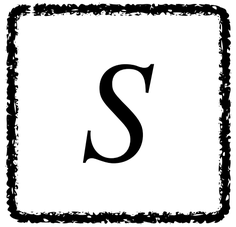
```
i   : 20      size:  8
              sec : 42
```

Registers
Stack Variables

Speculative Execution

Directive:Leakage →

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



Speculative Execution

Directive:Leakage

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

Directive

Speculative Execution

Directive:Leakage →

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

Speculative Execution

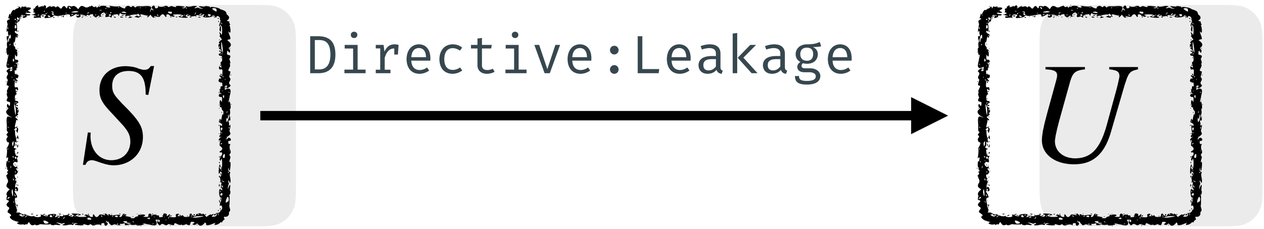
Directive:Leakage →

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

Speculative Execution

Directive:Leakage →

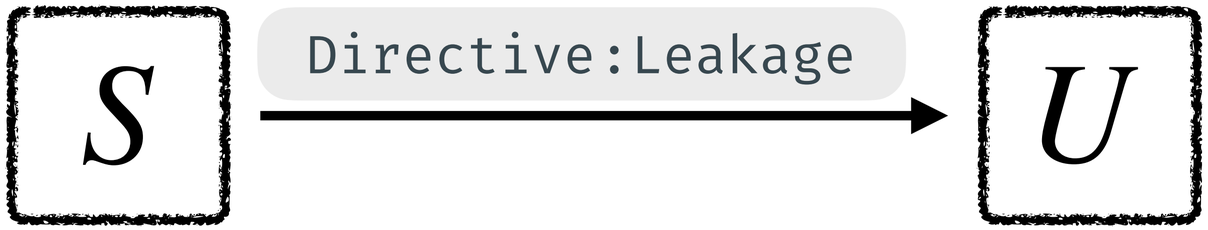
Directive		Leakage
miss	`if (i < size)`	BR false
oob sec	`a = buf[i];`	LD 20
step	`_ = buf2[a];`	LD 42



Speculative Execution

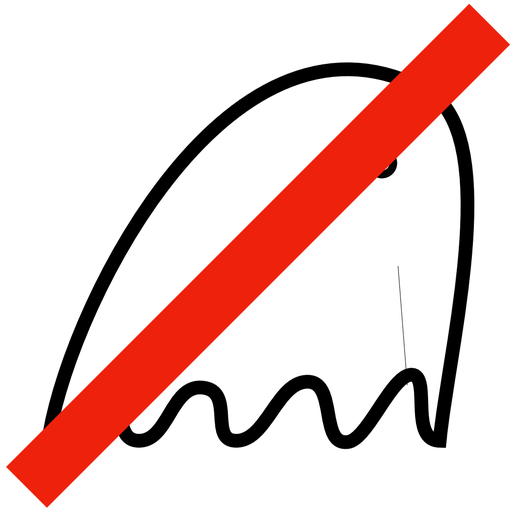
Directive:Leakage →

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

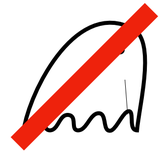


S Non-Interference

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

$P \models$ SNI

IF

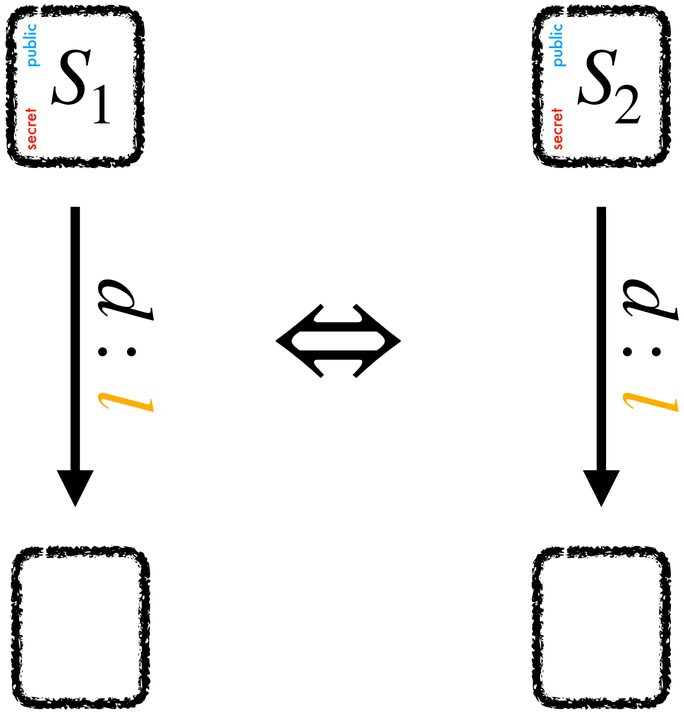
$$\forall \; \boxed{\overset{\text{public}}{\underset{\text{secret}}{S_1}}} = \boxed{\overset{\text{public}}{\underset{\text{secret}}{S_2}}} \;:\; d \in \texttt{Dir}^*$$

S Non-Interference

$P \models \mathbf{SNI}$

IF

$$\forall \; \boxed{\overset{\text{public}}{\underset{\text{secret}}{S_1}}} = \boxed{\overset{\text{public}}{\underset{\text{secret}}{S_2}}} \;\vdots \quad d \in \texttt{Dir}^*$$



SNI Preservation

SNIP

[.] : Compiler Pass

SNI Preservation

SNIP

$P \models$ SNI

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

SNI Preservation

SNIP

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

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

SNI Preservation

SNIP

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

IF $\forall P \quad P \models$ SNI $\implies [P] \models$ SNI

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

$P \vDash \mathbf{SNI}$ 

$\Longrightarrow [P] \vDash \mathbf{SNI}$ 

$P \vDash$ **SNI**

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

$P \models \mathbf{SNI}$

$\overset{?}{\Longrightarrow}$

$[P]_{\mathsf{ra}} \models \mathbf{SNI}$

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

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

 _ = buf[ind]
```

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

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

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

  _ = buf[ind]
```

$P \models$ **SNI**

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

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

 _ = buf[ind]
```

$\overset{?}{\Longrightarrow}$

$[P]_{\text{ra}} \models$ **SNI**

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

$P \models \mathbf{SNI}$

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

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

 _ = buf[ind]
```

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

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

$P \models \mathbf{SNI}$

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

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

 _ = buf[ind]
```

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

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

$P \models \mathbf{SNI}$

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

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

 _ = buf[ind]
```

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

```
                fn( public ind, secret sec42, public …)
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
```

$P \models \mathbf{SNI}$

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

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

 _ = buf[ind]
```

$[P]_{\text{ra}} \models \mathbf{SNI}$

```
                fn( public ind, secret sec42, public …)
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                 stk :  0
```

$P \models \mathbf{SNI}$

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

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

 _  = buf[ind]
```

$[P]_{\mathrm{ra}} \models \mathbf{SNI}$

```
                fn( public ind, secret sec, public …)
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                 stk : 42
```

$P \models \mathbf{SNI}$

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

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

 _  = buf[ind]
```

$[P]_{\mathrm{ra}} \models \mathbf{SNI}$

```
                fn( public ind, secret sec42, 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
```

$P \models \mathbf{SNI}$

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

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

 _  = buf[ind]
```

$\overset{?}{\Rightarrow}$ $[P]_{\mathsf{ra}} \models \mathbf{SNI}$

```
          fn( public ind, secret sec42, public …)
step      stk = ind;
miss      if (b < size)
oob stk    buf[b] =42sec;
step      ind =42stk;
step      _  = buf2[ind42]   LD 42
```

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

LLVM ≠ SNIP

LLVM ⊭ SNIP

Each of LLVM's 4 allocators!

LLVM ⊭ 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

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

Can we fix Register Allocation so that

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

SNI Preservation

DeadCode

SNI Preservation

DeadCode

$$[\,\cdot\,]_{dc} \vDash \textbf{SNIP}$$

~~SNI~~ Preservation

Safety

DeadCode

$$[\,\cdot\,]_{dc} \vDash \textbf{SNIP}$$

~~SNI~~ Preservation

Safety

DeadCode

~~SNI~~ Preservation

Safety

DeadCode

P

```
fn( public i, size, secret sec)
  if (i < size)
    a = buf[i];
    a = 0;
  ret;
```

~~SNI~~ Preservation

Safety

DeadCode

P

```
fn( public i, size, secret sec)
 if (i < size)
  a = buf[i];
  a = 0;
 ret;
```

$[P]_{\mathsf{dc}}$

```
fn( public i, size, secret sec)
 if (i < size)
  nop;
  a = 0;
 ret;
```

~~SNI~~ Preservation
Safety

DeadCode

P

```
fn( public i, size, secret sec)
 if (i < size)
  a = buf[i];
  a = 0;
 ret;
```

$[P]_{\text{dc}}$

```
fn( public i, size, secret sec)
 if (i < size)
  nop;
  a = 0;
 ret;
```

$\succ$

$\succ \subseteq States_P \times States_{[P]_{\text{dc}}}$

~~SNI~~ Preservation

Safety

DeadCode

P

```
fn( public i, size, secret sec)
 if (i < size)
  a = buf[i];
  a = 0;
 ret;
```

$[P]_{\text{dc}}$

```
fn( public i, size, secret sec)
 if (i < size)
  nop;
  a = 0;
 ret;
```

Equal except on dead locations

$\succ \subseteq States_P \times States_{[P]_{\text{dc}}}$

~~SNI~~ Preservation
Safety

DeadCode

P

```
fn( public i, size, secret sec)
 if (i < size)
  a = buf[i];
  a = 0;
 ret;
```

$[P]_{\text{dc}}$

```
fn( public i, size, secret sec)
 if (i < size)
  nop;
  a = 0;
 ret;
```

Equal except on dead locations

$\succ \subseteq States_P \times States_{[P]_{\text{dc}}}$

~~SNI~~ Preservation
Safety

DeadCode

P

```
fn( public i, size, secret sec)
 if (i < size)
  a = buf[i];
  a = 0;
 ret;
```

$[P]_{\text{dc}}$

```
fn( public i, size, secret sec)
 if (i < size)
  nop;
  a = 0;
 ret;
```

Equal except on dead locations

```
i :  4 size : 8
a : 15 sec : 42
```

```
i :  4 size : 8
a : 11 sec : 42
```

```
i :  4 size : 8
a :  0 sec : 42
```

$\succ \subseteq States_P \times States_{[P]_{\text{dc}}}$

~~SNI~~ Preservation
Safety

DeadCode

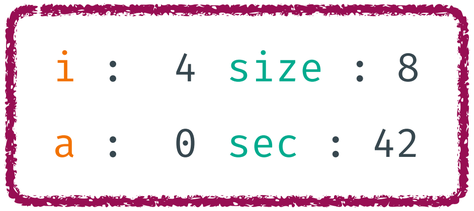
P

```
fn( public i, size, secret sec)
 if (i < size)
  a = buf[i];
  a = 0;
ret;
```

$[P]_{\text{dc}}$

```
fn( public i, size, secret sec)
 if (i < size)
  nop;
  a = 0;
ret;
```

Equal except on dead locations



$\succ \subseteq States_P \times States_{[P]_{\text{dc}}}$

~~SNI~~ Preservation
Safety

DeadCode

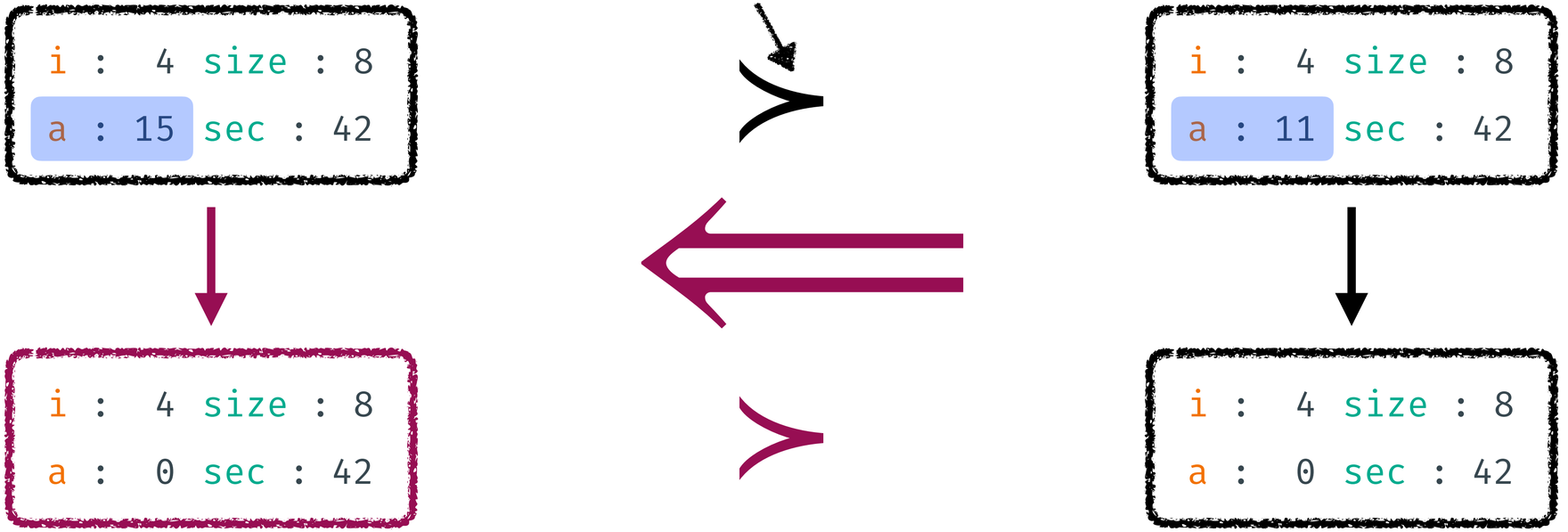
P

```
fn( public i, size, secret sec)
 if (i < size)
  a = buf[i];
  a = 0;
ret;
```

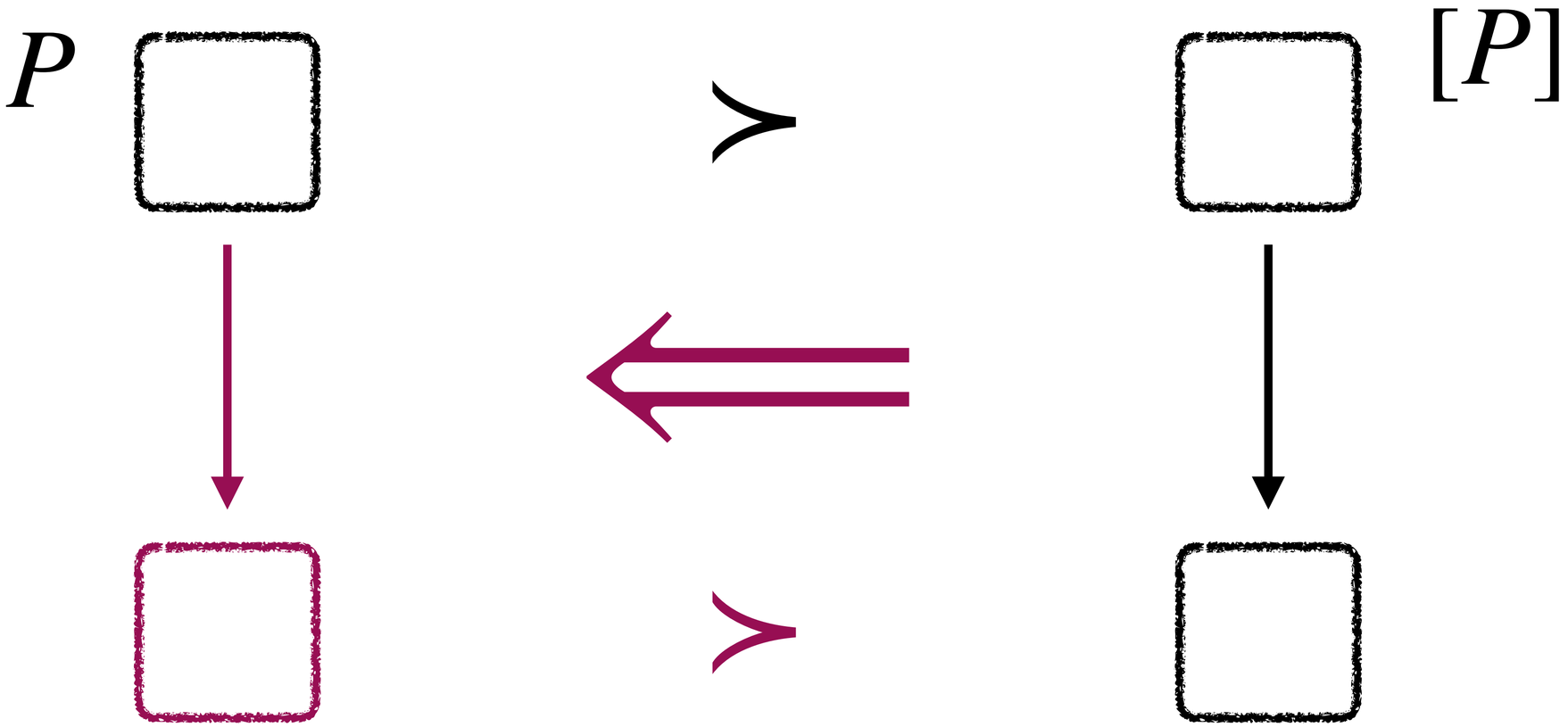
$[P]_{\text{dc}}$

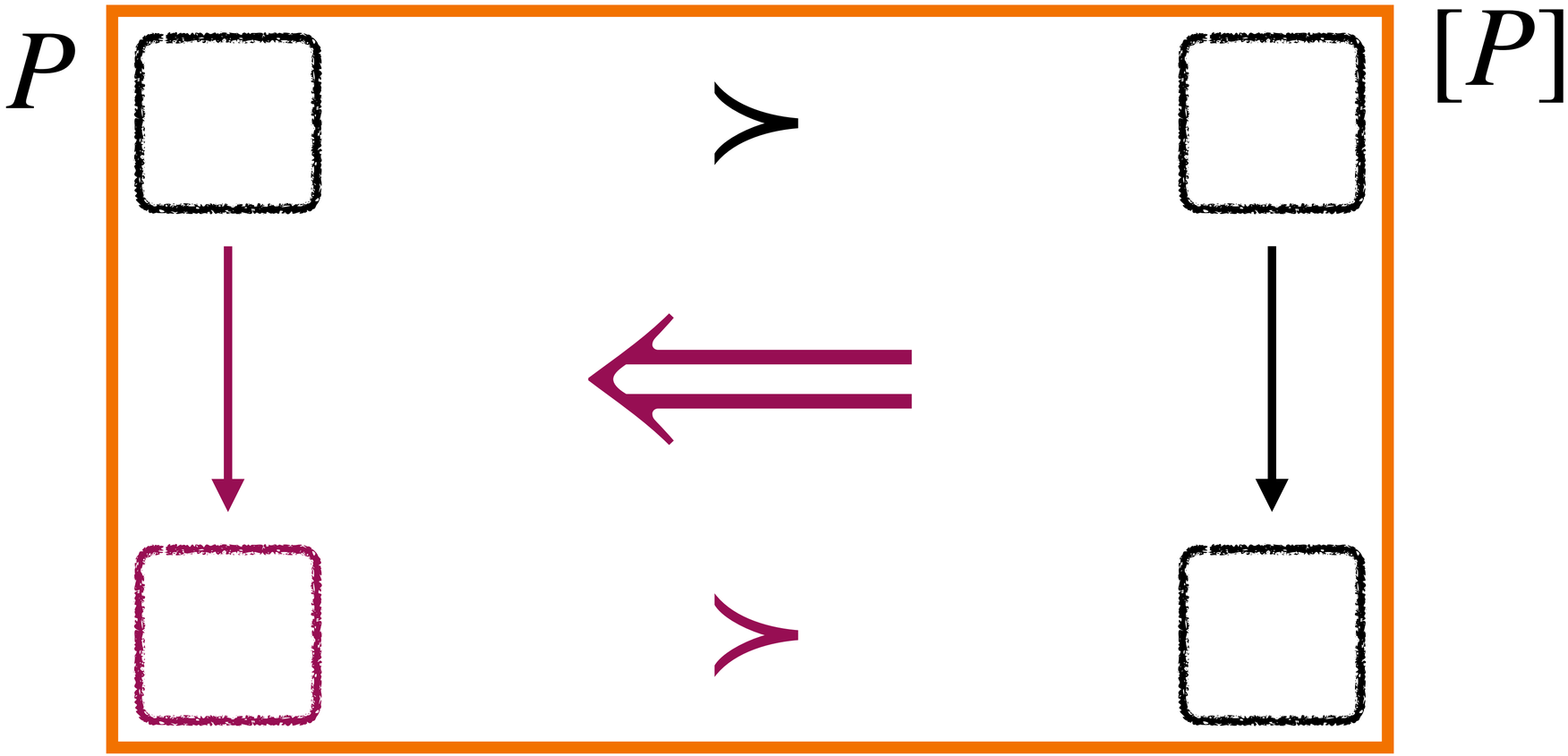
```
fn( public i, size, secret sec)
 if (i < size)
  nop;
  a = 0;
ret;
```

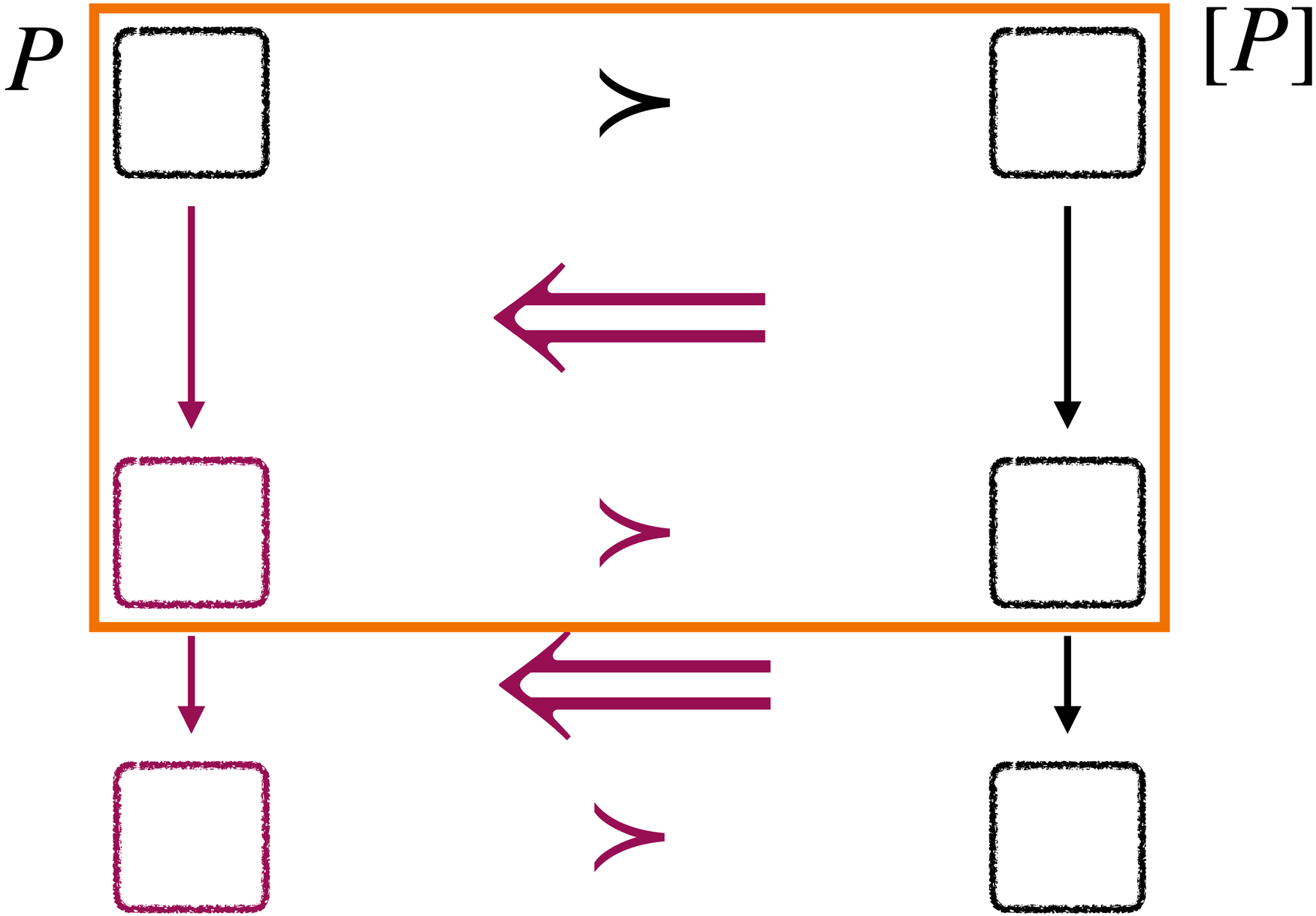
Equal except on dead locations

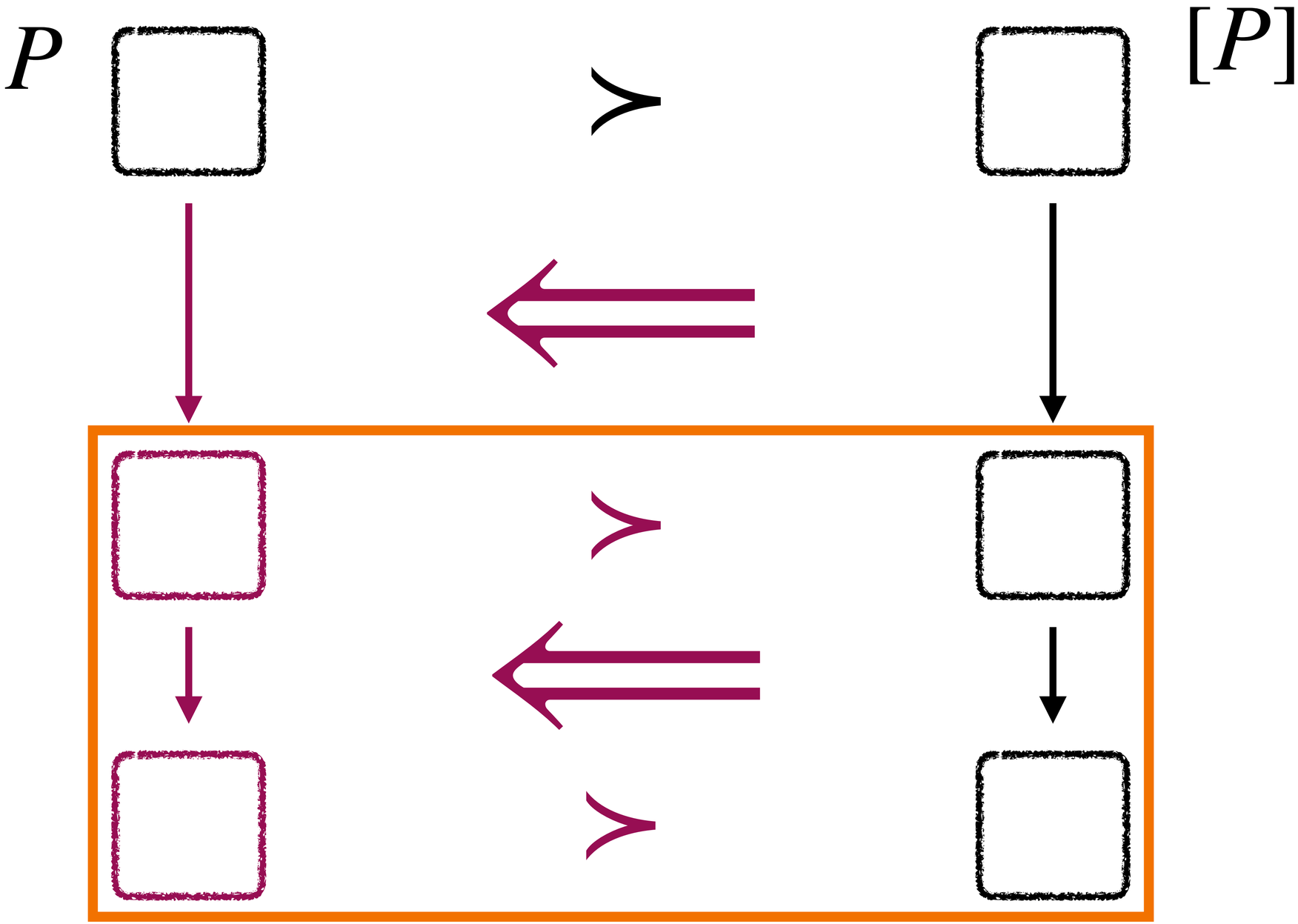


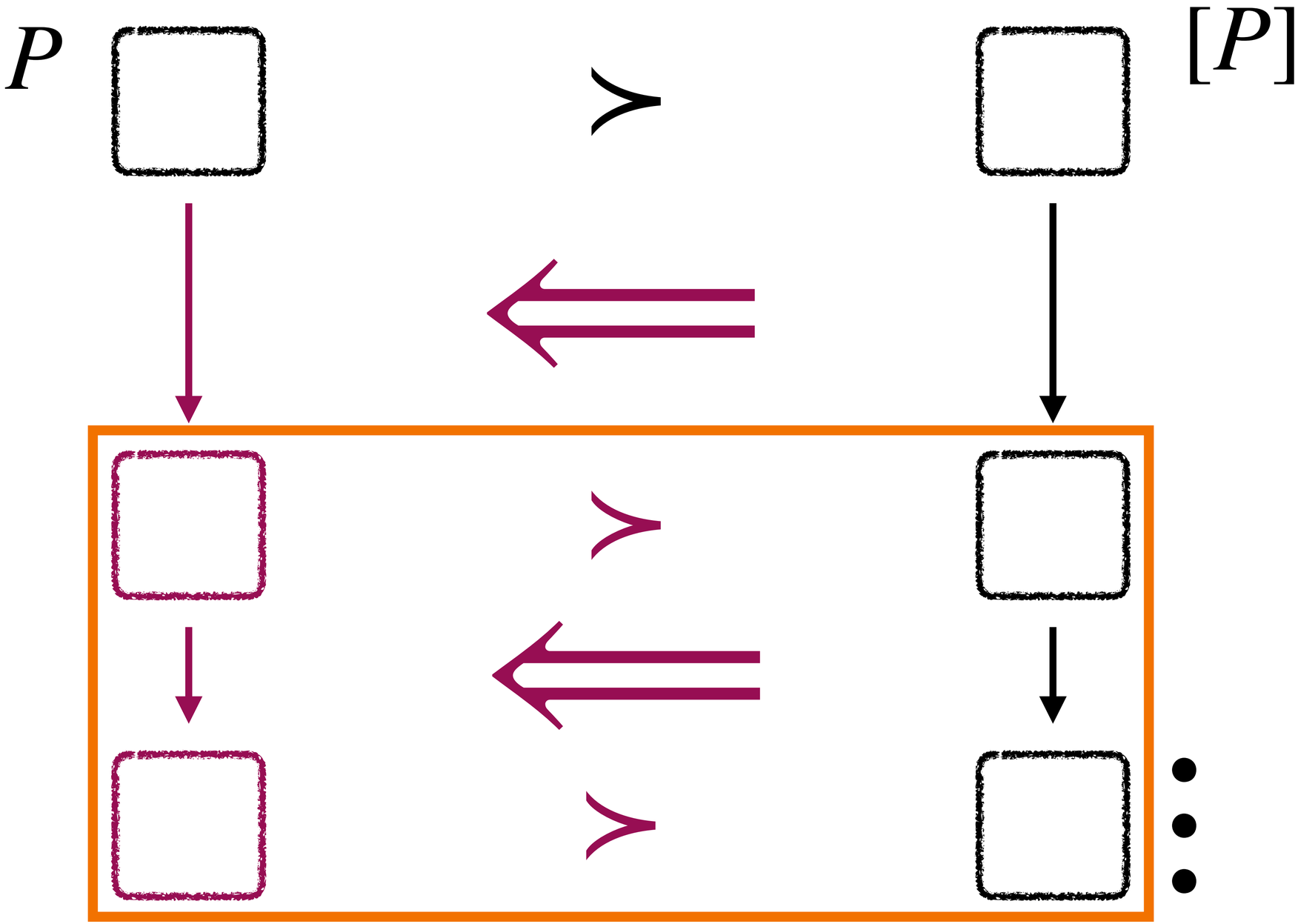
$\succ \subseteq States_P \times States_{[P]_{\text{dc}}}$


P
[P]
≻
≻


P
≻
[P]
≻


P
[P]
≻
≻
≻


P
≻
[P]
≻
≻


P
≻
[P]
≻
≻

~~SNI~~ Preservation

Safety

Correctness

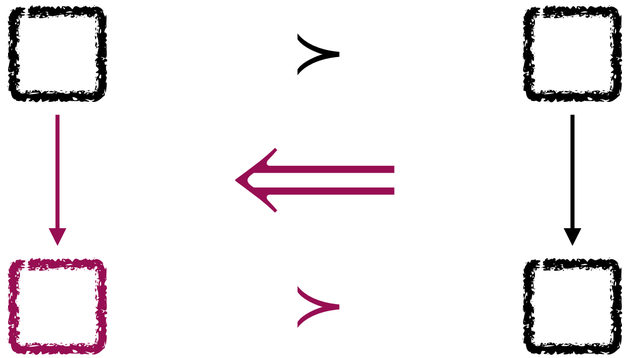
Theorem *If* $[\,.\,]$ *has a simulation* $\succ$ *between any* P *and* $[P]$*, so that*

~~SNI~~ Preservation

Safety

Correctness

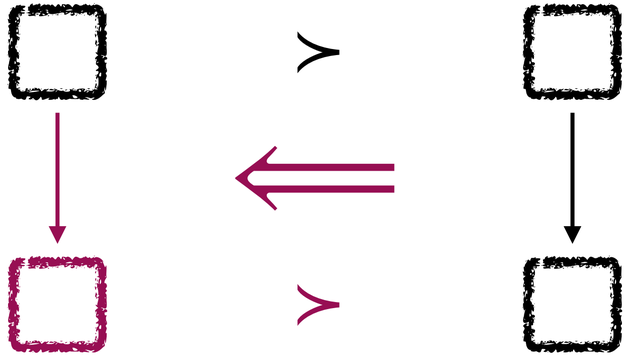
Theorem *If* $[\,.\,]$ *has a simulation* $\succ$ *between any* P *and* $[P]$*, so that*



~~SNI~~ Preservation

Safety

Theorem *If* $[\,.\,]$ *has a simulation* $\succ$ *between any* P *and* $[P]$*, so that*



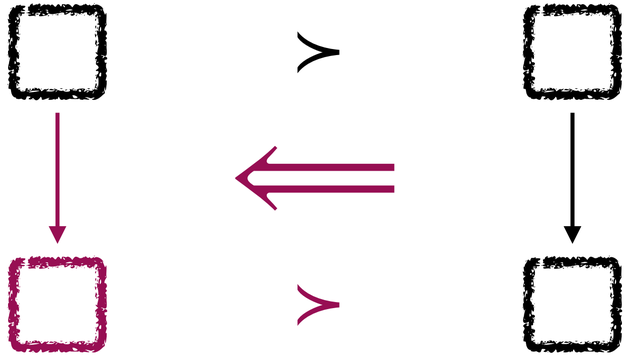
then for any safety property A*:* $P \models \mathsf{A} \implies [P] \models \mathsf{A}$

~~SNI~~ Preservation

Safety

Correctness

Theorem *If* $[\,.\,]$ *has a simulation* $\succ$ *between any* P *and* $[P]$*, so that*



then for any safety property A*:* $P \models \mathsf{A} \implies [P] \models \mathsf{A}$

Usually, $\succ$ is parametric in P

~~SNI~~ Preservation

Correctness

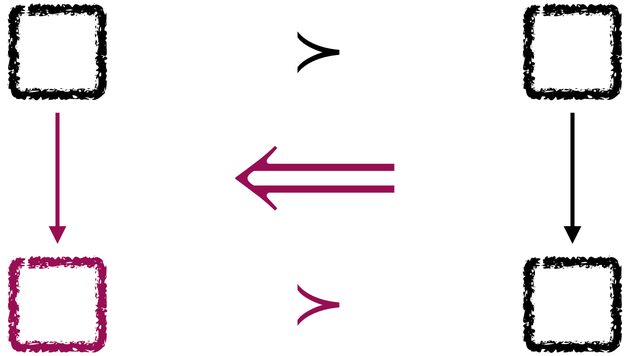
Safety

Theorem *If* $[\,.\,]$ *has a simulation* $\succ$ *between any* P *and* $[P]$*, so that*

$P \models \mathsf{A}$

Not needed

to construct $\succ$



then for any safety property A*:* $P \models \mathsf{A} \implies [P] \models \mathsf{A}$

Usually, $\succ$ is parametric in P

~~SNI~~ Preservation

Safety

DeadCode

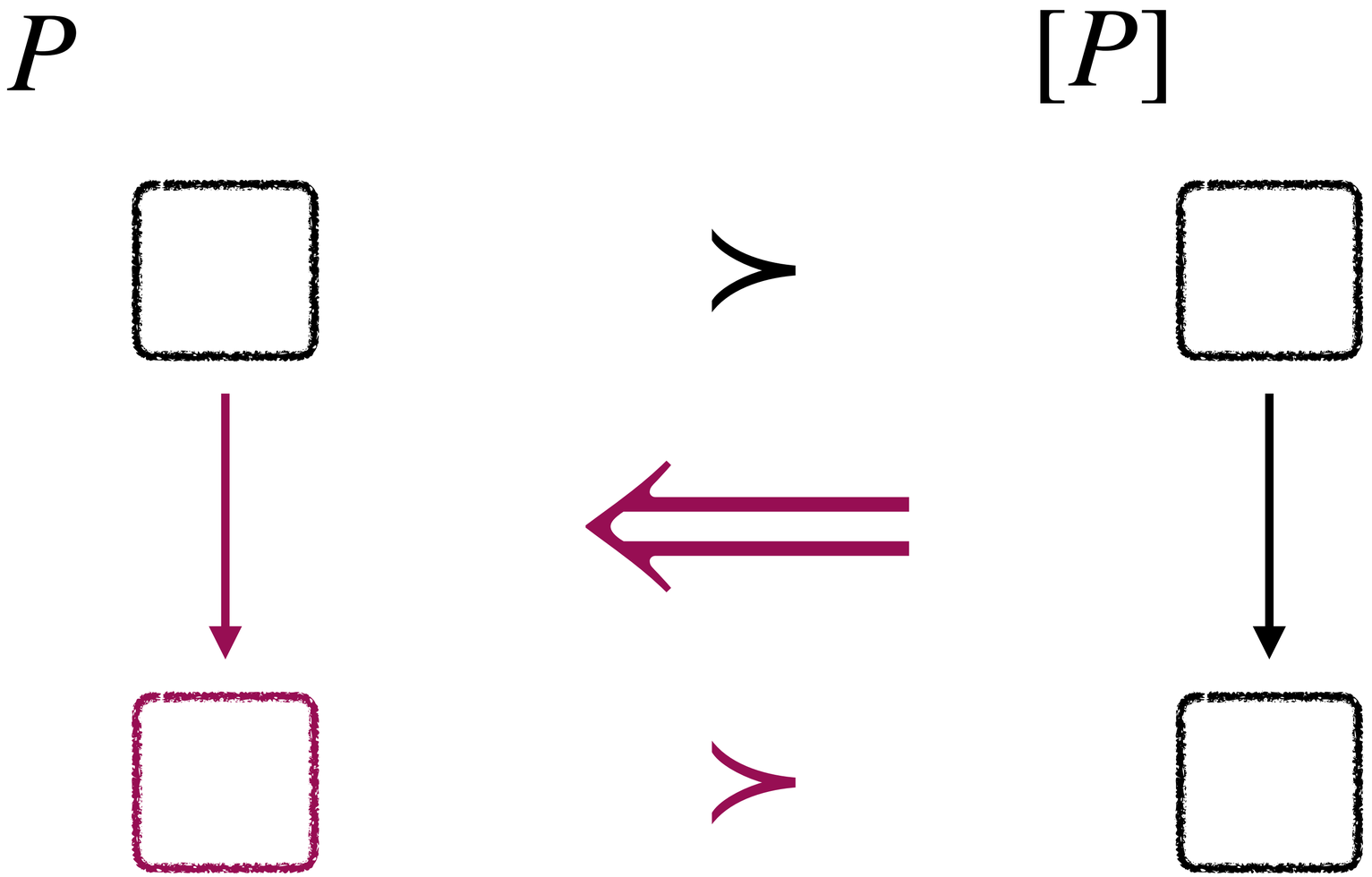
SNI Preservation

Safety

DeadCode

SNI Preservation

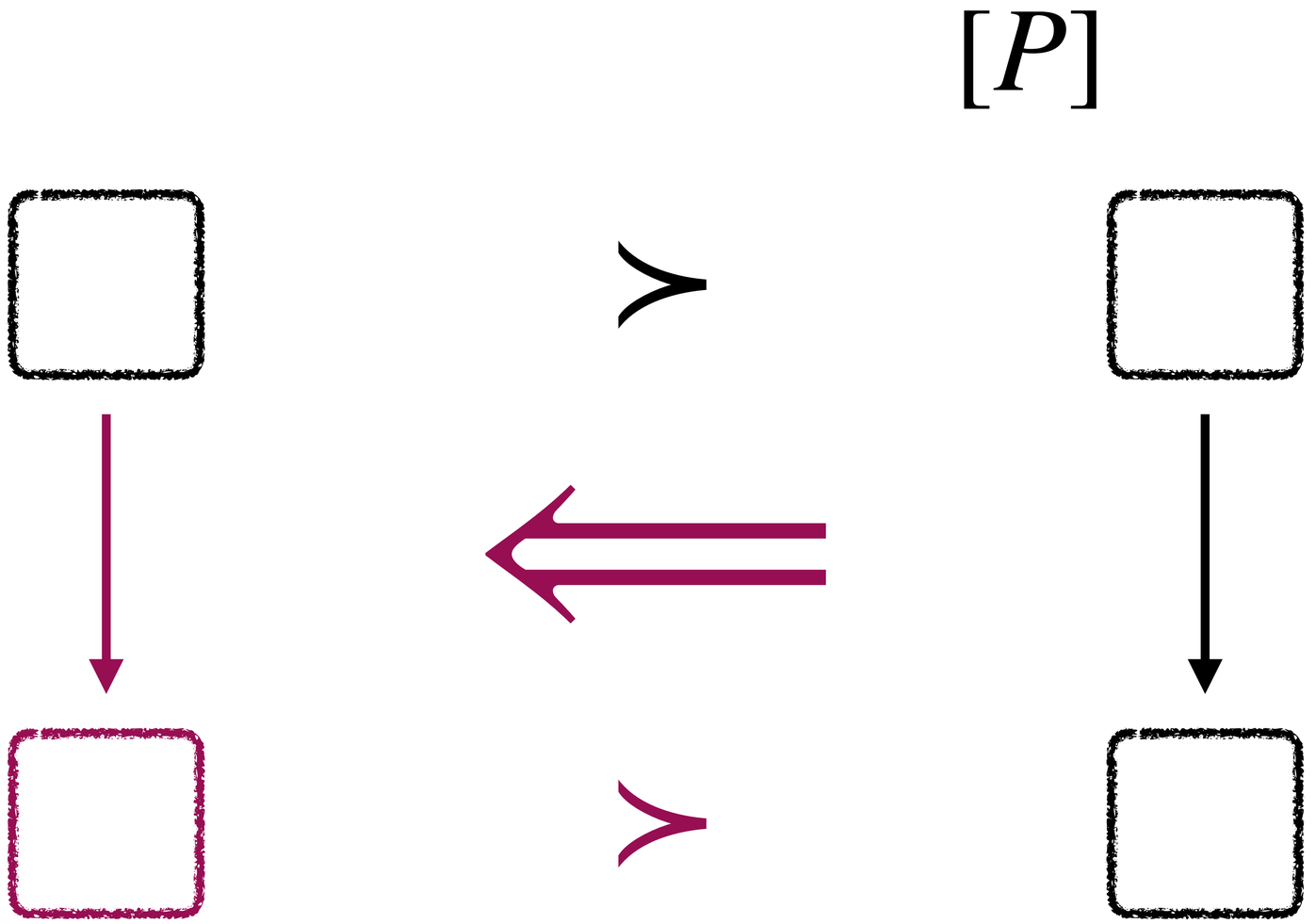
Safety



SNI Preservation

Safety

Method



SNI Preservation

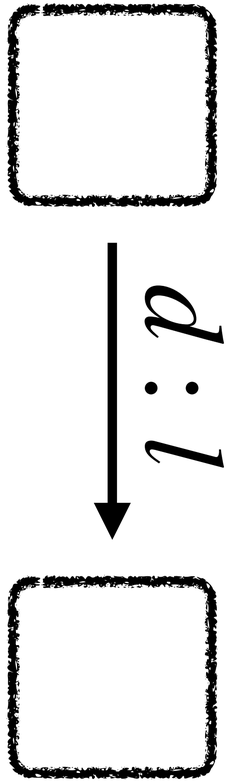
Safety

P

γ

γ

$[P]$



P

```
fn( public i, size, secret sec)
 if (i < size)
  a = buf[i];
  a = 0;
 ret;
```

$[P]_{\mathsf{dc}}$

```
   fn( public i, size, secret sec)
    if (i < size)
     nop;
     a = 0;
    ret;
```

P

```
fn( public i, size, secret sec)
 if (i < size)
  a = buf[i];
  a = 0;
 ret;
```

$[P]_{\mathsf{dc}}$

```
   fn( public i, size, secret sec)
    if (i < size)
     nop;
     a = 0;
    ret;
```

P

```
fn( public i, size, secret sec)
  if (i < size)
    a = buf[i];
    a = 0;
  ret;
```

$[P]_{\mathsf{dc}}$

```
fn( public i, size, secret sec)
  if (i < size)
    nop;
    a = 0;
  ret;
```

i	4
a	15
size	8
sec	42

$\succ$

i	4
a	11
size	8
sec	42

SNI Preservation

DeadCode

P

```
fn( public i, size, secret sec)
 if (i < size)
  a = buf[i];
  a = 0;
 ret;
```

$[P]_{\mathsf{dc}}$

```
fn( public i, size, secret sec)
 if (i < size)
  nop;
  a = 0;
 ret;
```

i	4
a	15
size	8
sec	42

$\succ$

i	4
a	11
size	8
sec	42

Directive Leakage

DeadCode

P

```
fn( public i, size, secret sec)
 if (i < size)
  a = buf[i];
  a = 0;
 ret;
```

$[P]_{\text{dc}}$

```
fn( public i, size, secret sec)
 if (i < size)
  nop;
  a = 0;
 ret;
```

i	4
a	15
size	8
sec	42

$\succ$

i	4
a	11
size	8
sec	42

Directive	Leakage
correct	BR true

DeadCode

P

```
fn( public i, size, secret sec)
 if (i < size)
  a = buf[i];
  a = 0;
 ret;
```

$[P]_{\text{dc}}$

```
fn( public i, size, secret sec)
 if (i < size)
  nop;
  a = 0;
 ret;
```

i	4
a	15
size	8
sec	42

$\succ$

i	4
a	11
size	8
sec	42

Directive	Leakage
correct	BR true
step	

P

```
fn( public i, size, secret sec)
 if (i < size)
  a = buf[i];
  a = 0;
 ret;
```

$[P]_{dc}$

```
fn( public i, size, secret sec)
 if (i < size)
  nop;
  a = 0;
 ret;
```

i	: 4
a	: 15
size	: 8
sec	: 42

Directive	Leakage

i	: 4
a	: 11
size	: 8
sec	: 42

Directive	Leakage
correct	BR true
step	

P

```
fn( public i, size, secret sec)
 if (i < size)
  a = buf[i];
  a = 0;
 ret;
```

$[P]_{dc}$

```
fn( public i, size, secret sec)
 if (i < size)
  nop;
  a = 0;
 ret;
```

i	: 4
a	: 15
size	: 8
sec	: 42

Directive	Leakage
correct	BR true

i	: 4
a	: 11
size	: 8
sec	: 42

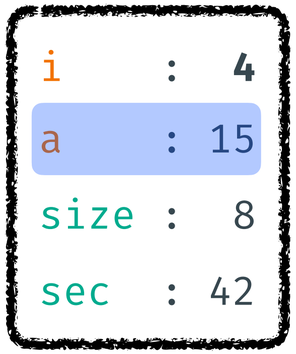
Directive	Leakage
correct	BR true
step	

P

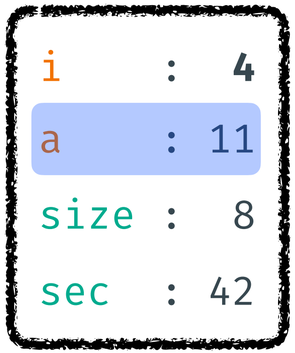
```
fn( public i, size, secret sec)
 if (i < size)
  a = buf[i];
  a = 0;
 ret;
```

$[P]_{dc}$

```
fn( public i, size, secret sec)
 if (i < size)
  nop;
  a = 0;
 ret;
```



Directive	Leakage
correct	BR true
step	LD 4



Directive	Leakage
correct	BR true
step	

SNI Preservation
Safety

DeadCode

P

```
fn( public i, size, secret sec)
 if (i < size)
  a = buf[i];
  a = 0;
 ret;
```

$[P]_{dc}$

```
fn( public i, size, secret sec)
 if (i < size)
  nop;
  a = 0;
 ret;
```

i	: 4
a	: 15
size	: 8
sec	: 42

Directive	Leakage
correct	BR true
step	LD 4

i	: 4
a	: 11
size	: 8
sec	: 42

Directive	Leakage
correct	BR true
step	

P

```
fn( public i, size, secret sec)
 if (i < size)
  a = buf[i];
  a = 0;
 ret;
```

i	: 20
a	: 15
size	: 8
sec	: 42

Directive	Leakage
miss	BR false

$[P]_{dc}$

```
fn( public i, size, secret sec)
 if (i < size)
  nop;
  a = 0;
 ret;
```

i	: 20
a	: 11
size	: 8
sec	: 42

Directive	Leakage
miss	BR false

P

```
fn( public i, size, secret sec)
 if (i < size)
  a = buf[i];
  a = 0;
 ret;
```

i	: 20
a	: 15
size	: 8
sec	: 42

Directive	Leakage
miss	BR false

$[P]_{dc}$

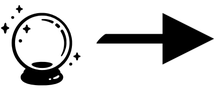
```
fn( public i, size, secret sec)
 if (i < size)
  nop;
  a = 0;
 ret;
```

i	: 20
a	: 11
size	: 8
sec	: 42

Directive	Leakage
miss	BR false
step	

P

```
fn( public i, size, secret sec)
 if (i < size)
  a = buf[i];
  a = 0;
 ret;
```

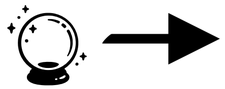


i	: 20
a	: 15
size	: 8
sec	: 42

Directive	Leakage
miss	BR false
oob …	LD 4

$[P]_{\text{dc}}$

```
fn( public i, size, secret sec)
 if (i < size)
  nop;
  a = 0;
 ret;
```



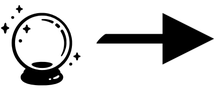
i	: 20
a	: 11
size	: 8
sec	: 42

Directive	Leakage
miss	BR false
step	

DeadCode

P

```
fn( public i, size, secret sec)
 if (i < size)
  a = buf[i];
  a = 0;
 ret;
```

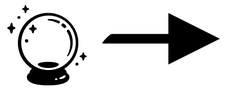


i	: 20
a	: 15
size	: 8
sec	: 42

Directive	Leakage
miss	BR false
oob …	LD 4

$[P]_{\text{dc}}$

```
fn( public i, size, secret sec)
 if (i < size)
  nop;
  a = 0;
 ret;
```



i	: 20
a	: 11
size	: 8
sec	: 42

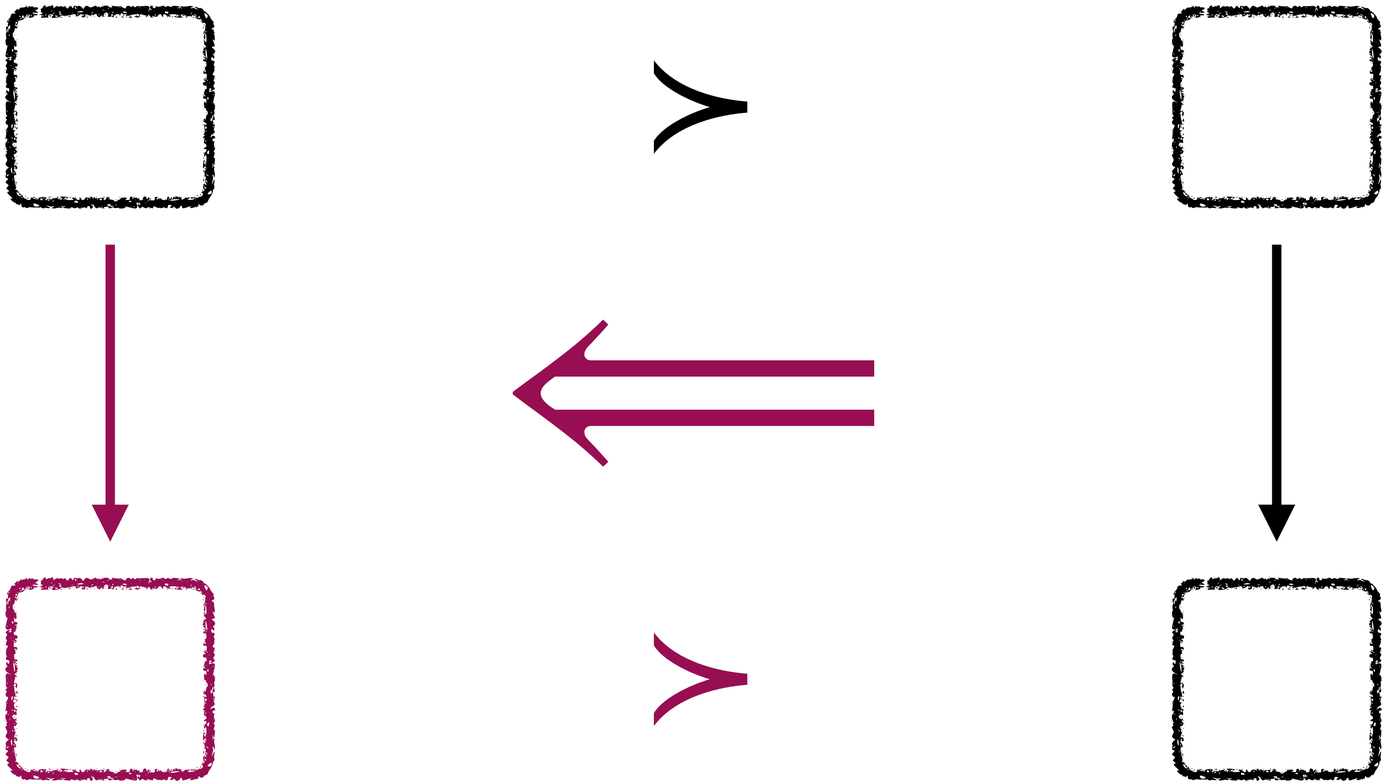
Directive	Leakage
miss	BR false
step	

SNI Preservation

Safety

P

$[P]$



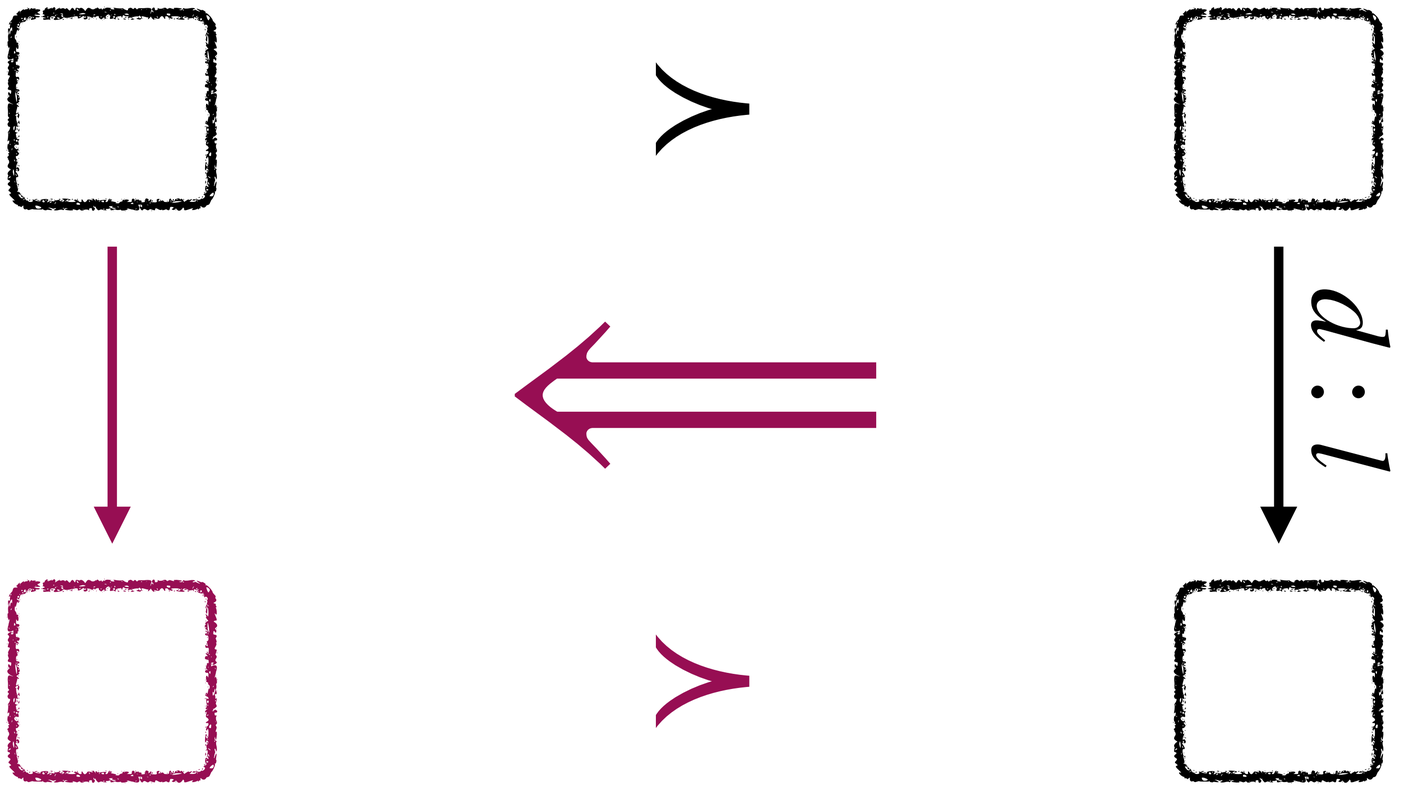
SNI Preservation

Safety

Method

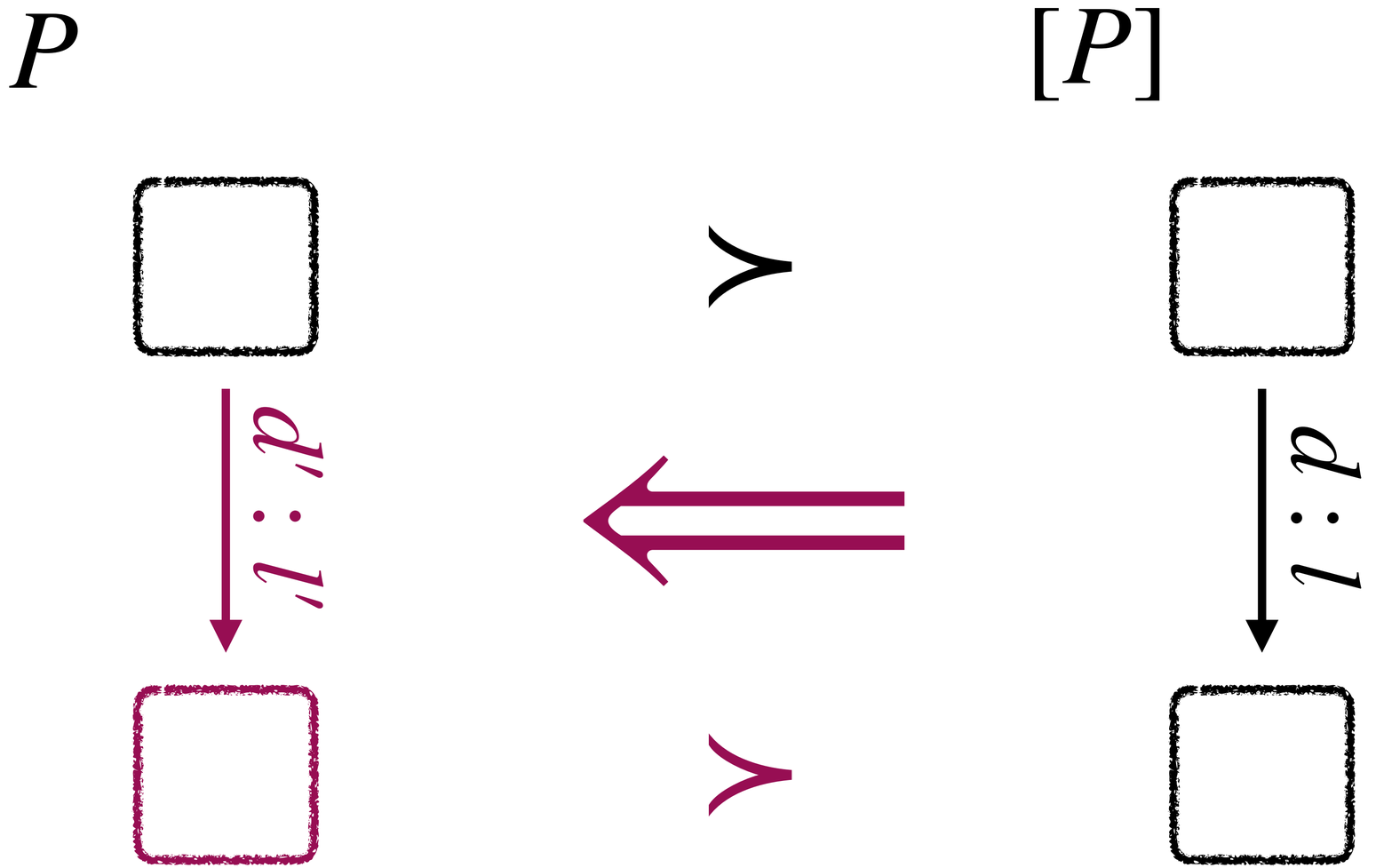
P

$[P]$



SNI Preservation

Safety



SNI Preservation

Safety

Method

SNI Preservation

SNI Preservation

Method

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

IF $\forall P \quad P \models$ **SNI** $\Longrightarrow [P] \models$ **SNI**

$P \models \mathbf{SNI}$

```
fn( public i, size, secret sec)
 if (i < size)
  a = buf[i];
  a = 0;
 ret;
```

```
fn( public i, size, secret sec)
 if (i < size)
  nop;
  a = 0;
 ret;
```

$P \models \mathbf{SNI}$

```
fn( public i, size, secret sec)
 if (i < size)
  a = buf[i];
  a = 0;
 ret;
```

$\overset{?}{\Longrightarrow}$

$[P]_{\mathsf{dc}} \models \mathbf{SNI}$

```
fn( public i, size, secret sec)
 if (i < size)
  nop;
  a = 0;
 ret;
```

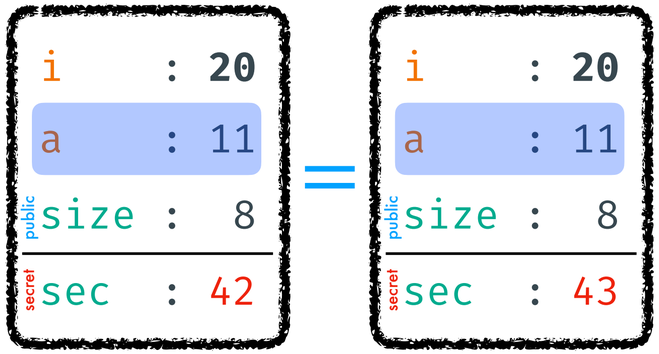
$P \vDash \mathbf{SNI}$

```
fn( public i, size, secret sec)
 if (i < size)
  a = buf[i];
  a = 0;
 ret;
```

$\overset{?}{\Longrightarrow}$

$[P]_{\mathrm{dc}} \vDash \mathbf{SNI}$

```
fn( public i, size, secret sec)
 if (i < size)
  nop;
  a = 0;
 ret;
```



$P \vDash \mathbf{SNI}$

```
fn( public i, size, secret sec)
 if (i < size)
  a = buf[i];
  a = 0;
 ret;
```

$\overset{?}{\Longrightarrow}$

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

```
fn( public i, size, secret sec)
 if (i < size)
  nop;
  a = 0;
 ret;
```

i	: 20
a	: 11
public size	: 8
secret sec	: 42

=

i	: 20
a	: 11
public size	: 8
secret sec	: 43

Directive	Leakage
miss	BR false
step	
step	

SNI Preservation **Method**

$P \models \text{SNI}$

```
fn( public i, size, secret sec)
 if (i < size)
  a = buf[i];
  a = 0;
 ret;
```

$\overset{?}{\Longrightarrow}$

$[P]_{\text{dc}} \models \text{SNI}$

```
fn( public i, size, secret sec)
 if (i < size)
  nop;
  a = 0;
 ret;
```

i	: 20
a	: 11
public size	: 8
secret sec	: 42

=

i	: 20
a	: 11
public size	: 8
secret sec	: 43

Directive	Leakage	
miss	BR false	BR false
step		
step		

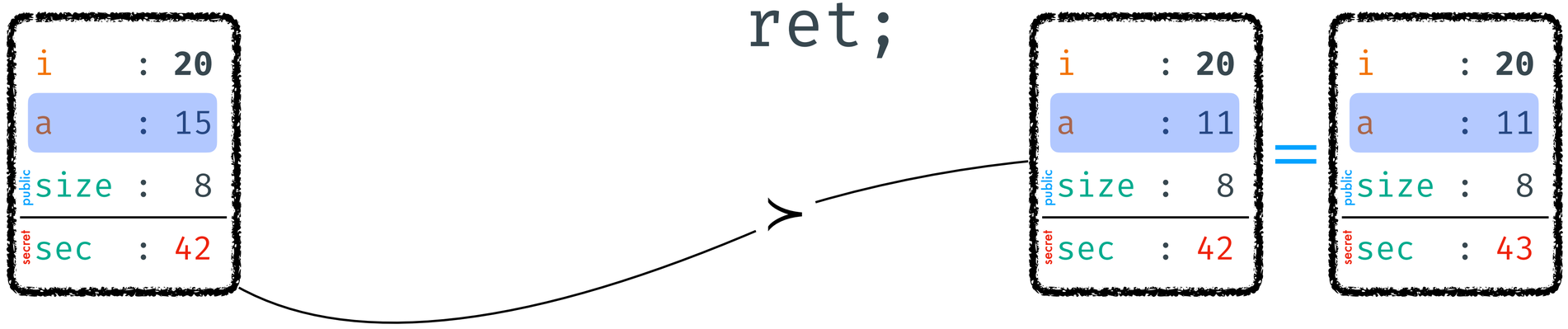
SNI Preservation **Method**

$P \models \mathbf{SNI}$ $\overset{?}{\Longrightarrow}$ $[P]_{\mathrm{dc}} \models \mathbf{SNI}$

```
fn( public i, size, secret sec)
 if (i < size)
  a = buf[i];
  a = 0;
 ret;
```

```
fn( public i, size, secret sec)
 if (i < size)
  nop;
  a = 0;
 ret;
```

$\prec$: Equal up to dead locations



Directive	Leakage	
miss	BR false	BR false
step		
step		

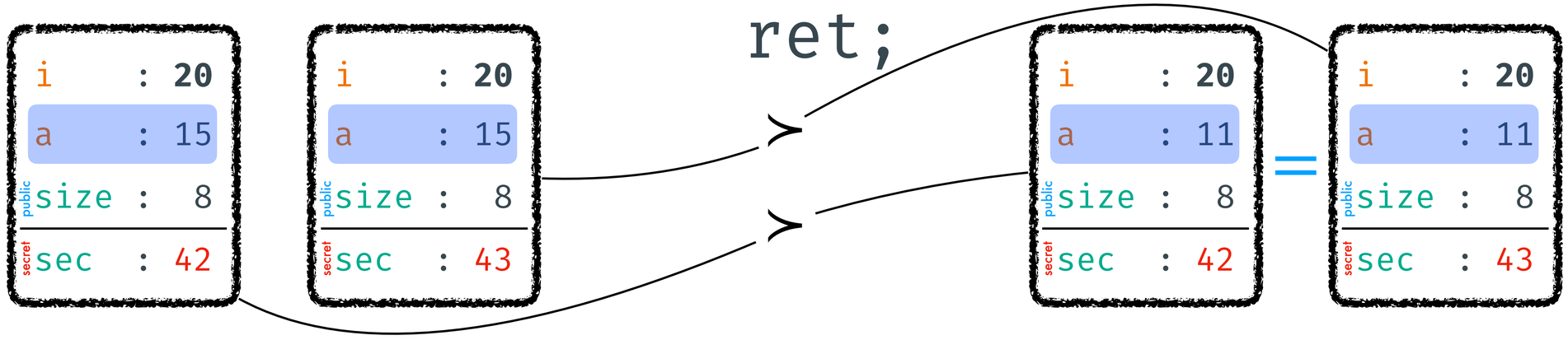
SNI Preservation **Method**

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

```
fn( public i, size, secret sec)
 if (i < size)
  a = buf[i];
  a = 0;
 ret;
```

```
fn( public i, size, secret sec)
 if (i < size)
  nop;
  a = 0;
 ret;
```

$\prec$: Equal up to dead locations



Directive	Leakage	
miss	BR false	BR false
step		
step		

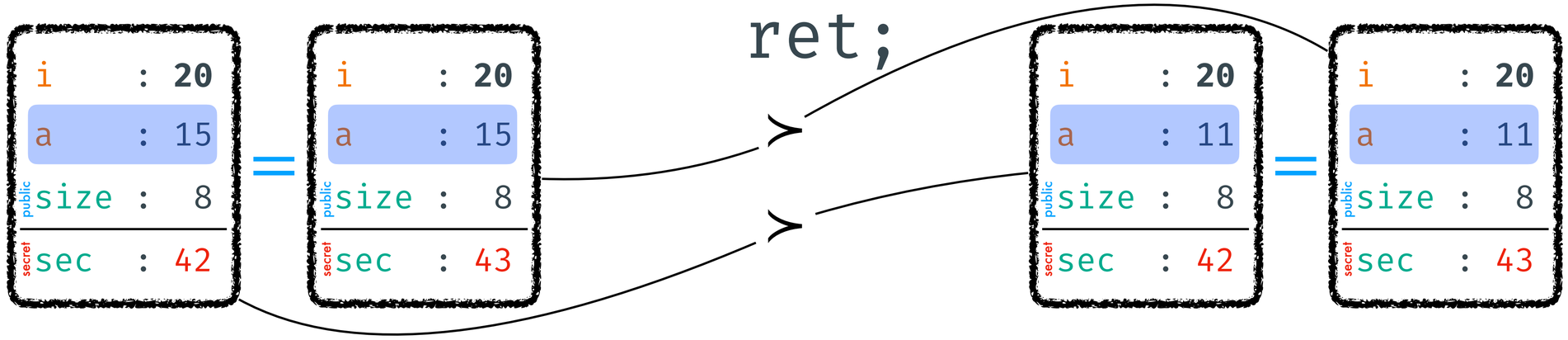
SNI Preservation **Method**

$P \models \mathbf{SNI}$ $\overset{?}{\Longrightarrow}$ $[P]_{dc} \models \mathbf{SNI}$

```
fn( public i, size, secret sec)
 if (i < size)
  a = buf[i];
  a = 0;
 ret;
```

```
fn( public i, size, secret sec)
 if (i < size)
  nop;
  a = 0;
 ret;
```

$\prec$: Equal up to dead locations



Directive	Leakage	
miss	BR false	BR false
step		
step		

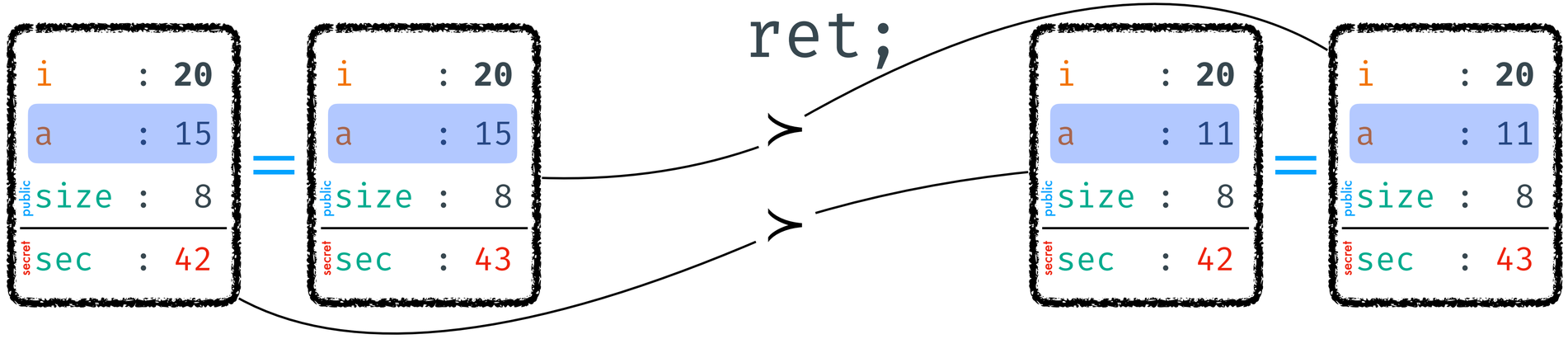
SNI Preservation **Method**

$P \vDash \mathbf{SNI}$ $\overset{?}{\Longrightarrow}$ $[P]_{\mathrm{dc}} \vDash \mathbf{SNI}$

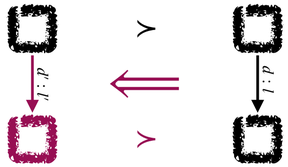
```
fn( public i, size, secret sec)
 if (i < size)
  a = buf[i];
  a = 0;
 ret;
```

```
fn( public i, size, secret sec)
 if (i < size)
  nop;
  a = 0;
 ret;
```

$\prec$: Equal up to dead locations



Directive	Leakage	
miss	BR false	BR false
step		
step		

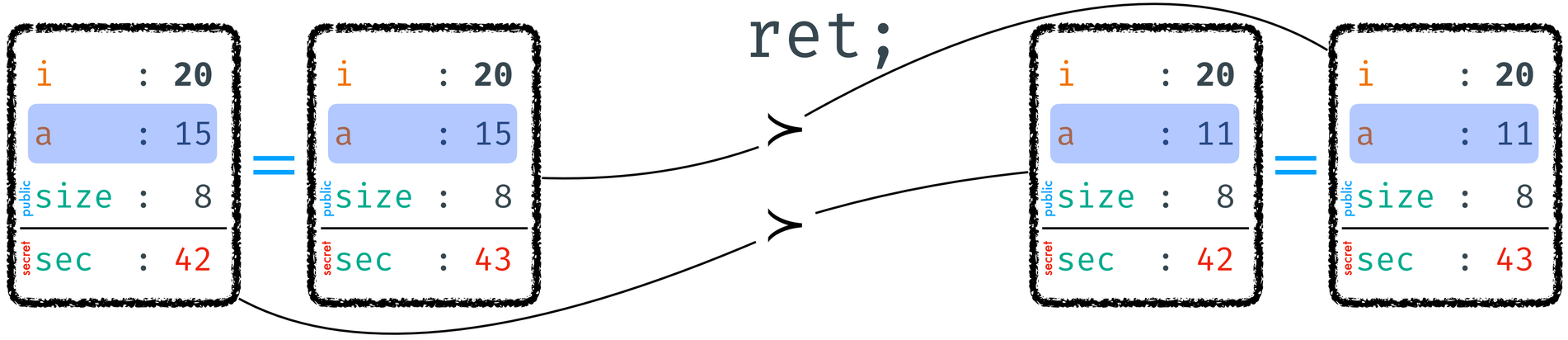


$P \models \mathbf{SNI}$ $\overset{?}{\Longrightarrow}$ $[P]_{dc} \models \mathbf{SNI}$

```
fn( public i, size, secret sec)
 if (i < size)
  a = buf[i];
  a = 0;
 ret;
```

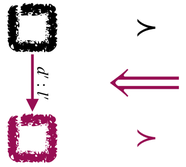
```
fn( public i, size, secret sec)
 if (i < size)
  nop;
  a = 0;
 ret;
```

$\asymp$: Equal up to dead locations



Directive	Leakage
miss	
oob sec	
step	

Directive	Leakage	
miss	BR false	BR false
step		
step		

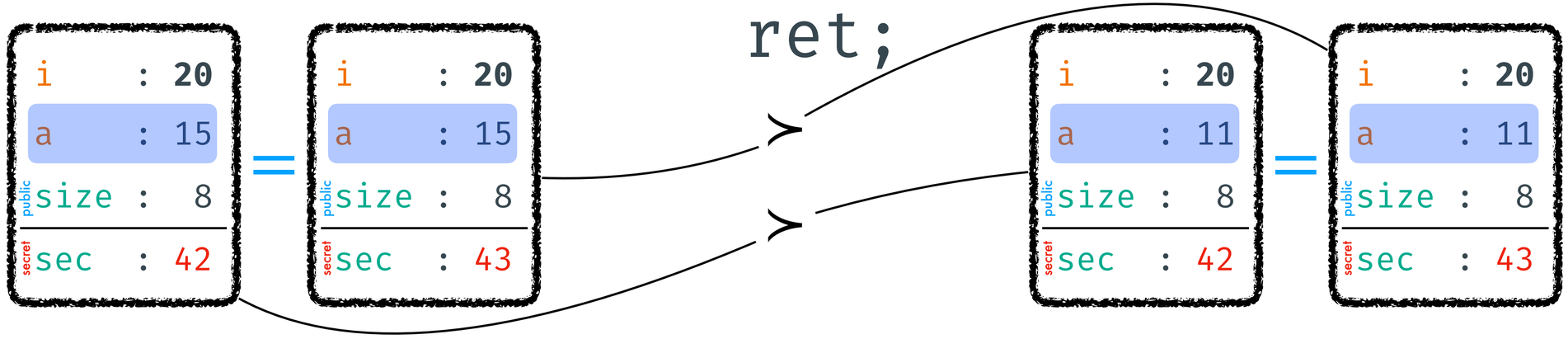


$P \models \mathbf{SNI}$ $\overset{?}{\Longrightarrow}$ $[P]_{\mathrm{dc}} \models \mathbf{SNI}$

```
fn( public i, size, secret sec)
 if (i < size)
  a = buf[i];
  a = 0;
 ret;
```

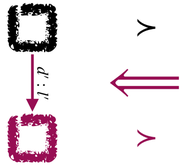
```
fn( public i, size, secret sec)
 if (i < size)
  nop;
  a = 0;
 ret;
```

$\asymp$: Equal up to dead locations



Directive	Leakage
miss	BR false
oob sec	LD 20
step	

Directive	Leakage	
miss	BR false	BR false
step		
step		

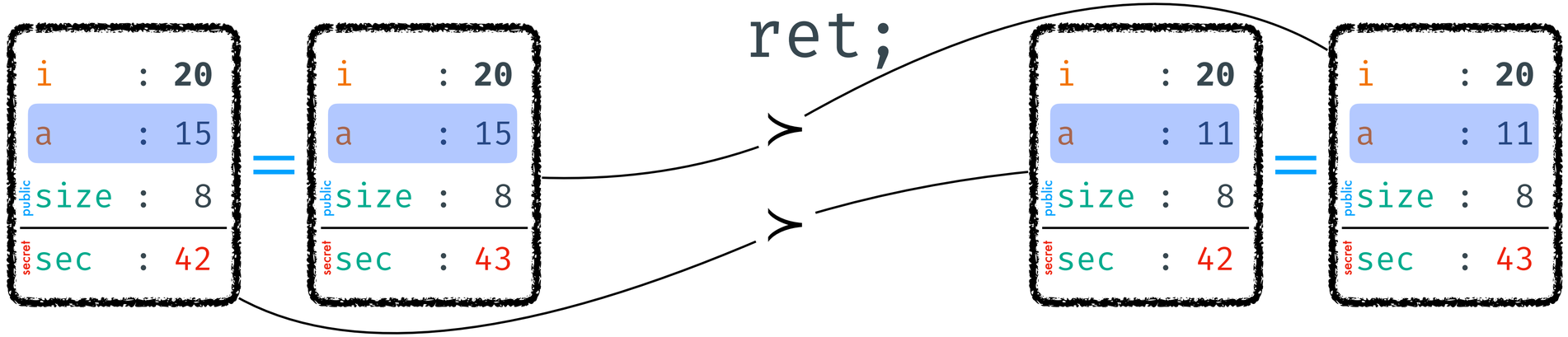


$P \models \mathbf{SNI}$ $\overset{?}{\Longrightarrow}$ $[P]_{\mathrm{dc}} \models \mathbf{SNI}$

```
fn( public i, size, secret sec)
 if (i < size)
  a = buf[i];
  a = 0;
 ret;
```

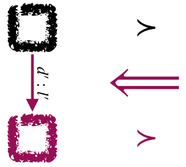
```
fn( public i, size, secret sec)
 if (i < size)
  nop;
  a = 0;
 ret;
```

$\sim$: Equal up to dead locations



Directive	Leakage	Leakage
miss	BR false	BR false
oob sec	LD 20	LD 20
step		

Directive	Leakage	Leakage
miss	BR false	BR false
step		
step		



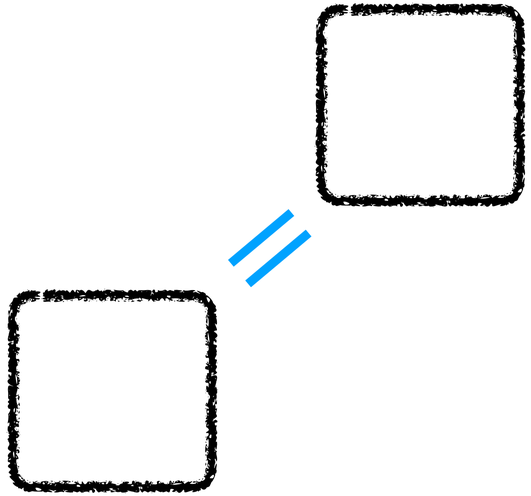
$P \models \mathbf{SNI}$

$[P] \models \mathbf{SNI}$?

SNI Preservation

$P \models \mathbf{SNI}$

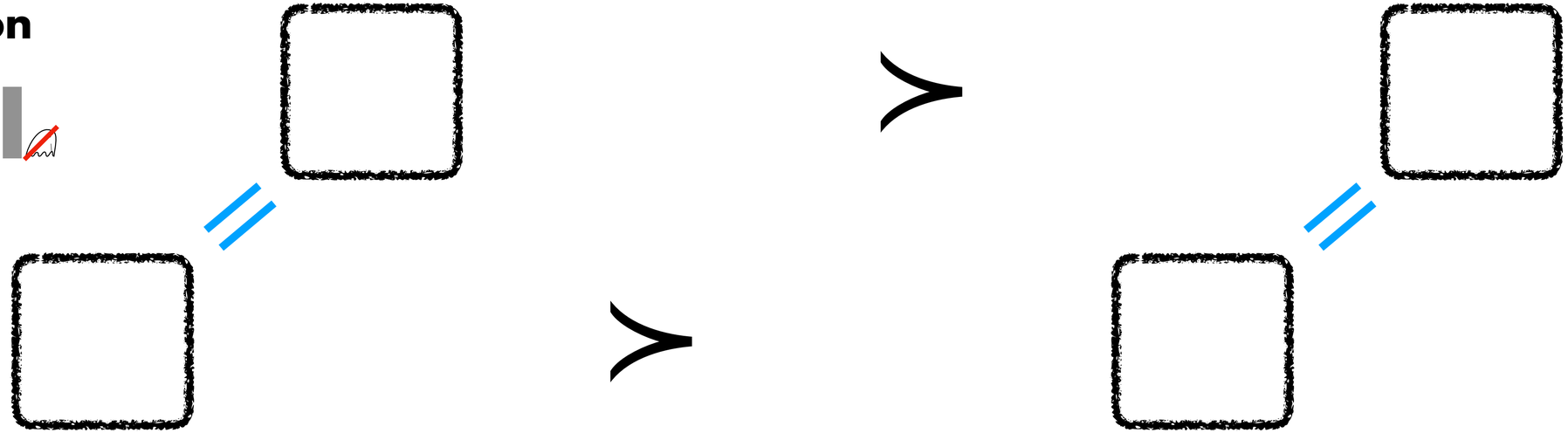
Method



$[P] \models \mathbf{SNI}$?

SNI Preservation

$P \models \mathbf{SNI}$

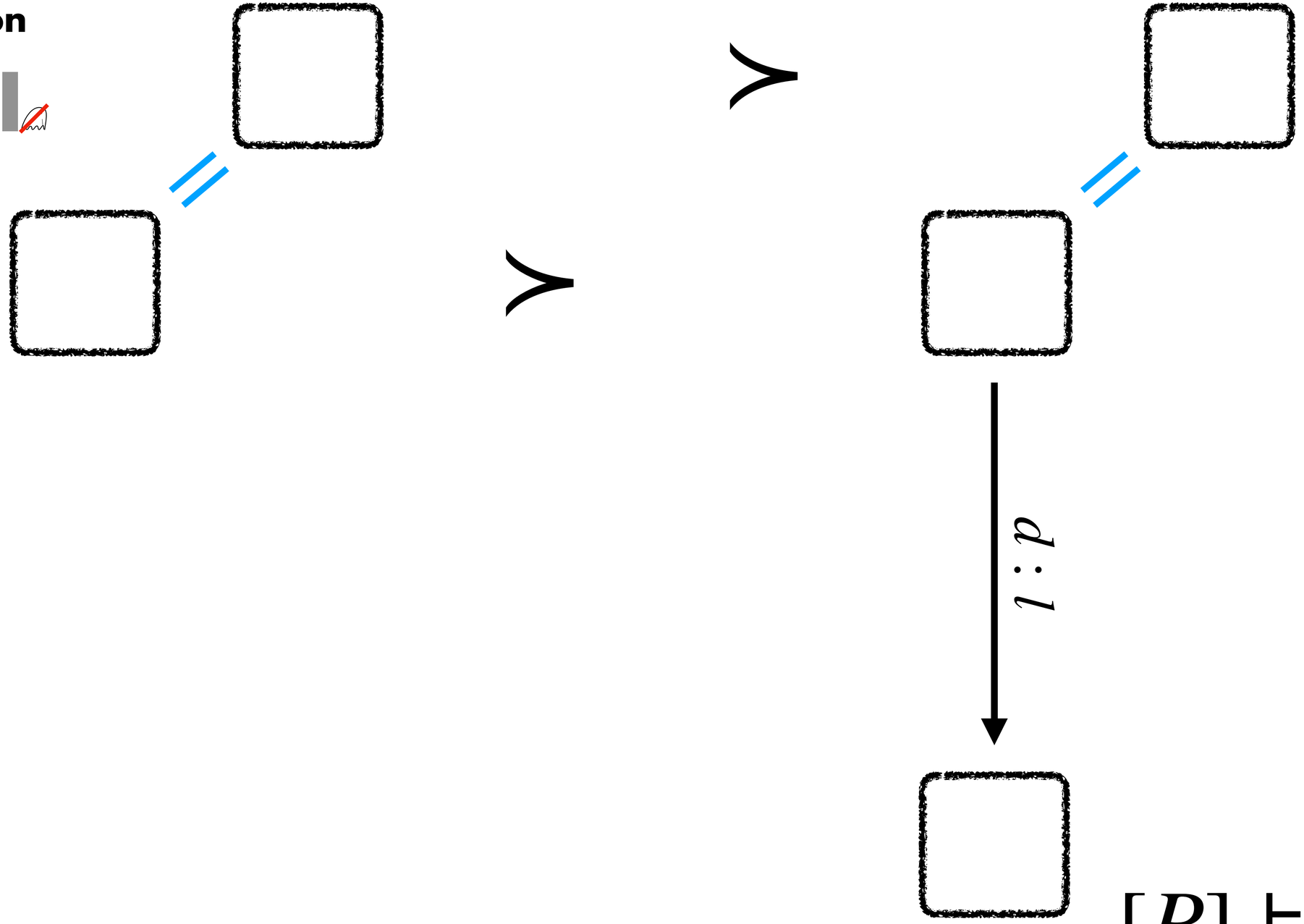


Method

$[P] \models \mathbf{SNI}$?

SNI Preservation

$P \models \mathbf{SNI}$



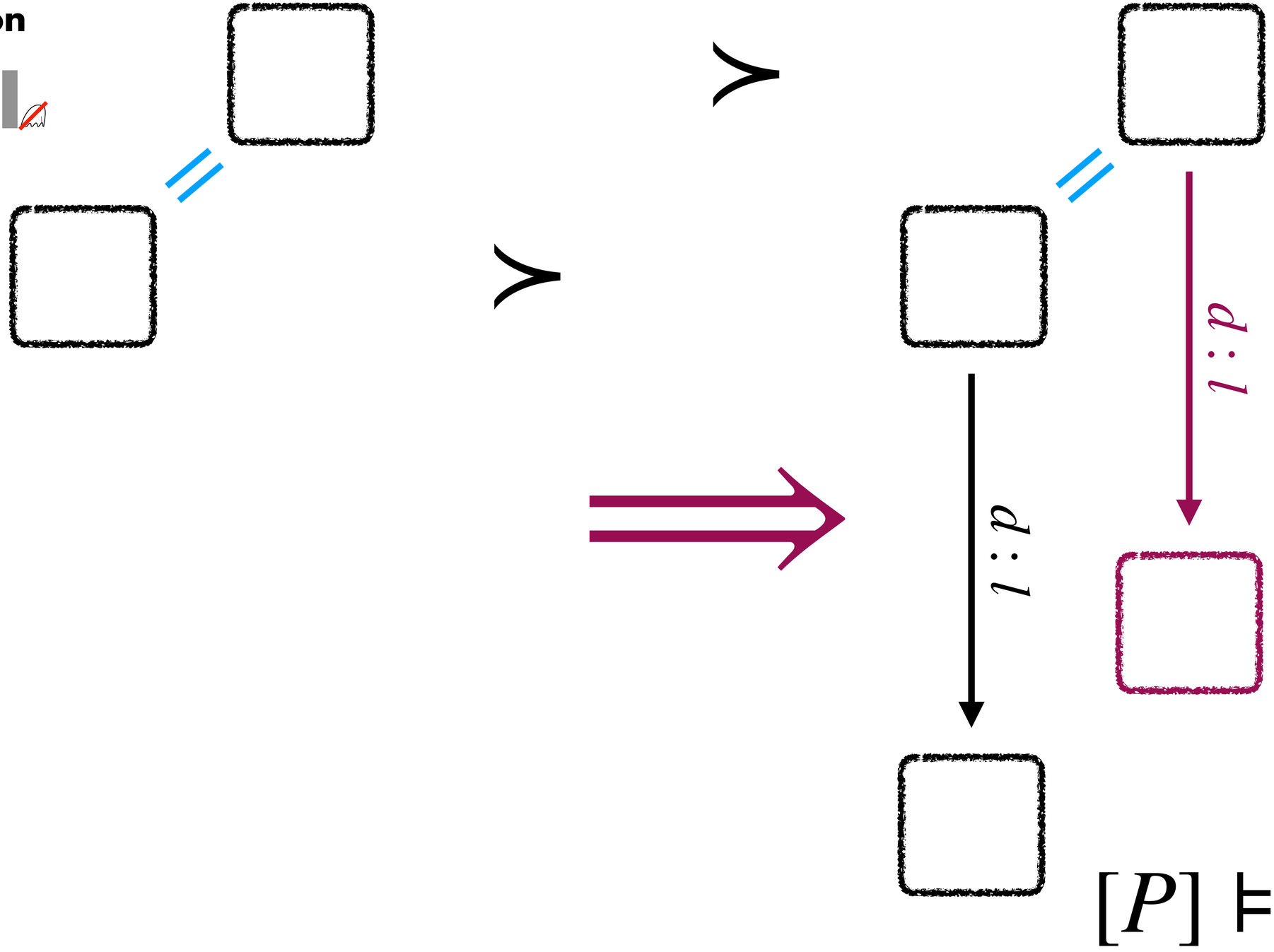
Method

$[P] \models \mathbf{SNI}$?

SNI Preservation

$P \models \mathbf{SNI}$

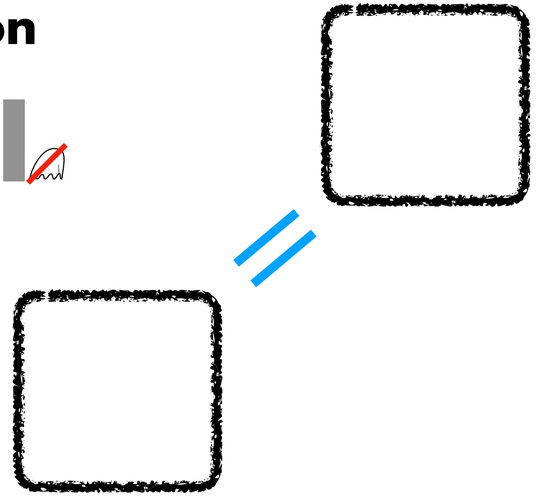
Method



$[P] \models \mathbf{SNI}$?

SNI Preservation

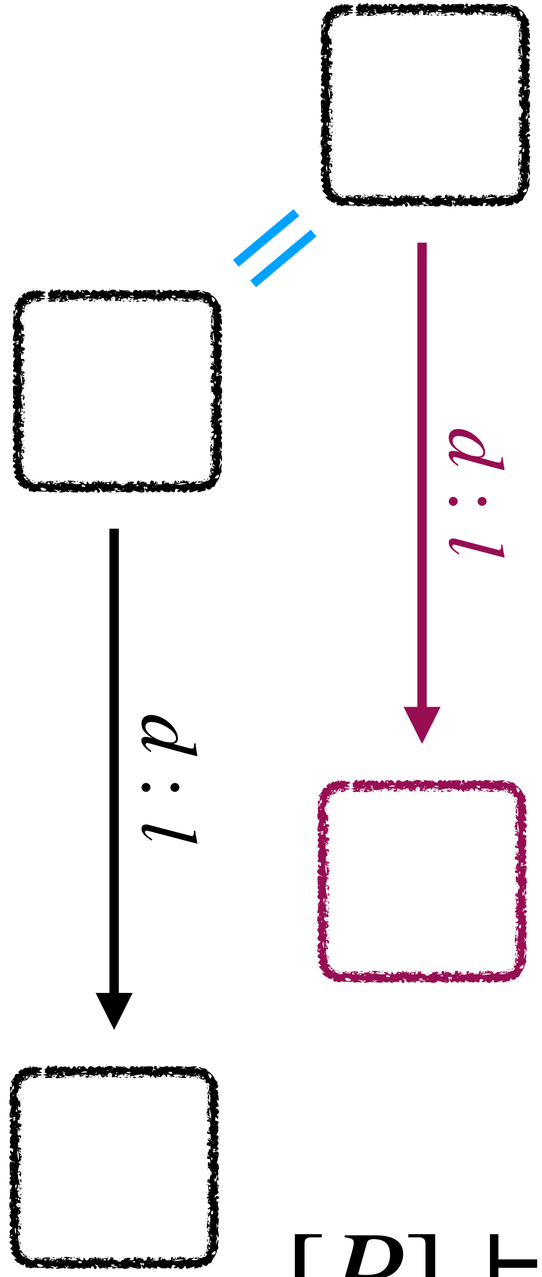
$P \models \mathbf{SNI}$



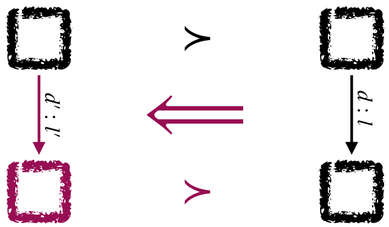
γ

γ

Method



$[P] \models \mathbf{SNI}$?



SNI Preservation

$P \models \mathbf{SNI}$

Method

$[P] \models \mathbf{SNI}$?

SNI Preservation

$P \models \mathbf{SNI}$

Method

$[P] \models \mathbf{SNI}$?

SNI Preservation

$P \models \mathbf{SNI}$

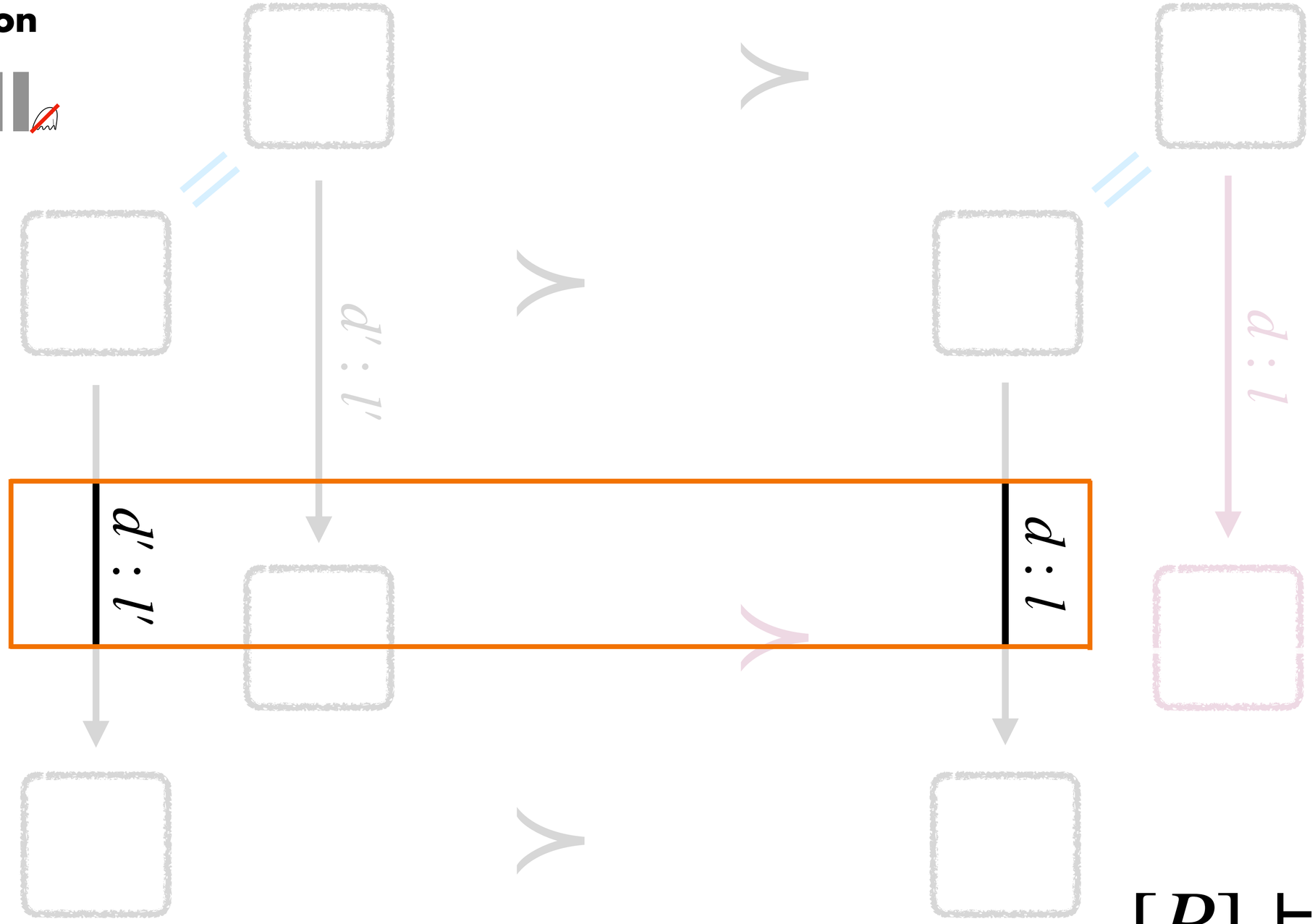
Method

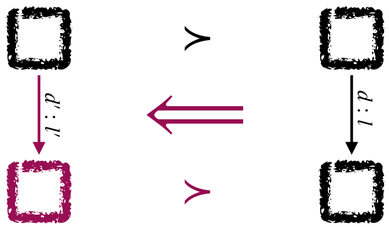
$[P] \models \mathbf{SNI}$?

SNI Preservation

$P \models \mathbf{SNI}$

Method



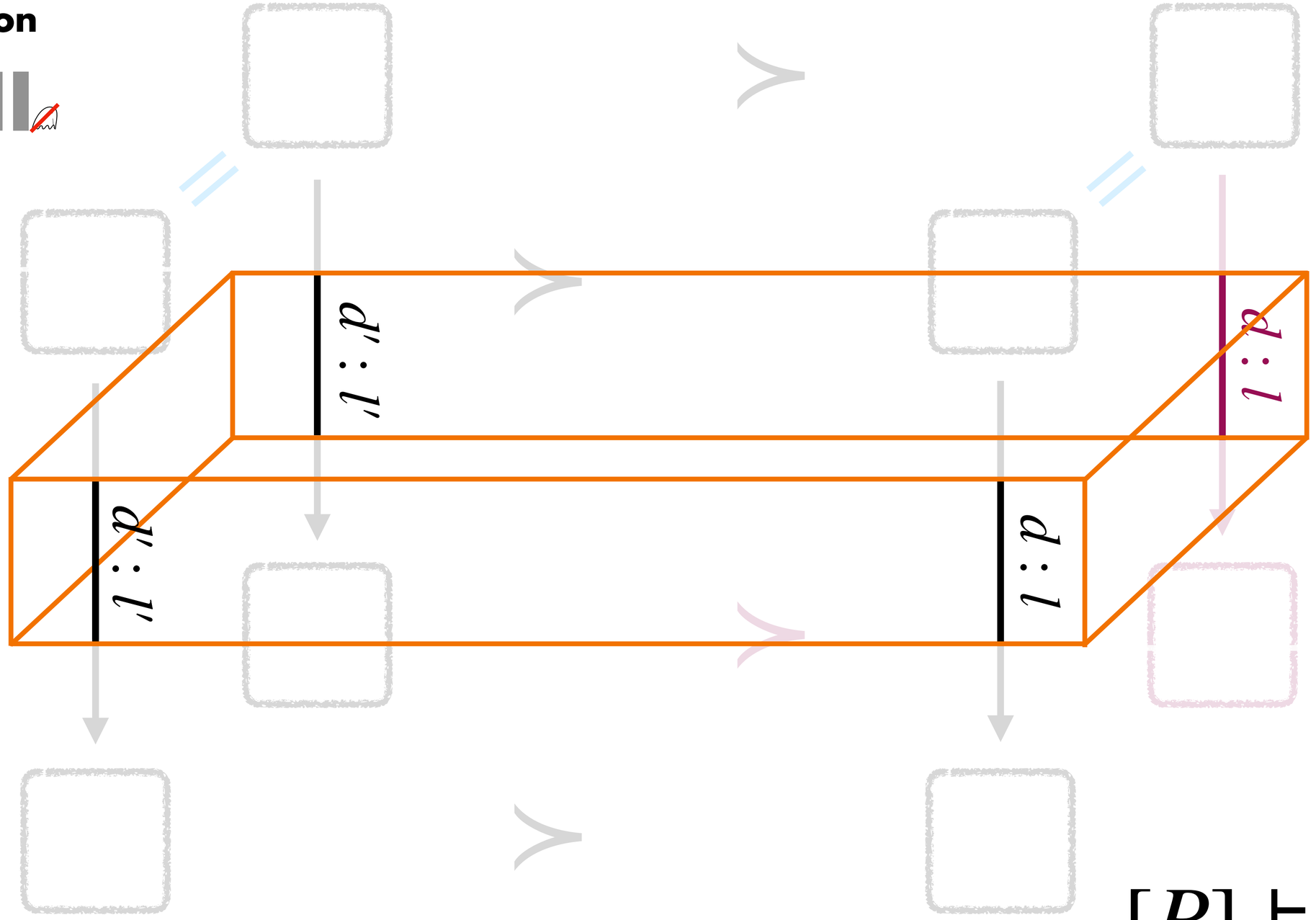


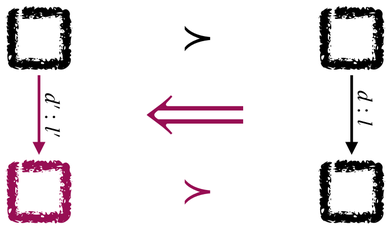
$[P] \models \mathbf{SNI}$?

SNI Preservation

Method

$P \models \mathbf{SNI}$

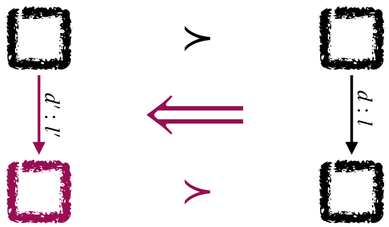




$[P] \models \mathbf{SNI}$?

P

$[P]$



P
[P]
Y
Y
Y
Y
d' : 1'
1' : d'
1 : d
1 : d
d : 1
Y
Y

P

$[P]$

P

γ

γ

$\Rightarrow$

γ

γ

$[P]$

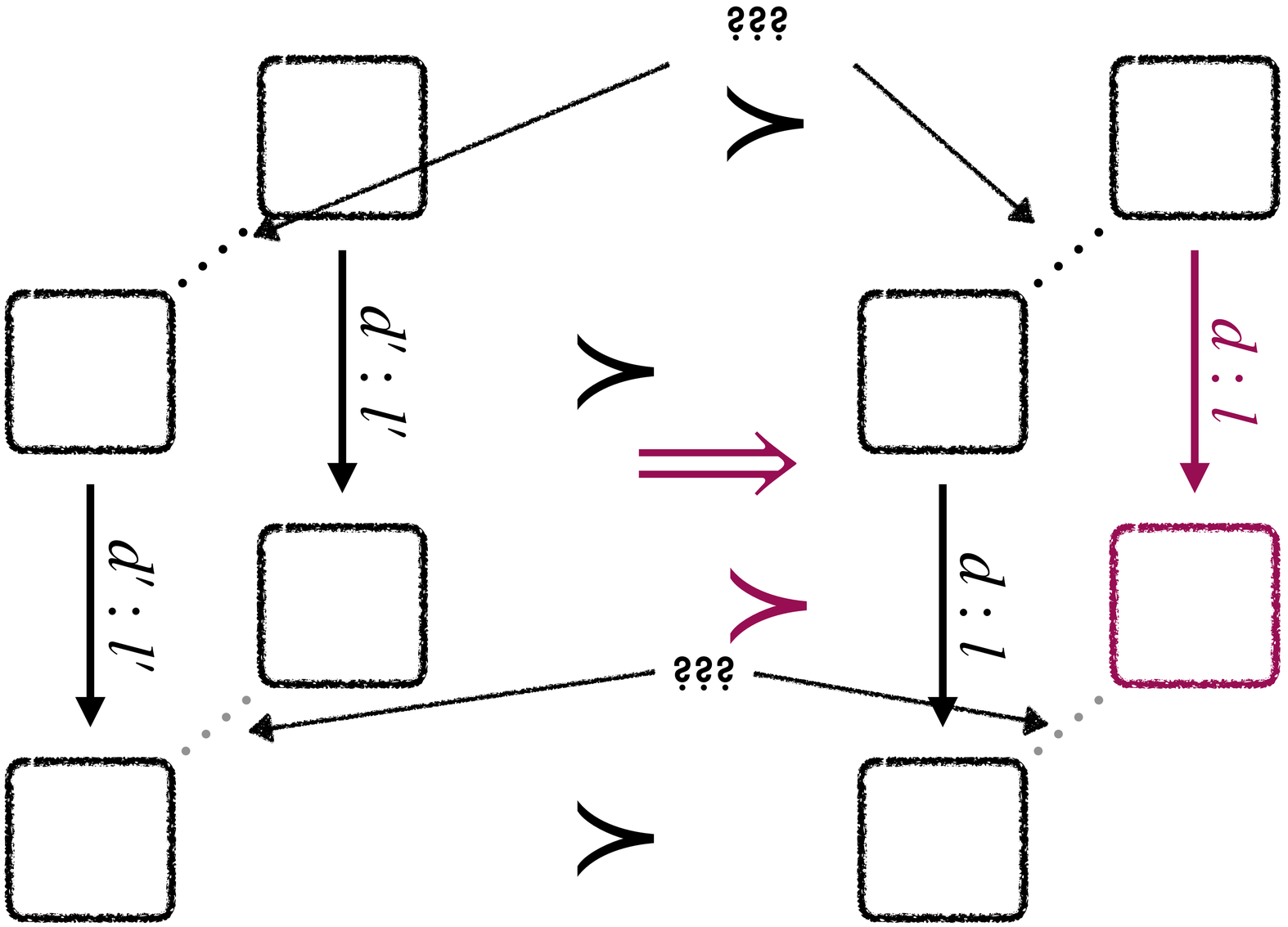
P

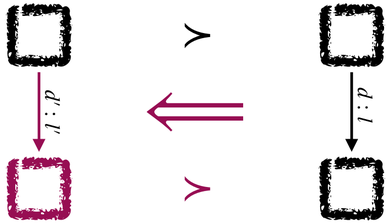
$[P]$

P

$[P]$

Initial only!



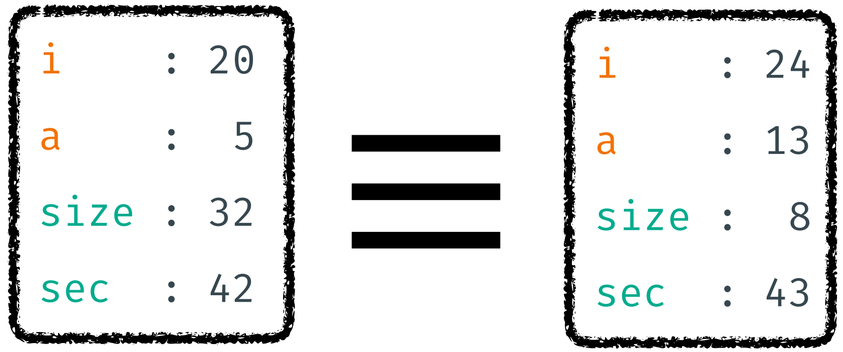


??? = Same Program Counter ≡

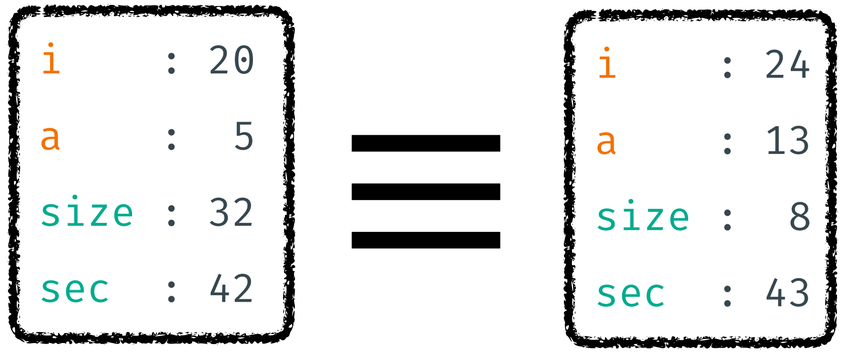
Method

??? = Same Program Counter ≡

Method



```
→ if (i < size) ←
    a = buf[i];
    a = 0;
```



```
→ if (i < size) ←
    a = buf[i];
    a = 0;
```

i	:	20	≡	i	:	24
a	:	5		a	:	13
size	:	32		size	:	8
sec	:	42		sec	:	43

$$S = U \implies S \equiv U$$

$[P]$

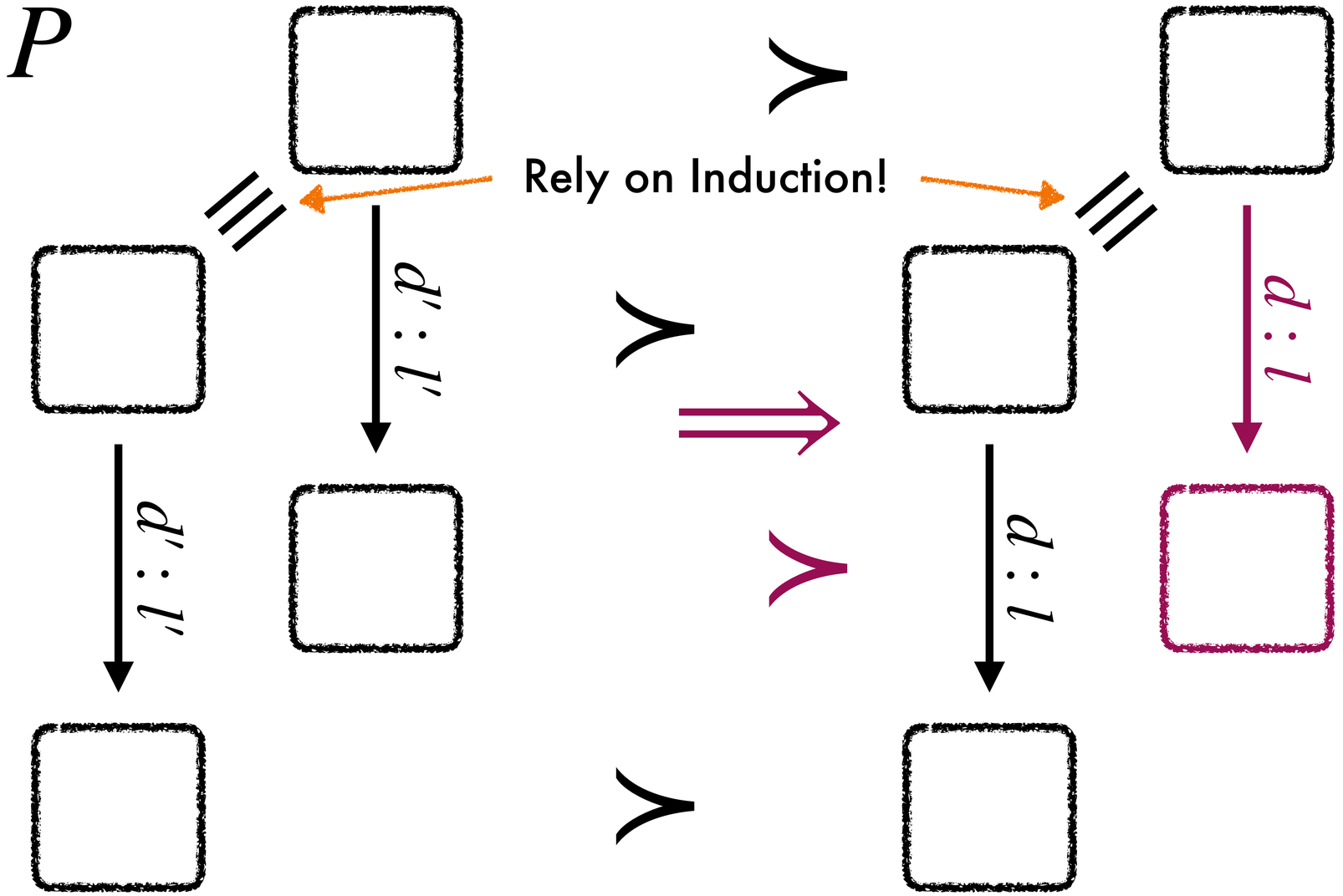
γ
$d : 1$
γ
$\Rightarrow$
γ
$1 : p$
γ

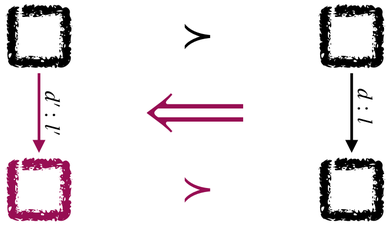
P

$d' : 1'$
$1' : p'$

γ
$1 : p$
$\Rightarrow$
γ

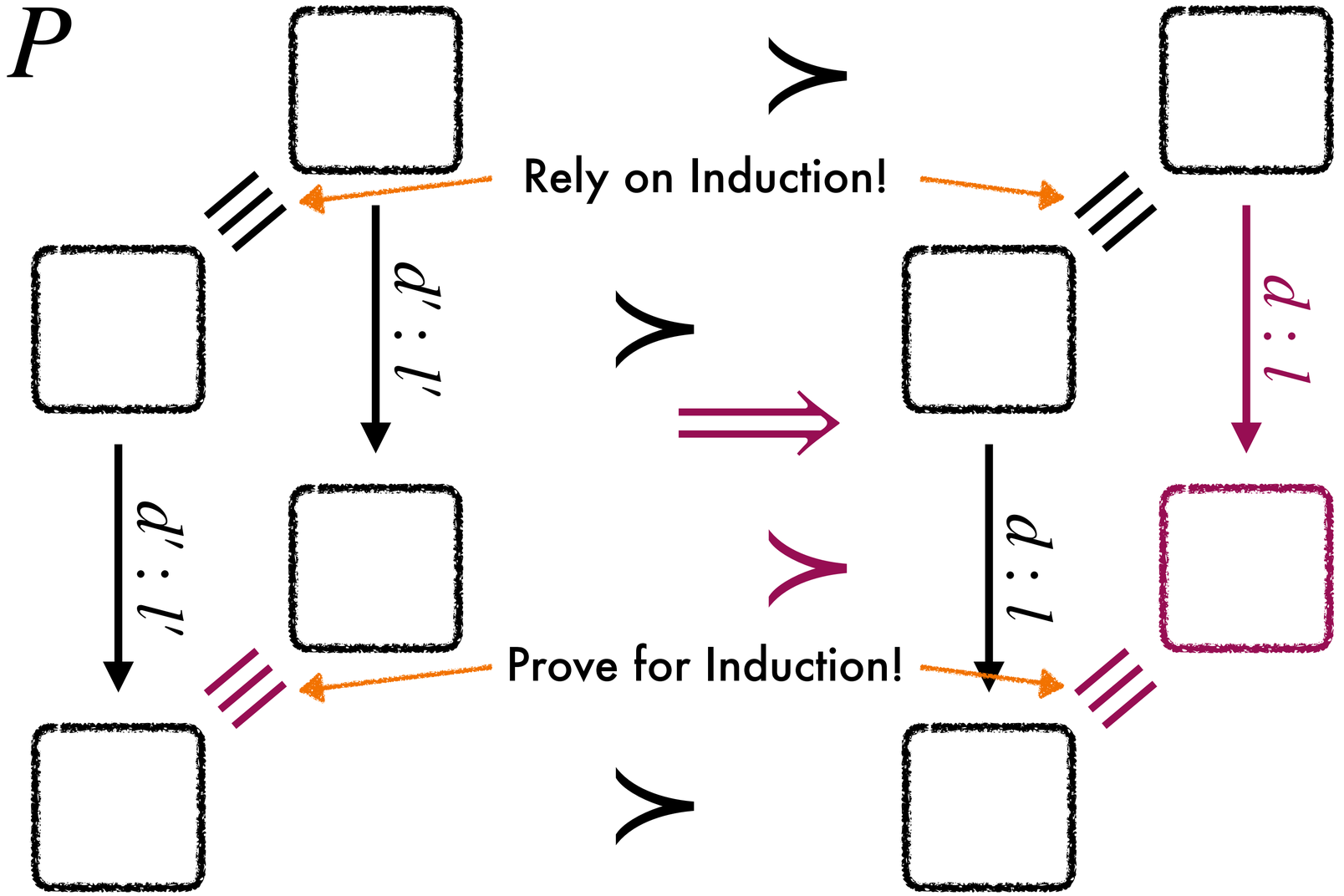
$[P]$

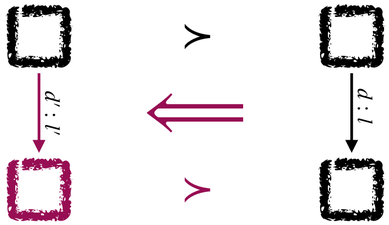




P

$[P]$





$[P]$

P

SNI Preservation

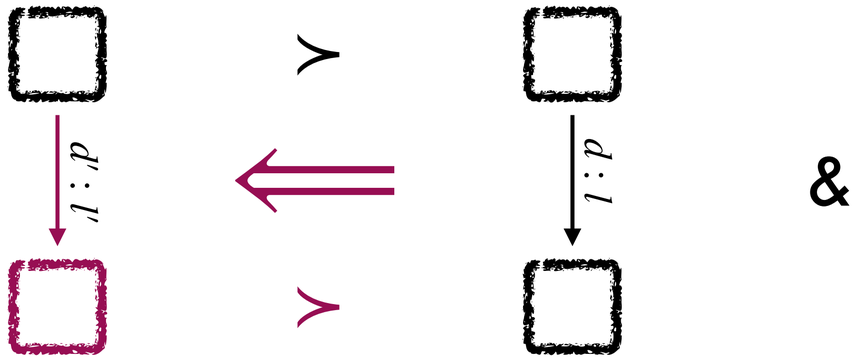
Theorem *If* $[\,.\,]$ *has a simulation* $\succ$ *between any* P *and* $[P]$, *so that*

&

then

SNI Preservation

Theorem *If* $[\,.\,]$ *has a simulation* $\succ$ *between any* P *and* $[P]$*, so that*

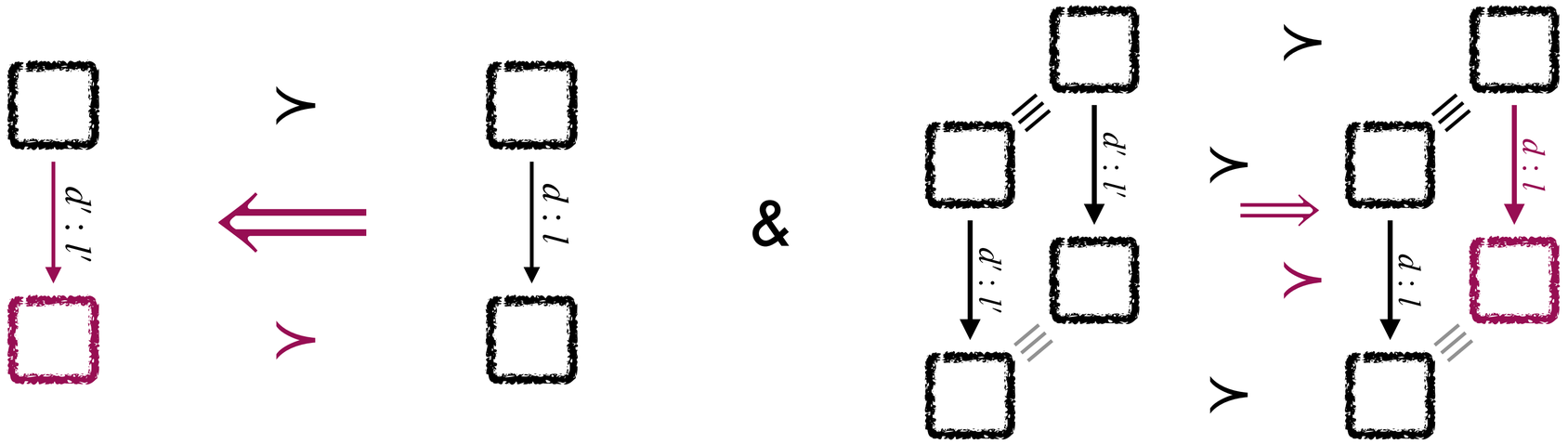


&

then

SNI Preservation

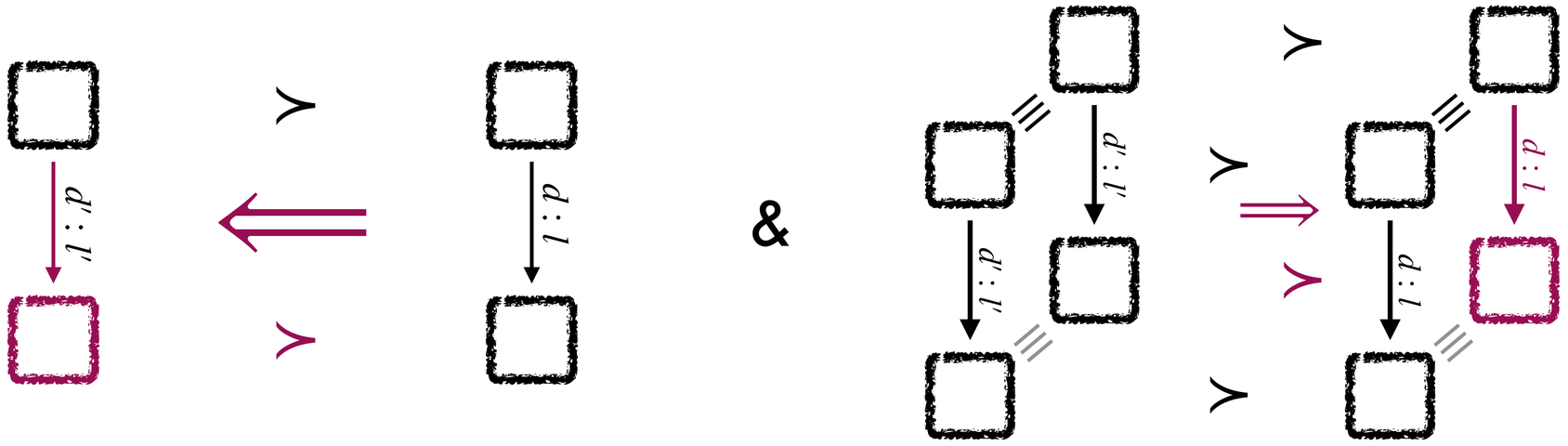
Theorem *If* $[\,.\,]$ *has a simulation* $\succ$ *between any* P *and* $[P]$*, so that*



then

SNI Preservation

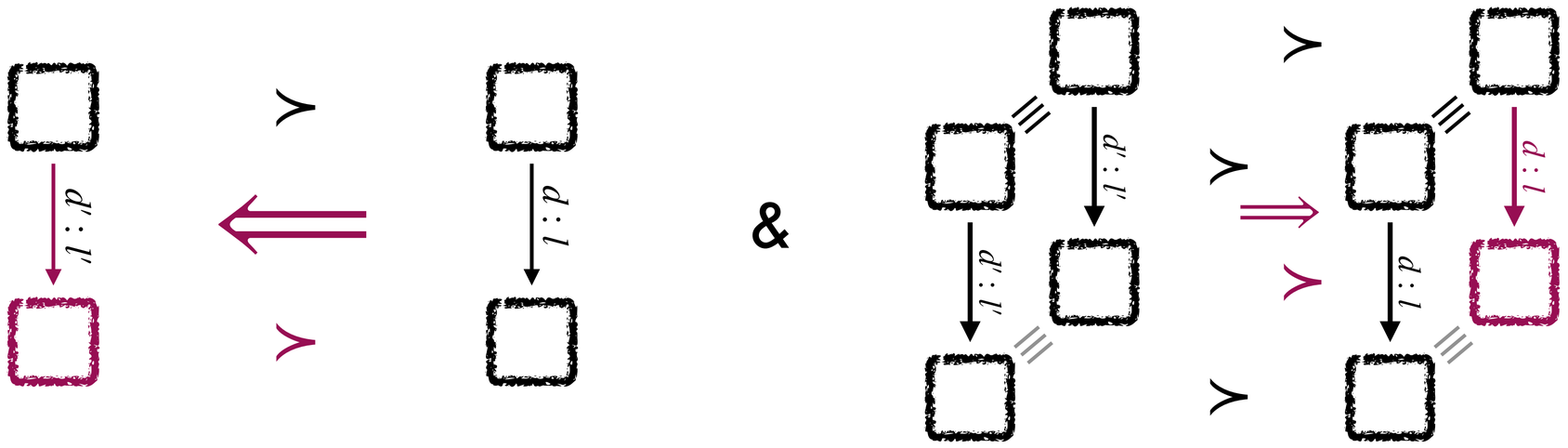
Theorem *If* $[\,.\,]$ *has a simulation* $\succ$ *between any* P *and* $[P]$*, so that*



then $[\,.\,] \vDash$ **SNIP**

SNI Preservation

Theorem *If* $[\,.\,]$ *has a simulation* $\succ$ *between any* P *and* $[P]$*, so that*

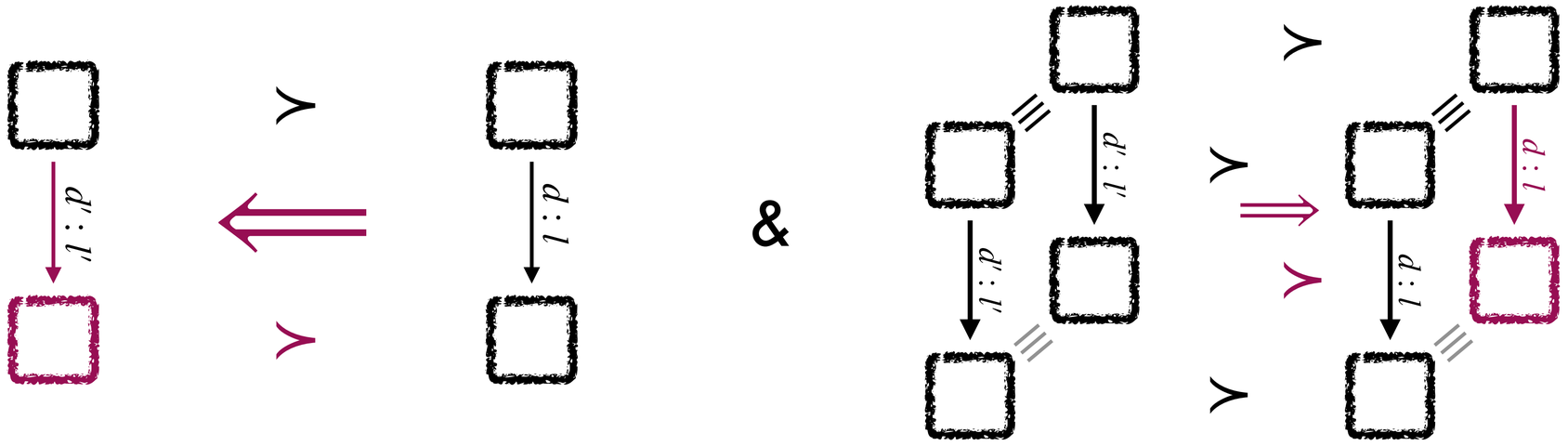


$P \models$ SNI
Not needed!

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

SNI Preservation

Theorem *If* $[\,.\,]$ *has a simulation* $\succ$ *between any* P *and* $[P]$*, so that*



$P \models$ SNI
Not needed!

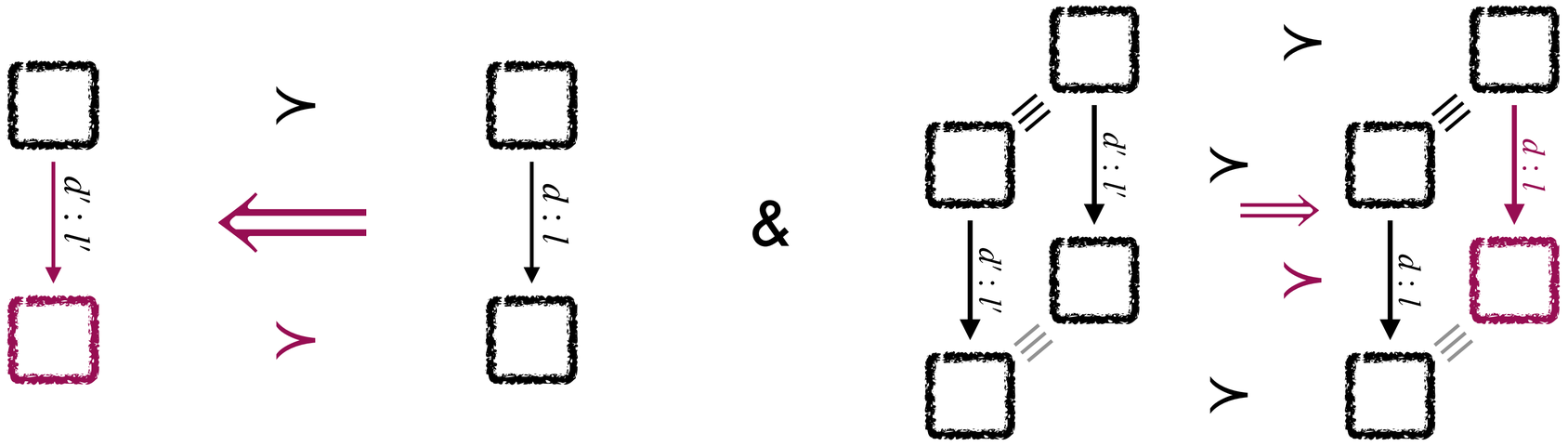
then $[\,.\,] \models$ **SNIP**

Usually, $\succ$ is parametric in P

SNI Preservation

Proof Effort

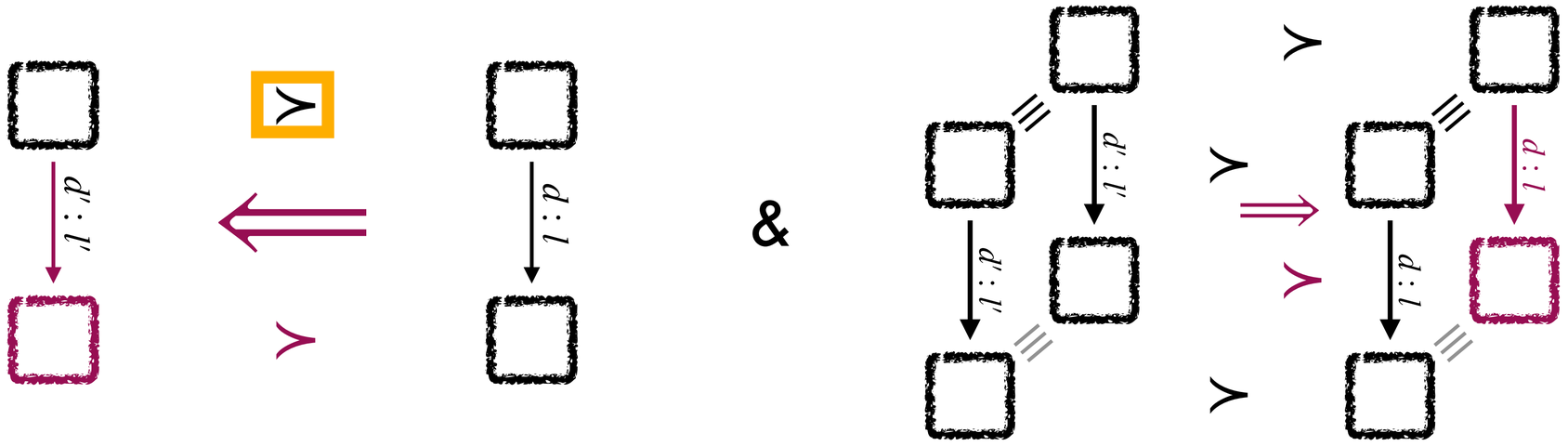
Theorem *If* $[\,.\,]$ *has a simulation* $\succ$ *between any* P *and* $[P]$*, so that*



SNI Preservation

Proof Effort

Theorem *If* $[\,.\,]$ *has a simulation* $\succ$ *between any* P *and* $[P]$*, so that*

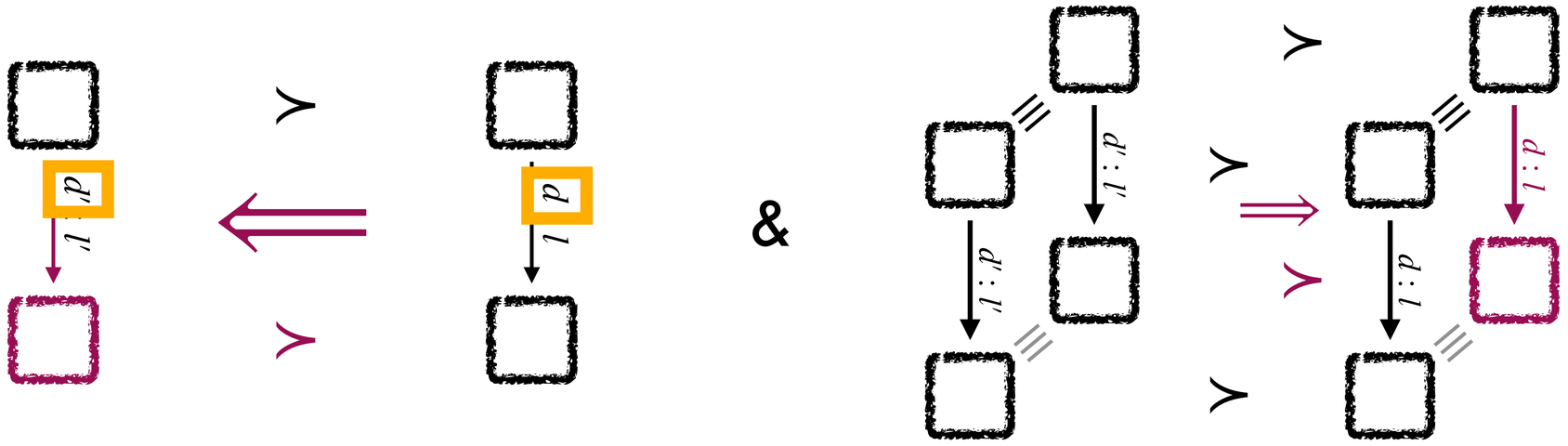


Define $\succ$

SNI Preservation

Proof Effort

Theorem *If* $[\,.\,]$ *has a simulation* $\succ$ *between any* P *and* $[P]$*, so that*



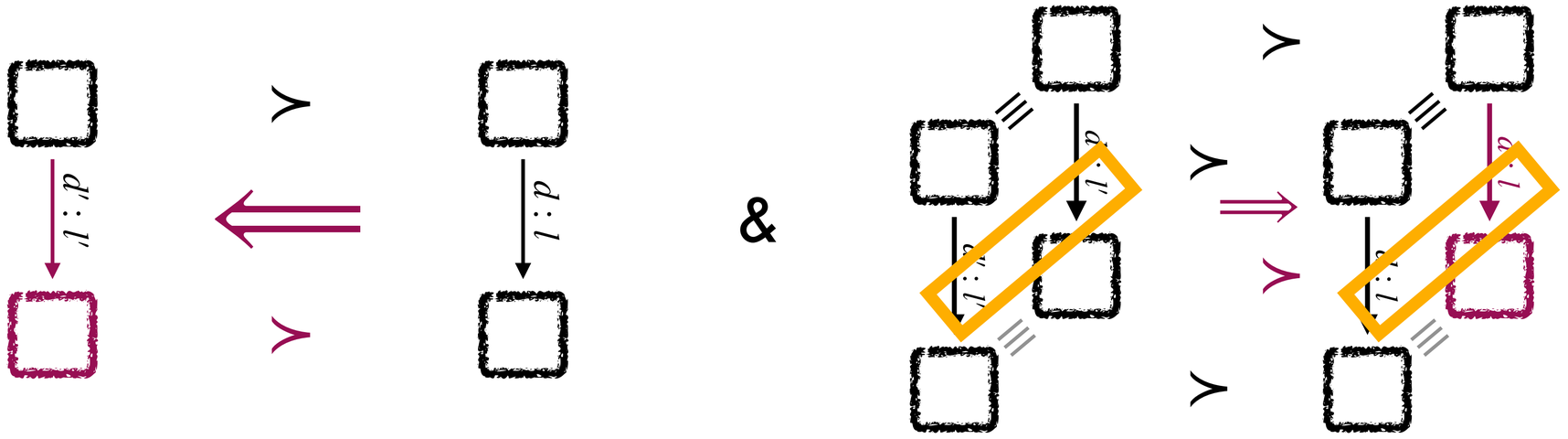
Define $\succ$

Translate Directives

SNI Preservation

Proof Effort

Theorem *If* $[\,.\,]$ *has a simulation* $\succ$ *between any* P *and* $[P]$*, so that*

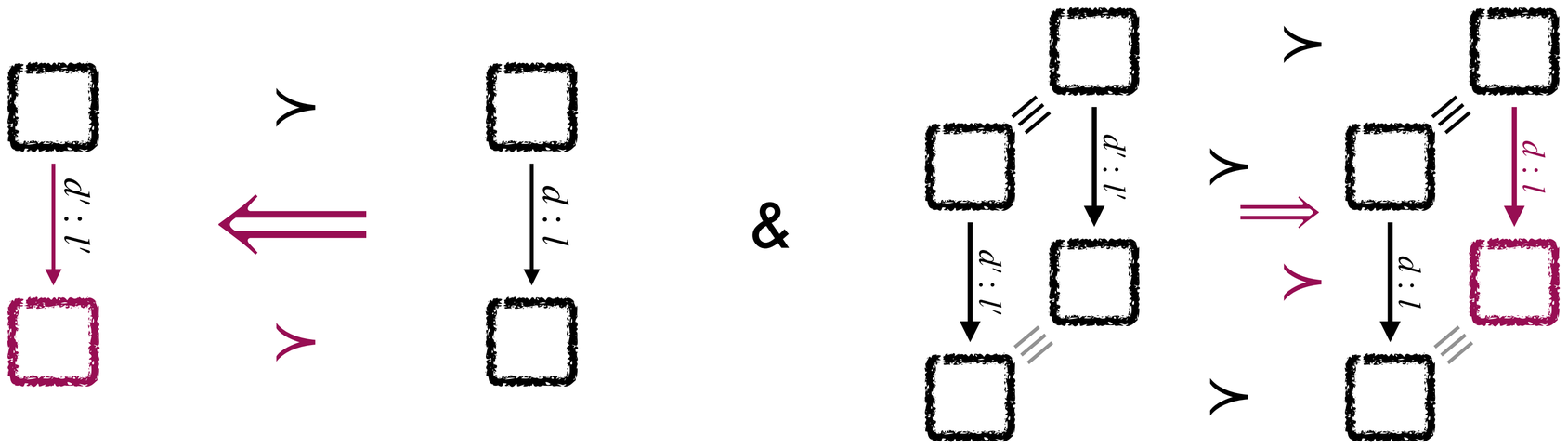


Define $\succ$ | Translate Directives | Equal Source Leakage → Equal Target Leakage

SNI Preservation

Proof Effort

Theorem *If* $[\,.\,]$ *has a simulation* $\succ$ *between any* P *and* $[P]$*, so that*



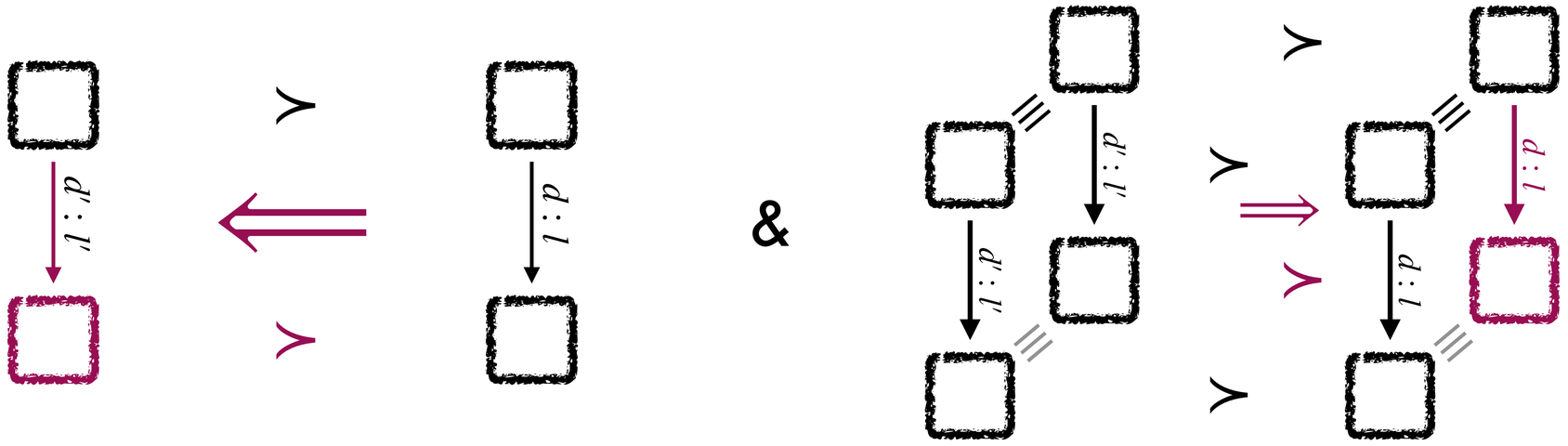
	Define $\succ$	Translate Directives	Equal Source Leakage → Equal Target Leakage
DeadCode	✓	✓	✓

$[\,.\,]_{dc} \vDash$ SNIP

SNI Preservation

Proof Effort

Theorem *If* $[\,.\,]$ *has a simulation* $\succ$ *between any* P *and* $[P]$*, so that*



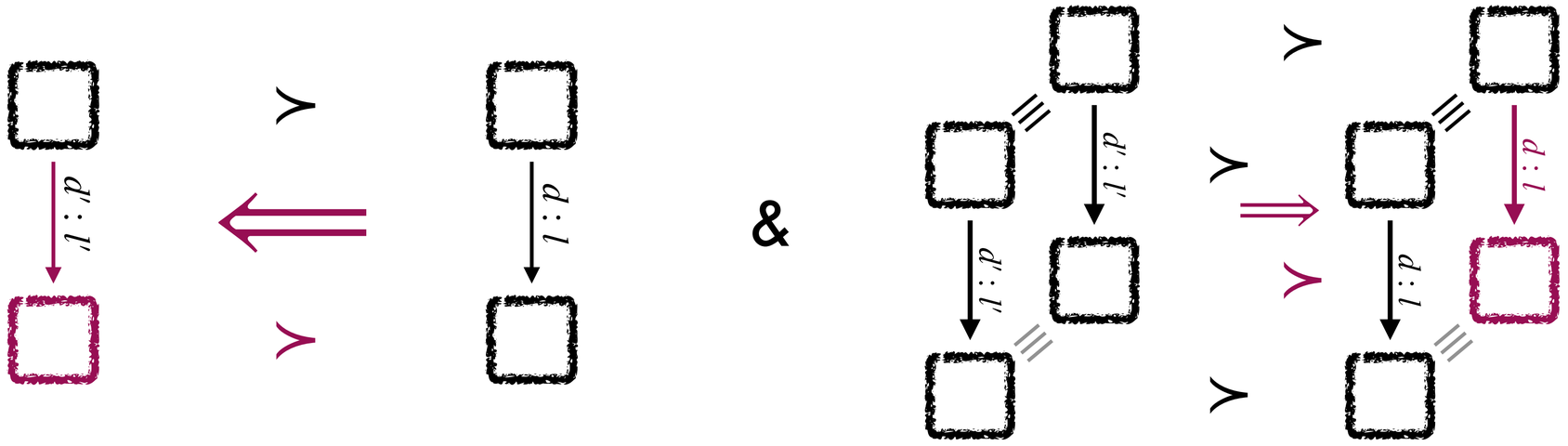
	Define $\succ$	Translate Directives	Equal Source Leakage → Equal Target Leakage
DeadCode	✓	✓	✓
RegAlloc	(✓)	✗	✗

$[\,.\,]_{dc} \vDash$ SNIP

SNI Preservation

Proof Effort

Theorem *If* $[\,.\,]$ *has a simulation* $\succ$ *between any* P *and* $[P]$*, so that*



	Define $\succ$	Translate Directives	Equal Source Leakage → Equal Target Leakage	
DeadCode	✓	✓	✓	$[\,.\,]_{dc} \models$ SNIP
RegAlloc	(✓)	✗	✗	$[\,.\,]_{ra} \not\models$ SNIP

Goals

How do we prove

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

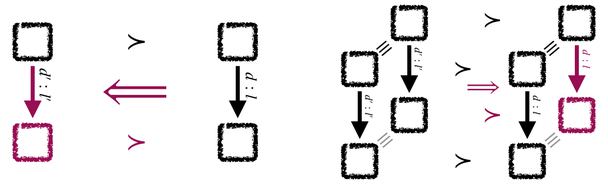
Can we fix Register Allocation so that

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

Goals

How do we prove

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

$\succ$ 

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

Can we fix Register Allocation so that

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

Goals

How do we prove

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

then $[\,\cdot\,] \models \mathbf{SNIP}$

Can we fix Register Allocation so that

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

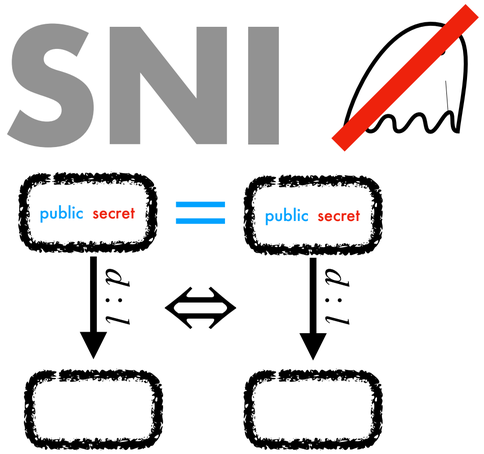
PriSC

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

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

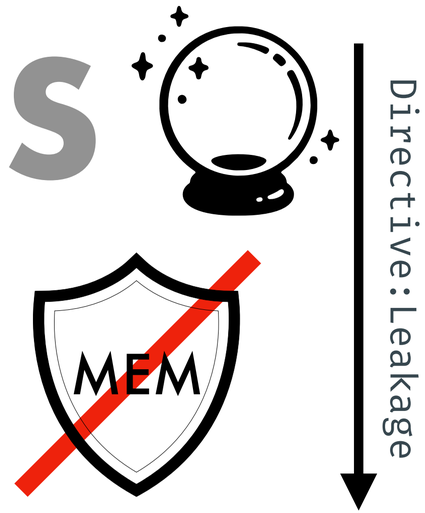
SNIP: Speculative Execution and Non-Interference Preservation for Compiler Transformations

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



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

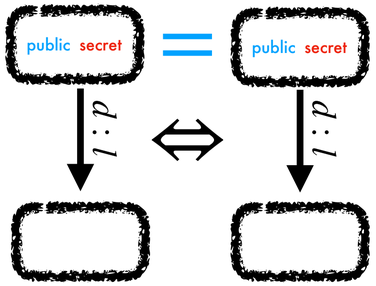
s.van-der-wall@tu-bs.de



SNIP

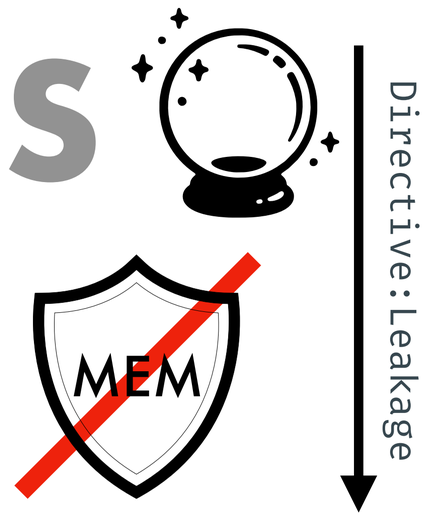
$$P \vDash \textbf{SNI}_{\not s} \implies [P] \vDash \textbf{SNI}_{\not s}$$

SNI



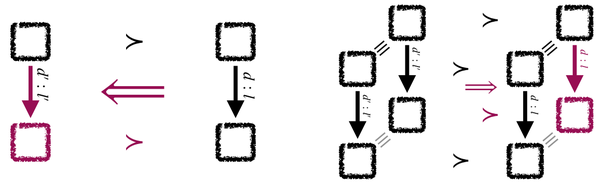
SNIP: Speculative Execution and Non-Interference Preservation for Compiler Transformations

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



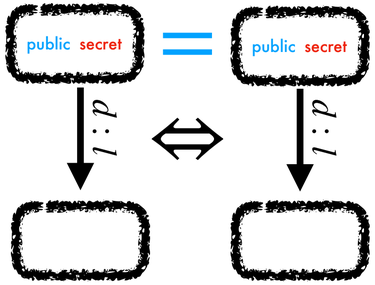
SNIP

$$P \models \text{SNI}_{\not\lightning} \implies [P] \models \text{SNI}_{\not\lightning}$$

$\succ$ 

then $[\,.\,] \models \text{SNIP}$

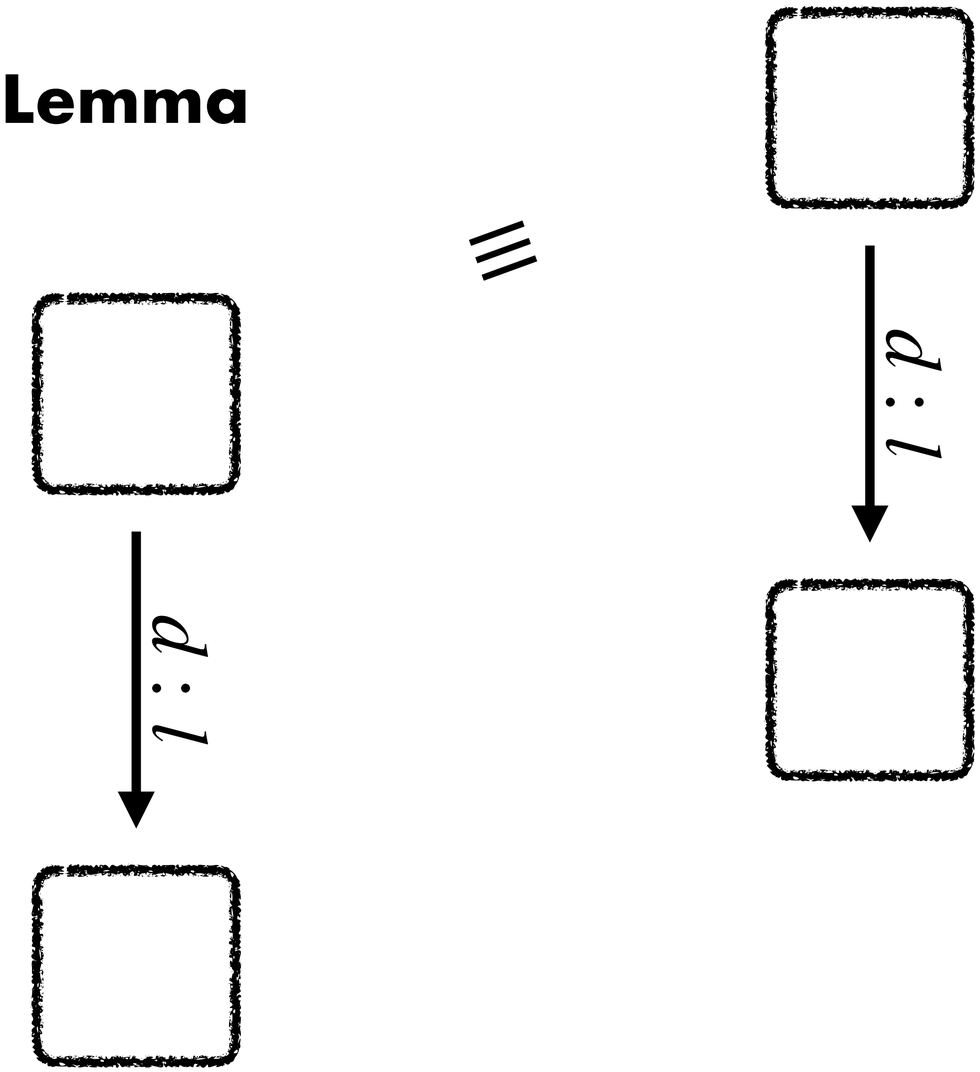
SNI $_{\not\lightning}$



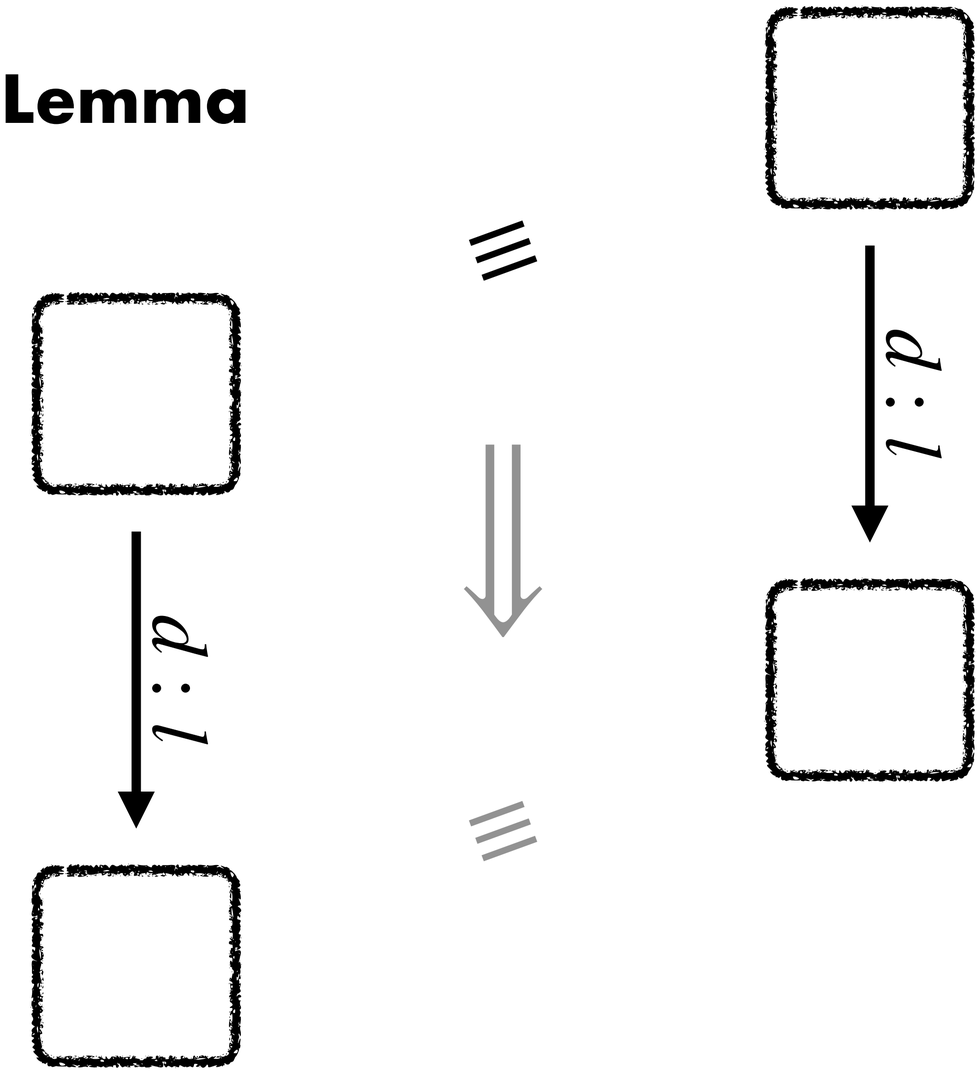
SNIP: Speculative Execution and Non-Interference Preservation for Compiler Transformations

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

Lemma



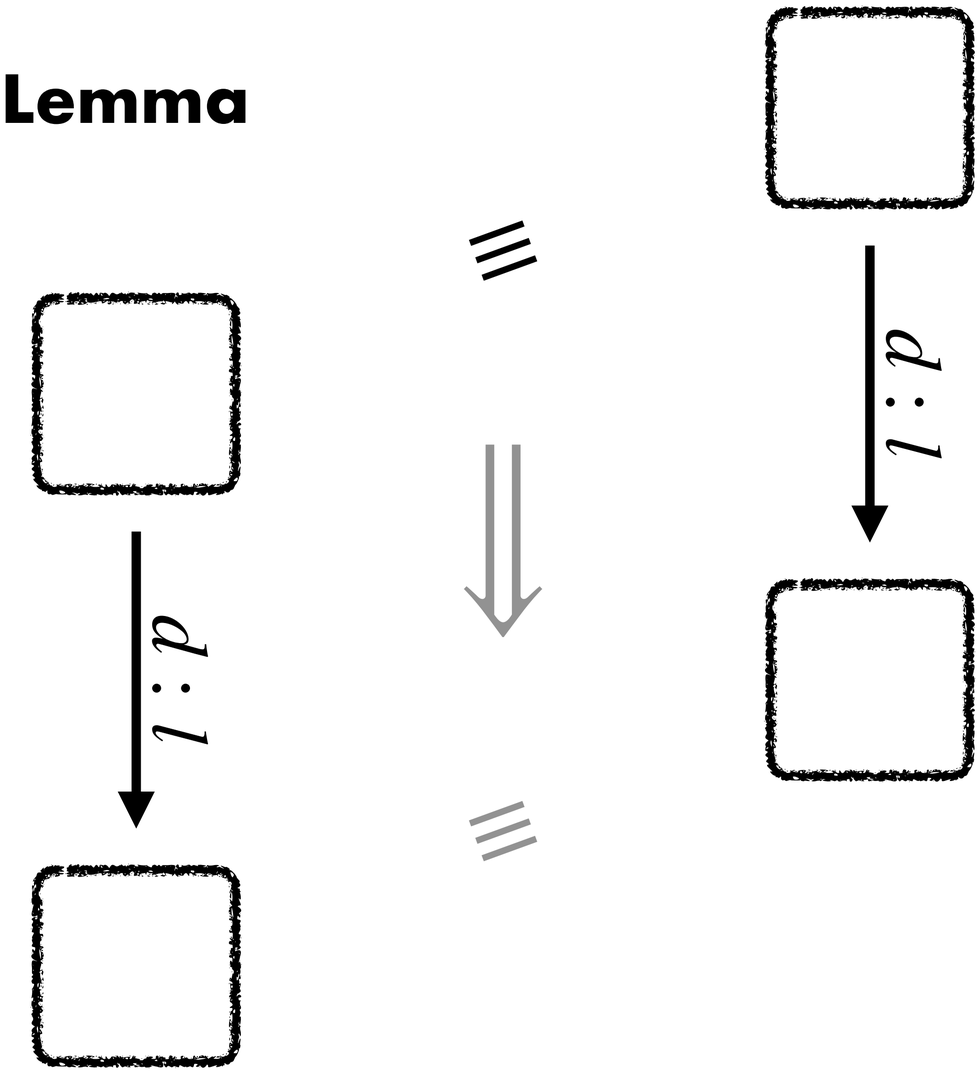
Lemma



Directive + Leakage determines program counter!

Method

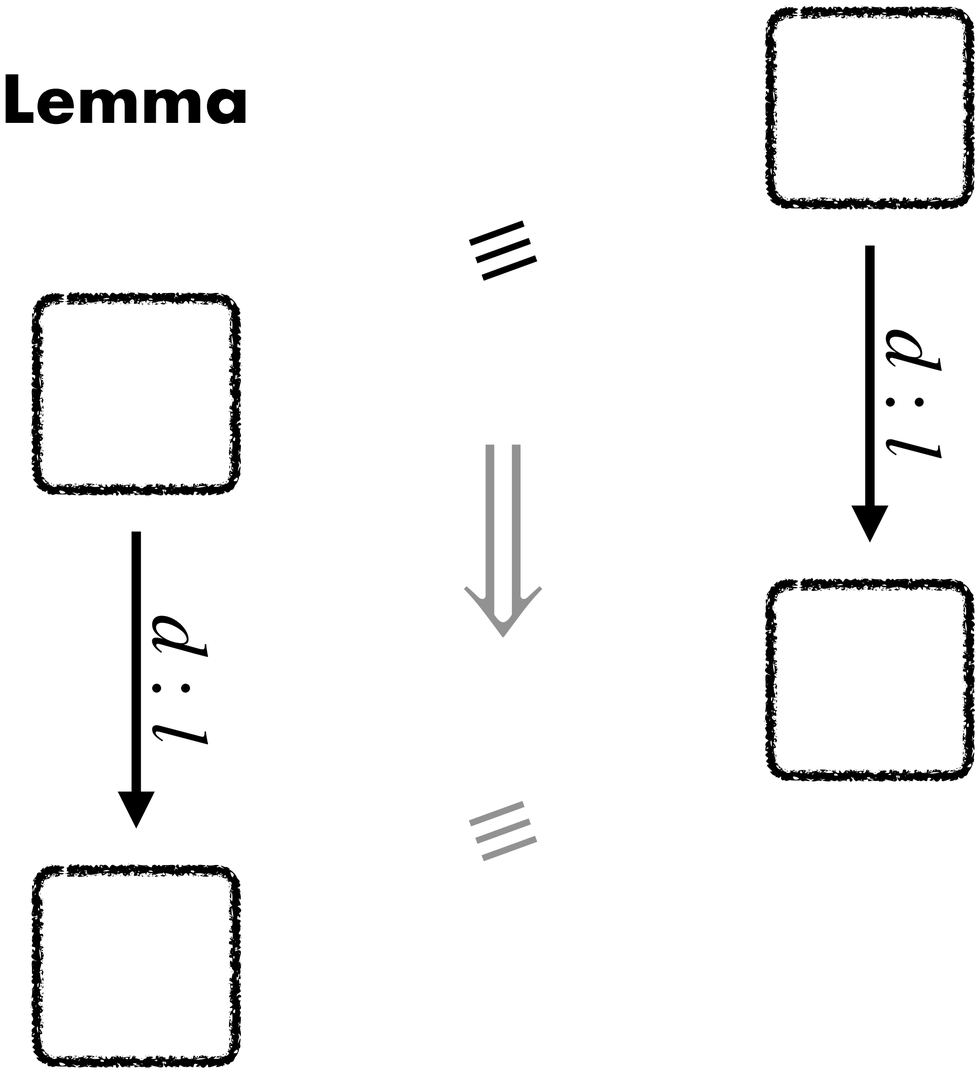
Lemma



Reason:

```
if (i < size)
```

Lemma

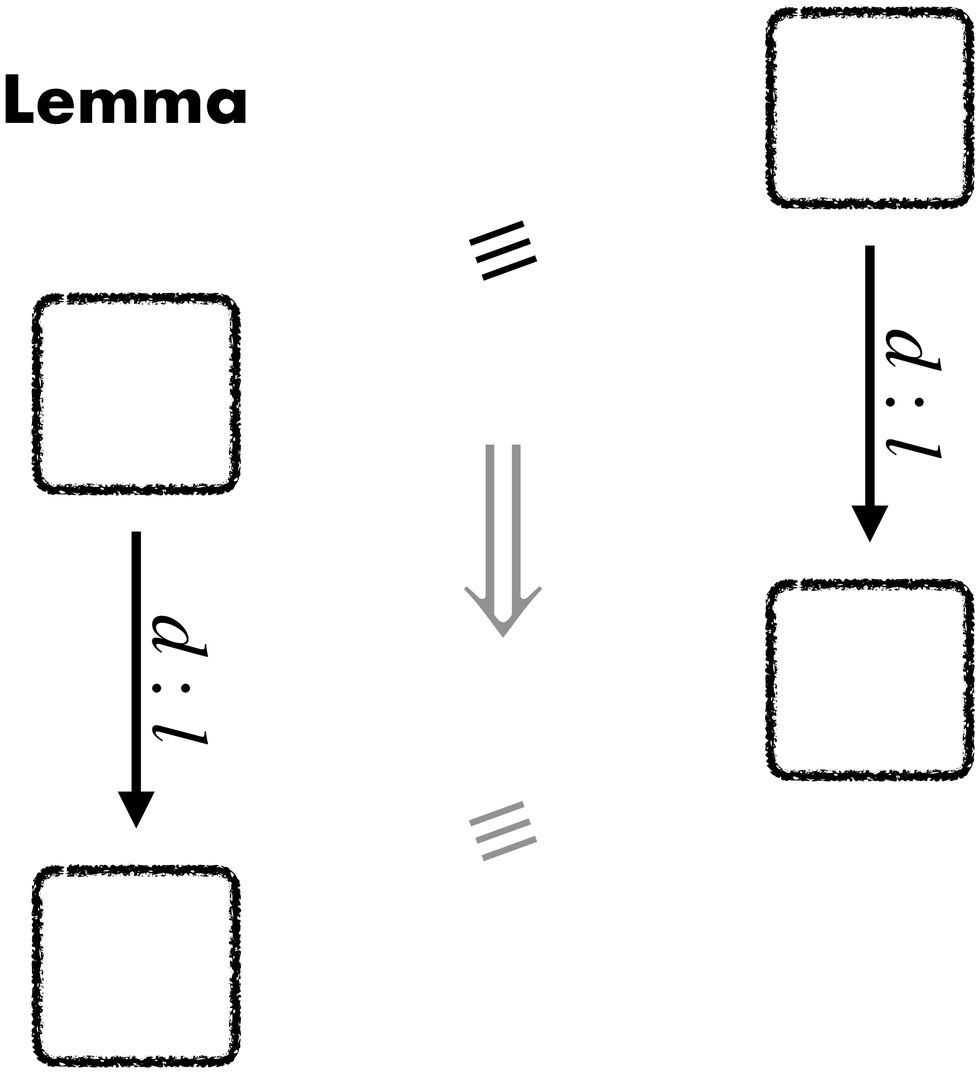


Reason:

```
if (i < size)
```

Directive	Leakage
`miss`	`BR false`

Lemma



Reason:

```
if (i < size)
```

Directive: `miss`

Leakage: `BR false`

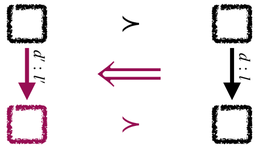
Combination determines Program Counter!

P

```
a = buf[i];
a = 0;
```

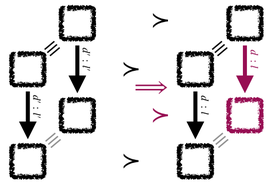
$[P]_{dc}$

```
nop;
a = 0;
```



Define $\prec$

Translate Directives



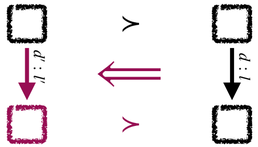
Equal Source Leakage → Equal Target Leakage

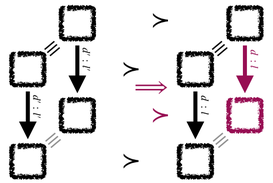
P

```
a = buf[i];
a = 0;
```

$[P]_{dc}$

```
nop;
a = 0;
```

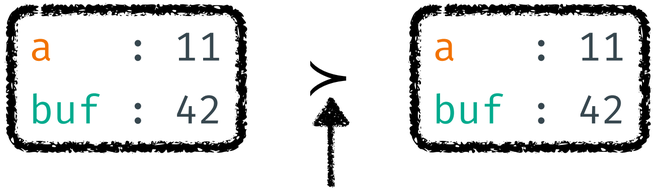




Define $\succ$

Translate Directives

Equal Source Leakage → Equal Target Leakage



Equal up to
dead locations

P

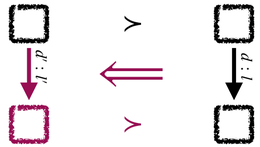
```
a = buf[i];
a = 0;
```

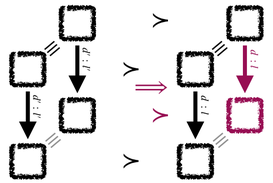
d'

$[P]_{dc}$

d

```
nop;
a = 0;
```



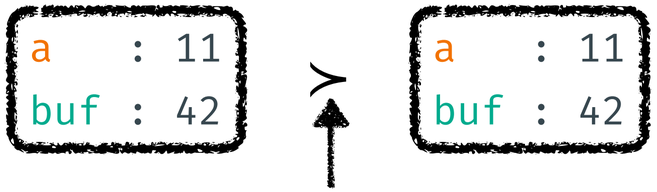


Define $\succ$

Translate Directives

Equal Source Leakage → Equal Target Leakage

d' d



Equal up to
dead locations

P

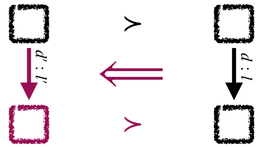
```
a = buf[i];
a = 0;
```

d'

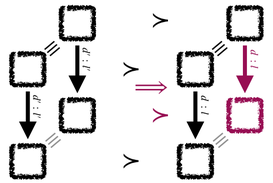
$[P]_{dc}$

d

```
nop;
a = 0;
```

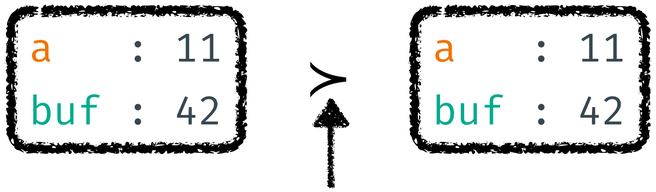


Translate Directives



Equal Source Leakage → Equal Target Leakage

Define $\succ$



Equal up to dead locations

d' d

```
a=buf[i];

a=b+c;

buf[i]=a;
```

P

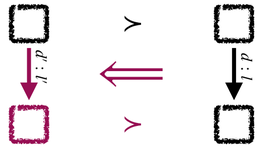
```
a = buf[i];
a = 0;
```

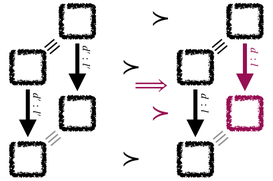
d'

$[P]_{dc}$

d

```
nop;
a = 0;
```

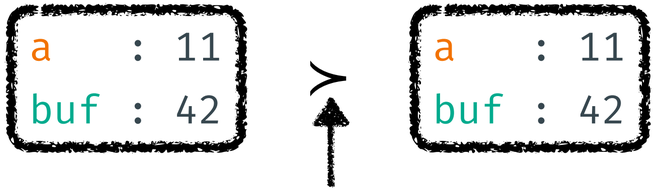




Define $\succ$

Translate Directives

Equal Source Leakage → Equal Target Leakage



Equal up to dead locations

d' d

```
a=buf[i];  step / oob  <-- step nop;
a=b+c;
buf[i]=a;
```

DeadCode

P

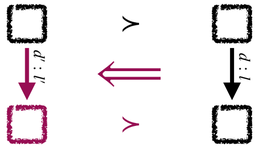
```
a = buf[i];
a = 0;
```

d'

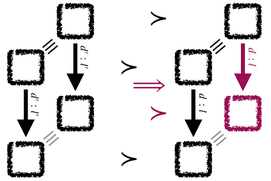
$[P]_{dc}$

d

```
nop;
a = 0;
```

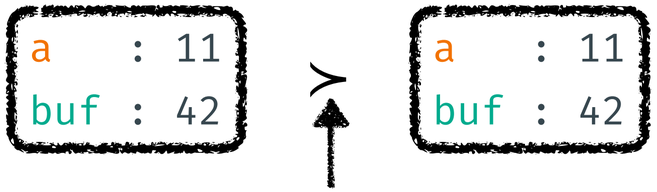


Define $\succ$



Translate Directives

Equal Source Leakage → Equal Target Leakage



Equal up to dead locations

	d'		d
`a=buf[i];`	`step` `oob`	◀--	`step nop;`
`a=b+c;`	`step`	◀--	`step nop;`
`buf[i]=a;`	`step` `oob`	◀--	`step nop;`

DeadCode

P

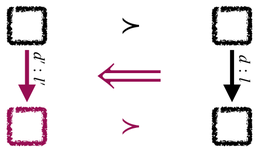
```
a = buf[i];
a = 0;
```

d'

$[P]_{dc}$

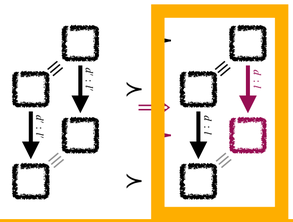
d

```
nop;
a = 0;
```

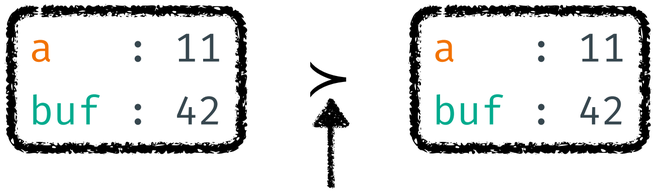


Define $\succ$

Translate Directives



Equal Source Leakage → Equal Target Leakage



Equal up to dead locations

d'		d
`a=buf[i];` step oob	◀--	`step nop;`
`a=b+c;` step	◀--	`step nop;`
`buf[i]=a;` step oob	◀--	`step nop;`

DeadCode

P

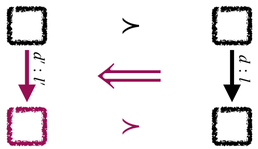
```
a = buf[i];
a = 0;
```

d'

$[P]_{dc}$

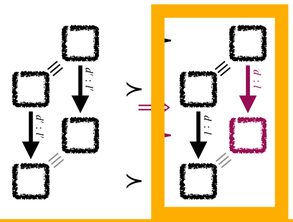
d `nop;`

```
a = 0;
```

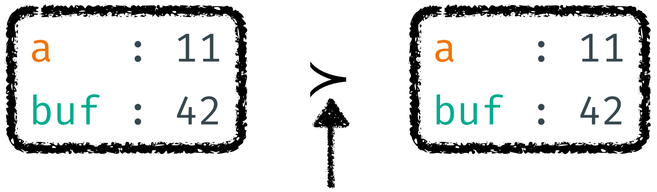


Define $\succ$

Translate Directives

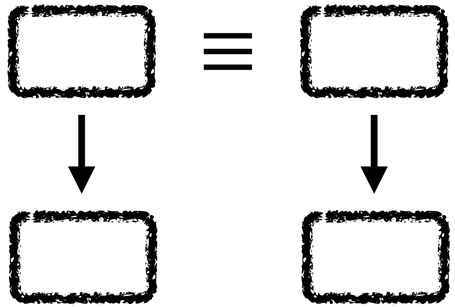


Equal Source Leakage → Equal Target Leakage



Equal up to dead locations

d'		d
`a=buf[i];`	step oob ◀--	`step nop;`
`a=b+c;`	step ◀--	`step nop;`
`buf[i]=a;`	step oob ◀--	`step nop;`



P

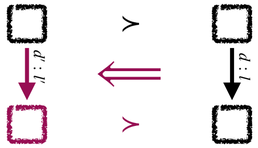
```
a = buf[i];
a = 0;
```

d'

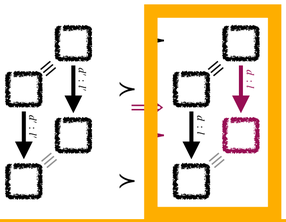
$[P]_{dc}$

d `nop;`

```
a = 0;
```

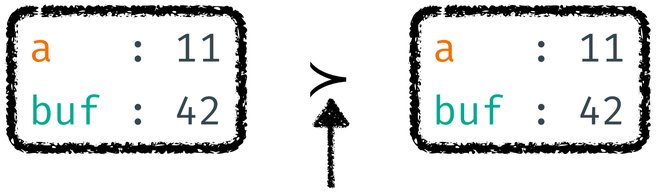


Define $\succ$



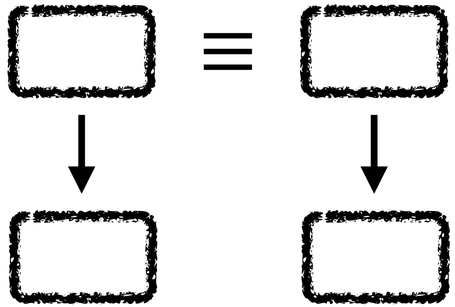
Translate Directives

Equal Source Leakage → Equal Target Leakage



Equal up to dead locations

d'		d
`a=buf[i];`	`step` `oob` ◀--	`step nop;`
`a=b+c;`	`step` ◀--	`step nop;`
`buf[i]=a;`	`step` `oob` ◀--	`step nop;`



DeadCode

P

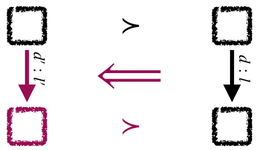
```
a = buf[i];
a = 0;
```

d'

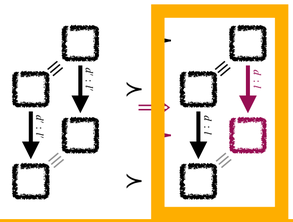
$[P]_{dc}$

d

```
nop;
a = 0;
```

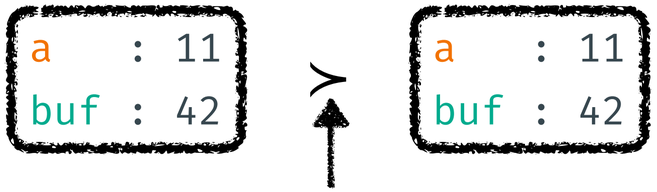


Translate Directives



Equal Source Leakage → Equal Target Leakage

Define $\succ$



Equal up to dead locations

d'		d
`a=buf[i];`	`step` `oob` ◀--	`step nop;`
`a=b+c;`	`step` ◀--	`step nop;`
`buf[i]=a;`	`step` `oob` ◀--	`step nop;`