

Quantitative Static Timing Analysis

Denis Mazzucato, Marco Campion, and Caterina Urban

21st October 2024

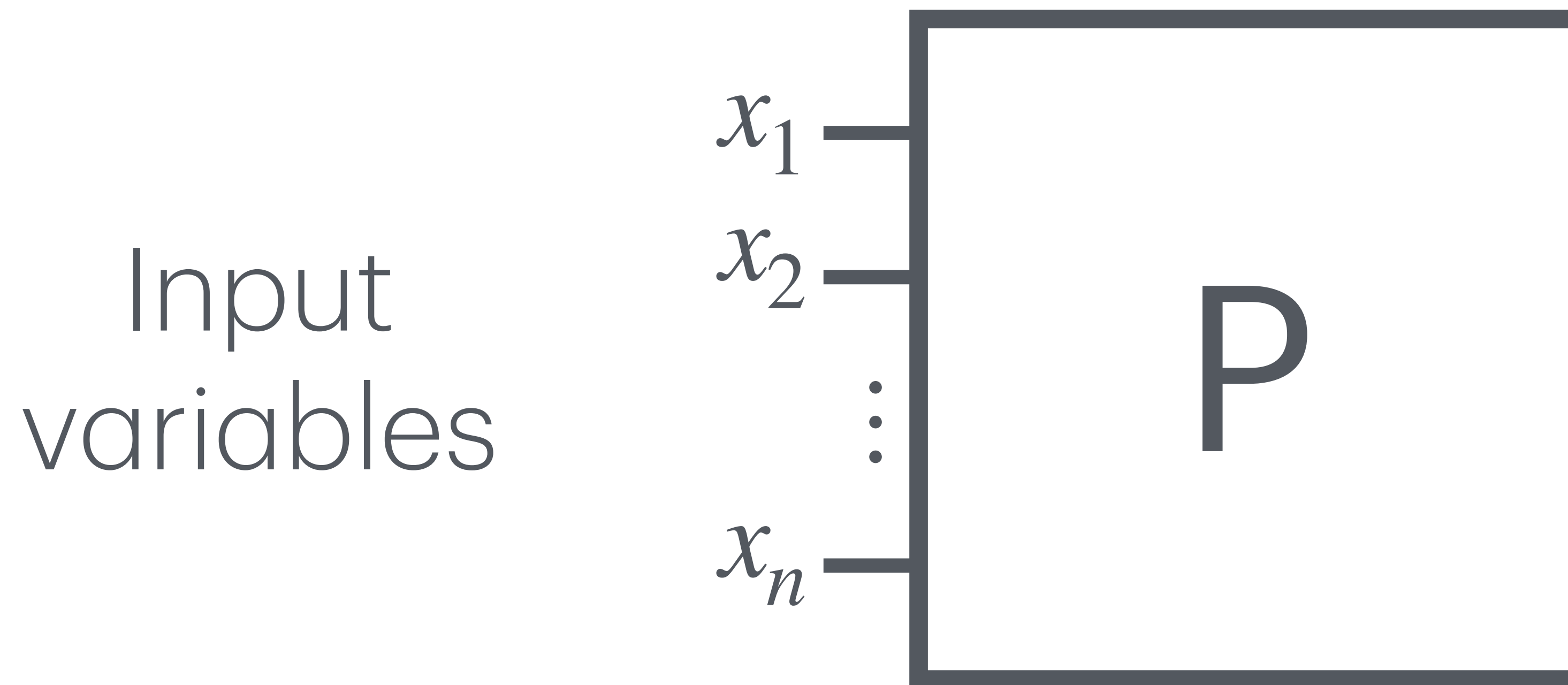


Quantifying the impact of input variables on the number of iterations of a program

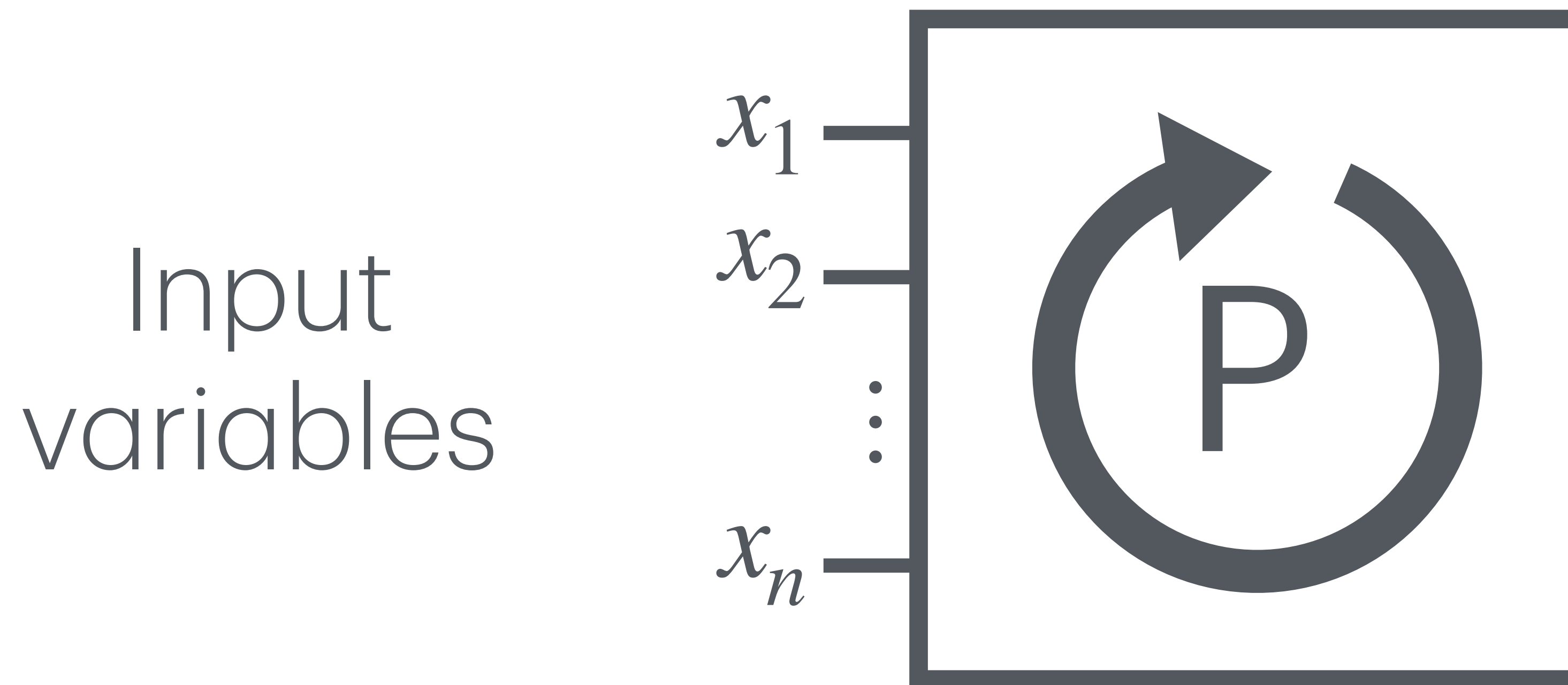
Quantifying the impact of input variables on the number of iterations of a program



Quantifying the impact of input variables on the number of iterations of a program

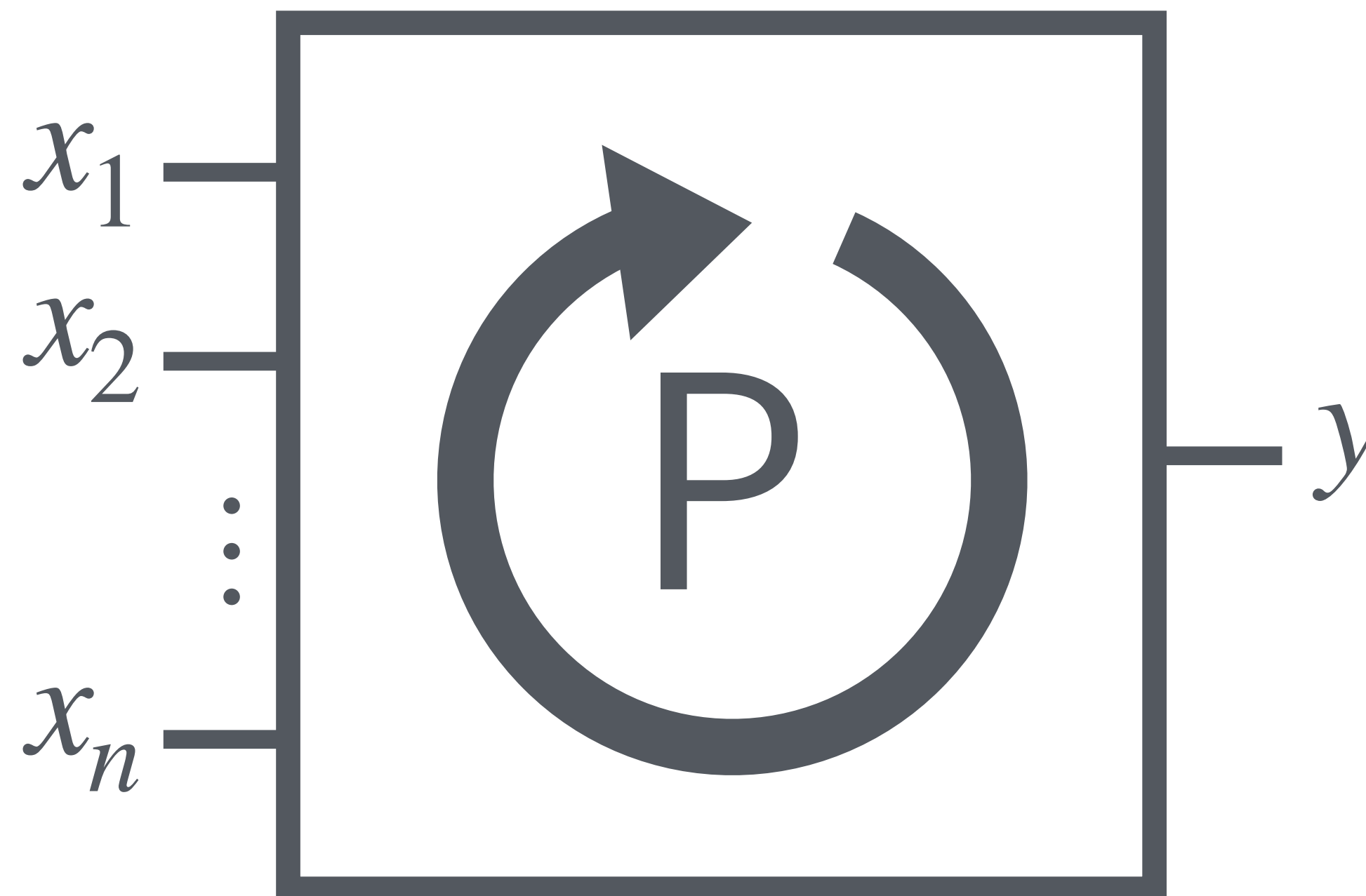


Quantifying the impact of input variables on the number of iterations of a program



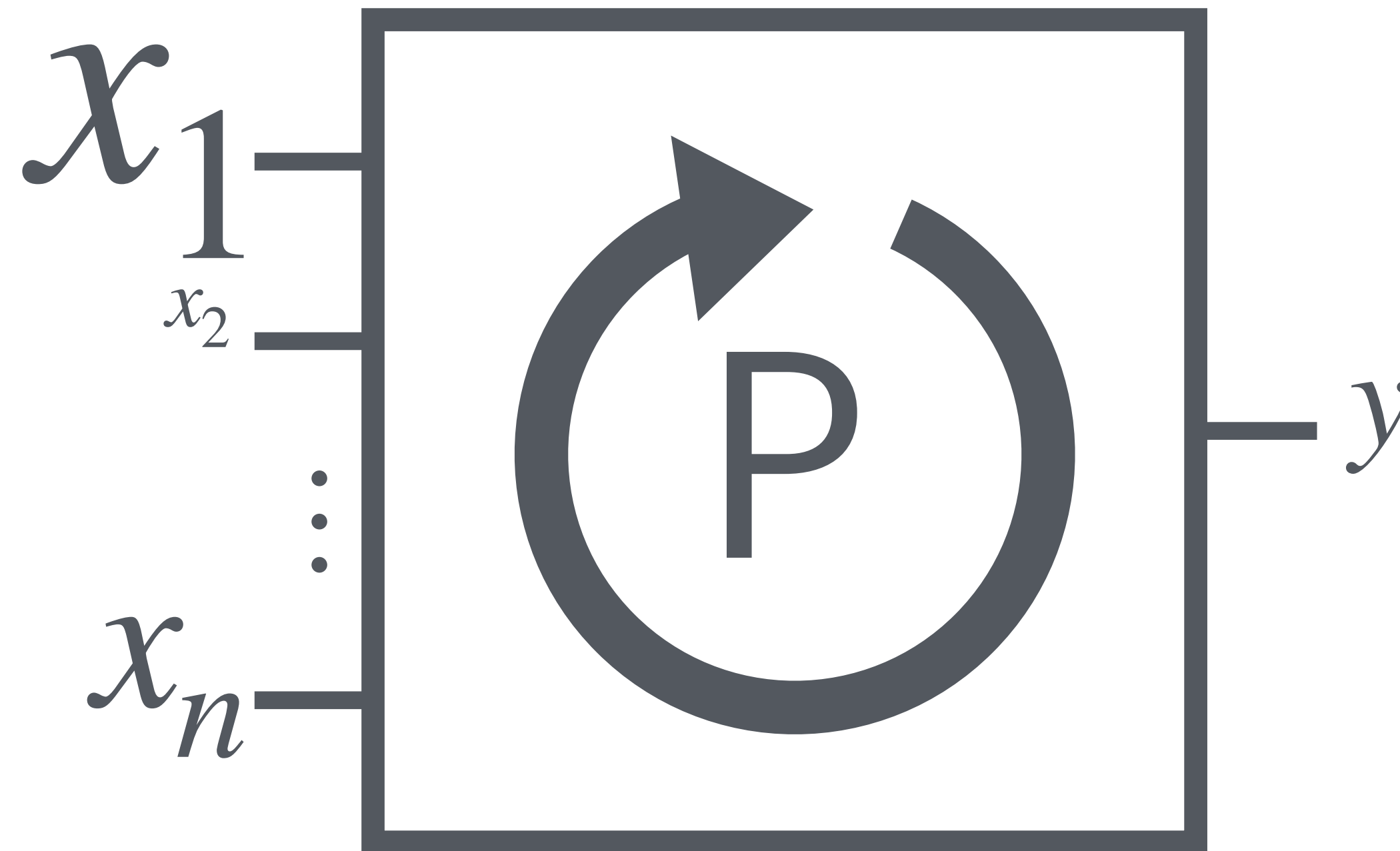
Quantifying the impact of input variables on the number of iterations of a program

Input
variables

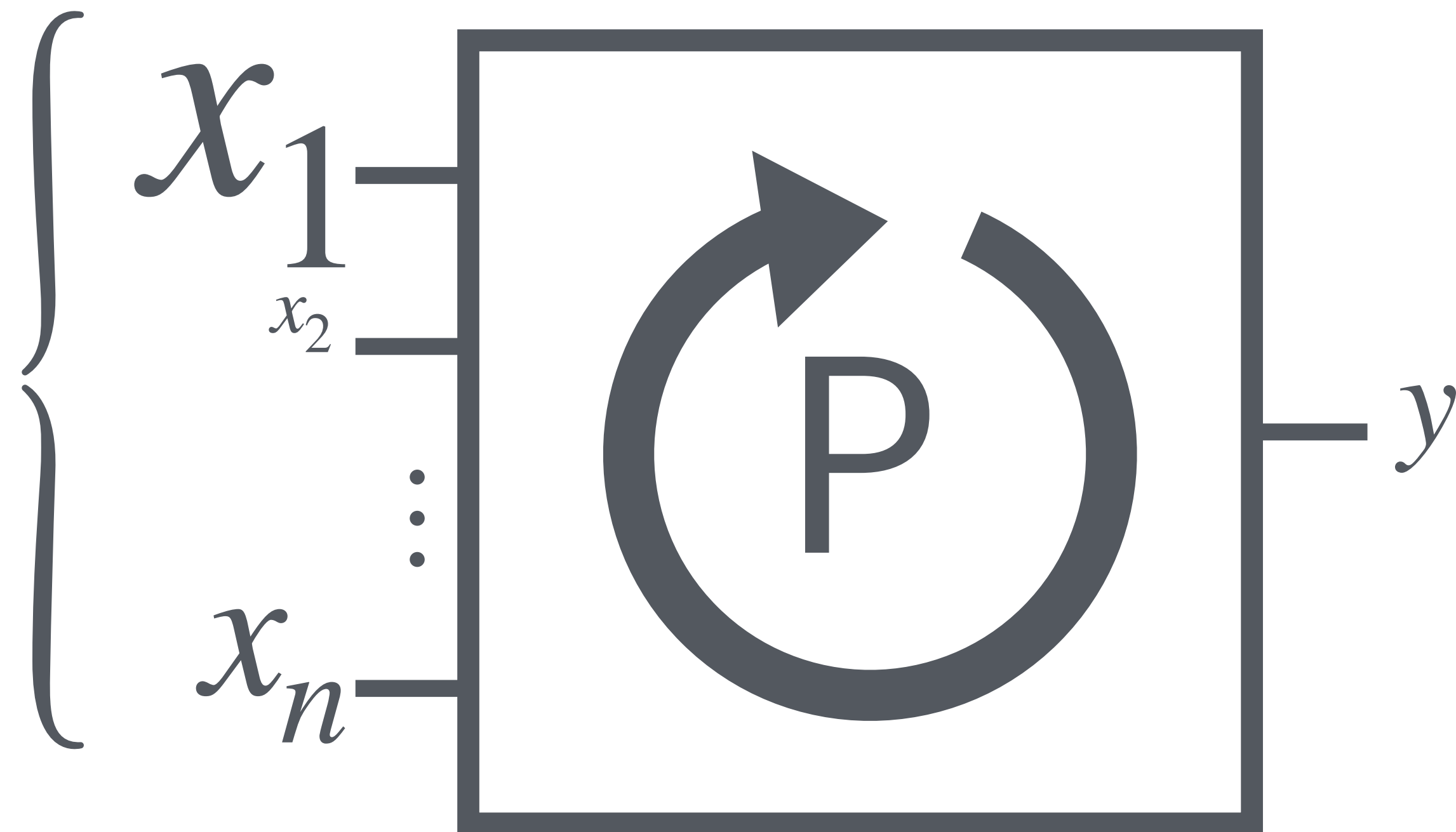


Quantifying the impact of input variables on the number of iterations of a program

How much
impact on
loop iterations?

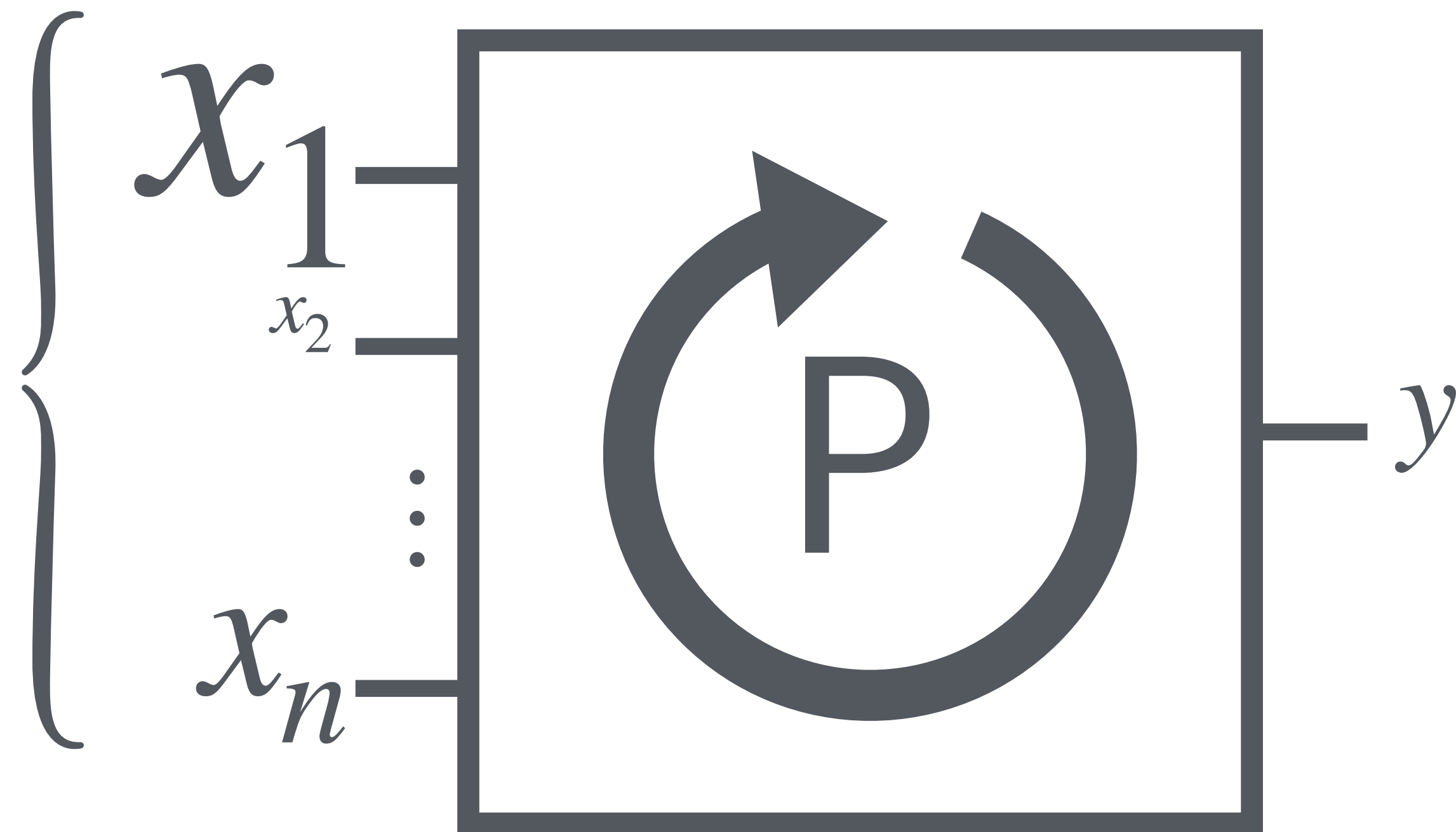


Why?



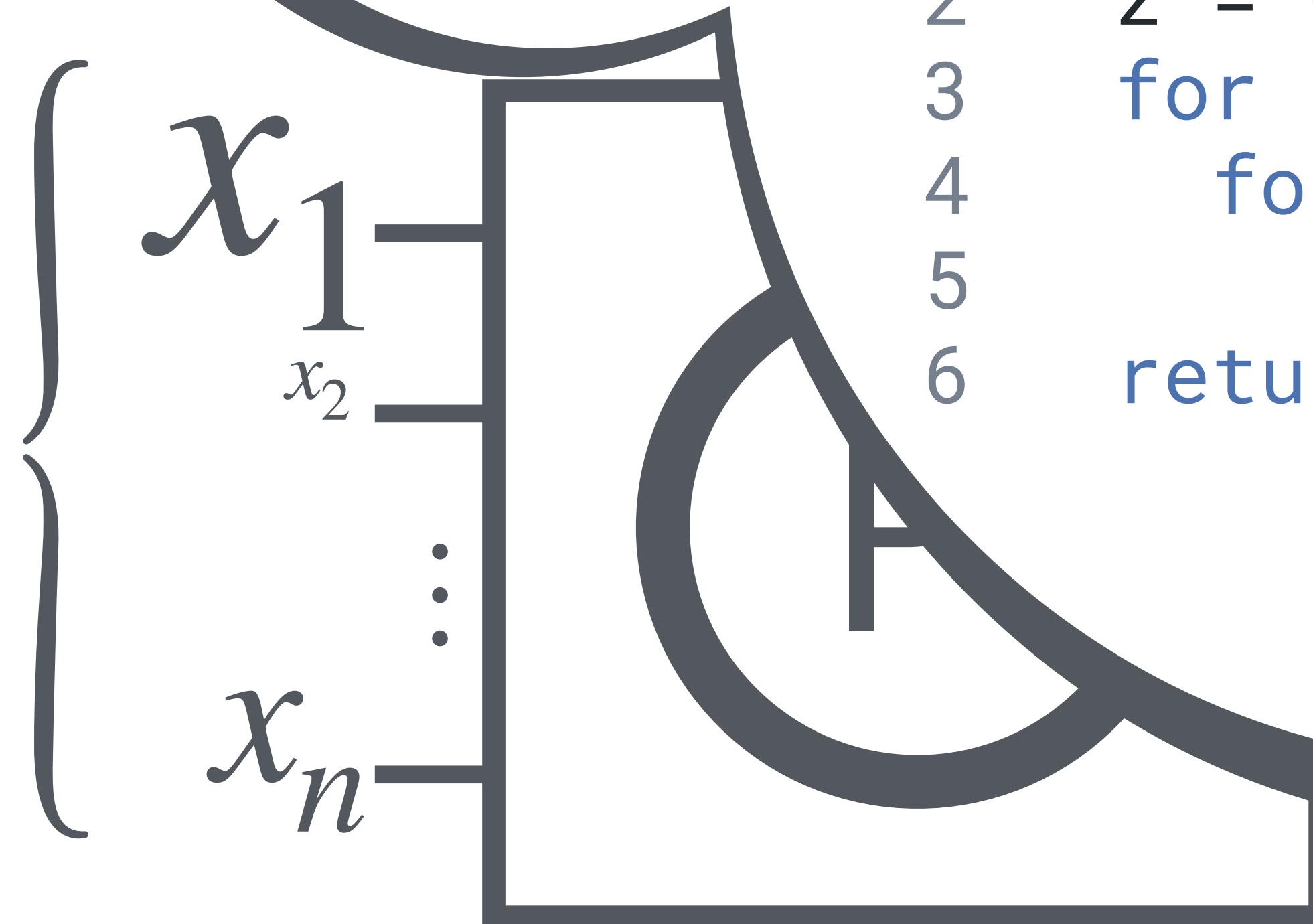
Why?

- Correct Loop Behaviour



Why?

- Correct Loop Behaviour



```
1 def Mul(x, y):  
2     z = 0  
3     for (; x > 0; x--):  
4         for (; y > 0; y--):  
5             z++  
6     return z
```

Why?

- Correct Loop Behaviour

$\text{IMPACT}_x(\text{Mul})$

```
1 def Mul(x, y):  
2     z = 0  
3     for (; x > 0; x--):  
4         for (; y > 0; y--):  
5             z++  
6     return z
```

x_1
 x_2

⋮

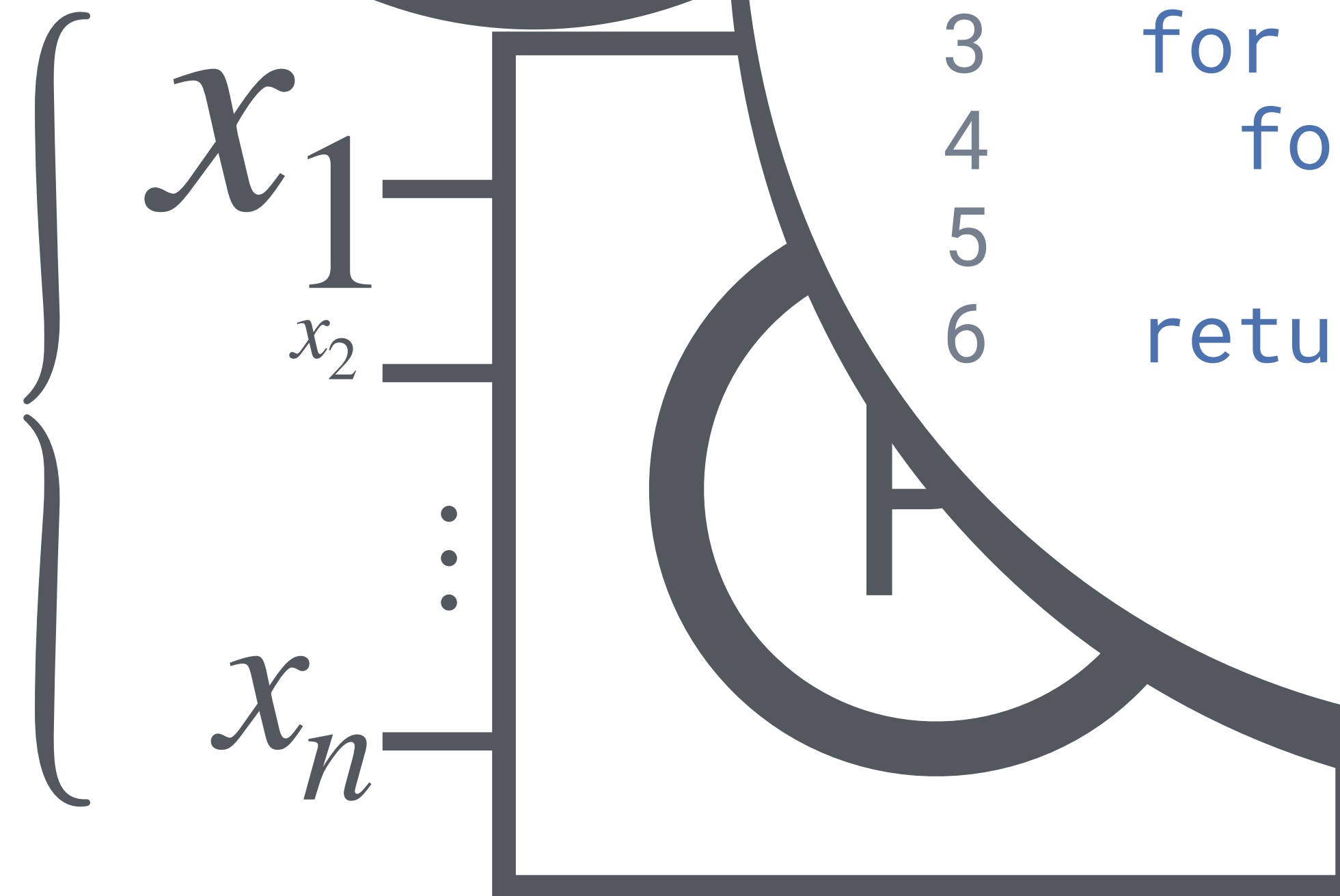
x_n

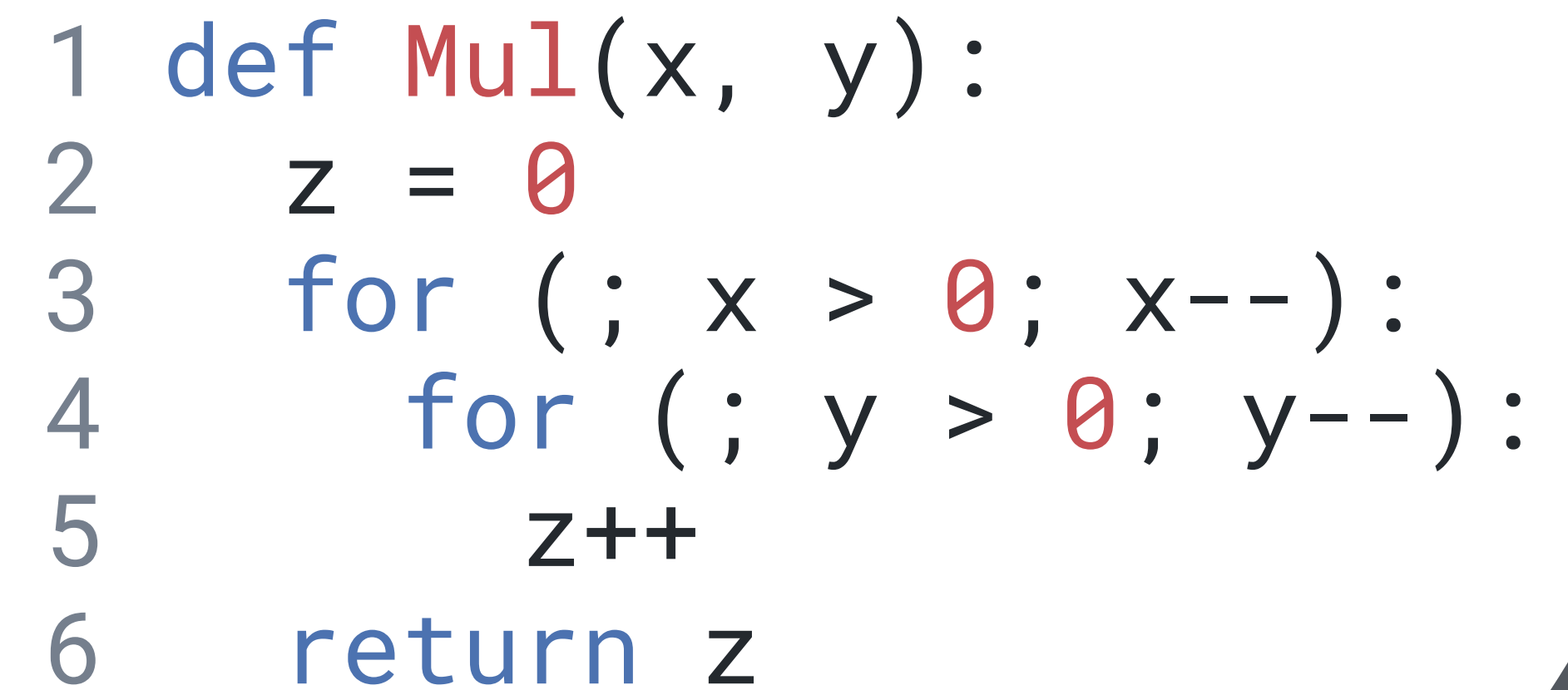
Why?

- Correct Loop Behaviour

$$\text{IMPACT}_x(\text{Mul}) = 2 \cdot \text{IMPACT}_y(\text{Mul})$$

```
1 def Mul(x, y):  
2     z = 0  
3     for (; x > 0; x--):  
4         for (; y > 0; y--):  
5             z++  
6     return z
```

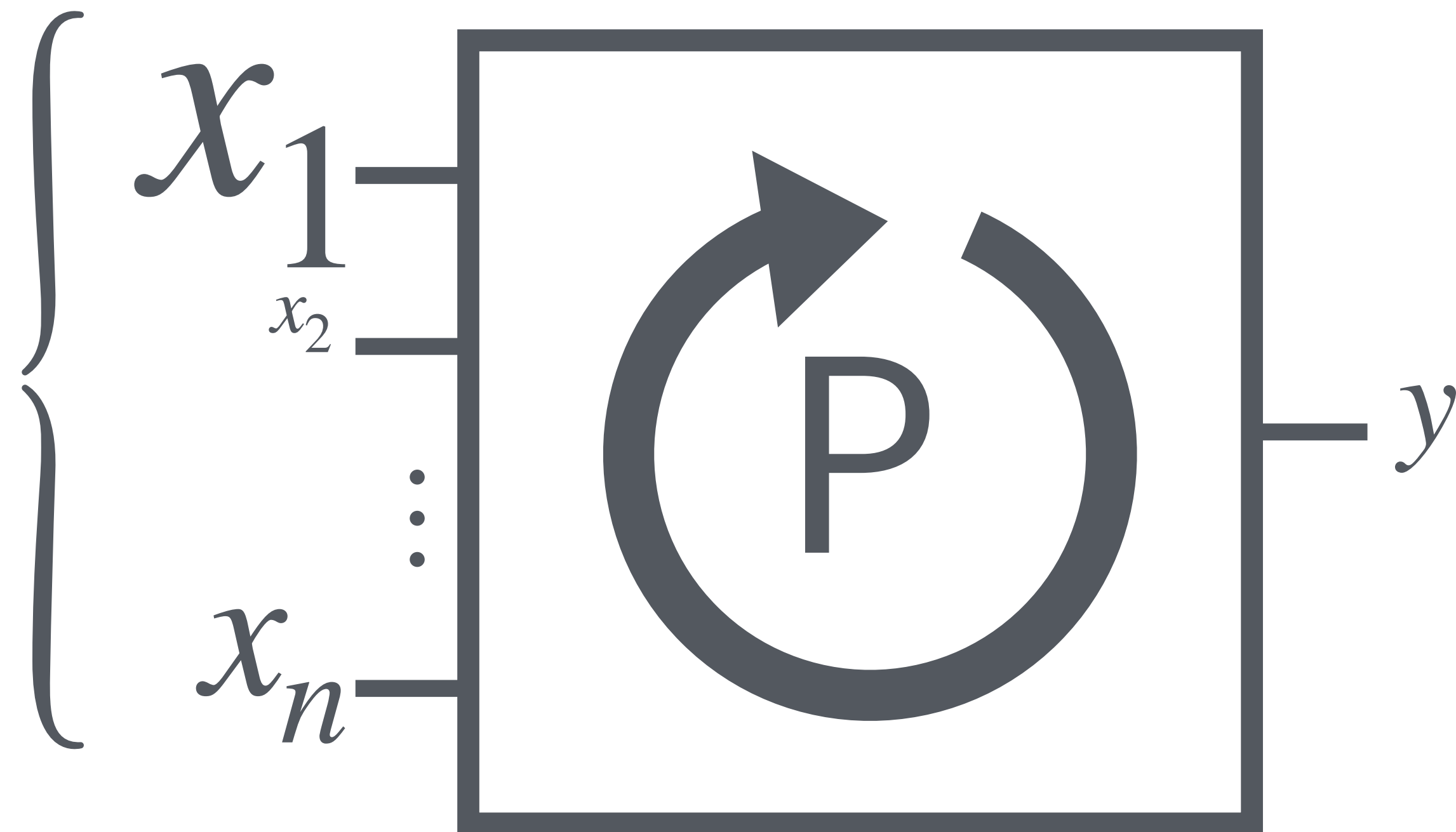


$$\text{IMPACT}_x(\text{Mu1}) = 2 \cdot \text{IMPACT}_y(\text{Mu1})$$


-
- Diagram illustrating a neural network layer structure. The input layer consists of nodes x_1, x_2, \dots, x_n . The output layer consists of nodes 2, 3, 4, 5, 6. A curved arrow labeled A indicates the transformation from the input layer to the output layer.

Why?

- Correct Loop Behaviour
- Performance



Why?

- Correct Loop Behaviour
- Performance



Why?

- Correct Loop Behaviour
- Performance

$$\text{IMPACT}_x(P) \leq \text{IMPACT}_x(P')$$



Why?

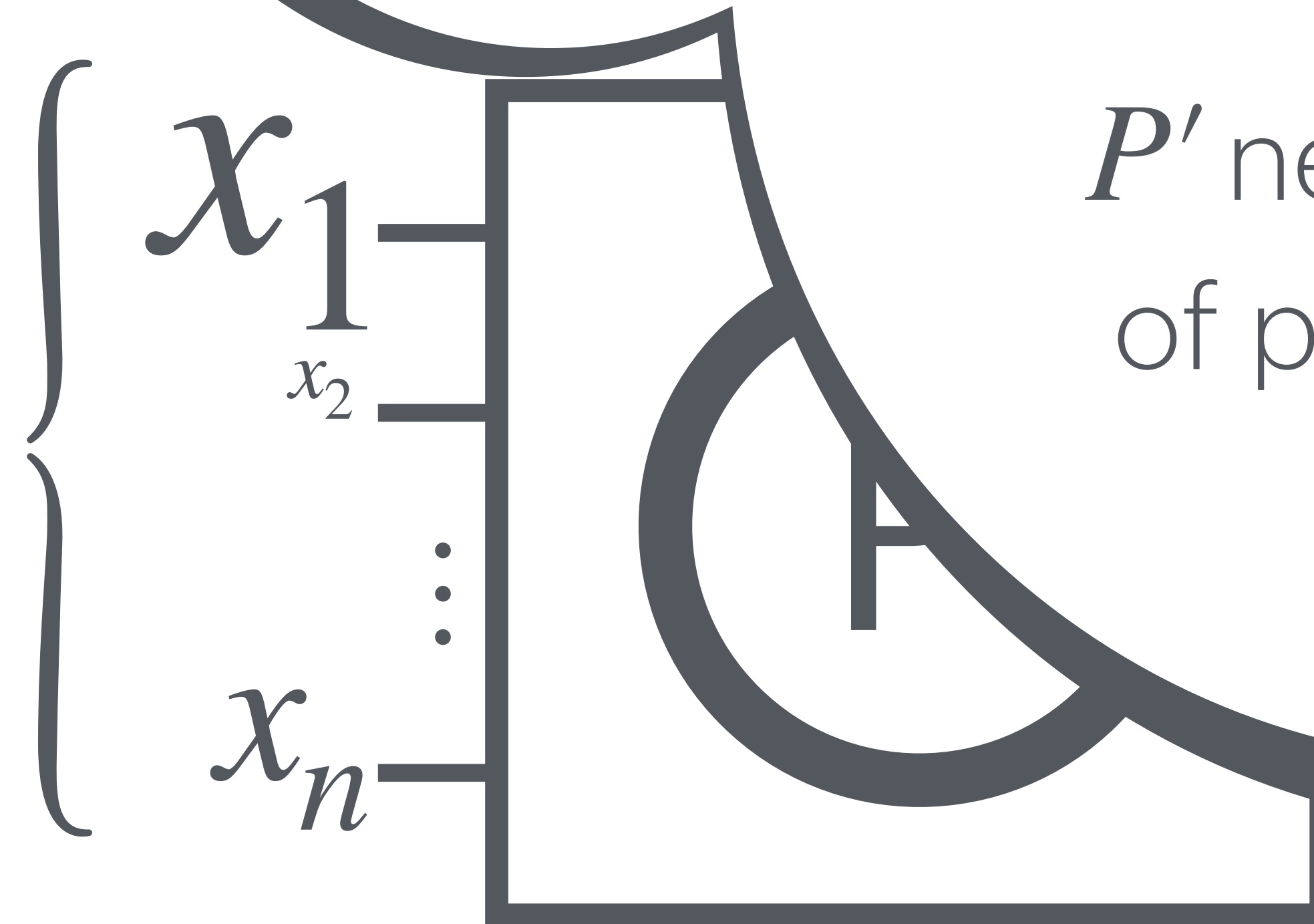
- Correct Loop Behaviour
- Performance

$$\text{IMPACT}_x(P) \leq \text{IMPACT}_x(P')$$



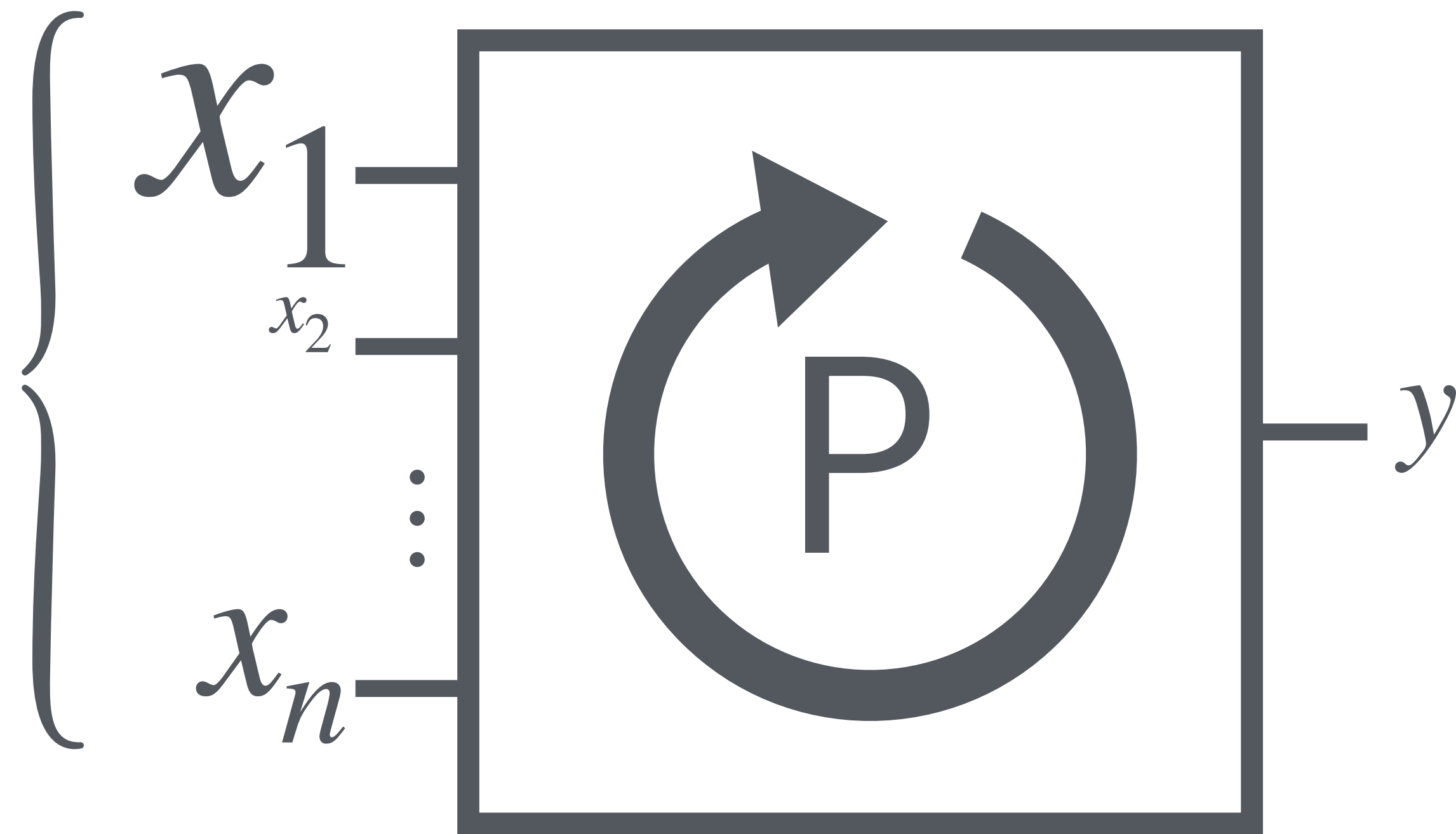
Regression!

P' new version
of program P



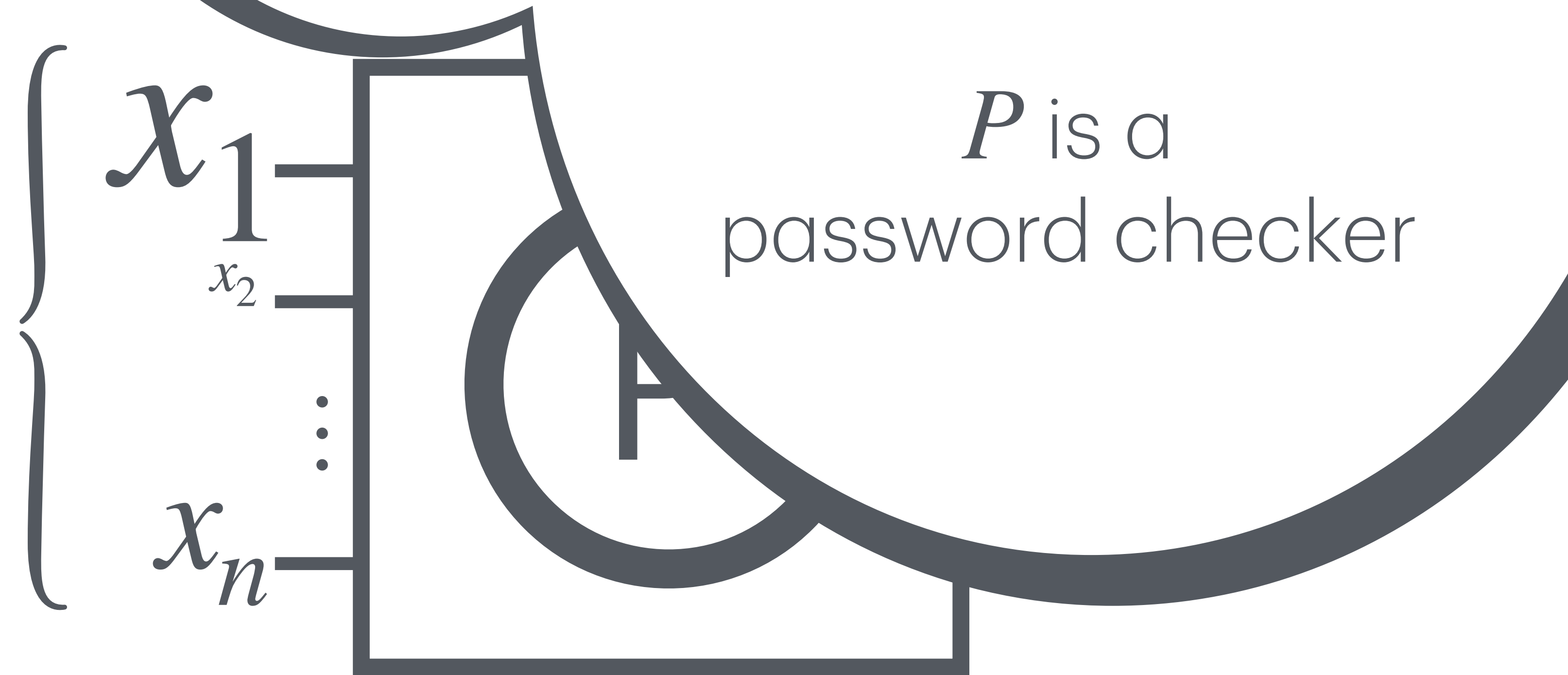
Why?

- Correct Loop Behaviour
- Performance
- Security



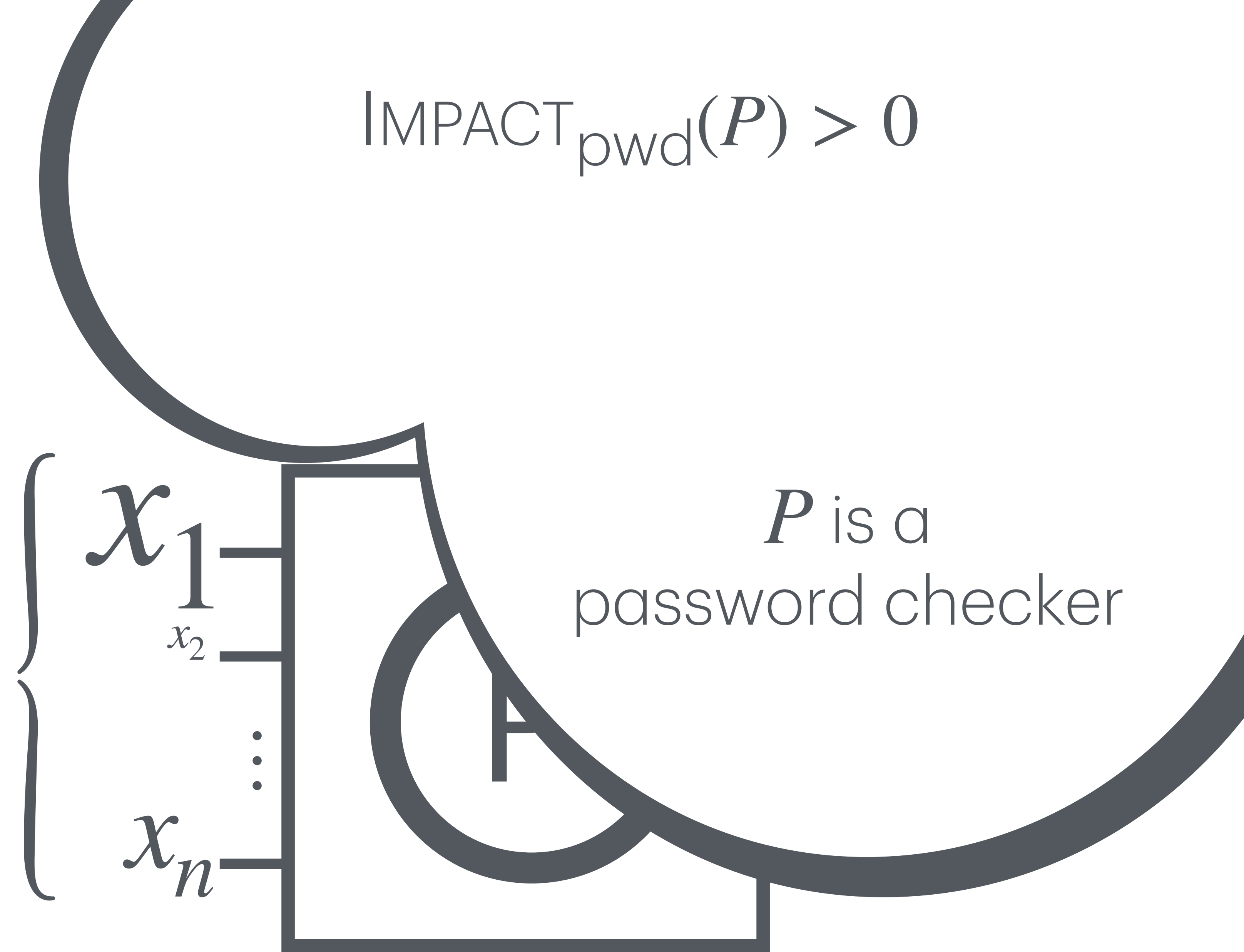
Why?

- Correct Loop Behaviour
- Performance
- Security



Why?

- Correct Loop Behaviour
- Performance
- Security



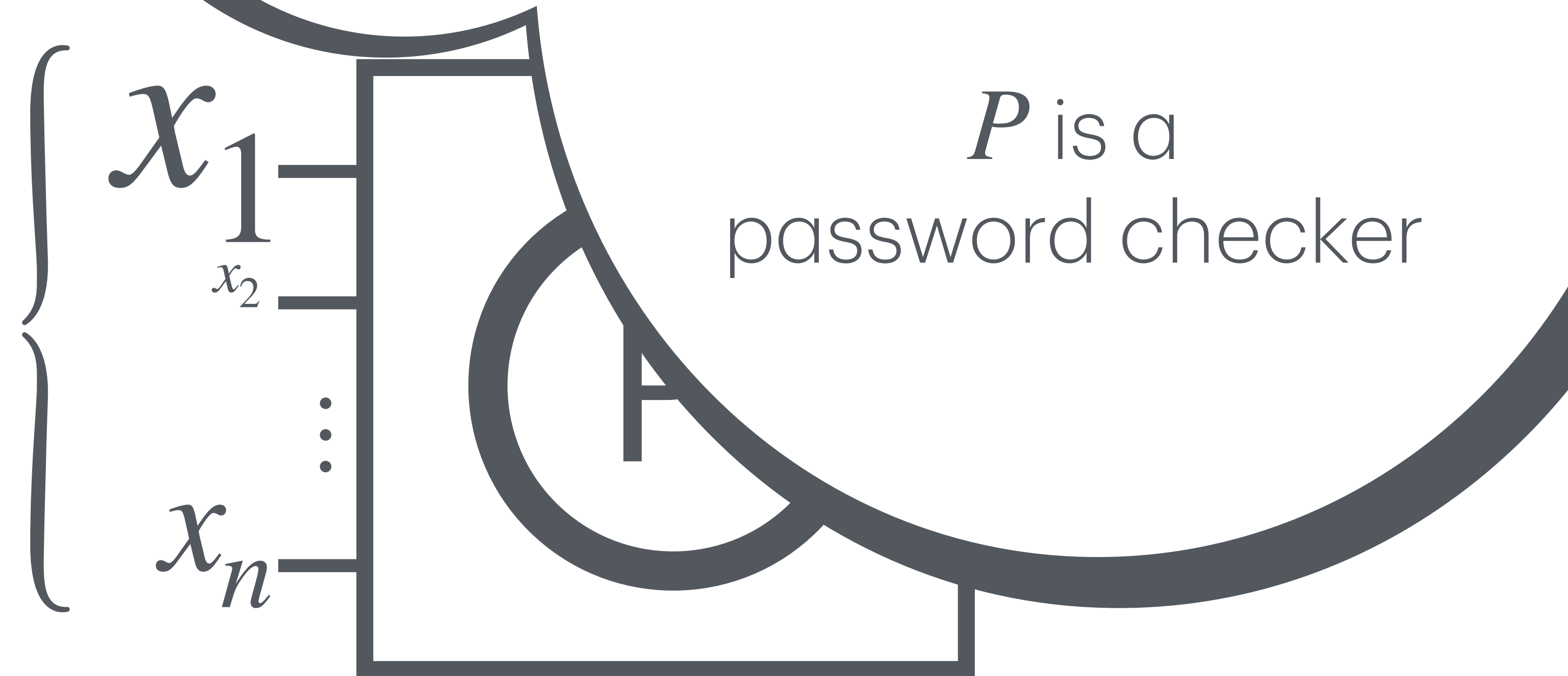
Why?

- Correct Loop Behaviour
- Performance
- Security

$$\text{IMPACT}_{\text{pwd}}(P) > 0$$



Timing side-channels!



Add Function

S2N-Bignum

3 8
4

Add Function

$$\begin{array}{r} 42 = \\ \hline 38 + \\ 4 \end{array}$$

Add Function

array

$$\begin{array}{r} [0, 4, 2] = \\ \hline [3, 8] + \\ [4] \end{array}$$

Add Function

length array

| | | |
|---|-----------|---|
| 3 | [0, 4, 2] | = |
| 2 | [3, 8] | + |
| 1 | [4] | |

Add Function

| length | array |
|--------|-----------|
| 3 | [0, 4, 2] |
| 2 | [3, 8] |
| 1 | [4] |

```
1 def Add(p, z, m, x, n, y): 26
2   r = min(p, m)           27
3   s = min(p, n)           28
4   if (r < s):              29
5       t = p - s            30
6       q = s - r            31
7       i = 0                32
8       a = 0                33
9       for (; r > 0; r--):  34
10          s = x[i]          35
11          w = y[i]          36
12          z[i] = s + w + a  37
13          i = i + 1         38
14          a = (w < a) ||    39
15              (s + w < s) || 40
16              (s + w + a < s) 41
17          do:               42
18              r = y[i]      43
19              b = (r < a) || 44
20                  (r + a < r) 45
21              z[i] = r + a   46
22              i = i + 1     47
23              q--           48
24              a = b         49
25          while (q > 0)      50
```

```
else:
    t = p - r
    q = r - s
    i = 0
    b = 0
    for (; s > 0; s--):
        r = x[i]
        w = y[i]
        z[i] = r + w + b
        i = i + 1
        b = (w < b) ||
            (r + w < r) ||
            (r + w + b < r)
    for (; q > 0; q--):
        r = x[i]
        z[i] = r + b
        i = i + 1
        b = (r < b) ||
            (r + b < r)
    if (t > 0):
        z[i] = b
        while (t > 0):
            i = i + 1
            t--
            if (t > 0):
                z[i] = 0
```

Add Function

Add $\left(\begin{matrix} p, \\ m, \\ n, \end{matrix} \right. \left. \begin{matrix} z, \\ x, \\ y \end{matrix} \right)$

```
1 def Add(p, z, m, x, n, y): 26
2     r = min(p, m)          27
3     s = min(p, n)          28
4     if (r < s):             29
5         t = p - s          30
6         q = s - r          31
7         i = 0              32
8         a = 0              33
9         for (; r > 0; r--): 34
10            s = x[i]        35
11            w = y[i]        36
12            z[i] = s + w + a 37
13            i = i + 1        38
14            a = (w < a) ||   39
15                (s + w < s) || 40
16                (s + w + a < s) 41
17        do:                 42
18            r = y[i]         43
19            b = (r < a) ||   44
20                (r + a < r)    45
21            z[i] = r + a     46
22            i = i + 1        47
23            q--              48
24            a = b            49
25        while (q > 0)        50
                                51
                                else:
                                t = p - r
                                q = r - s
                                i = 0
                                b = 0
                                for (; s > 0; s--):
                                    r = x[i]
                                    w = y[i]
                                    z[i] = r + w + b
                                    i = i + 1
                                    b = (w < b) ||
                                        (r + w < r) ||
                                        (r + w + b < r)
                                for (; q > 0; q--):
                                    r = x[i]
                                    z[i] = r + b
                                    i = i + 1
                                    b = (r < b) ||
                                        (r + b < r)
                                if (t > 0):
                                    z[i] = b
                                    while (t > 0):
                                        i = i + 1
                                        t--
                                        if (t > 0):
                                            z[i] = 0
```

Add Function

Add $\begin{pmatrix} 1, z, \\ 1, [4], \\ 1, [2] \end{pmatrix}$

Add $\begin{pmatrix} p, & z, \\ m, & x, \\ n, & y \end{pmatrix}$

Add Function

Add $\begin{pmatrix} 1, z, \\ 1, [4], \\ 1, [2] \end{pmatrix}$ — 1 iteration

Add $\begin{pmatrix} p, & z, \\ m, & x, \\ n, & y \end{pmatrix}$

Add Function

Add $\left(\begin{array}{l} 1, z, \\ 1, [4], \\ 1, [2] \end{array} \right) \xrightarrow{\text{1 iteration}} z = [6]$

Add $\left(\begin{array}{l} p, \\ m, \\ n, \end{array} \begin{array}{l} z, \\ x, \\ y \end{array} \right)$

Add Function

Add $\left(\begin{array}{l} 1, z, \\ 1, [4], \\ 1, [2] \end{array} \right) \xrightarrow{\text{1 iteration}} z = [6]$

Add $\left(\begin{array}{l} p, \\ m, \\ n, \end{array} \begin{array}{l} z, \\ x, \\ y \end{array} \right) \xrightarrow{\text{2 iterations}} z = [1, 4]$

Add Function

Add $\left(\begin{array}{l} 1, z, \\ 1, [4], \\ 1, [2] \end{array} \right)$ $\xrightarrow{\text{1 iteration}}$ $z = [6]$

Add $\left(\begin{array}{l} 2, z, \\ 1, [2], \\ 2, [1, 2] \end{array} \right)$ $\xrightarrow{\text{2 iterations}}$ $z = [1, 4]$

Add $\left(\begin{array}{l} 2, z, \\ 2, [3, 8], \\ 2, [0, 4] \end{array} \right)$ $\xrightarrow{\text{2 iterations}}$ $z = [4, 2]$

Add $\left(\begin{array}{l} p, \\ m, \\ n, \end{array} \begin{array}{l} z, \\ x, \\ y \end{array} \right)$

Add Function

Add $\left(\begin{array}{l} 1, z, \\ 1, [4], \\ 1, [2] \end{array} \right)$ $\xrightarrow{\text{1 iteration}}$ $z = [6]$

Add $\left(\begin{array}{l} 2, z, \\ 1, [2], \\ 2, [1, 2] \end{array} \right)$ $\xrightarrow{\text{2 iterations}}$ $z = [1, 4]$

Add $\left(\begin{array}{l} 2, z, \\ 2, [3, 8], \\ 2, [0, 4] \end{array} \right)$ $\xrightarrow{\text{2 iterations}}$ $z = [4, 2]$

Add $\left(\begin{array}{l} 3, z, \\ 1, [4], \\ 1, [2] \end{array} \right)$ $\xrightarrow{\text{3 iterations}}$ $z = [0, 0, 6]$

Add Function

$$\text{Add} \begin{pmatrix} p, \\ m, \\ n, \end{pmatrix} \quad \begin{pmatrix} z, \\ x, \\ y \end{pmatrix}$$

$$\text{Add} \begin{pmatrix} 1, z, \\ 1, [4], \\ 1, [2] \end{pmatrix}$$

— 1 iteration $\rightarrow z = [6]$

$$\text{Add} \begin{pmatrix} 2, z, \\ 1, [2], \\ 2, [1, 2] \end{pmatrix}$$

— 2 iterations $\rightarrow z = [1, 4]$

$$\text{Add} \begin{pmatrix} 2, z, \\ 2, [3, 8], \\ 2, [0, 4] \end{pmatrix}$$

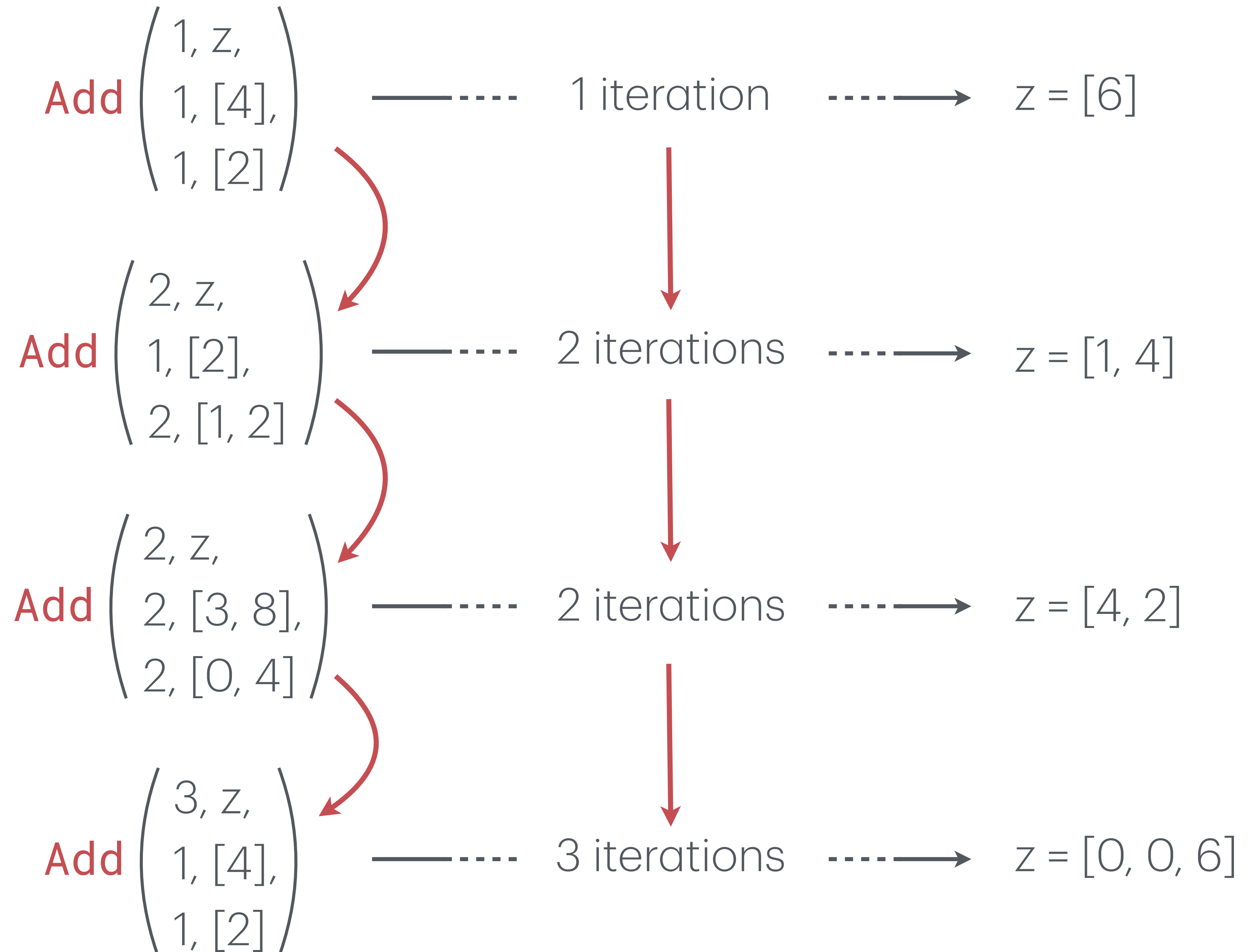
— 2 iterations $\rightarrow z = [4, 2]$

$$\text{Add} \begin{pmatrix} 3, z, \\ 1, [4], \\ 1, [2] \end{pmatrix}$$

— 3 iterations $\rightarrow z = [0, 0, 6]$

Add Function

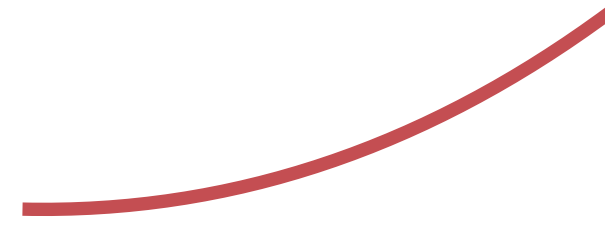
$$\text{Add} \begin{pmatrix} p, \\ m, \\ n, \end{pmatrix} \quad \begin{pmatrix} z, \\ x, \\ y \end{pmatrix}$$



How we compute $\text{IMPACT}_p(\text{Add})$?

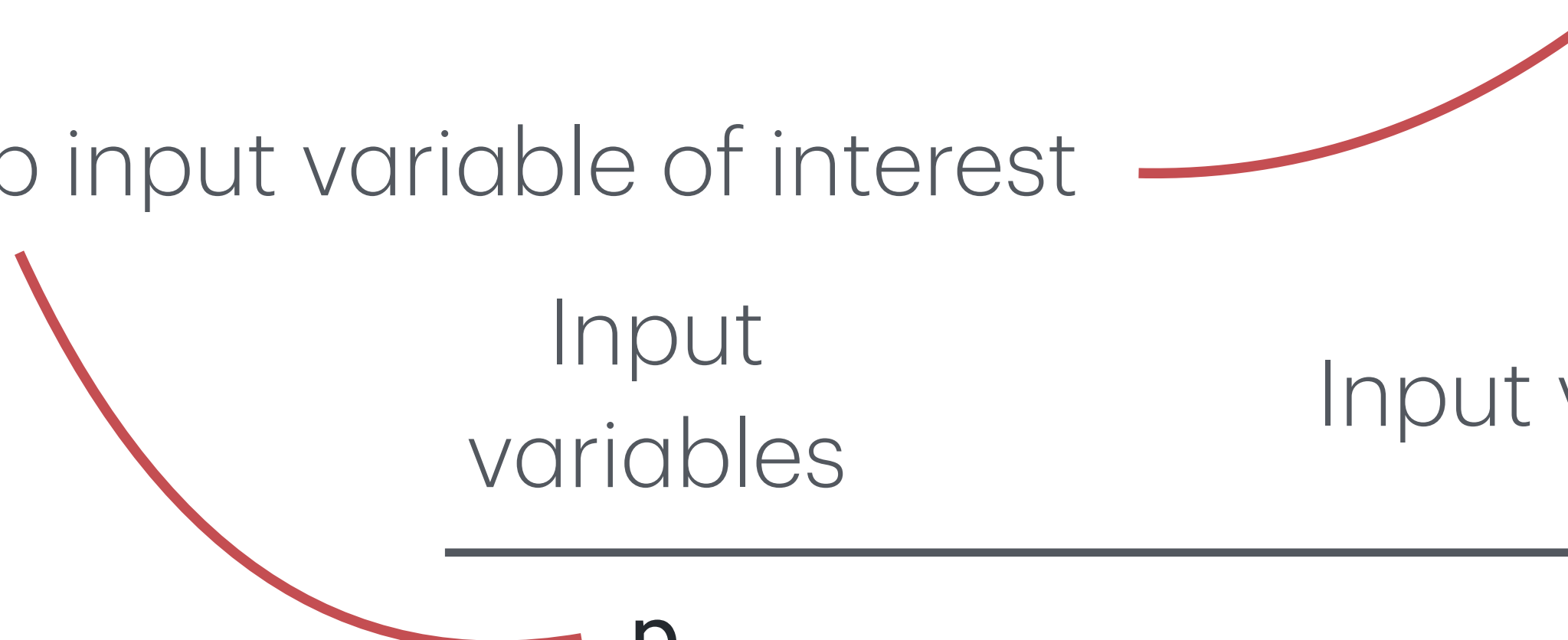
How we compute $\text{IMPACT}_p(\text{Add})$?

p input variable of interest



How we compute $\text{IMPACT}_p(\text{Add})$?

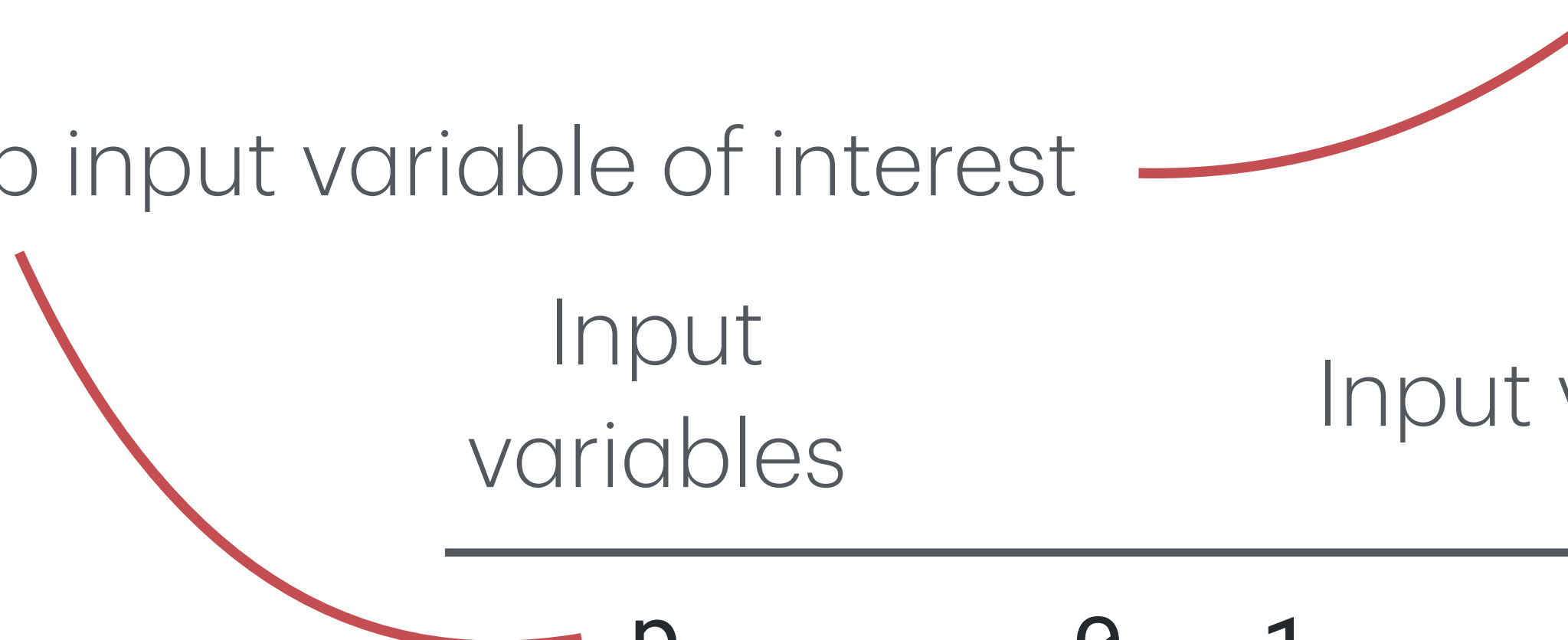
p input variable of interest



| Input variables | Input values |
|-----------------|--------------|
| p | |
| z | z |
| m | 2 |
| x | [3, 8] |
| n | 1 |
| y | [4] |

How we compute $\text{IMPACT}_p(\text{Add})$?

p input variable of interest



| Input variables | Input values | | | | | |
|-----------------|---------------|---|-----|---|-----|-----|
| p | \emptyset | 1 | ... | 3 | ... | u |
| z | z | | | | | |
| m | 2 | | | | | |
| x | [3, 8] | | | | | |
| n | 1 | | | | | |
| y | [4] | | | | | |

How we compute $\text{IMPACT}_p(\text{Add})$?

p input variable of interest

| Input variables | Input values | | | | | |
|-----------------|--------------|---|-----|---|-----|-----|
| p | 0 | 1 | ... | 3 | ... | u |
| z | z | | | | | |
| m | 2 | | | | | |
| x | [3, 8] | | | | | |
| n | 1 | | | | | |
| y | [4] | | | | | |
| # it. | 0 | 1 | ... | 3 | ... | u |

How we compute $\text{IMPACT}_p(\text{Add})$?

p input variable of interest

| Input variables | Input values | | | | | |
|-----------------|--------------|---|-----|---|-----|-----|
| p | 0 | 1 | ... | 3 | ... | u |
| z | z | | | | | |
| m | 2 | | | | | |
| x | [3, 8] | | | | | |
| n | 1 | | | | | |
| y | [4] | | | | | |
| # it. | 0 | 1 | ... | 3 | ... | u |

For all
input values!

How we compute $\text{IMPACT}_p(\text{Add})$?

p input variable of interest

| Input variables | Input values | | | | | |
|-----------------|---------------------------|---|-----|---|-----|-----|
| p | 0 | 1 | ... | 3 | ... | u |
| z | z | | | | | |
| m | 2 | | | | | |
| x | [3, 8] | | | | | |
| n | 1 | | | | | |
| y | [4] | | | | | |
| # it. | {0, 1, ..., 3, ..., u } | | | | | |

For all
input values!

How we compute $\text{IMPACT}_p(\text{Add})$?

p input variable of interest

| Input variables | Input values | | | | | |
|-----------------|--------------|---|-----|---|-----|-----|
| p | 0 | 1 | ... | 3 | ... | u |
| z | z | | | | | |
| m | 2 | | | | | |
| x | [3, 8] | | | | | |
| n | 1 | | | | | |
| y | [4] | | | | | |

For all
input values!

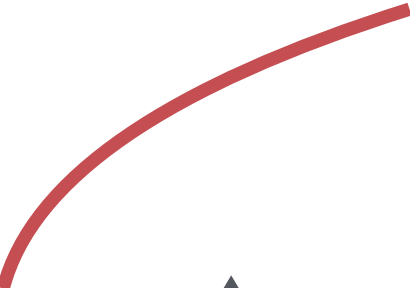
$$\text{Range}(\{0, 1, \dots, 3, \dots, u\}) = u$$

How we compute $\text{IMPACT}_p(\text{Add})$?

$$\text{IMPACT}_x(P) \triangleq \max_s \text{Range}(\{\text{iterations of } \sigma \mid \sigma \in P \wedge \sigma_0(\text{variables} \setminus x) = s\})$$

How we compute $\text{IMPACT}_p(\text{Add})$?

set of traces


$$\text{IMPACT}_x(P) \triangleq \max_s \text{Range}(\{\text{iterations of } \sigma \mid \sigma \in P \wedge \sigma_0(\text{variables} \setminus x) = s\})$$

How we compute $\text{IMPACT}_p(\text{Add})$?

set of traces

$$\text{IMPACT}_x(P) \triangleq$$
$$\max_s \text{Range}(\{\text{iterations of } \sigma \mid \sigma \in P \wedge \sigma_0(\text{variables} \setminus x) = s\})$$

input values

How we compute $\text{IMPACT}_p(\text{Add})$?

set of traces

for all traces

$$\text{IMPACT}_x(P) \triangleq \max_s \text{Range}(\{\text{iterations of } \sigma \mid \sigma \in P \wedge \sigma_0(\text{variables} \setminus x) = s\})$$

input values

The diagram illustrates the components of the IMPACT_x(P) definition. A red arc connects the text 'set of traces' to the variable P in the formula. Another red arc connects the text 'for all traces' to the set of iterations {iterations of σ | σ ∈ P ∧ σ₀(variables \ x) = s}. A third red arc connects the text 'input values' to the variable s in the formula.

How we compute $\text{IMPACT}_p(\text{Add})$?

set of traces

for all traces

$$\text{IMPACT}_x(P) \triangleq \max_s \text{Range}(\{\text{iterations of } \sigma \mid \sigma \in P \wedge \sigma_0(\text{variables} \setminus x) = s\})$$

input values

all variables but x

How we compute $\text{IMPACT}_p(\text{Add})$?

set of traces

for all traces

$$\text{IMPACT}_x(P) \triangleq \max_s \text{Range}(\{\text{iterations of } \sigma \mid \sigma \in P \wedge \sigma_0(\text{variables} \setminus x) = s\})$$

input values

all variables but x

all perturbations of variable x

Add Function

Add $\left(\begin{array}{l} p, \\ m, \\ n, \end{array} \right)$ $\left(\begin{array}{l} z, \\ x, \\ y \end{array} \right)$
size data

Security Requirement:
no timing side-channels
on **data** input variables

Add Function

Add $\left(\begin{array}{l} p, \\ m, \\ n, \\ \text{size} \end{array} \right) \left(\begin{array}{l} z, \\ x, \\ y \\ \text{data} \end{array} \right)$

Security Requirement:
no timing side-channels
on **data** input variables

- $\text{IMPACT}_{\{p,m,n\}}(\text{Add}) \geq 0$

Add Function

Add $\left(\begin{array}{l} p, \\ m, \\ n, \\ \text{size} \end{array} \right) \quad \left(\begin{array}{l} z, \\ x, \\ y \\ \text{data} \end{array} \right)$

Security Requirement:
no timing side-channels
on **data** input variables

- $\text{IMPACT}_{\{p,m,n\}}(\text{Add}) \geq 0$
- $\text{IMPACT}_{\{z,x,y\}}(\text{Add}) = 0$

How?

$\text{IMPACT}_{\{z,x,y\}}(\text{Add})$

How?

abstract

concrete

$$\text{Impact}_{\{z,x,y\}}^{\sharp}(\text{Add}) \geq \text{IMPACT}_{\{z,x,y\}}(\text{Add})$$

How?

abstract

concrete

$\text{Impact}_{\{z,x,y\}}^{\sharp}(\text{Add}) = 0$ then $\text{IMPACT}_{\{z,x,y\}}(\text{Add}) = 0$

$\text{Impact}_{\{z,x,y\}}^{\sharp}(\text{Add}) \geq \text{IMPACT}_{\{z,x,y\}}(\text{Add})$

Computing $\text{Impact}_x^{\sharp}(P)$

In three steps:

(i) Remove irrelevant instructions

Computing $\text{Impact}_x^{\sharp}(P)$

In three steps:

(i) Remove irrelevant instructions

Syntactic Dependency Analysis

Computing $\text{Impact}_x^{\sharp}(P)$

In three steps:

(i) Remove irrelevant instructions

Syntactic Dependency Analysis

(ii) Abstract Interpretation

Computing $\text{Impact}_x^{\sharp}(P)$

In three steps:

(i) Remove irrelevant instructions

Syntactic Dependency Analysis

(ii) Abstract Interpretation

Invariant on **input variables** + **iteration counter**

Computing $\text{Impact}_x^{\sharp}(P)$

In three steps:

(i) Remove irrelevant instructions

Syntactic Dependency Analysis

(ii) Abstract Interpretation

Invariant on **input variables + iteration counter**

(iii) Impact quantification

Computing $\text{Impact}_x^{\sharp}(P)$

In three steps:

(i) Remove irrelevant instructions

Syntactic Dependency Analysis

(ii) Abstract Interpretation

Invariant on **input variables + iteration counter**

(iii) Impact quantification

Mixed-integer linear programming

(i) Irrelevant Instructions

```
1 def Add(p, z, m, x, n, y): 26
2   r = min(p, m)           27
3   s = min(p, n)           28
4   if (r < s):              29
5       t = p - s           30
6       q = s - r           31
7       i = 0               32
8       a = 0               33
9       for (; r > 0; r--):  34
10          s = x[i]         35
11          w = y[i]         36
12          z[i] = s + w + a 37
13          i = i + 1        38
14          a = (w < a) ||   39
15              (s + w < s) || 40
16              (s + w + a < s) 41
17      do:                  42
18          r = y[i]         43
19          b = (r < a) ||   44
20              (r + a < r)   45
21          z[i] = r + a     46
22          i = i + 1        47
23          q--              48
24          a = b            49
25      while (q > 0)        50
```

```
else:
    t = p - r
    q = r - s
    i = 0
    b = 0
    for (; s > 0; s--):
        r = x[i]
        w = y[i]
        z[i] = r + w + b
        i = i + 1
        b = (w < b) ||
            (r + w < r) ||
            (r + w + b < r)
        for (; q > 0; q--):
            r = x[i]
            z[i] = r + b
            i = i + 1
            b = (r < b) ||
                (r + b < r)
    if (t > 0):
        z[i] = b
        while (t > 0):
            i = i + 1
            t--
            if (t > 0):
                z[i] = 0
    51
```

(i) Irrelevant Instructions

```
1 def Add(p, z, m, x, n, y): 26
2   r = min(p, m)           27
3   s = min(p, n)           28
4   if (r < s):              29
5       t = p - s           30
6       q = s - r           31
7       i = 0               32
8       a = 0               33
9       for (; r > 0; r--): 34
10           s = x[i]        35
11           w = y[i]        36
12           z[i] = s + w + a 37
13           i = i + 1        38
14           a = (w < a) ||   39
15               (s + w < s) || 40
16               (s + w + a < s) 41
17       do:                 42
18           r = y[i]         43
19           b = (r < a) ||   44
20               (r + a < r)    45
21           z[i] = r + a     46
22           i = i + 1        47
23           q--              48
24           a = b            49
25       while (q > 0)        50
```

```
else:
    t = p - r
    q = r - s
    i = 0
    b = 0
    for (; s > 0; s--):
        r = x[i]
        w = y[i]
        z[i] = r + w + b
        i = i + 1
        b = (w < b) ||
            (r + w < r) ||
            (r + w + b < r)
        for (; q > 0; q--):
            r = x[i]
            z[i] = r + b
            i = i + 1
            b = (r < b) ||
                (r + b < r)
    if (t > 0):
        z[i] = b
        while (t > 0):
            i = i + 1
            t--
            if (t > 0):
                z[i] = 0
```

(i) Irrelevant Instructions

```
1 def Add(p, z, m, x, n, y): 26
2   r = min(p, m)           27
3   s = min(p, n)           28
4   if (r < s):              29
5       t = p - s           30
6       q = s - r           31
7       i = 0               32
8       a = 0               33
9       for (; r > 0; r--): 34
10          s = x[i]         35
11          w = y[i]         36
12          z[i] = s + w + a 37
13          i = i + 1        38
14          a = (w < a) ||   39
15              (s + w < s) || 40
16              (s + w + a < s) 41
17      do:                  42
18          r = y[i]         43
19          b = (r < a) ||   44
20              (r + a < r)   45
21          z[i] = r + a     46
22          i = i + 1        47
23          q--              48
24          a = b            49
25      while (q > 0)        50
```

```
else:
    t = p - r
    q = r - s
    i = 0
    b = 0
    for (; s > 0; s--):
        r = x[i]
        w = y[i]
        z[i] = r + w + b
        i = i + 1
        b = (w < b) ||
            (r + w < r) ||
            (r + w + b < r)
        for (; q > 0; q--):
            r = x[i]
            z[i] = r + b
            i = i + 1
            b = (r < b) ||
                (r + b < r)
    if (t > 0):
        z[i] = b
        while (t > 0):
            i = i + 1
            t--
            if (t > 0):
                z[i] = 0
```


(i) Irrelevant Instructions

```
1 def Add(p, z, m, x, n, y): 26
2   r = min(p, m)           27
3   s = min(p, n)           28
4   if (r < s):              29
5       t = p - s           30
6       q = s - r           31
7       i = 0               32
8       a = 0               33
9       for (; r > 0; r--): 34
10           s = x[i]        35
11           w = y[i]        36
12           z[i] = s + w + a 37
13           i = i + 1       38
14           a = (w < a) ||   39
15               (s + w < s) || 40
16               (s + w + a < s) 41
17       do:                 42
18           r = y[i]         43
19           b = (r < a) ||   44
20               (r + a < r)    45
21           z[i] = r + a     46
22           i = i + 1       47
23           q--             48
24           a = b           49
25       while (q > 0)       50
```

```
else:
    t = p - r
    q = r - s
    i = 0
    b = 0
    for (; s > 0; s--):
        r = x[i]
        w = y[i]
        z[i] = r + w + b
        i = i + 1
        b = (w < b) ||
            (r + w < r) ||
            (r + w + b < r)
        for (; q > 0; q--):
            r = x[i]
            z[i] = r + b
            i = i + 1
            b = (r < b) ||
                (r + b < r)
    if (t > 0):
        z[i] = b
        while (t > 0):
            i = i + 1
            t--
            if (t > 0):
                z[i] = 0
```

(i) Irrelevant Instructions

```
1 def Add(p, z, m, x, n, y): 26
2   r = min(p, m)           27
3   s = min(p, n)           28
4   if (r < s):              29
5       t = p - s           30
6       q = s - r           31
7       i = 0               32
8       a = 0               33
9       for (; r > 0; r--):  34
10          s = x[i]         35
11          w = y[i]         36
12          z[i] = s + w + a  37
13          i = i + 1        38
14          a = (w < a) ||    39
15              (s + w < s) || 40
16              (s + w + a < s) 41
17      do:                  42
18          r = y[i]         43
19          b = (r < a) ||    44
20              (r + a < r)    45
21          z[i] = r + a     46
22          i = i + 1        47
23          q--              48
24          a = b            49
25      while (q > 0)         50
```

```
else:
    t = p - r
    q = r - s
    i = 0
    b = 0
    for (; s > 0; s--):
        r = x[i]
        w = y[i]
        z[i] = r + w + b
        i = i + 1
        b = (w < b) ||
            (r + w < r) ||
            (r + w + b < r)
        for (; q > 0; q--):
            r = x[i]
            z[i] = r + b
            i = i + 1
            b = (r < b) ||
                (r + b < r)
    if (t > 0):
        z[i] = b
        while (t > 0):
            i = i + 1
            t--
            if (t > 0):
                z[i] = 0
```

(i) Irrelevant Instructions

```
1 def Add(p, z, m, x, n, y): 26
2   r = min(p, m)           27
3   s = min(p, n)           28
4   if (r < s):              29
5       t = p - s           30
6       q = s - r           31
7       i = 0               32
8       a = 0               33
9       for (; r > 0; r--): 34
10          s = x[i]         35
11          w = y[i]         36
12          z[i] = s + w + a 37
13          i = i + 1        38
14          a = (w < a) ||    39
15              (s + w < s) || 40
16              (s + w + a < s) 41
17      do:                  42
18          r = y[i]         43
19          b = (r < a) ||    44
20              (r + a < r)    45
21          z[i] = r + a      46
22          i = i + 1        47
23          q--              48
24          a = b            49
25      while (q > 0)         50
                               51
else:
    t = p - r
    q = r - s
    i = 0
    b = 0
    for (; s > 0; s--):
        r = x[i]
        w = y[i]
        z[i] = r + w + b
        i = i + 1
        b = (w < b) ||
            (r + w < r) ||
            (r + w + b < r)
        for (; q > 0; q--):
            r = x[i]
            z[i] = r + b
            i = i + 1
            b = (r < b) ||
                (r + b < r)
    if (t > 0):
        z[i] = b
        while (t > 0):
            i = i + 1
            t--
            if (t > 0):
                z[i] = 0
```

(i) Irrelevant Instructions

```
1 def Add(p, z, m, x, n, y): 26
2   r = min(p, m)           27
3   s = min(p, n)           28
4   if (r < s):              29
5       t = p - s           30
6       q = s - r           31
7       i = 0               32
8       a = 0               33
9       for (; r > 0; r--):  34
10          s = x[i]         35
11          w = y[i]         36
12          z[i] = s + w + a  37
13          i = i + 1        38
14          a = (w < a) ||    39
15              (s + w < s) || 40
16              (s + w + a < s) 41
17      do:                  42
18          r = y[i]         43
19          b = (r < a) ||    44
20              (r + a < r)    45
21          z[i] = r + a     46
22          i = i + 1        47
23          q--              48
24          a = b            49
25      while (q > 0)         50
```

```
else:
    t = p - r
    q = r - s
    i = 0
    b = 0
    for (; s > 0; s--):
        r = x[i]
        w = y[i]
        z[i] = r + w + b
        i = i + 1
        b = (w < b) ||
            (r + w < r) ||
            (r + w + b < r)
    for (; q > 0; q--):
        r = x[i]
        z[i] = r + b
        i = i + 1
        b = (r < b) ||
            (r + b < r)
    if (t > 0):
        z[i] = b
        while (t > 0):
            i = i + 1
            t--
            if (t > 0):
                z[i] = 0
```

(i) Irrelevant Instructions

Syntactic Dependency Analysis

Caterina Urban and Peter Müller, *An Abstract Interpretation Framework for Input Data Usage*, ESOP 2018

```
1 def Add(p, z, m, x, n, y): 26
2   r = min(p, m)           27
3   s = min(p, n)           28
4   if (r < s):              29
5       t = p - s           30
6       q = s - r           31
7       i = 0               32
8       a = 0               33
9       for (; r > 0; r--): 34
10          s = x[i]         35
11          w = y[i]         36
12          z[i] = s + w + a 37
13          i = i + 1        38
14          a = (w < a) ||   39
15              (s + w < s) || 40
16              (s + w + a < s) 41
17      do:                  42
18          r = y[i]         43
19          b = (r < a) ||   44
20              (r + a < r)   45
21          z[i] = r + a     46
22          i = i + 1        47
23          q--              48
24          a = b            49
25      while (q > 0)        50
```

```
else:
    t = p - r
    q = r - s
    i = 0
    b = 0
    for (; s > 0; s--):
        r = x[i]
        w = y[i]
        z[i] = r + w + b
        i = i + 1
        b = (w < b) ||
            (r + w < r) ||
            (r + w + b < r)
        for (; q > 0; q--):
            r = x[i]
            z[i] = r + b
            i = i + 1
            b = (r < b) ||
                (r + b < r)
    if (t > 0):
        z[i] = b
        while (t > 0):
            i = i + 1
            t--
            if (t > 0):
                z[i] = 0
```

(i) Irrelevant Instructions

Syntactic Dependency Analysis

Caterina Urban and Peter Müller, *An Abstract Interpretation Framework for Input Data Usage*, ESOP 2018

```
1  def Add(p, z, m, x, n, y):
2      r = min(p, m)
3      s = min(p, n)
4      if (r < s):
5          t = p - s
6          q = s - r
9      for (; r > 0; r--):
--          T skip;
17     do:
23         I q--;
25         while (q > 0)
26     else:
27         t = p - r
28         q = r - s
31         for (; s > 0; s--):
--             T skip;
39         for (; q > 0; q--):
--             T skip;
45     if (t > 0):
47         while (t > 0):
49             T t--;
```

(ii) Abstract Interpretation

Intuitively:

```
1  def Add(p, z, m, x, n, y):
2      r = min(p, m)
3      s = min(p, n)
4      if (r < s):
5          t = p - s
6          q = s - r
9          for (; r > 0; r--):
--              skip;
17         do:
23             q--;
25         while (q > 0)
26     else:
27         t = p - r
28         q = r - s
31         for (; s > 0; s--):
--             skip;
39         for (; q > 0; q--):
--             skip;
45     if (t > 0):
47         while (t > 0):
49             t--;
```


(ii) Abstract Interpretation

```
1 def Add(p, z, m, x, n, y):
2     r = min(p, m)
3     s = min(p, n)
4     if (r < s):
5         t = p - s
6         q = s - r
9         for (; r > 0; r--):
--             T skip; counter--
17        do:
23            I q--; counter--
25        while (q > 0)
26    else:
27        t = p - r
28        q = r - s
31        for (; s > 0; s--):
--            T skip; counter--
39        for (; q > 0; q--):
--            T skip; counter--
45    if (t > 0):
47        T while (t > 0):
49        T t--; counter--
```

Intuitively:

Augment each
loop body with a **counter**
for **iterations**

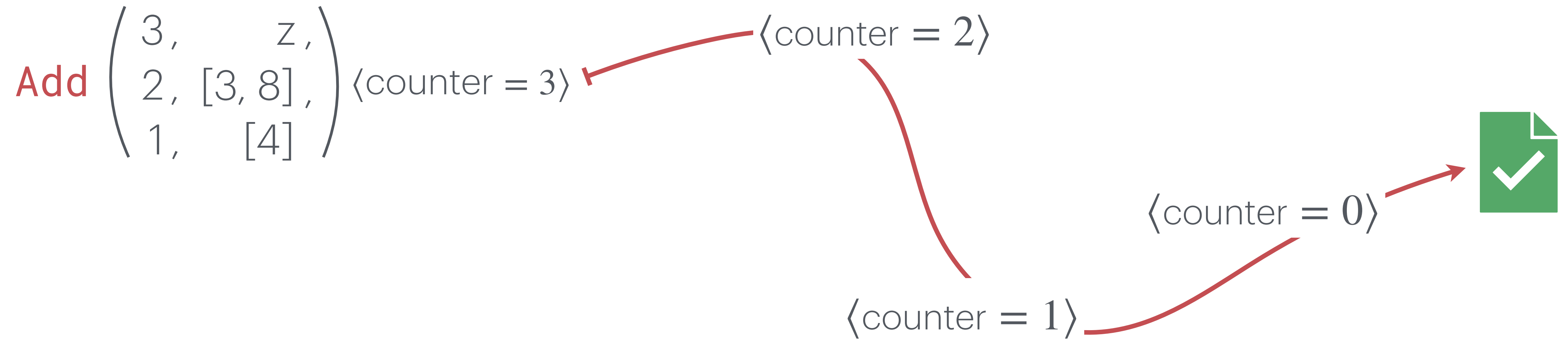
(ii) Abstract Interpretation

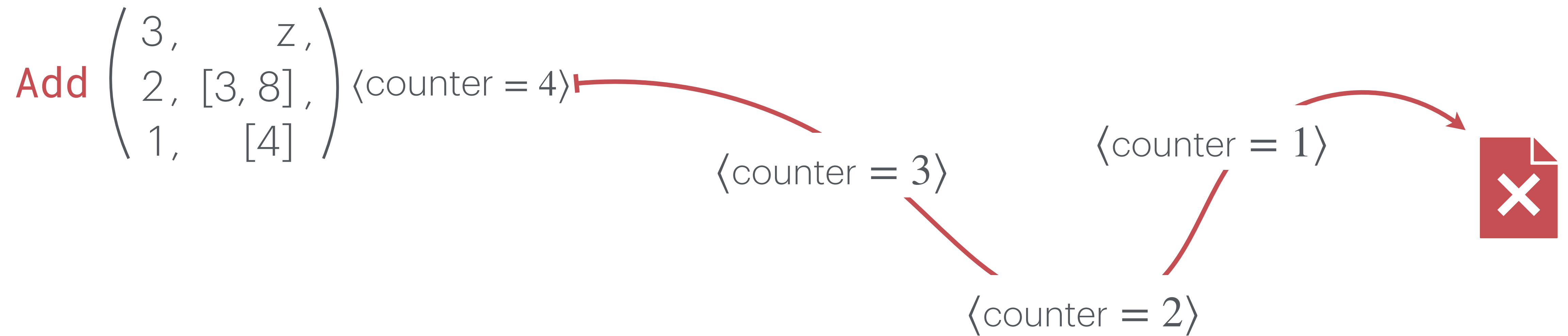
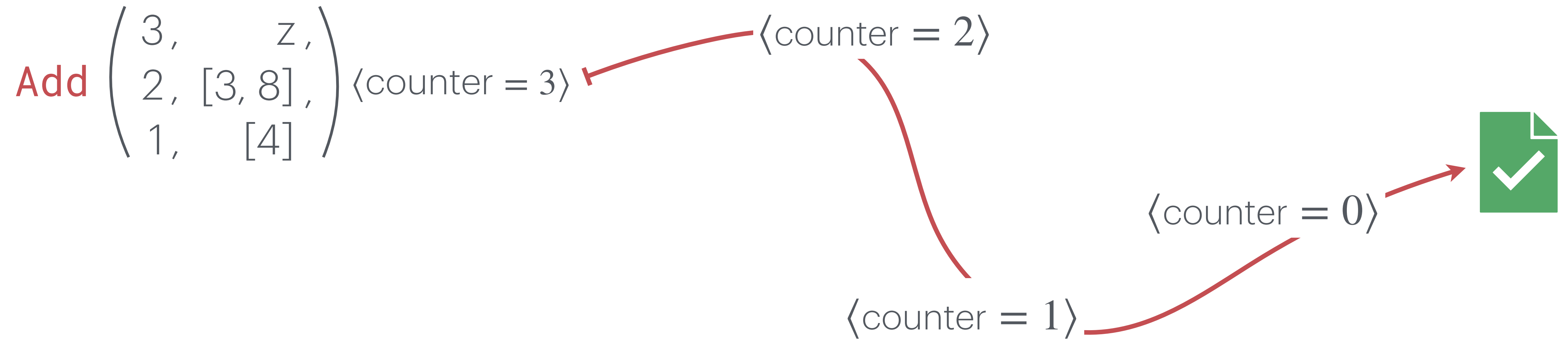
```
1 def Add(p, z, m, x, n, y):
2     r = min(p, m)
3     s = min(p, n)
4     if (r < s):
5         t = p - s
6         q = s - r
9         for (; r > 0; r--):
--             T skip; counter--
17        do:
23            I q--; counter--
25        while (q > 0)
26    else:
27        t = p - r
28        q = r - s
31        for (; s > 0; s--):
--            T skip; counter--
39        for (; q > 0; q--):
--            T skip; counter--
45    if (t > 0):
47        while (t > 0):
49            T t--; counter--
--    assert counter == 0
```

Intuitively:

Augment each
loop body with a **counter**
for **iterations**

Backwards starting
from zero!





(ii) Abstract Interpretation

```
1 def Add(p, z, m, x, n, y):
2     r = min(p, m)
3     s = min(p, n)
4     if (r < s):
5         t = p - s
6         q = s - r
9         for (; r > 0; r--):
--             skip; counter--
17        do:
23            q--; counter--
25        while (q > 0)
26    else:
27        t = p - r
28        q = r - s
31        for (; s > 0; s--):
--            skip; counter--
39        for (; q > 0; q--):
--            skip; counter--
45    if (t > 0):
47        while (t > 0):
49            t--; counter--
--    assert counter == 0
```

Intuitively:

Augment each
loop body with a **counter**
for **iterations**

Backward abstract
analysis

Backwards starting
from zero!

(ii) Abstract Interpretation

Without rewritings!

Iteration counter is handled

semantically

```
1 def Add(p, z, m, x, n, y):
2     r = min(p, m)
3     s = min(p, n)
4     if (r < s):
5         T t = p - r
6         q = r - s
9         for (; r > 0; r--):
--             T skip; counter--
17        do:
23            I q--; counter--
25            while (q > 0)
26        else:
27            T t = p - r
28            q = r - s
31            for (; s > 0; s--):
--                counter--
39            if (t > 0):
--                T skip; counter--
45            while (t > 0):
47                T t--; counter--
49            assert counter == 0
--
```

Backward Abstract Semantics

starts from $\Lambda^{\sharp}[[P]](\text{counter} = 0)$

Backward Abstract Semantics

starts from $\Lambda^{\natural}[[P]](\text{counter} = 0)$

$$\Lambda^{\natural}[[stmt; stmt']]d^{\natural} \triangleq \Lambda^{\natural}[[stmt]](\Lambda^{\natural}[[stmt']]d^{\natural})$$

Backward Abstract Semantics

starts from $\Lambda^{\sharp}[[P]](\text{counter} = 0)$

$$\Lambda^{\sharp}[[stmt; stmt']]d^{\sharp} \triangleq \Lambda^{\sharp}[[stmt]](\Lambda^{\sharp}[[stmt']]d^{\sharp})$$

$$\Lambda^{\sharp}[[\text{skip}]]d^{\sharp} \triangleq d^{\sharp}$$

Backward Abstract Semantics

starts from $\Lambda^{\natural}[[P]](\text{counter} = 0)$

$$\Lambda^{\natural}[[stmt; stmt']]d^{\natural} \triangleq \Lambda^{\natural}[[stmt]](\Lambda^{\natural}[[stmt']]d^{\natural})$$

$$\Lambda^{\natural}[[\text{skip}]]d^{\natural} \triangleq d^{\natural}$$

$$\Lambda^{\natural}[[x := e]]d^{\natural} \triangleq \text{Substitute}^{\natural}[[x \leftarrow e]]d^{\natural}$$

Backward Abstract Semantics

starts from $\Lambda^{\sharp}[[P]](\text{counter} = 0)$

$$\Lambda^{\sharp}[[stmt; stmt']d^{\sharp} \triangleq \Lambda^{\sharp}[[stmt]](\Lambda^{\sharp}[[stmt']]d^{\sharp})$$

$$\Lambda^{\sharp}[[\text{skip}]]d^{\sharp} \triangleq d^{\sharp}$$

$$\Lambda^{\sharp}[[x := e]]d^{\sharp} \triangleq \text{Substitute}^{\sharp}[[x \leftarrow e]]d^{\sharp}$$

$$\Lambda^{\sharp}[[\text{if } b \text{ then } stmt \text{ else } stmt']]d^{\sharp} \triangleq$$

$$\text{Filter}^{\sharp}[[b]](\Lambda^{\sharp}[[stmt]]) \sqcup^{\sharp} \text{Filter}^{\sharp}[[\neg b]](\Lambda^{\sharp}[[stmt']])$$

Backward Abstract Semantics

starts from $\Lambda^{\sharp}[[P]](\text{counter} = 0)$

$$\Lambda^{\sharp}[[stmt; stmt']d^{\sharp} \triangleq \Lambda^{\sharp}[[stmt]](\Lambda^{\sharp}[[stmt']]d^{\sharp})$$

$$\Lambda^{\sharp}[[\text{skip}]]d^{\sharp} \triangleq d^{\sharp}$$

$$\Lambda^{\sharp}[[x := e]]d^{\sharp} \triangleq \text{Substitute}^{\sharp}[[x \leftarrow e]]d^{\sharp}$$

$$\Lambda^{\sharp}[[\text{if } b \text{ then } stmt \text{ else } stmt']]d^{\sharp} \triangleq$$

$$\text{Filter}^{\sharp}[[b]](\Lambda^{\sharp}[[stmt]]) \sqcup^{\sharp} \text{Filter}^{\sharp}[[\neg b]](\Lambda^{\sharp}[[stmt']])$$

$$\Lambda^{\sharp}[[\text{while } b \text{ do } stmt]]d^{\sharp} \triangleq \lim_n F^n$$

Backward Abstract Semantics

starts from $\Lambda^{\flat}[[P]](\text{counter} = 0)$

$$\Lambda^{\flat}[[stmt; stmt']]d^{\flat} \triangleq \Lambda^{\flat}[[stmt]](\Lambda^{\flat}[[stmt']]d^{\flat})$$

$$\Lambda^{\flat}[[\text{skip}]]d^{\flat} \triangleq d^{\flat}$$

$$\Lambda^{\flat}[[x := e]]d^{\flat} \triangleq \text{Substitute}^{\flat}[[x \leftarrow e]]d^{\flat}$$

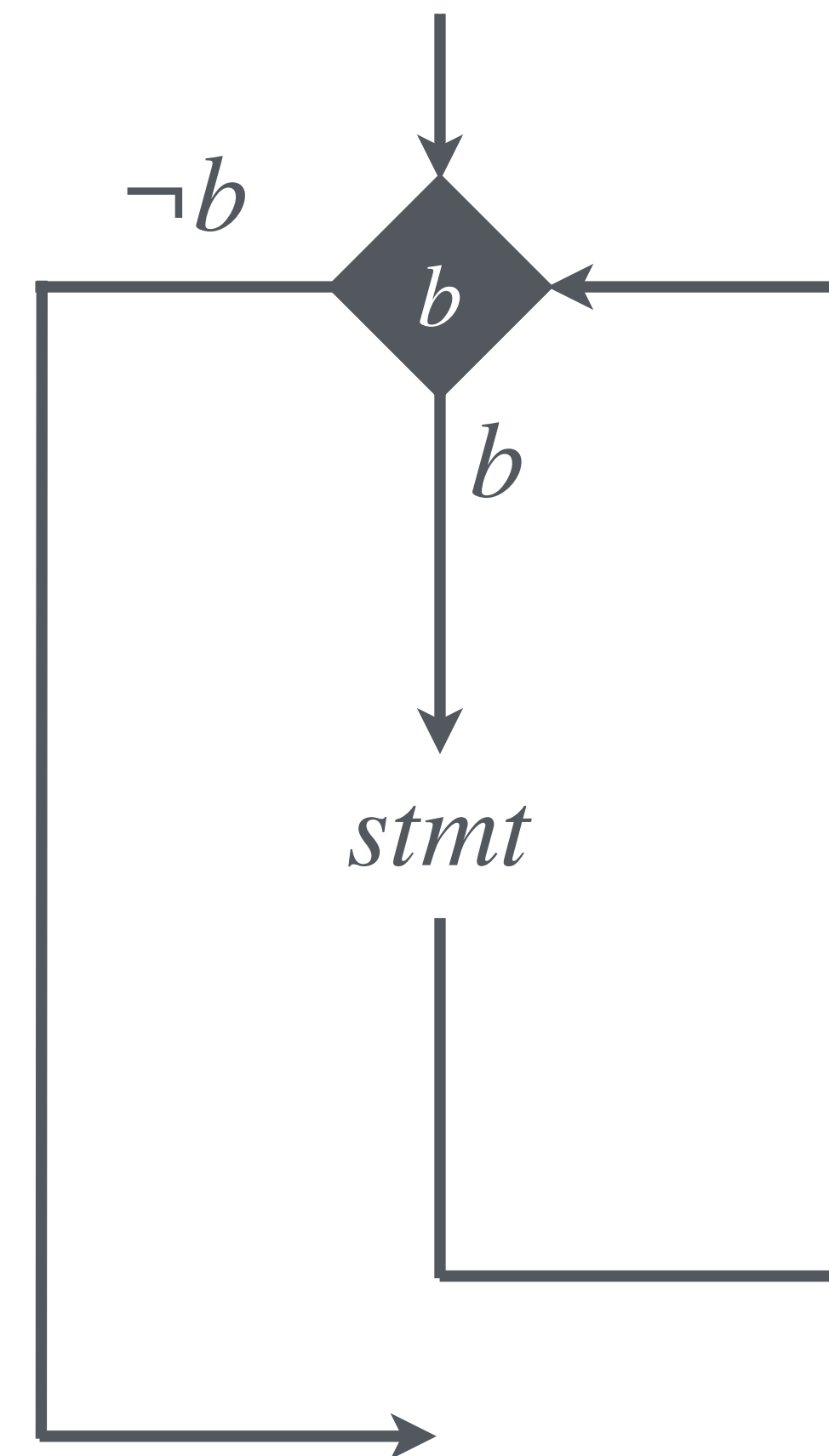
$$\Lambda^{\flat}[[\text{if } b \text{ then } stmt \text{ else } stmt']]d^{\flat} \triangleq$$

$$\text{Filter}^{\flat}[[b]](\Lambda^{\flat}[[stmt]]) \sqcup^{\flat} \text{Filter}^{\flat}[[\neg b]](\Lambda^{\flat}[[stmt']])$$

$$\Lambda^{\flat}[[\text{while } b \text{ do } stmt]]d^{\flat} \triangleq \lim_n F^n$$

$$F(x^{\flat}) \triangleq \text{Filter}^{\flat}[[\neg b]]d^{\flat} \sqcup^{\flat}$$

$$\text{Filter}^{\flat}[[b]](\Lambda^{\flat}[[stmt]](\text{Substitute}^{\flat}[[\text{counter} \leftarrow \text{counter} - 1]]x^{\flat})))$$



Backward Abstract Semantics

starts from $\Lambda^{\sharp}[[P]](\text{counter} = 0)$

$$\Lambda^{\sharp}[[stmt; stmt']]d^{\sharp} \triangleq \Lambda^{\sharp}[[stmt]](\Lambda^{\sharp}[[stmt']]d^{\sharp})$$

$$\Lambda^{\sharp}[[\text{skip}]]d^{\sharp} \triangleq d^{\sharp}$$

$$\Lambda^{\sharp}[[x := e]]d^{\sharp} \triangleq \text{Substitute}^{\sharp}[[x \leftarrow e]]d^{\sharp}$$

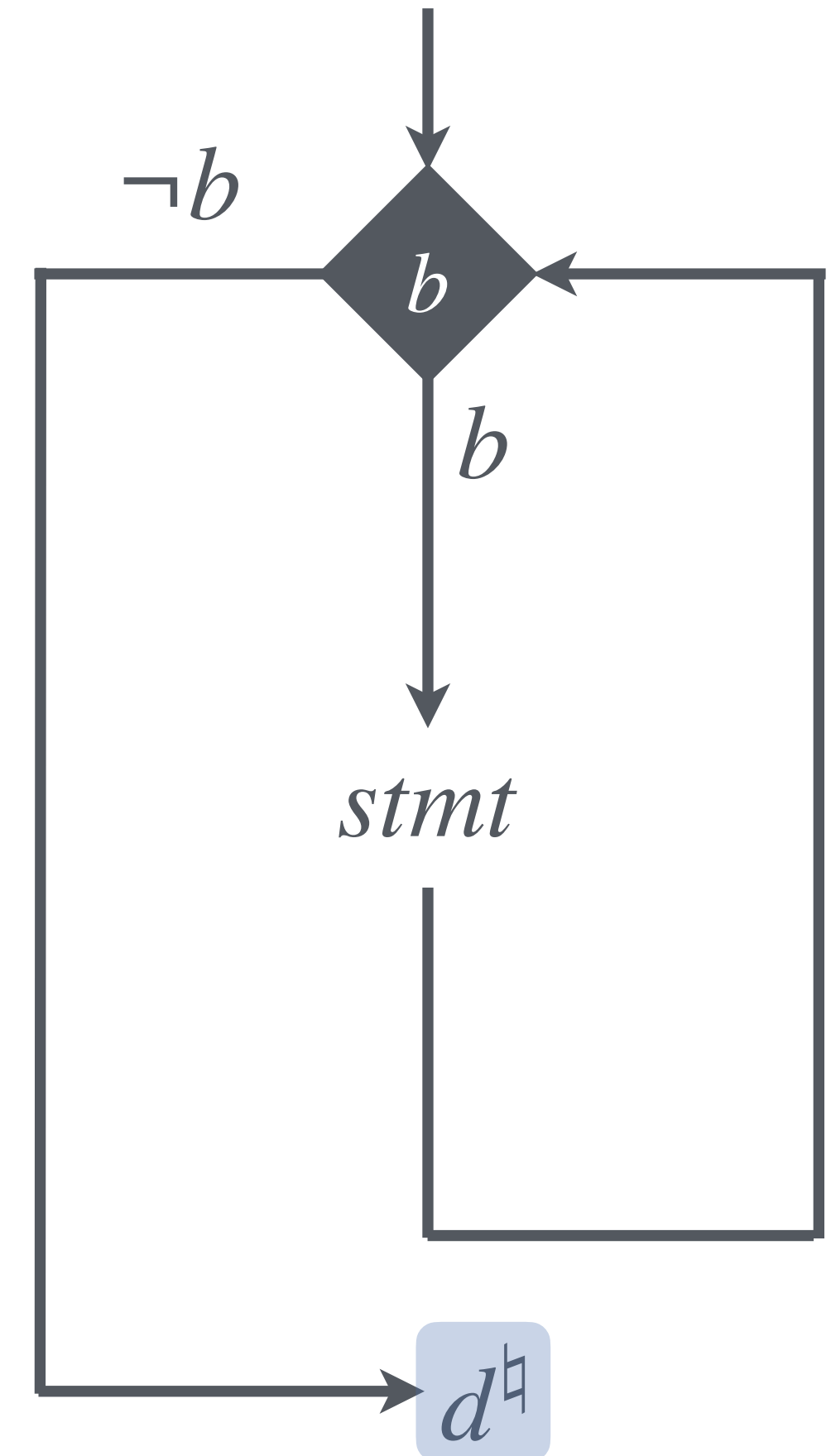
$$\Lambda^{\sharp}[[\text{if } b \text{ then } stmt \text{ else } stmt']]d^{\sharp} \triangleq$$

$$\text{Filter}^{\sharp}[[b]](\Lambda^{\sharp}[[stmt]]) \sqcup^{\sharp} \text{Filter}^{\sharp}[[\neg b]](\Lambda^{\sharp}[[stmt']])$$

$$\Lambda^{\sharp}[[\text{while } b \text{ do } stmt]]d^{\sharp} \triangleq \lim_n F^n$$

$$F(x^{\sharp}) \triangleq \text{Filter}^{\sharp}[[\neg b]]d^{\sharp} \sqcup^{\sharp}$$

$$\text{Filter}^{\sharp}[[b]](\Lambda^{\sharp}[[stmt]](\text{Substitute}^{\sharp}[[\text{counter} \leftarrow \text{counter} - 1]]x^{\sharp})))$$



Backward Abstract Semantics

starts from $\Lambda^{\sharp}[[P]](\text{counter} = 0)$

$$\Lambda^{\sharp}[[stmt; stmt']]d^{\sharp} \triangleq \Lambda^{\sharp}[[stmt]](\Lambda^{\sharp}[[stmt']]d^{\sharp})$$

$$\Lambda^{\sharp}[[\text{skip}]]d^{\sharp} \triangleq d^{\sharp}$$

$$\Lambda^{\sharp}[[x := e]]d^{\sharp} \triangleq \text{Substitute}^{\sharp}[[x \leftarrow e]]d^{\sharp}$$

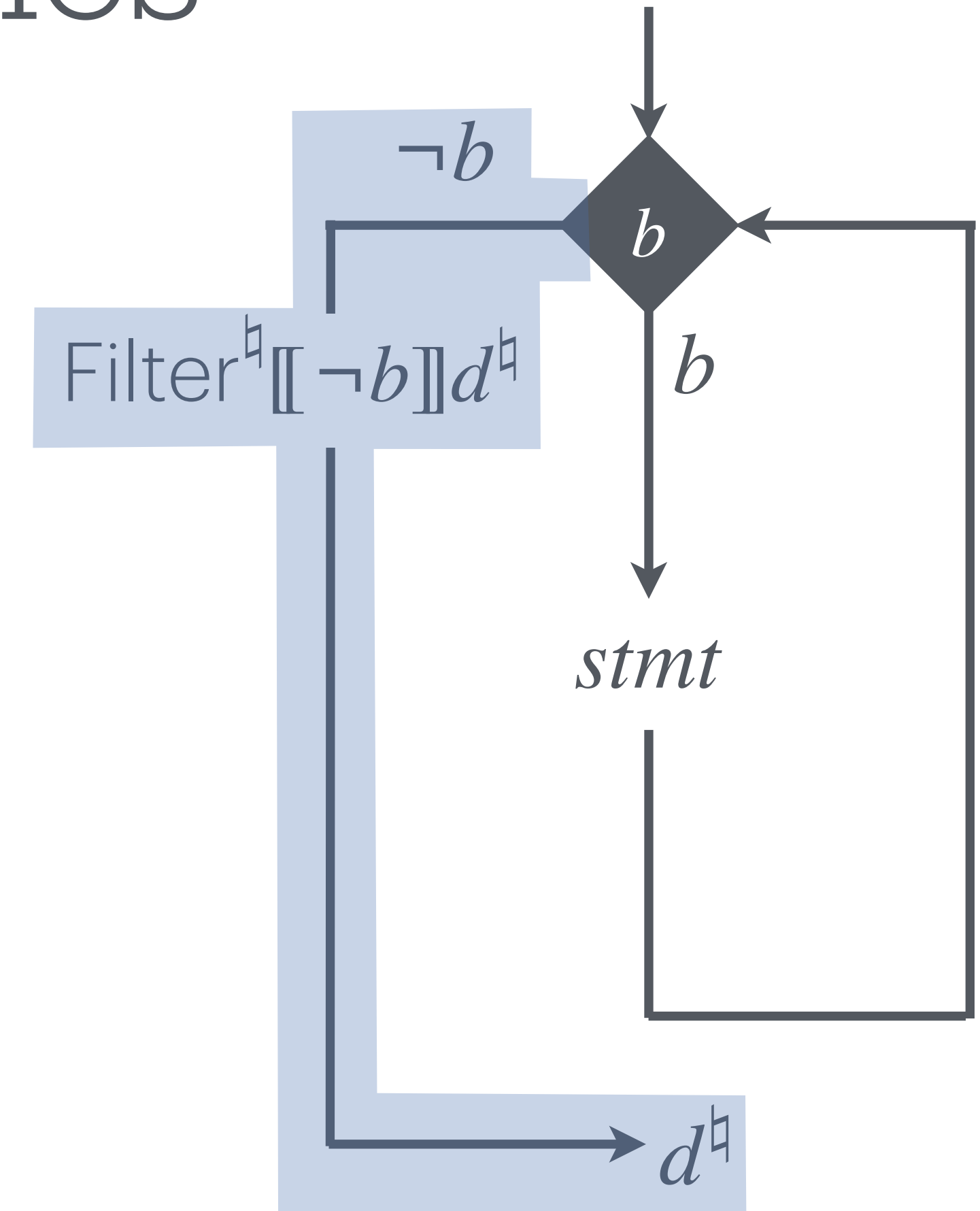
$$\Lambda^{\sharp}[[\text{if } b \text{ then } stmt \text{ else } stmt']]d^{\sharp} \triangleq$$

$$\text{Filter}^{\sharp}[[b]](\Lambda^{\sharp}[[stmt]]) \sqcup^{\sharp} \text{Filter}^{\sharp}[[\neg b]](\Lambda^{\sharp}[[stmt']])$$

$$\Lambda^{\sharp}[[\text{while } b \text{ do } stmt]]d^{\sharp} \triangleq \lim_n F^n$$

$$F(x^{\sharp}) \triangleq \text{Filter}^{\sharp}[[\neg b]]d^{\sharp} \sqcup^{\sharp}$$

$$\text{Filter}^{\sharp}[[b]](\Lambda^{\sharp}[[stmt]](\text{Substitute}^{\sharp}[[\text{counter} \leftarrow \text{counter} - 1]]x^{\sharp}))$$



Backward Abstract Semantics

starts from $\Lambda^{\sharp}[[P]](\text{counter} = 0)$

$$\Lambda^{\sharp}[[stmt; stmt']]d^{\sharp} \triangleq \Lambda^{\sharp}[[stmt]](\Lambda^{\sharp}[[stmt']]d^{\sharp})$$

$$\Lambda^{\sharp}[[\text{skip}]]d^{\sharp} \triangleq d^{\sharp}$$

$$\Lambda^{\sharp}[[x := e]]d^{\sharp} \triangleq \text{Substitute}^{\sharp}[[x \leftarrow e]]d^{\sharp}$$

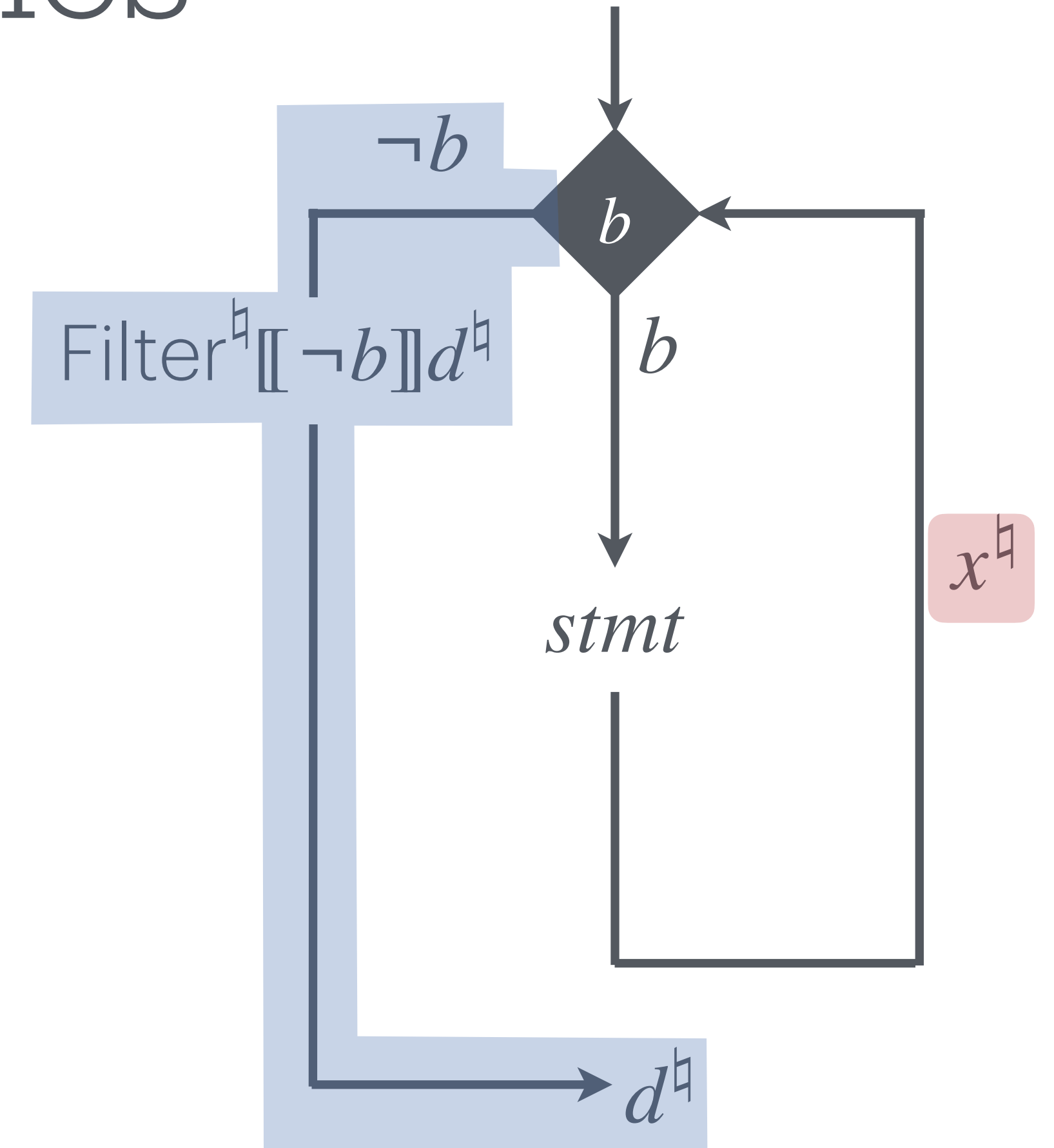
$$\Lambda^{\sharp}[[\text{if } b \text{ then } stmt \text{ else } stmt']]d^{\sharp} \triangleq$$

$$\text{Filter}^{\sharp}[[b]](\Lambda^{\sharp}[[stmt]]) \sqcup^{\sharp} \text{Filter}^{\sharp}[[\neg b]](\Lambda^{\sharp}[[stmt']])$$

$$\Lambda^{\sharp}[[\text{while } b \text{ do } stmt]]d^{\sharp} \triangleq \lim_n F^n$$

$$F(x^{\sharp}) \triangleq \text{Filter}^{\sharp}[[\neg b]]d^{\sharp} \sqcup^{\sharp}$$

$$\text{Filter}^{\sharp}[[b]](\Lambda^{\sharp}[[stmt]](\text{Substitute}^{\sharp}[[\text{counter} \leftarrow \text{counter} - 1]]x^{\sharp})))$$



Backward Abstract Semantics

starts from $\Lambda^{\sharp}[[P]](\text{counter} = 0)$

$$\Lambda^{\sharp}[[stmt; stmt']]d^{\sharp} \triangleq \Lambda^{\sharp}[[stmt]](\Lambda^{\sharp}[[stmt']]d^{\sharp})$$

$$\Lambda^{\sharp}[[\text{skip}]]d^{\sharp} \triangleq d^{\sharp}$$

$$\Lambda^{\sharp}[[x := e]]d^{\sharp} \triangleq \text{Substitute}^{\sharp}[[x \leftarrow e]]d^{\sharp}$$

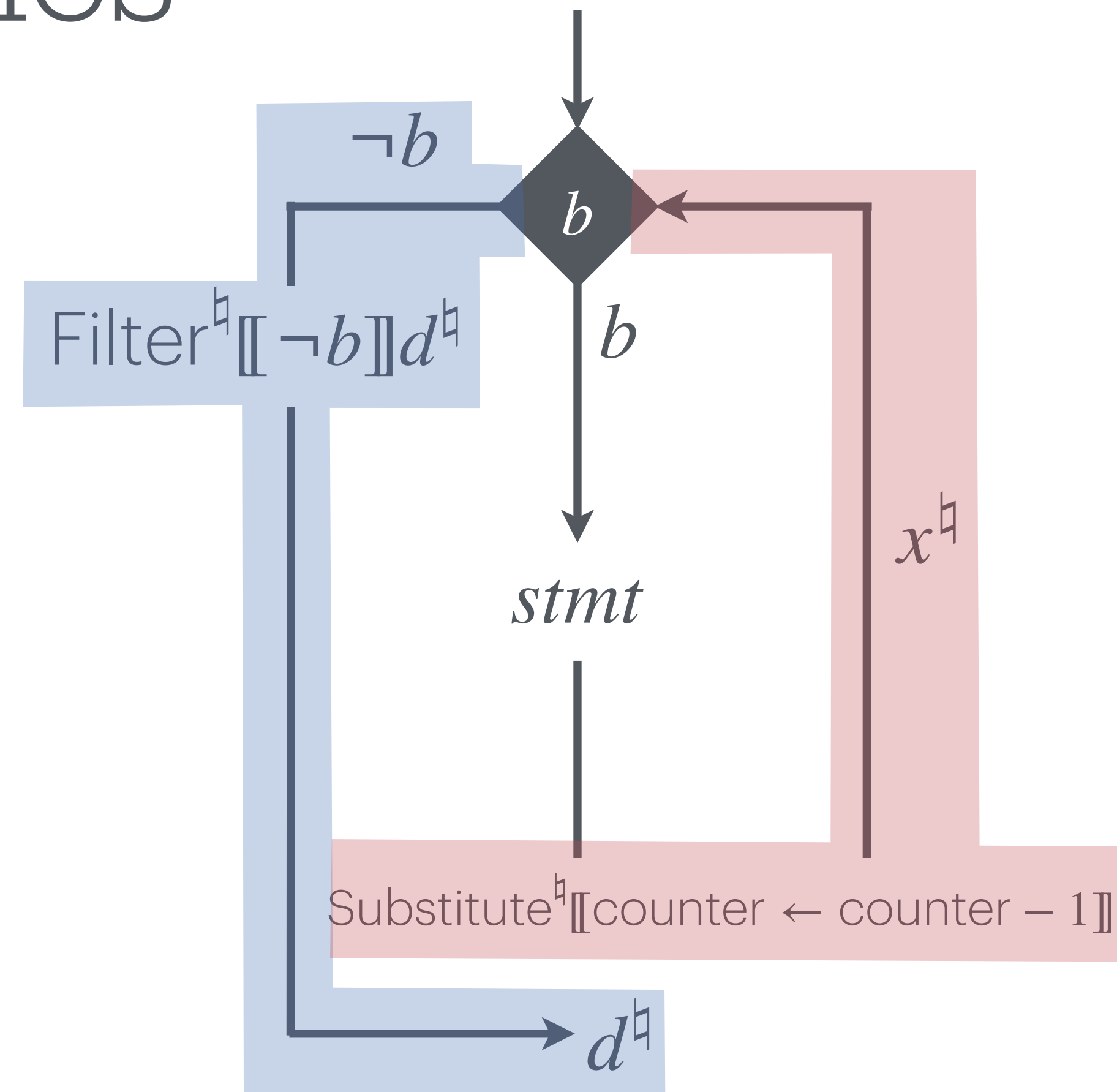
$$\Lambda^{\sharp}[[\text{if } b \text{ then } stmt \text{ else } stmt']]d^{\sharp} \triangleq$$

$$\text{Filter}^{\sharp}[[b]](\Lambda^{\sharp}[[stmt]]) \sqcup^{\sharp} \text{Filter}^{\sharp}[[\neg b]](\Lambda^{\sharp}[[stmt']])$$

$$\Lambda^{\sharp}[[\text{while } b \text{ do } stmt]]d^{\sharp} \triangleq \lim_n F^n$$

$$F(x^{\sharp}) \triangleq \text{Filter}^{\sharp}[[\neg b]]d^{\sharp} \sqcup^{\sharp}$$

$$\text{Filter}^{\sharp}[[b]](\Lambda^{\sharp}[[stmt]](\text{Substitute}^{\sharp}[[\text{counter} \leftarrow \text{counter} - 1]]x^{\sharp})))$$



Backward Abstract Semantics

starts from $\Lambda^{\sharp}[[P]]$ **counter = 0**

$$\Lambda^{\sharp}[[stmt; stmt']]d^{\sharp} \triangleq \Lambda^{\sharp}[[stmt]](\Lambda^{\sharp}[[stmt']]d^{\sharp})$$

$$\Lambda^{\sharp}[[skip]]d^{\sharp} \triangleq d^{\sharp}$$

$$\Lambda^{\sharp}[[x := e]]d^{\sharp} \triangleq \text{Substitute}^{\sharp}[[x \leftarrow e]]d^{\sharp}$$

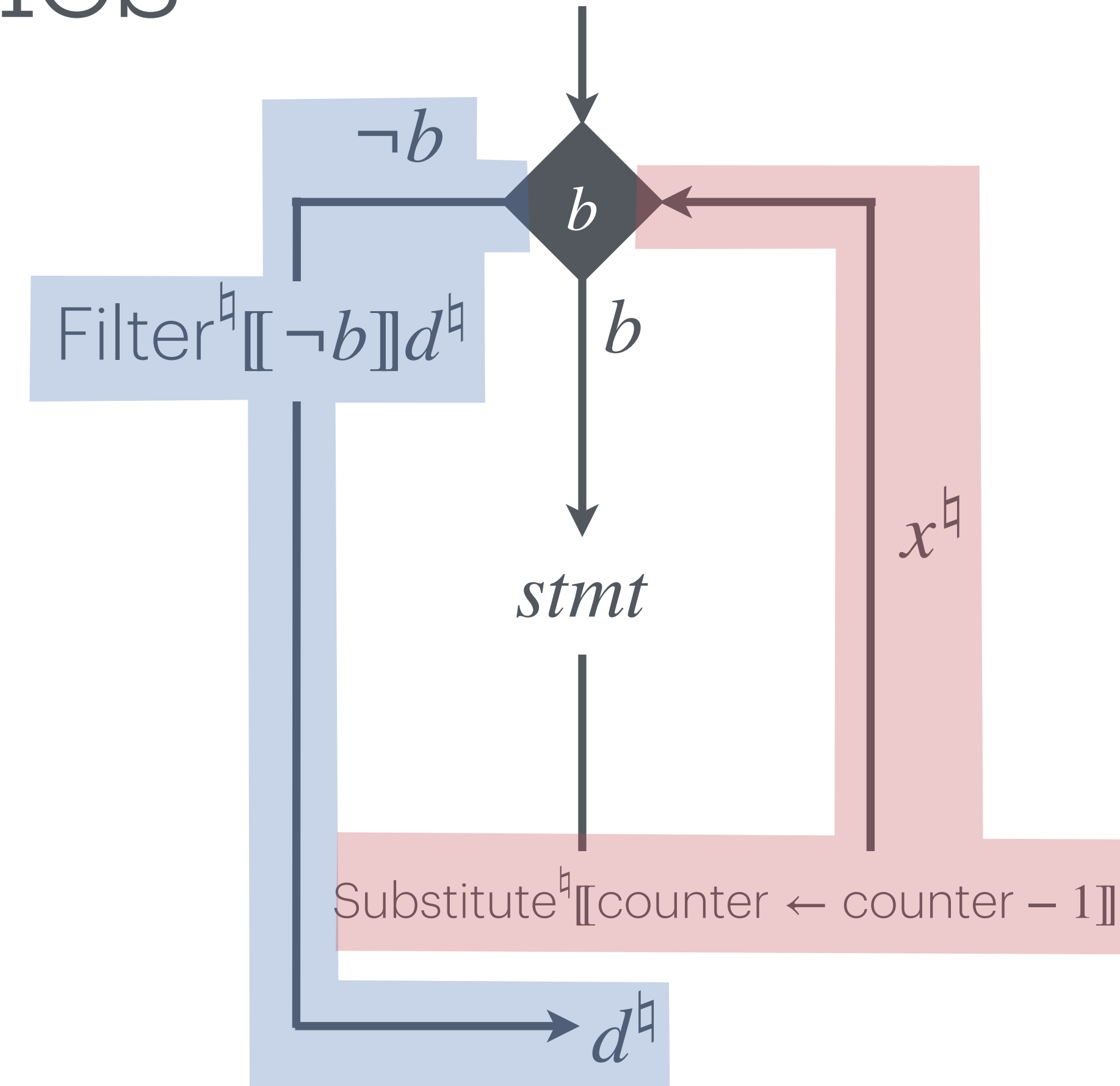
$$\Lambda^{\sharp}[[\text{if } b \text{ then } stmt \text{ else } stmt']]d^{\sharp} \triangleq$$

$$\text{Filter}^{\sharp}[[b]](\Lambda^{\sharp}[[stmt]]) \sqcup^{\sharp} \text{Filter}^{\sharp}[[\neg b]](\Lambda^{\sharp}[[stmt']])$$

$$\Lambda^{\sharp}[[\text{while } b \text{ do } stmt]]d^{\sharp} \triangleq \lim_n F^n$$

$$F(x^{\sharp}) \triangleq \text{Filter}^{\sharp}[[\neg b]]d^{\sharp} \sqcup^{\sharp}$$

$$\text{Filter}^{\sharp}[[b]](\Lambda^{\sharp}[[stmt]](\text{Substitute}^{\sharp}[[\text{counter} \leftarrow \text{counter} - 1]]x^{\sharp})))$$



Backward Abstract Semantics

starts from $\Lambda^{\sharp}[[P]](\text{counter} = 0)$

$$\Lambda^{\sharp}[[stmt; stmt']]d^{\sharp} \triangleq \Lambda^{\sharp}[[stmt]](\Lambda^{\sharp}[[stmt']]d^{\sharp})$$

$$\Lambda^{\sharp}[[\text{skip}]]d^{\sharp} \triangleq d^{\sharp}$$

$$\Lambda^{\sharp}[[x := e]]d^{\sharp} \triangleq \text{Substitute}^{\sharp}[[x \leftarrow e]]d^{\sharp}$$

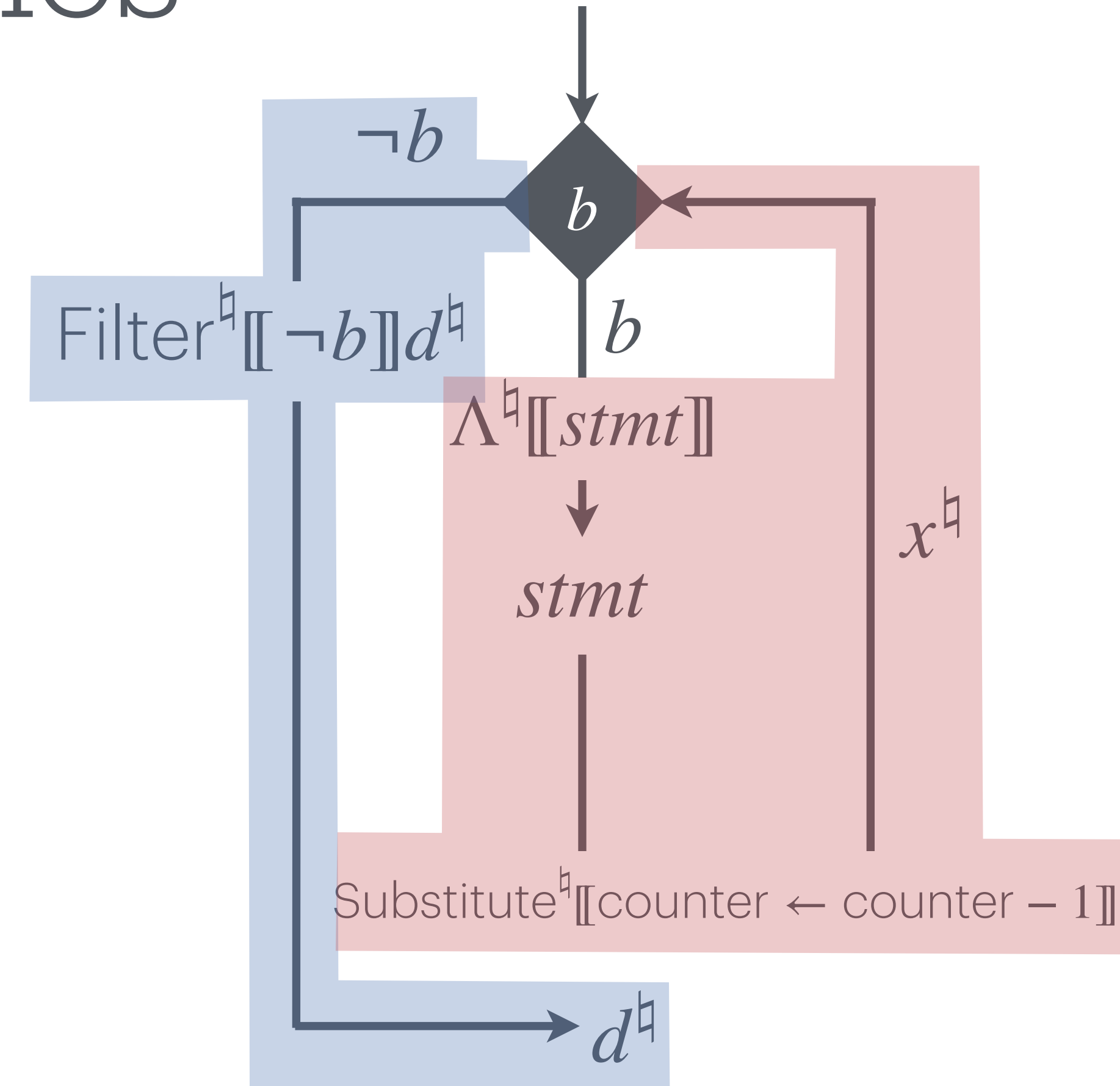
$$\Lambda^{\sharp}[[\text{if } b \text{ then } stmt \text{ else } stmt']]d^{\sharp} \triangleq$$

$$\text{Filter}^{\sharp}[[b]](\Lambda^{\sharp}[[stmt]]) \sqcup^{\sharp} \text{Filter}^{\sharp}[[\neg b]](\Lambda^{\sharp}[[stmt']])$$

$$\Lambda^{\sharp}[[\text{while } b \text{ do } stmt]]d^{\sharp} \triangleq \lim_n F^n$$

$$F(x^{\sharp}) \triangleq \text{Filter}^{\sharp}[[\neg b]]d^{\sharp} \sqcup^{\sharp}$$

$$\text{Filter}^{\sharp}[[b]](\Lambda^{\sharp}[[stmt]](\text{Substitute}^{\sharp}[[\text{counter} \leftarrow \text{counter} - 1]]x^{\sharp}))$$



Backward Abstract Semantics

starts from $\Lambda^{\sharp}[[P]](\text{counter} = 0)$

$$\Lambda^{\sharp}[[stmt; stmt']]d^{\sharp} \triangleq \Lambda^{\sharp}[[stmt]](\Lambda^{\sharp}[[stmt']]d^{\sharp})$$

$$\Lambda^{\sharp}[[\text{skip}]]d^{\sharp} \triangleq d^{\sharp}$$

$$\Lambda^{\sharp}[[x := e]]d^{\sharp} \triangleq \text{Substitute}^{\sharp}[[x \leftarrow e]]d^{\sharp}$$

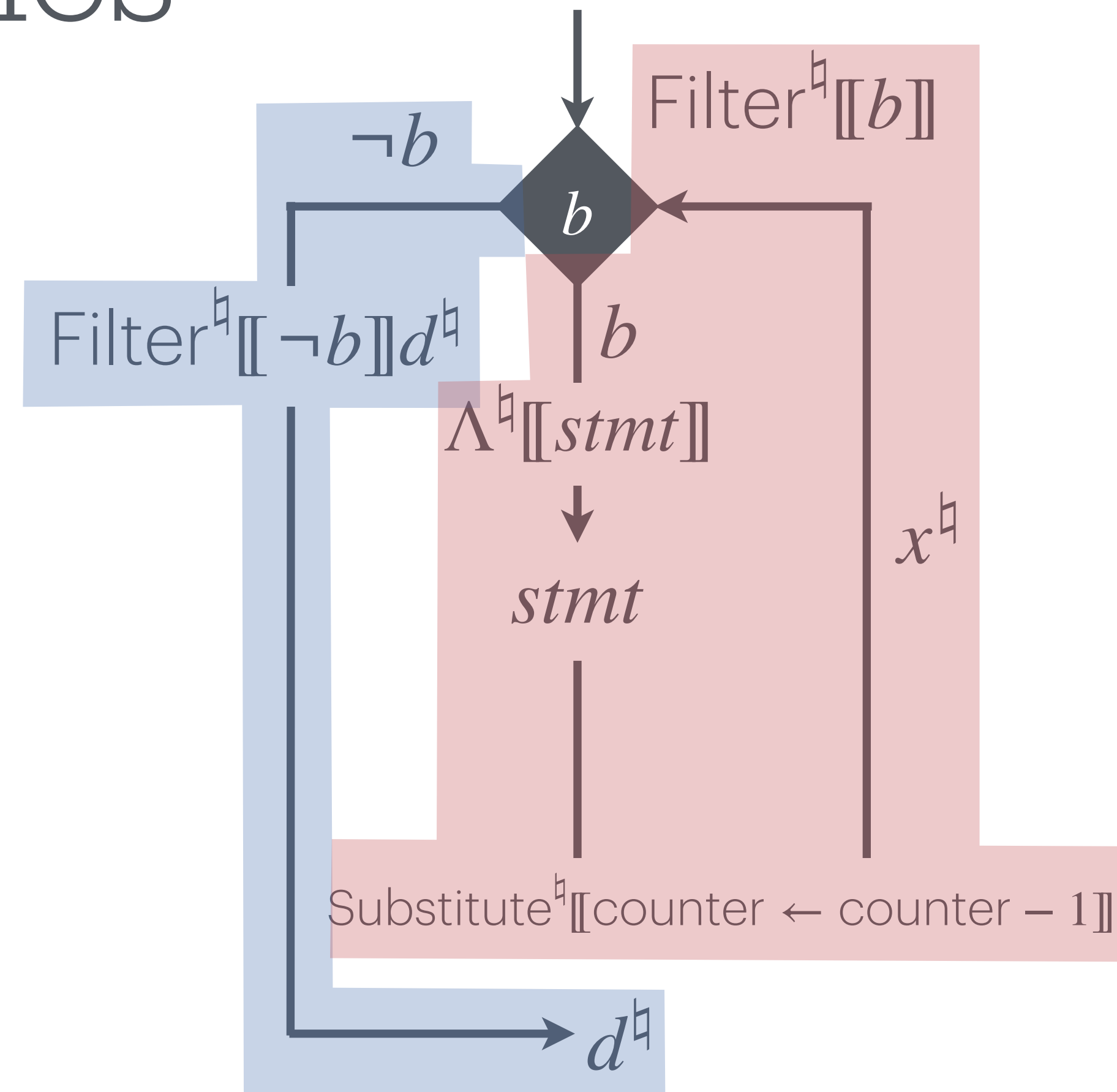
$$\Lambda^{\sharp}[[\text{if } b \text{ then } stmt \text{ else } stmt']]d^{\sharp} \triangleq$$

$$\text{Filter}^{\sharp}[[b]](\Lambda^{\sharp}[[stmt]]) \sqcup^{\sharp} \text{Filter}^{\sharp}[[\neg b]](\Lambda^{\sharp}[[stmt']])$$

$$\Lambda^{\sharp}[[\text{while } b \text{ do } stmt]]d^{\sharp} \triangleq \lim_n F^n$$

$$F(x^{\sharp}) \triangleq \text{Filter}^{\sharp}[[\neg b]]d^{\sharp} \sqcup^{\sharp}$$

$$\text{Filter}^{\sharp}[[b]](\Lambda^{\sharp}[[stmt]](\text{Substitute}^{\sharp}[[\text{counter} \leftarrow \text{counter} - 1]]x^{\sharp}))$$



(ii) Abstract Interpretation

Abstract invariant on the
input variables + counter

Backward abstract
analysis

```
1 def Add(p, z, m, x, n, y):
2     r = min(p, m)
3     s = min(p, n)
4     if (r < s):
5         t = p - s
6         q = s - r
9         for (; r > 0; r--):
--             skip; counter--
17        do:
23            q--; counter--
25        while (q > 0)
26    else:
27        t = p - r
28        q = r - s
31        for (; s > 0; s--):
--            skip; counter--
39        for (; q > 0; q--):
--            skip; counter--
45        if (t > 0):
47            while (t > 0):
49                t--; counter--
--    assert counter == 0
```

Augment each
loop body with a **counter**
for **iterations**

Backwards starting
from zero!

At the beginning, the
counter yields the **global**
number of iterations

(ii) Abstract Interpretation

Abstract invariant on the
input variables + counter

Forward +

Backward abstract
analysis

```
1 def Add(p, z, m, x, n, y):
2   r = min(p, m)
3   s = min(p, n)
4   if (r < s):
5     t = p - s
6     q = s - r
9     for (; r > 0; r--):
--      skip; counter--
17    do:
23      q--; counter--
25      while (q > 0)
26    else:
27      t = p - r
28      q = r - s
31      for (; s > 0; s--):
--        skip; counter--
39      for (; q > 0; q--):
--        skip; counter--
45      if (t > 0):
47        while (t > 0):
49          t--; counter--
--      assert counter == 0
```

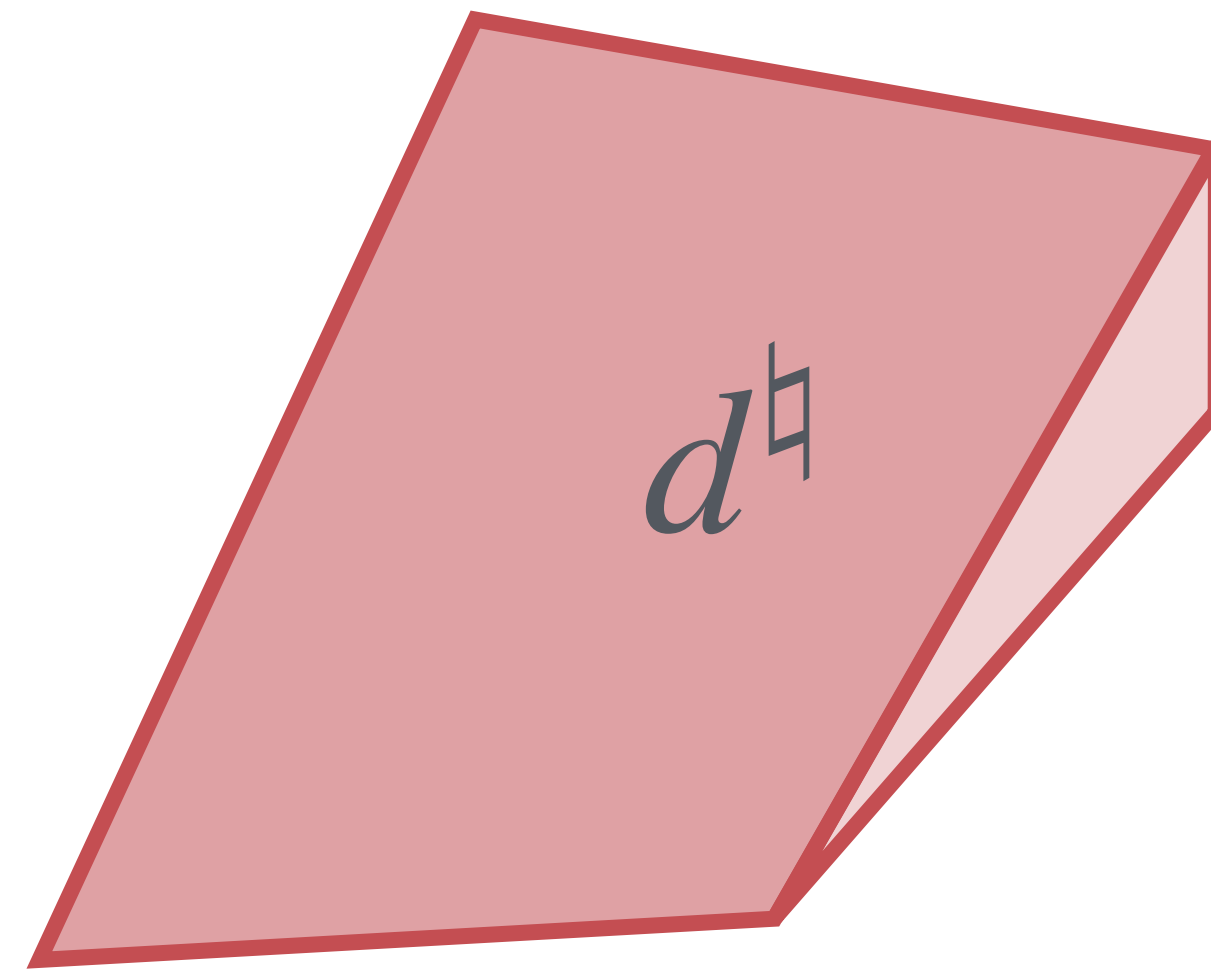
Augment each
loop body with a **counter**
for **iterations**

Backwards starting
from zero!

At the beginning, the
counter yields the **global**
number of iterations

(iii) Impact Quantification

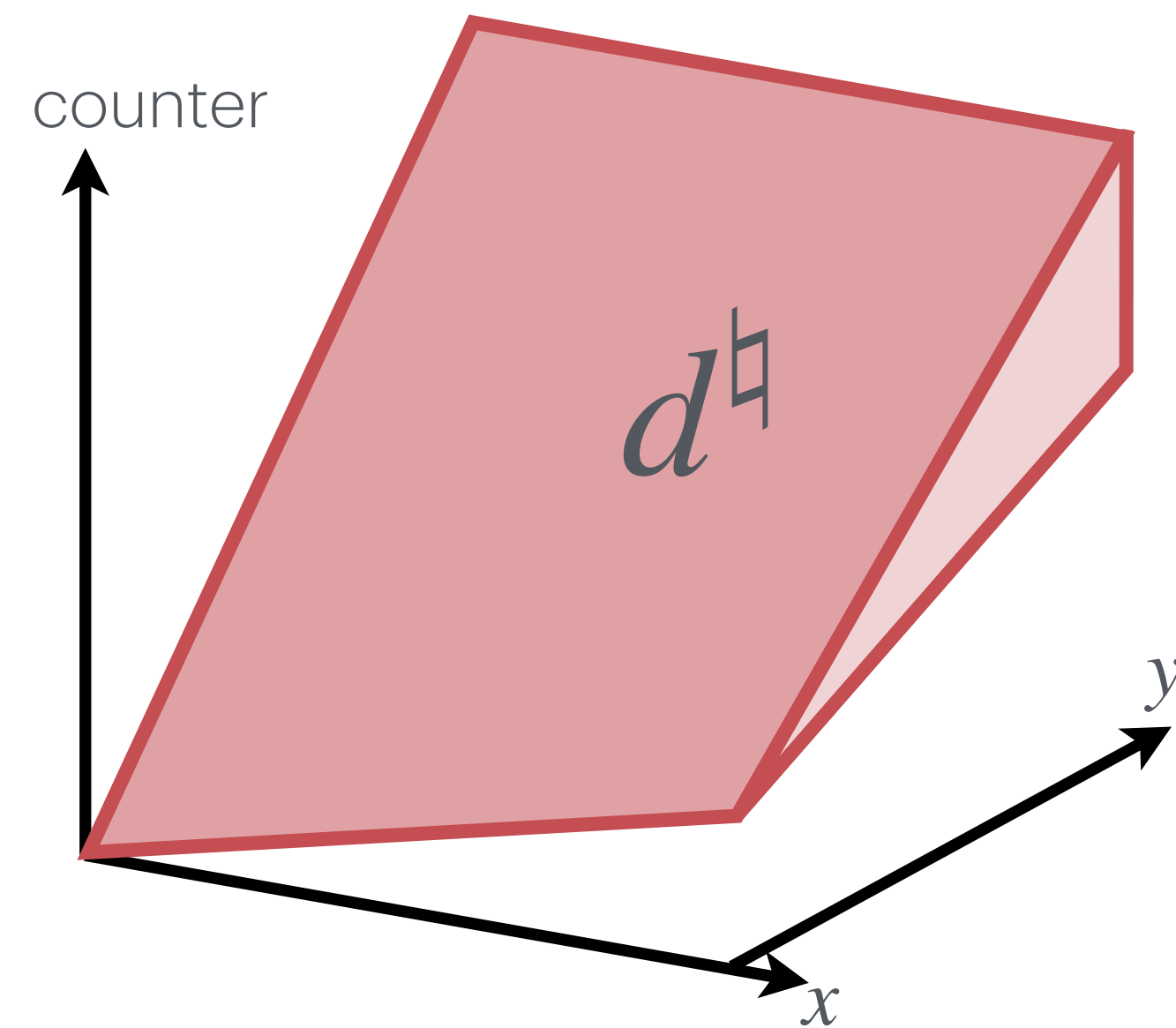
Abstract invariant on the
input variables + counter



(iii) Impact Quantification

Abstract invariant on the
input variables + counter

(i) Assume **input variable** of interest x

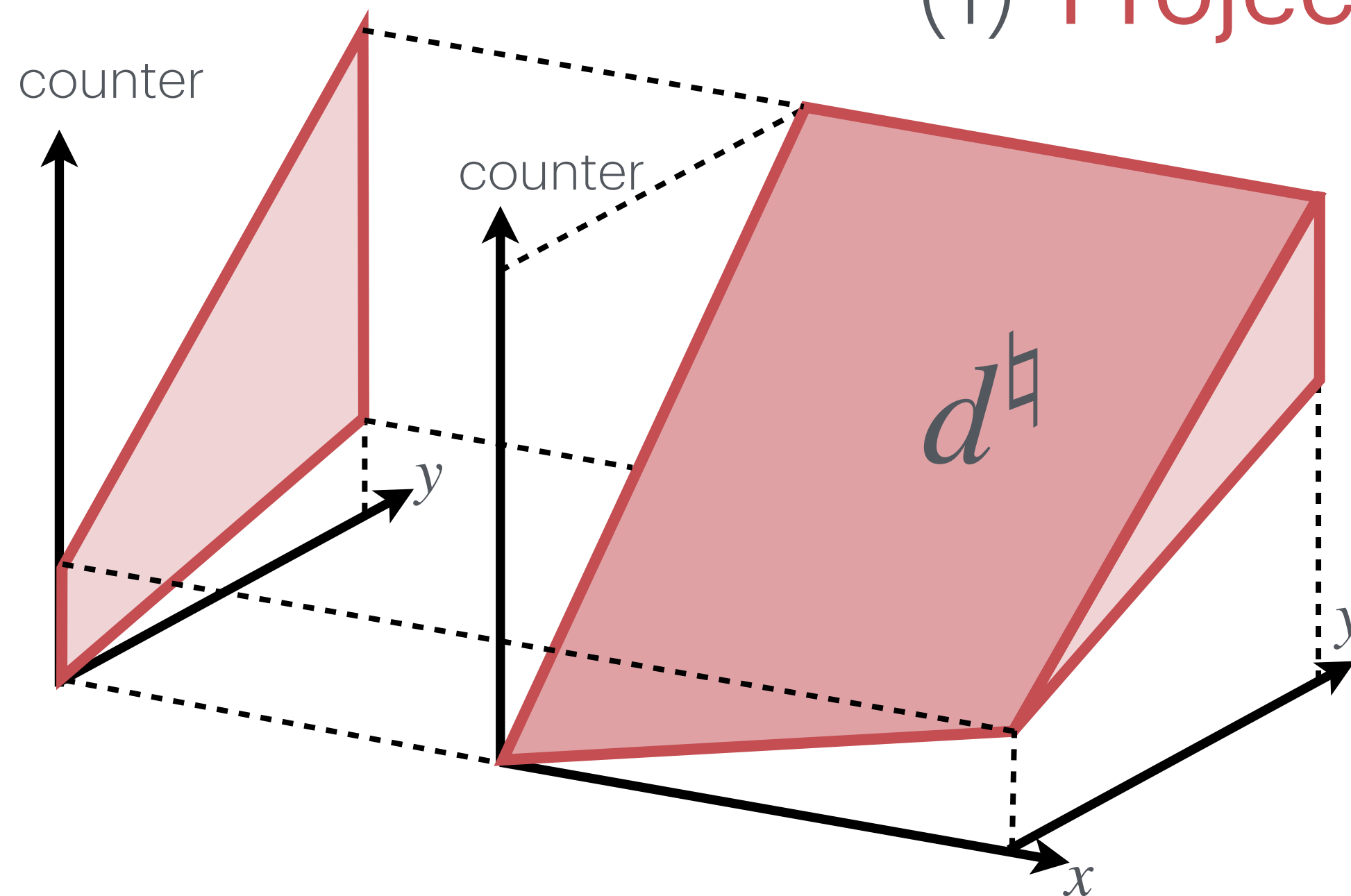


(iii) Impact Quantification

Abstract invariant on the
input variables + counter

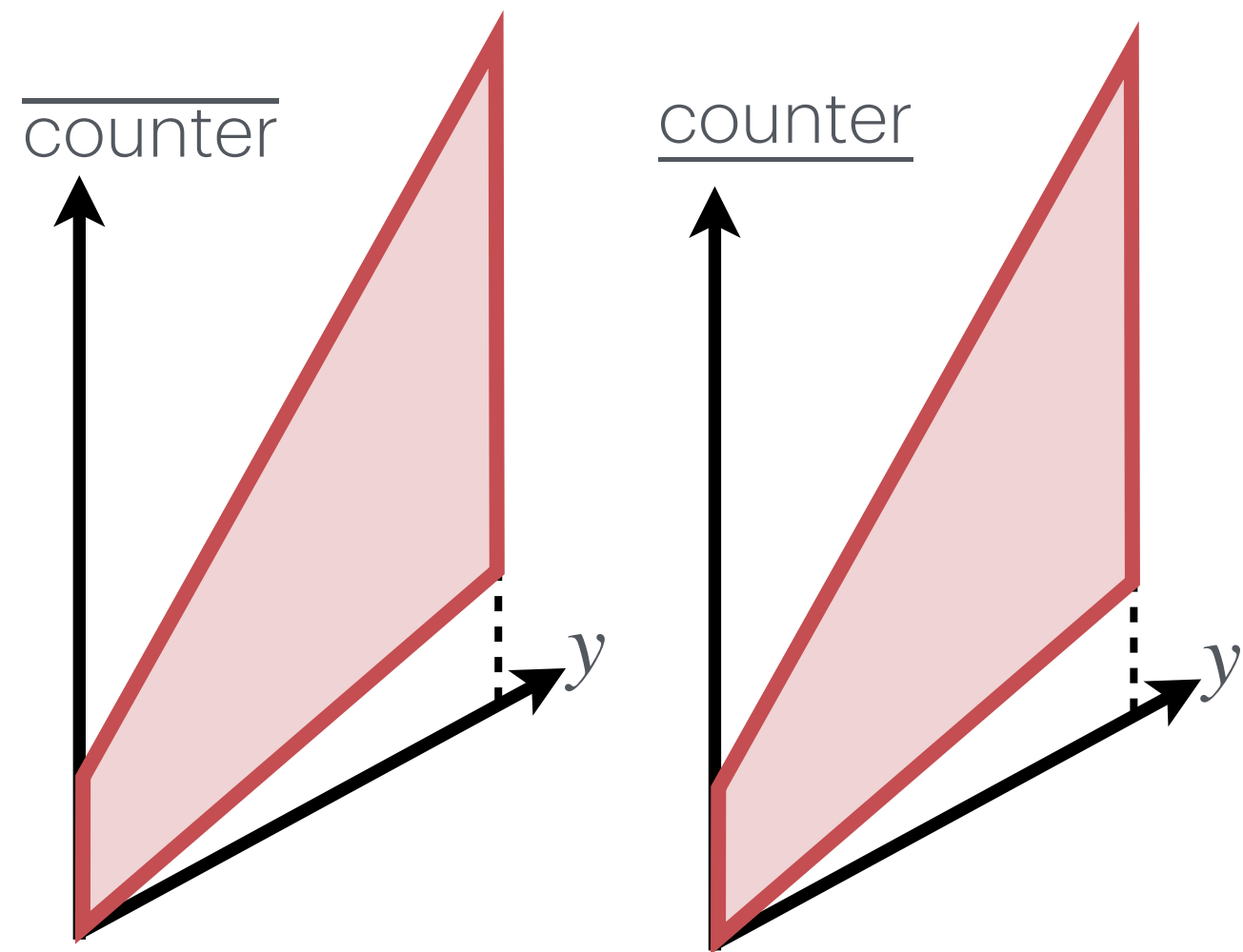
(i) Assume **input variable** of interest x

(1) **Project** away x



(iii) Impact Quantification

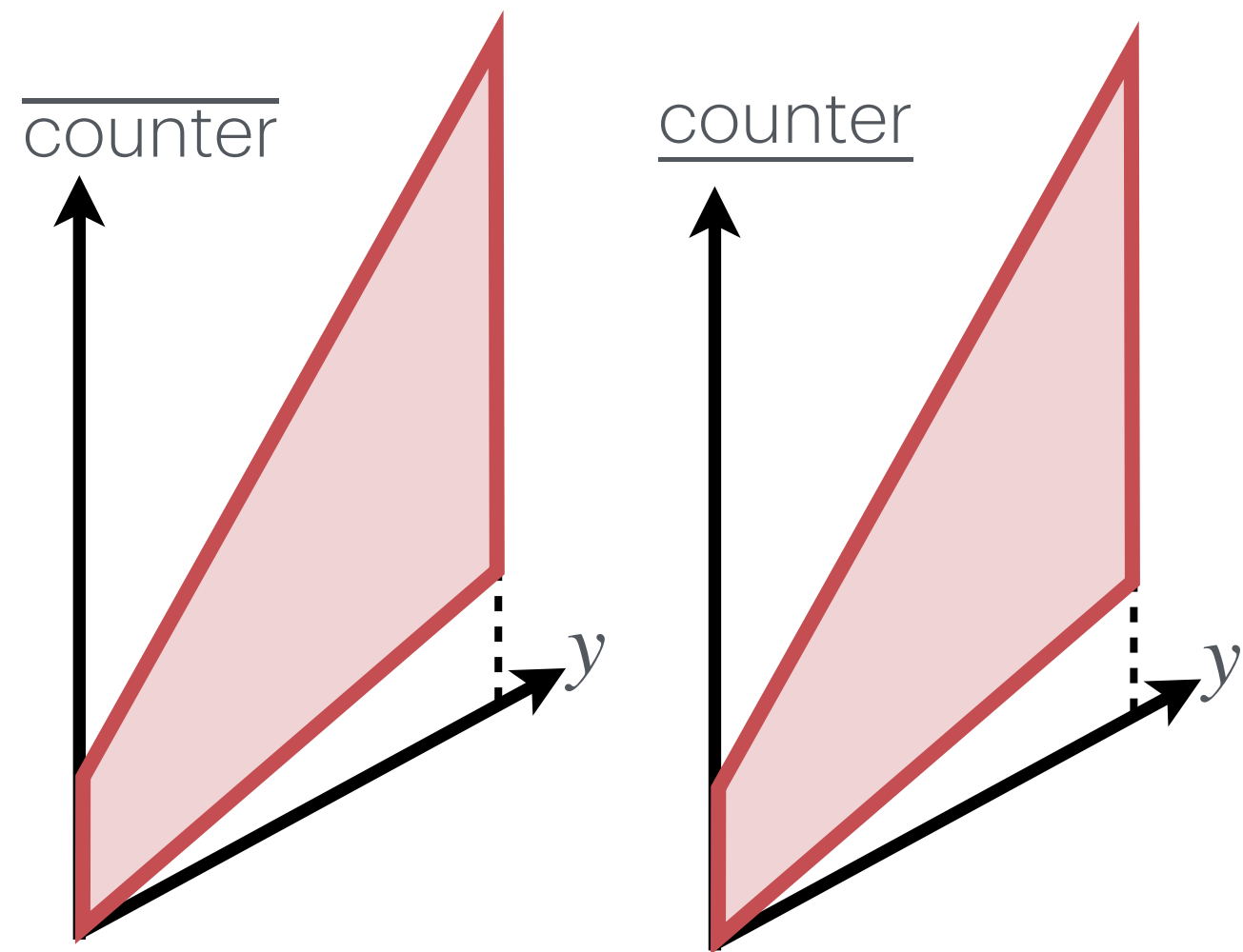
Abstract invariant on the
input variables + counter



- (i) Assume **input variable** of interest x
- (1) **Project** away x
- (2) **Duplicate** the invariant and **substitute** the counter with counter and counter

(iii) Impact Quantification

Abstract invariant on the
input variables + counter

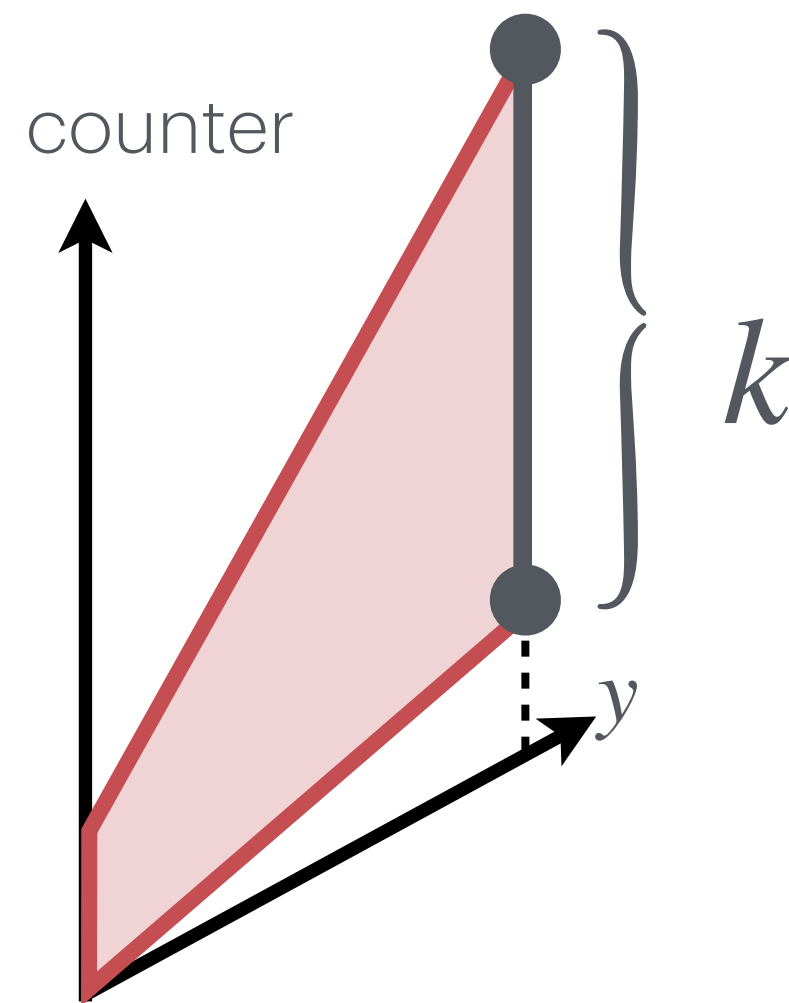


$$0 \leq k \leq \overline{\text{counter}} - \underline{\text{counter}}$$

- (i) Assume **input variable** of interest x
- (1) **Project** away x
- (2) **Duplicate** the invariant and **substitute** the counter with $\underline{\text{counter}}$ and $\overline{\text{counter}}$
- (3) **Maximize** the distance between the two

(iii) Impact Quantification

Abstract invariant on the
input variables + counter



$$0 \leq k \leq \overline{\text{counter}} - \underline{\text{counter}}$$

- (i) Assume **input variable** of interest x
- (1) **Project** away x
- (2) **Duplicate** the invariant and **substitute** the counter with counter and $\overline{\text{counter}}$
- (3) **Maximize** the distance between the two

k is the impact of x

(iii) Impact Quantification

$$\text{Impact}_x^{\sharp}(d^{\sharp})$$

(iii) Impact Quantification

$$\text{Impact}_x^{\sharp}(d^{\sharp}) = \max k \text{ subject to}$$

$$0 \leq k \leq \overline{\text{counter}} - \underline{\text{counter}}$$

(iii) Impact Quantification

$\text{Impact}_x^{\sharp}(d^{\sharp}) = \max k$ subject to

$\text{Project}_x(d^{\sharp})$

$$0 \leq k \leq \overline{\text{counter}} - \underline{\text{counter}}$$

(iii) Impact Quantification

$\text{Impact}_x^{\sharp}(d^{\sharp}) = \max k$ subject to

$\text{Substitute}[\text{counter} \leftarrow \overline{\text{counter}}](\text{Project}_x(d^{\sharp})) \wedge$

$\text{Substitute}[\text{counter} \leftarrow \underline{\text{counter}}](\text{Project}_x(d^{\sharp})) \wedge$

$0 \leq k \leq \overline{\text{counter}} - \underline{\text{counter}}$

abstract

concrete

$$\text{Impact}_x^{\sharp}(\mathbf{P}) \geq \text{IMPACT}_x(\mathbf{P})$$

abstract

concrete

$$\text{Impact}_x^{\Downarrow}(\mathbf{P}) \leq k$$



input variable x has an
impact below k on the
iterations of the program \mathbf{P}

$$\text{Impact}_x^{\Downarrow}(\mathbf{P}) \geq \text{IMPACT}_x(\mathbf{P})$$

Add Function

d^{\sharp} is

$$p = \text{counter} \wedge 0 \leq p \leq u$$

Forward +
Backward abstract
analysis

```
1 def Add(p, z, m, x, n, y):
2   r = min(p, m)
3   s = min(p, n)
4   if (r < s):
5     t = p - s
6     q = s - r
9     for (; r > 0; r--):
--      T skip; counter--
17    do:
23      T q--; counter--
25      while (q > 0)
26    else:
27      T t = p - r
28      q = r - s
31      for (; s > 0; s--):
--       T skip; counter--
39      for (; q > 0; q--):
--       T skip; counter--
45      if (t > 0):
47        T while (t > 0):
49          T t--; counter--
--      assert counter = 0
```

Add Function

d^\sharp is

$$p = \text{counter} \wedge 0 \leq p \leq u$$

$\text{Impact}_x^\sharp(d^\sharp) = \max k$ subject to

Substitute[[counter \leftarrow $\overline{\text{counter}}$]](Project _{x} (d^\sharp)) \wedge

Substitute[[counter \leftarrow counter]](Project _{x} (d^\sharp)) \wedge

$$0 \leq k \leq \overline{\text{counter}} - \underline{\text{counter}}$$

Add Function

d^{\sharp} is

$$p = \text{counter} \wedge 0 \leq p \leq u$$

$$\text{Impact}_{\text{p}}^{\sharp}(p = \text{counter} \wedge 0 \leq p \leq u) =$$

max k subject to

$$0 \leq \overline{\text{counter}} \leq u \wedge$$

$$0 \leq \underline{\text{counter}} \leq u \wedge$$

$$0 \leq k \leq \overline{\text{counter}} - \underline{\text{counter}}$$

Add Function

d^{\sharp} is

$$p = \text{counter} \wedge 0 \leq p \leq u$$

$$\text{Impact}_p^{\sharp}(p = \text{counter} \wedge 0 \leq p \leq u) = \left. \begin{array}{l} \text{max } k \text{ subject to} \\ 0 \leq \overline{\text{counter}} \leq u \wedge \\ 0 \leq \underline{\text{counter}} \leq u \wedge \\ 0 \leq k \leq \overline{\text{counter}} - \underline{\text{counter}} \end{array} \right\} u$$

Add Function

d^\sharp is

$$p = \text{counter} \wedge 0 \leq p \leq u$$

$$\text{Impact}_n^\sharp(p = \text{counter} \wedge 0 \leq p \leq u) =$$

$\max k$ subject to

$$p = \overline{\text{counter}} \wedge$$

$$p = \underline{\text{counter}} \wedge$$

$$0 \leq p \leq u \wedge$$

$$0 \leq k \leq \overline{\text{counter}} - \underline{\text{counter}}$$

Add Function

d^\sharp is

$$p = \text{counter} \wedge 0 \leq p \leq u$$

$$\text{Impact}_n^\sharp(p = \text{counter} \wedge 0 \leq p \leq u) = \left. \begin{array}{l} \max k \text{ subject to} \\ p = \overline{\text{counter}} \wedge \\ p = \underline{\text{counter}} \wedge \\ 0 \leq p \leq u \wedge \\ 0 \leq k \leq \overline{\text{counter}} - \underline{\text{counter}} \end{array} \right\} 0$$

Add Function

d^\sharp is

$$p = \text{counter} \wedge 0 \leq p \leq u$$

$$\text{Impact}_n^\sharp(p = \text{counter} \wedge 0 \leq p \leq u) = \left. \begin{array}{l} \text{max } k \text{ subject to} \\ p = \overline{\text{counter}} \wedge \\ p = \underline{\text{counter}} \wedge \\ 0 \leq p \leq u \wedge \\ 0 \leq k \leq \overline{\text{counter}} - \underline{\text{counter}} \end{array} \right\} 0$$

Add Function

d^\sharp is

$$p = \text{counter} \wedge 0 \leq p \leq u$$

$$\text{Impact}_n^\sharp(p = \text{counter} \wedge 0 \leq p \leq u) = \left. \begin{array}{l} \text{max } k \text{ subject to} \\ p = \overline{\text{counter}} \wedge \\ p = \underline{\text{counter}} \wedge \\ 0 \leq p \leq u \wedge \\ 0 \leq k \leq \overline{\text{counter}} - \underline{\text{counter}} \end{array} \right\} \begin{array}{l} 0 \\ \text{And to all the other} \\ \text{input variables} \end{array}$$

Add Function

$$\text{Impact}_p^{\sharp}(d^{\sharp}) = u$$

d^{\sharp} is

$$p = \text{counter} \wedge 0 \leq p \leq u$$

```
1 def Add(p, z, m, x, n, y):
2     r = min(p, m)
3     s = min(p, n)
4     if (r < s):
5         t = p - s
6         q = s - r
9         for (; r > 0; r--):
--             T skip; counter--
17        do:
23            I q--; counter--
25        while (q > 0)
26    else:
27        t = p - r
28        q = r - s
31        for (; s > 0; s--):
--            T skip; counter--
39        for (; q > 0; q--):
--            T skip; counter--
45    if (t > 0):
47        T while (t > 0):
49            T t--; counter--
--    assert counter = 0
```

Add Function

$$\text{Impact}_p^{\sharp}(d^{\sharp}) = u$$

$$\text{Impact}_m^{\sharp}(d^{\sharp}) = 0$$

$$\text{Impact}_n^{\sharp}(d^{\sharp}) = 0$$

$$\text{Impact}_x^{\sharp}(d^{\sharp}) = 0$$

$$\text{Impact}_y^{\sharp}(d^{\sharp}) = 0$$

$$\text{Impact}_z^{\sharp}(d^{\sharp}) = 0$$

d^{\sharp} is

$$p = \text{counter} \wedge 0 \leq p \leq u$$

```
1 def Add(p, z, m, x, n, y):
2     r = min(p, m)
3     s = min(p, n)
4     if (r < s):
5         t = p - s
6         q = s - r
9         for (; r > 0; r--):
--             T skip; counter--
17        do:
23            I q--; counter--
25            while (q > 0)
26        else:
27            t = p - r
28            q = r - s
31            for (; s > 0; s--):
--                T skip; counter--
39            for (; q > 0; q--):
--                T skip; counter--
45            if (t > 0):
47                T while (t > 0):
49                    T t--; counter--
--            assert counter = 0
```

Add Function

$$\text{Impact}_p^{\sharp}(d^{\sharp}) = u$$

$$\text{Impact}_m^{\sharp}(d^{\sharp}) = 0$$

$$\text{Impact}_n^{\sharp}(d^{\sharp}) = 0$$

$$\text{Impact}_x^{\sharp}(d^{\sharp}) = 0$$

$$\text{Impact}_y^{\sharp}(d^{\sharp}) = 0$$

$$\text{Impact}_z^{\sharp}(d^{\sharp}) = 0$$

d^{\sharp} is

$$p = \text{counter} \wedge 0 \leq p \leq u$$

```
1 def Add(p, z, m, x, n, y):
2     r = min(p, m)
3     s = min(p, n)
4     if (r < s):
5         t = p - s
6         q = s - r
9         for (; r > 0; r--):
--             T skip; counter--
17        do:
23            I q--; counter--
25            while (q > 0)
26        else:
27            t = p - r
28            q = r - s
31            for (; s > 0; s--):
--                T skip; counter--
39            for (; q > 0; q--):
--                T skip; counter--
45            if (t > 0):
47                T while (t > 0):
49                    T t--; counter--
--            assert counter = 0
```

Add Function

$$\text{Impact}_p^{\sharp}(d^{\sharp}) = u$$

$$\text{Impact}_m^{\sharp}(d^{\sharp}) = 0$$

$$\text{Impact}_n^{\sharp}(d^{\sharp}) = 0$$

$$\text{Impact}_x^{\sharp}(d^{\sharp}) = 0$$

$$\text{Impact}_y^{\sharp}(d^{\sharp}) = 0$$

$$\text{Impact}_z^{\sharp}(d^{\sharp}) = 0$$

```
1 def Add(p, z, m, x, n, y):
2     r = min(p, m)
3     s = min(p, n)
4     if (r < s):
5         t = p - s
6         q = s - r
9         for (; r > 0; r--):
--             T skip; counter--
17        do:
23            I q--; counter--
25            while (q > 0)
26        else:
27            t = p - r
28            q = r - s
31            for (; s > 0; s--):
--                T skip; counter--
39            for (; q > 0; q--):
--                T skip; counter--
45            if (t > 0):
47                T while (t > 0):
49                    T t--; counter--
--            assert counter = 0
```

d^{\sharp} is

$$p = \text{counter} \wedge 0 \leq p \leq u$$

The **Add** function is **safe**
from **timing side-channels**
on **input variables**

m, n, x, y, z

Add Function

$$\text{Impact}_p^{\sharp}(d^{\sharp}) = u$$

$$\text{Impact}_m^{\sharp}(d^{\sharp}) = 0$$

$$\text{Impact}_n^{\sharp}(d^{\sharp}) = 0$$

$$\text{Impact}_x^{\sharp}(d^{\sharp}) = 0$$

$$\text{Impact}_y^{\sharp}(d^{\sharp}) = 0$$

$$\text{Impact}_z^{\sharp}(d^{\sharp}) = 0$$

```
1 def Add(p, z, m, x, n, y):
2     r = min(p, m)
3     s = min(p, n)
4     if (r < s):
5         t = p - s
6         q = s - r
9         for (; r > 0; r--):
--             T skip; counter--
17        do:
23            I q--; counter--
25            while (q > 0)
26        else:
27            t = p - r
28            q = r - s
31            for (; s > 0; s--):
--                T skip; counter--
39            for (; q > 0; q--):
--                T skip; counter--
45            if (t > 0):
47                T while (t > 0):
49                    T t--; counter--
--            assert counter = 0
```

d^{\sharp} is

$$p = \text{counter} \wedge 0 \leq p \leq u$$

The **Add** function is **safe**
from **timing side-channels**
on **input variables**

m, n, **x, y, z**

data input
variables

github.com/denismazzucato/**timesec**

Python + Abstract Domain Library Apron

github.com/denismazzucato/**timesec**

Python + Abstract Domain Library Apron

S2N-Bignum

<https://github.com/aws-labs/s2n-bignum>

- used in cryptographic applications
- 72 disassembled c routines, 5984 loc
- 1172 variables (272 input variables)

github.com/denismazzucato/timesec

Python + Abstract Domain Library Apron

S2N-Bignum

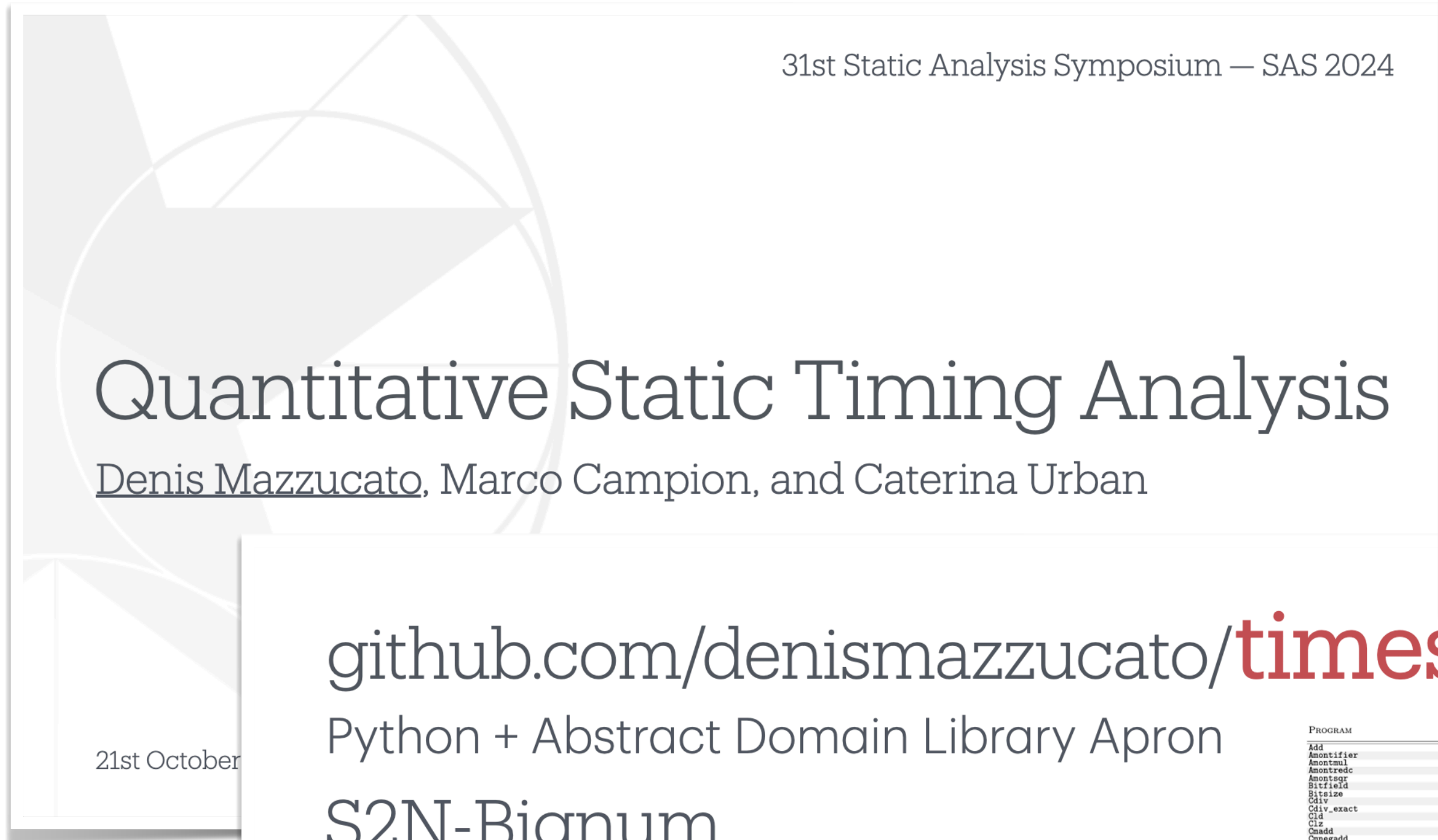
<https://github.com/aws-labs/s2n-bignum>

- used in cryptographic applications
- 72 disassembled c routines, 5984 loc
- 1172 variables (272 input variables)



Verified that the **S2N-Bignum** library is **timing side-channel free** for **data** input variables

| PROGRAM | INPUT SAFE $\Delta _S$ | VARIABLES Δ NUMERICAL $\Delta _N$ | MAYBE DANGEROUS | ZERO IMPACT |
|-----------------------------|---------------------------|---|--------------------|---------------------------|
| Add | s_1, s_3, s_5 | n_2, n_4, n_6 | s_1 | s_3, s_5, n_2, n_4, n_6 |
| Amontifier | s_1 | n_2, n_3, n_4 | s_1 | n_2, n_3, n_4 |
| Amontmul | s_1 | n_2, n_3, n_4, n_5 | s_1 | n_2, n_3, n_4, n_5 |
| Amontredc | s_1, s_3, s_6 | n_2, n_4, n_5 | s_1, s_3, s_6 | n_2, n_4, n_5 |
| Amontsqr | s_1 | n_2, n_3, n_4 | s_1 | n_2, n_3, n_4 |
| Bitfield | s_1 | n_2, n_3, n_4, n_5 | s_1 | n_2, n_3, n_4, n_5 |
| Bitsize | s_1 | n_2 | s_1 | n_2 |
| Cdiv | s_1, s_3 | n_2, n_4, n_5 | s_1, s_3 | n_2, n_4, n_5 |
| Cdiv_exact | s_1, s_3 | n_2, n_4, n_5 | s_1 | n_2, s_3, n_4, n_5 |
| Clz | s_1 | n_2 | s_1 | n_2 |
| Clz | s_1 | n_2 | s_1 | n_2 |
| Cmadd | s_1, s_4 | n_2, n_3, n_5 | s_1, s_4 | n_2, n_3, n_5 |
| Cmnegadd | s_1, s_4 | n_2, n_3, n_5 | s_1, s_4 | n_2, n_3, n_5 |
| Cmod | s_1 | n_2, n_3 | s_1 | n_2, n_3 |
| Cmul | s_1, s_4 | n_2, n_3, n_5 | s_1, s_4 | n_2, n_3, n_5 |
| Coprime | s_1, s_3 | n_2, n_4, n_5 | s_1, s_3 | n_2, n_4, n_5 |
| Copy | s_1, s_3 | n_2, n_4 | s_1, s_3 | n_2, n_4 |
| Copy_row_from_table | s_3, s_4 | n_1, n_2, n_5 | s_3, s_4 | n_1, n_2, n_5 |
| Copy_row_from_table_16_neon | s_3 | n_1, n_2, n_4 | s_3 | n_1, n_2, n_4 |
| Copy_row_from_table_32_neon | s_3 | n_1, n_2, n_4 | s_3 | n_1, n_2, n_4 |
| Copy_row_from_table_8n_neon | s_3, s_4 | n_1, n_2, n_5 | s_3, s_4 | n_1, n_2, n_5 |
| Ctd | s_1 | n_2 | s_1 | n_2 |
| Ctz | s_1 | n_2 | s_1 | n_2 |
| Demont | s_1 | n_2, n_3, n_4 | s_1 | n_2, n_3, n_4 |
| Digit | s_1 | n_2, n_3 | s_1 | n_2, n_3 |
| Digitsize | s_1 | n_2 | s_1 | n_2 |
| Divmod10 | s_1 | n_2 | s_1 | n_2 |
| Emontredc | s_1 | n_2, n_3, n_4 | s_1 | n_2, n_3, n_4 |
| Eq | s_1, s_3 | n_2, n_4 | s_1, s_3 | n_2, n_4 |
| Even | s_1 | n_2 | s_1 | n_2 |
| Ge | s_1, s_3 | n_2, n_4 | s_1, s_3 | n_2, n_4 |
| Gt | s_1, s_3 | n_2, n_4 | s_1, s_3 | n_2, n_4 |
| Iszero | s_1 | n_2 | s_1 | n_2 |
| Lt | s_1, s_3 | n_2, n_4 | s_1, s_3 | n_2, n_4 |
| Madd | s_1, s_3, s_5 | n_2, n_4, n_6 | s_1, s_3, s_5 | n_2, n_4, n_6 |
| Modadd | s_1 | n_2, n_3, n_4, n_5 | s_1 | n_2, n_3, n_4, n_5 |
| Moddouble | s_1 | n_2, n_3, n_4 | s_1 | n_2, n_3, n_4 |
| Modifier | s_1 | n_2, n_3, n_4 | s_1 | n_2, n_3, n_4 |
| Modinv | s_1 | n_2, n_3, n_4, n_5 | s_1 | n_2, n_3, n_4, n_5 |
| Modoptneg | s_1 | n_2, n_3, n_4, n_5 | s_1 | n_2, n_3, n_4, n_5 |
| Modsub | s_1 | n_2, n_3, n_4, n_5 | s_1 | n_2, n_3, n_4, n_5 |
| Montifier | s_1 | n_2, n_3, n_4 | s_1 | n_2, n_3, n_4 |
| Montmul | s_1 | n_2, n_3, n_4, n_5 | s_1 | n_2, n_3, n_4, n_5 |
| Montredc | s_1, s_3, s_6 | n_2, n_4, n_5 | s_1, s_3, s_6 | n_2, n_4, n_5 |
| Montsqr | s_1 | n_2, n_3, n_4 | s_1 | n_2, n_3, n_4 |
| Mul | s_1, s_3, s_5 | n_2, n_4, n_6 | s_1, s_3, s_5 | n_2, n_4, n_6 |
| Muladd10 | s_1 | n_2, n_3 | s_1 | n_2, n_3 |
| Mux | s_2 | n_1, n_3, n_4, n_5 | s_2 | n_1, n_3, n_4, n_5 |
| Mux16 | s_1 | n_2, n_3, n_4 | s_1 | n_2, n_3, n_4 |
| Negmodinv | s_1 | n_2, n_3 | s_1 | n_2, n_3 |
| Nonzero | s_1 | n_2 | s_1 | n_2 |
| Normalize | s_1 | n_2 | s_1 | n_2 |
| Odd | s_1 | n_2 | s_1 | n_2 |
| Of_word | s_1 | n_2, n_3 | s_1 | n_2, n_3 |
| Optadd | s_1 | n_2, n_3, n_4, n_5 | s_1 | n_2, n_3, n_4, n_5 |
| Optneg | s_1 | n_2, n_3, n_4 | s_1 | n_2, n_3, n_4 |
| Optsub | s_1 | n_2, n_3, n_4, n_5 | s_1 | n_2, n_3, n_4, n_5 |
| Optsubadd | s_1 | n_2, n_3, n_4, n_5 | s_1 | n_2, n_3, n_4, n_5 |
| Pow2 | s_1 | n_2, n_3 | s_1 | n_2, n_3 |
| Shl_small | s_1, s_3 | n_2, n_4, n_5 | s_1, s_3 | n_2, n_4, n_5 |
| Shr_small | s_1, s_3 | n_2, n_4, n_5 | s_1 | s_3, n_2, n_4, n_5 |
| Sqr | s_1, s_3 | n_2, n_4 | s_1, s_3 | n_2, n_4 |
| Sub | s_1, s_3, s_5 | n_2, n_4, n_6 | s_1 | s_3, s_5, n_2, n_4, n_6 |
| Word_bytereverse | | n_1 | | n_1 |
| Word_clz | | n_1 | | n_1 |
| Word_ctz | | n_1 | | n_1 |
| Word_divstep59 | | n_1, n_2, n_3, n_4 | | n_1, n_2, n_3, n_4 |
| Word_max | | n_1, n_2 | | n_1, n_2 |
| Word_min | | n_1, n_2 | | n_1, n_2 |
| Word_negmodinv | | n_1 | | n_1 |
| Word_recip | | n_1 | | n_1 |
| TOTAL VARIABLES: | 93 | 179 | 85 | 187 |



31st Static Analysis Symposium — SAS 2024

Quantitative Static Timing Analysis

Denis Mazzucato, Marco Campion, and Caterina Urban

github.com/denismazzucato/times

Python + Abstract Domain Library Apron

S2N-Bignum

21st October

PROGRAM

- Add
- Amontifier
- Amontail
- Amontredc
- Amontrop
- Bitfield
- Bitsize
- Cdiv
- Cdiv_exact
- Cid
- Cidp
- Cmadd
- Cmsgadd

Quantitative Static Timing Analysis

21st October

Python + Abstract Domain Library Apron

<https://github.com/aws-labs/s2n-bignum>

- 

Denis Mazzucato et al.

(i) Irrelevant Instructions

```

1 def Add(p, z, m, x, n, y): 26
2     r = min(p, m)          27
3     s = min(p, n)          28
4     if (r < s):             29
5         t = p - s           30
6         q = s - r           31
7         i = 0               32
8         a = 0               33
9         for (; r > 0; r--): 34

```

Sy
Dep
A

Caterina Urban
Interpretation Fro

Denis Mazzucato et al.

(ii) Abstract Interpretation

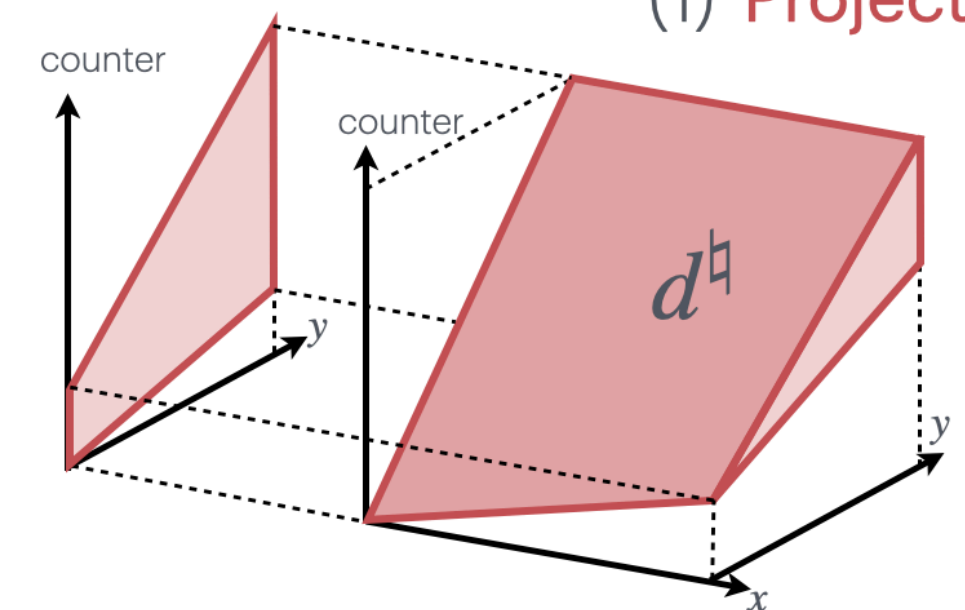
Abstract invariant on the
input variable

```
1 def Add(p, z, m, x, n, y):  Augment each iteration with a
2     r = min(p, m)          loop body with a
3     s = min(p, n)          for iteration
4     if (r < s):
5         t = p - s
```

Backward analysis

Abstract invariant on the
input variables + counter

- (i) Assume **input** v
- (1) **Project** away x



Denis Mazzucato et al.

| PROGRAM | INPUT VARIABLES | | | MAYBE DANGEROUS |
|-----------------------------|--------------------------------|--|--------------------------------|--------------------------------|
| | SAFE | NUMERICAL | Δ_N | |
| Add | $\Delta_1, \Delta_3, \Delta_5$ | $\Delta_2, \Delta_4, \Delta_6$ | Δ_1 | Δ_3, Δ_4 |
| Amantfor | Δ_1 | $\Delta_2, \Delta_4, \Delta_6$ | Δ_1 | Δ_3, Δ_4 |
| Amantmul | $\Delta_1, \Delta_2, \Delta_5$ | $\Delta_2, \Delta_4, \Delta_6, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Amantrec | Δ_1 | $\Delta_2, \Delta_4, \Delta_6$ | Δ_1 | $\Delta_3, \Delta_4, \Delta_5$ |
| Amantrot | Δ_1 | $\Delta_2, \Delta_4, \Delta_6$ | Δ_1 | Δ_3 |
| Bitfield | Δ_1 | $\Delta_2, \Delta_4, \Delta_6, \Delta_5$ | Δ_1 | Δ_3 |
| Bitize | Δ_1 | $\Delta_2, \Delta_4, \Delta_6$ | Δ_1 | Δ_3 |
| Cliv | Δ_1 | $\Delta_2, \Delta_4, \Delta_6$ | Δ_1 | Δ_3, Δ_4 |
| Cliv_exact | Δ_1, Δ_3 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Clz | Δ_1 | Δ_2 | Δ_1 | Δ_3 |
| Clz2 | Δ_1 | Δ_2 | Δ_1 | Δ_3 |
| Coad | Δ_1, Δ_4 | $\Delta_2, \Delta_3, \Delta_5, \Delta_6$ | Δ_1, Δ_4 | Δ_3 |
| Omegadd | Δ_1, Δ_4 | $\Delta_2, \Delta_3, \Delta_5, \Delta_6$ | Δ_1, Δ_4 | Δ_3 |
| Coad2 | Δ_1, Δ_4 | $\Delta_2, \Delta_3, \Delta_5, \Delta_6$ | Δ_1, Δ_4 | Δ_3 |
| Cmal | Δ_1, Δ_4 | $\Delta_2, \Delta_3, \Delta_5, \Delta_6$ | Δ_1, Δ_4 | Δ_3 |
| Cotrine | Δ_1 | $\Delta_2, \Delta_4, \Delta_6$ | Δ_1 | Δ_3, Δ_4 |
| Copy | Δ_1, Δ_3 | $\Delta_2, \Delta_4, \Delta_6$ | Δ_1, Δ_3 | Δ_3, Δ_4 |
| Copy_row_from_table | Δ_1 | $\Delta_2, \Delta_4, \Delta_6$ | Δ_1 | Δ_3, Δ_4 |
| Copy_row_from_table_16_neon | Δ_3 | $\Delta_1, \Delta_2, \Delta_4, \Delta_6$ | Δ_3 | Δ_3 |
| Copy_row_from_table_32_neon | Δ_3 | $\Delta_1, \Delta_2, \Delta_4, \Delta_6$ | Δ_3 | Δ_3 |
| Copy_row_from_table_8u_neon | Δ_3 | $\Delta_1, \Delta_2, \Delta_4, \Delta_6$ | Δ_3 | Δ_3 |
| Copy_row_from_table_8u_neon | Δ_1, Δ_4 | $\Delta_2, \Delta_3, \Delta_5, \Delta_6$ | Δ_1, Δ_4 | Δ_3, Δ_4 |
| Crc | Δ_1 | $\Delta_2, \Delta_4, \Delta_6$ | Δ_1 | Δ_3, Δ_4 |
| Ctz | Δ_1 | Δ_2 | Δ_1 | Δ_3 |
| Dmemset | Δ_1 | $\Delta_2, \Delta_4, \Delta_6$ | Δ_1 | Δ_3 |
| Digit | Δ_1 | $\Delta_2, \Delta_4, \Delta_6$ | Δ_1 | Δ_3, Δ_4 |
| Digitize | Δ_1 | $\Delta_2, \Delta_4, \Delta_6$ | Δ_1 | Δ_3, Δ_4 |
| Dvmod10 | Δ_1 | Δ_2 | Δ_1 | Δ_3 |
| E2 | Δ_1 | $\Delta_2, \Delta_4, \Delta_6$ | Δ_1 | Δ_3, Δ_4 |
| Emantrec | Δ_1, Δ_3 | $\Delta_2, \Delta_4, \Delta_6$ | Δ_1, Δ_3 | Δ_3, Δ_4 |
| Even | Δ_1 | $\Delta_2, \Delta_4, \Delta_6$ | Δ_1 | Δ_3, Δ_4 |
| Ex | Δ_1 | $\Delta_2, \Delta_4, \Delta_6$ | Δ_1 | Δ_3, Δ_4 |
| Ge | Δ_1 | $\Delta_2, \Delta_4, \Delta_6$ | Δ_1, Δ_3 | Δ_3, Δ_4 |
| Izzero | Δ_1 | $\Delta_2, \Delta_4, \Delta_6$ | Δ_1, Δ_3 | Δ_3, Δ_4 |
| Le | Δ_1 | $\Delta_2, \Delta_4, \Delta_6$ | Δ_1, Δ_3 | Δ_3, Δ_4 |
| Li | Δ_1, Δ_3 | $\Delta_2, \Delta_4, \Delta_6$ | Δ_1 | Δ_3, Δ_4 |
| Madd | $\Delta_1, \Delta_3, \Delta_5$ | $\Delta_2, \Delta_4, \Delta_6$ | $\Delta_1, \Delta_3, \Delta_5$ | $\Delta_3, \Delta_4, \Delta_5$ |
| Madd2 | $\Delta_1, \Delta_3, \Delta_5$ | $\Delta_2, \Delta_4, \Delta_6, \Delta_5$ | $\Delta_1, \Delta_3, \Delta_5$ | $\Delta_3, \Delta_4, \Delta_5$ |
| Madddouble | Δ_1 | $\Delta_2, \Delta_4, \Delta_6$ | Δ_1 | Δ_3, Δ_4 |
| Modifier | Δ_1 | $\Delta_2, \Delta_4, \Delta_6$ | Δ_1 | Δ_3, Δ_4 |
| Modinv | Δ_1 | $\Delta_2, \Delta_4, \Delta_6, \Delta_5$ | Δ_1 | $\Delta_3, \Delta_4, \Delta_5$ |
| Modnegneg | Δ_1 | $\Delta_2, \Delta_4, \Delta_6, \Delta_5$ | Δ_1 | $\Delta_3, \Delta_4, \Delta_5$ |
| Modadd | Δ_1 | $\Delta_2, \Delta_4, \Delta_6, \Delta_5$ | Δ_1 | $\Delta_3, \Delta_4, \Delta_5$ |
| Modsub | Δ_1 | $\Delta_2, \Delta_4, \Delta_6, \Delta_5$ | Δ_1 | $\Delta_3, \Delta_4, \Delta_5$ |
| Montmul | Δ_1 | $\Delta_2, \Delta_4, \Delta_6, \Delta_5$ | Δ_1 | $\Delta_3, \Delta_4, \Delta_5$ |
| Montrec | $\Delta_1, \Delta_3, \Delta_5$ | $\Delta_2, \Delta_4, \Delta_6, \Delta_5$ | $\Delta_1, \Delta_3, \Delta_5$ | $\Delta_3, \Delta_4, \Delta_5$ |
| Mul | Δ_1 | $\Delta_2, \Delta_4, \Delta_6$ | Δ_1 | Δ_3, Δ_4 |
| Muladd10 | Δ_1 | $\Delta_2, \Delta_4, \Delta_6$ | Δ_1 | Δ_3, Δ_4 |
| Mul2 | Δ_1 | $\Delta_2, \Delta_4, \Delta_6, \Delta_5$ | Δ_1 | $\Delta_3, \Delta_4, \Delta_5$ |
| Mul16 | Δ_1 | $\Delta_2, \Delta_4, \Delta_6$ | Δ_1 | Δ_3, Δ_4 |
| Negmodinv | Δ_1 | $\Delta_2, \Delta_4, \Delta_6$ | Δ_1 | Δ_3, Δ_4 |
| Nonzero | Δ_1 | Δ_2 | Δ_1 | Δ_3 |
| Normalize | Δ_1 | Δ_2 | Δ_1 | Δ_3 |
| Odd | Δ_1 | Δ_2 | Δ_1 | Δ_3 |
| Of_word | Δ_1 | Δ_2 | Δ_1 | Δ_3 |
| Ofpadd | Δ_1 | $\Delta_2, \Delta_4, \Delta_6, \Delta_5$ | Δ_1 | $\Delta_3, \Delta_4, \Delta_5$ |
| Ofpadd2 | Δ_1 | $\Delta_2, \Delta_4, \Delta_6, \Delta_5$ | Δ_1 | $\Delta_3, \Delta_4, \Delta_5$ |
| Ofpsub | Δ_1 | $\Delta_2, \Delta_4, \Delta_6, \Delta_5$ | Δ_1 | $\Delta_3, \Delta_4, \Delta_5$ |
| Ofpshadd | Δ_1 | $\Delta_2, \Delta_4, \Delta_6, \Delta_5$ | Δ_1 | $\Delta_3, \Delta_4, \Delta_5$ |
| Pov2 | Δ_1 | Δ_2, Δ_4 | Δ_1 | Δ_3 |
| Shr_small | Δ_1 | Δ_2, Δ_4 | Δ_1 | Δ_3 |
| Shr_small2 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1, Δ_3 | Δ_3, Δ_4 |
| Shr_small3 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small4 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small5 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small6 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small7 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small8 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small9 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small10 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small11 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small12 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small13 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small14 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small15 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small16 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small17 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small18 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small19 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small20 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small21 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small22 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small23 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small24 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small25 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small26 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small27 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small28 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small29 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small30 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small31 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small32 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small33 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small34 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small35 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small36 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small37 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small38 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small39 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small40 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small41 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small42 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small43 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small44 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small45 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small46 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small47 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small48 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small49 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small50 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small51 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small52 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small53 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small54 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small55 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small56 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small57 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small58 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small59 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small60 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small61 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small62 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small63 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small64 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small65 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small66 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small67 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small68 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small69 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small70 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small71 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small72 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small73 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small74 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small75 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small76 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small77 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small78 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small79 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small80 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small81 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small82 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small83 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small84 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small85 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small86 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small87 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small88 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small89 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small90 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small91 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small92 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small93 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small94 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small95 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small96 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small97 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small98 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small99 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | Δ_3, Δ_4 |
| Shr_small100 | Δ_1 | $\Delta_2, \Delta_4, \Delta_5$ | Δ_1 | $\Delta_$ |

Denis Mazzucato et al.