31st Static Analysis Symposium — SAS 2024

Quantitative Static Timing Analysis

Denis Mazzucato, Marco Campion, and Caterina Urban

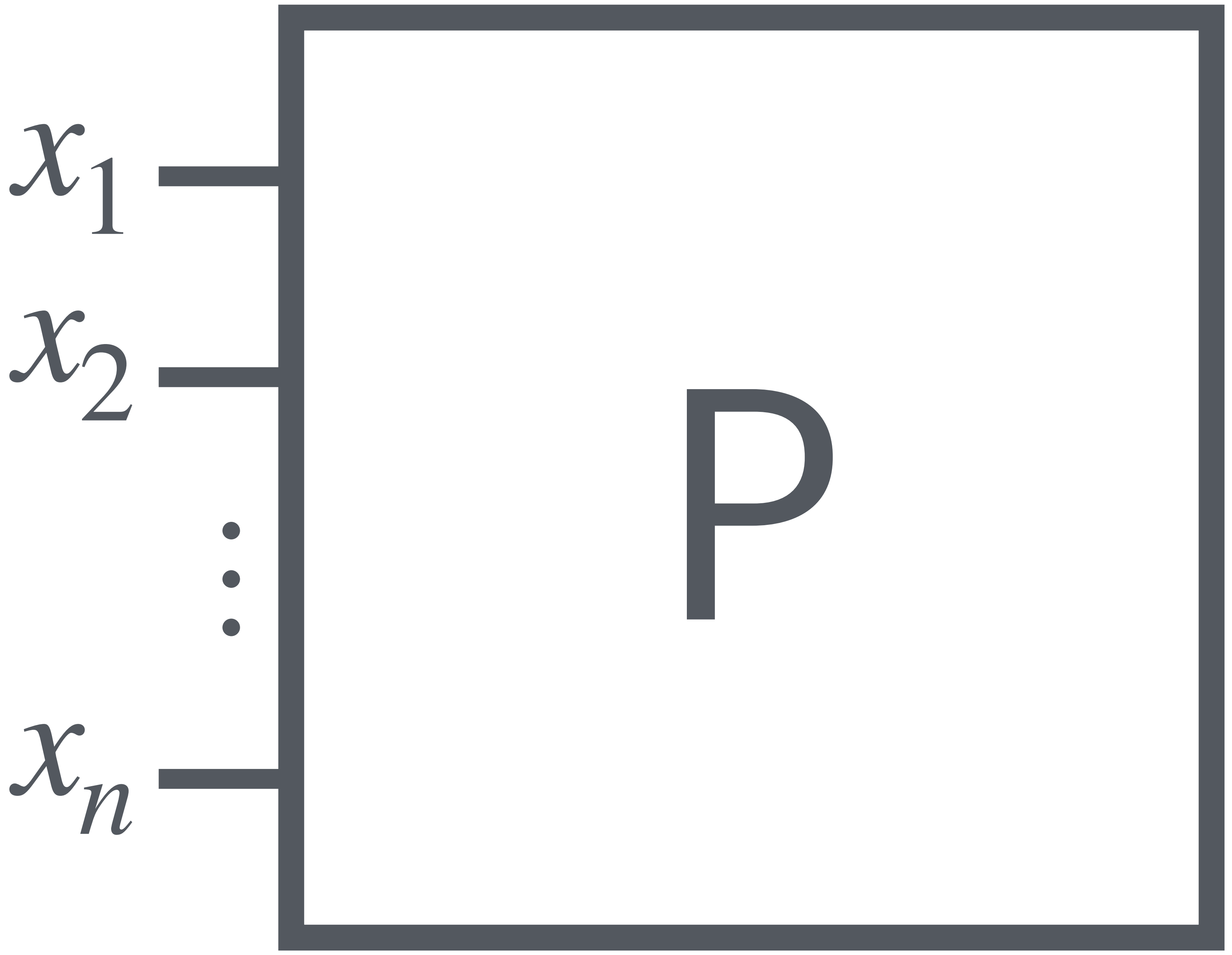
21st October 2024

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

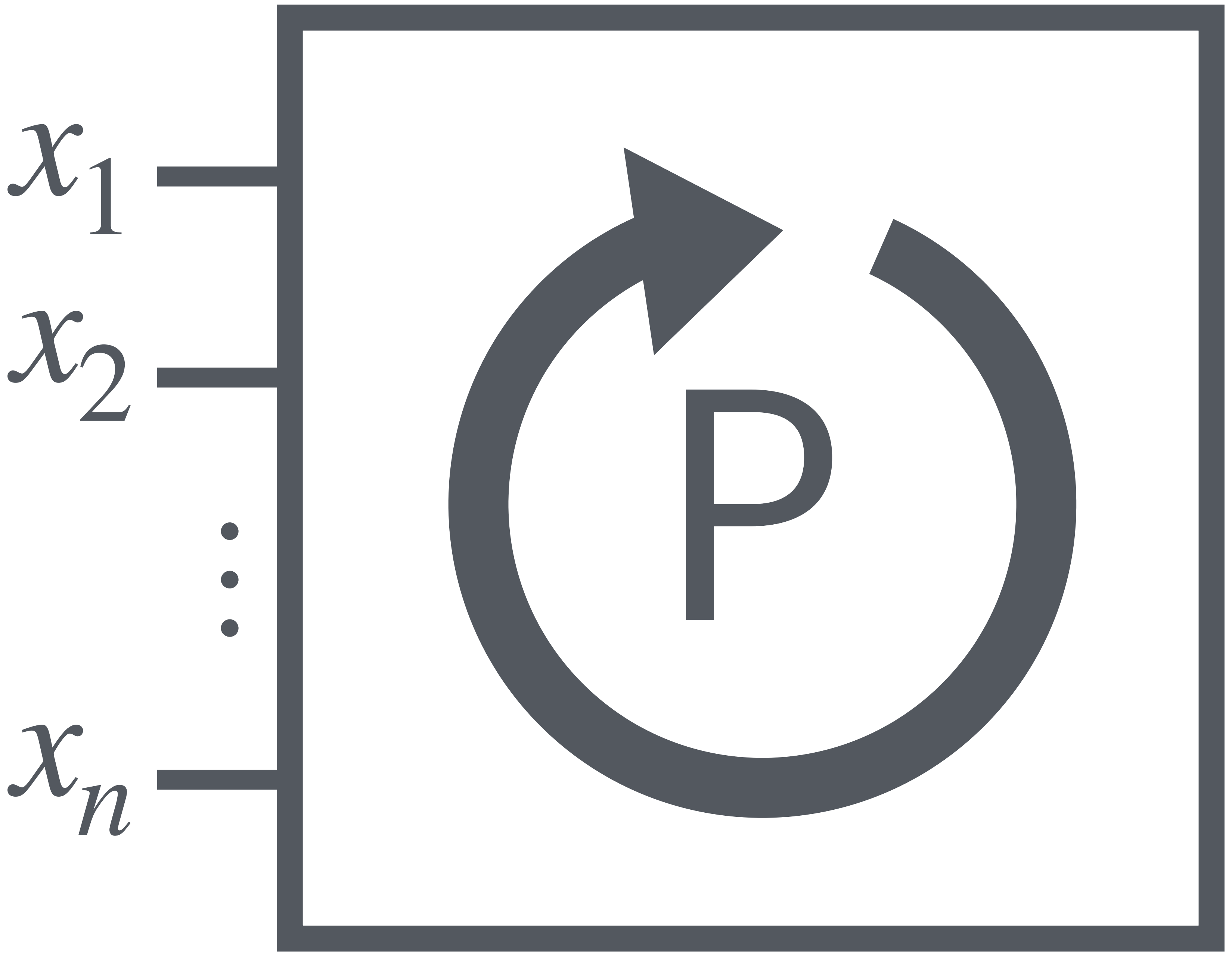
Quantifying the impact of input variables on the number of iterations of a program

Input variables



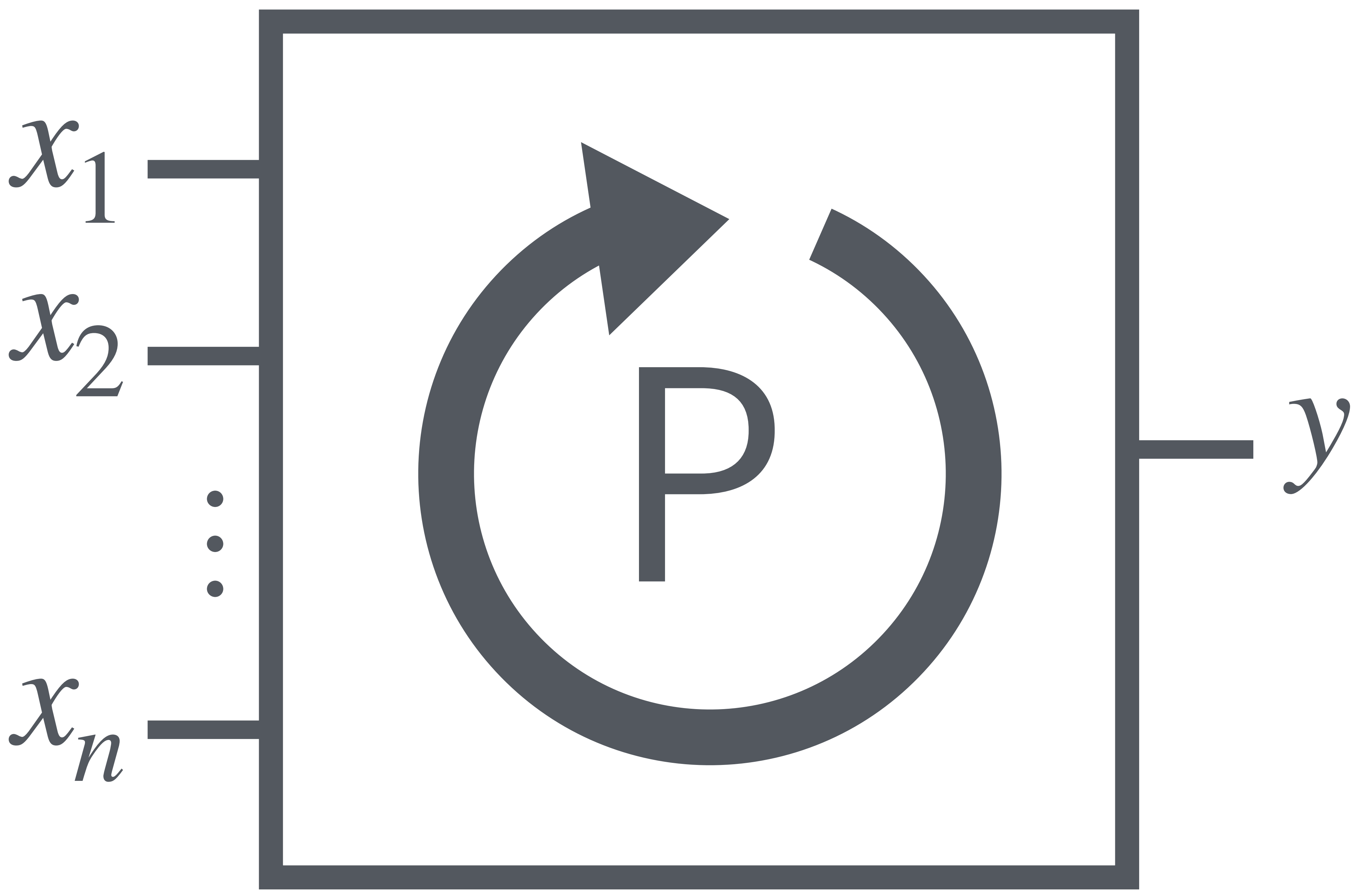
Quantifying the impact of input variables on the number of iterations of a program

Input variables



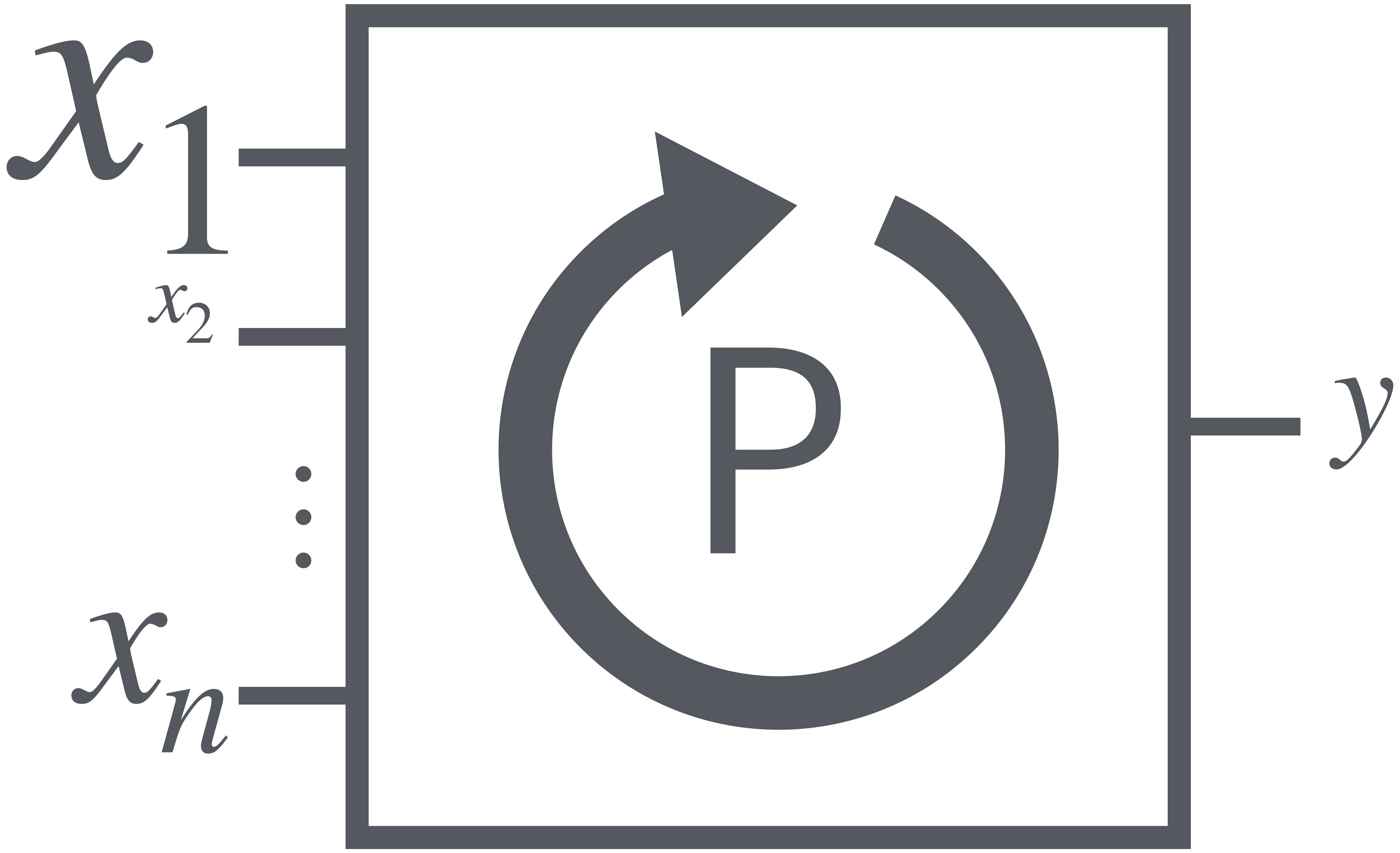
Quantifying the impact of input variables on the number of iterations of a program

Input variables

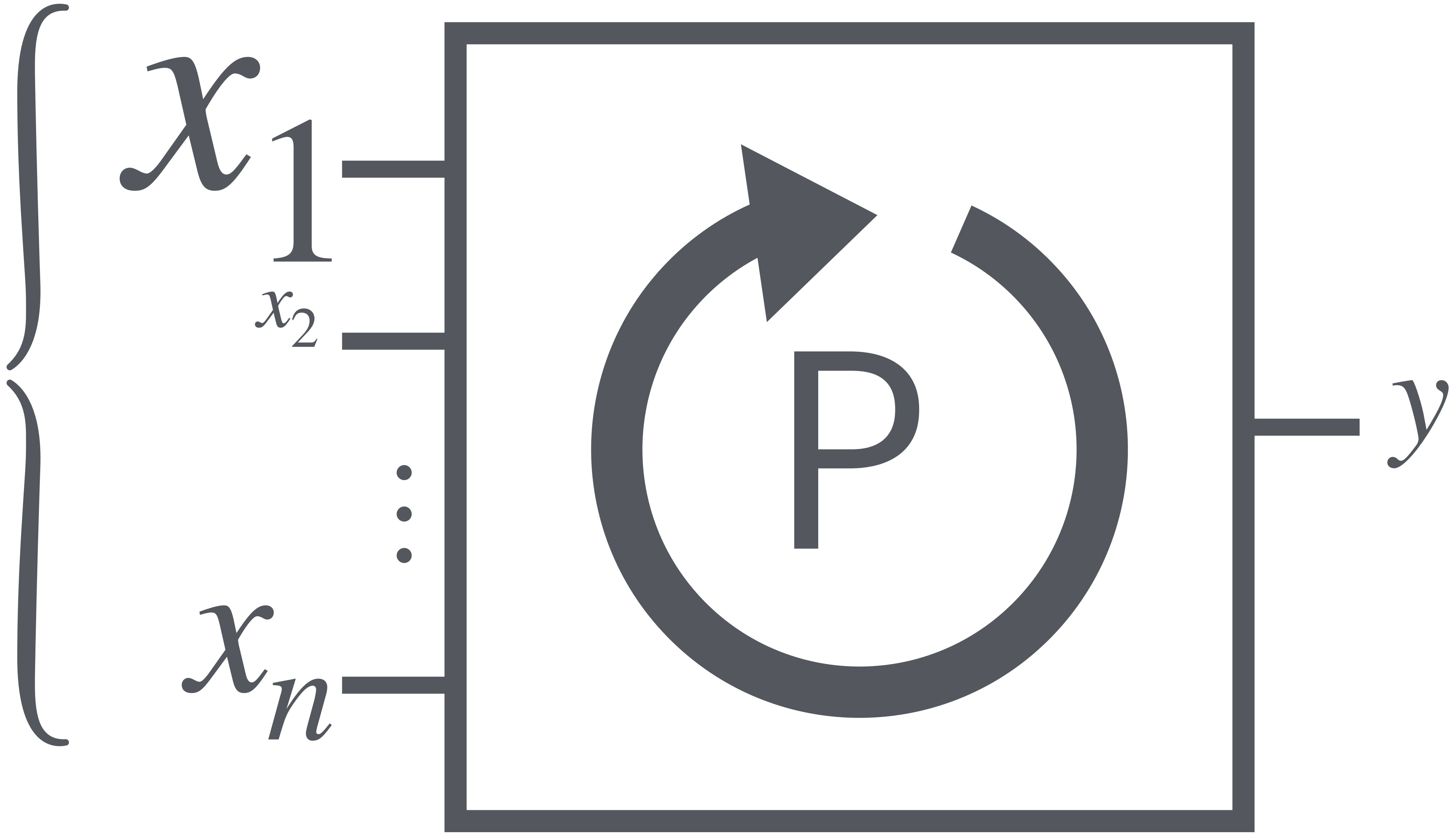


Quantifying the impact of input variables on the number of iterations of a program

How much impact on loop iterations?

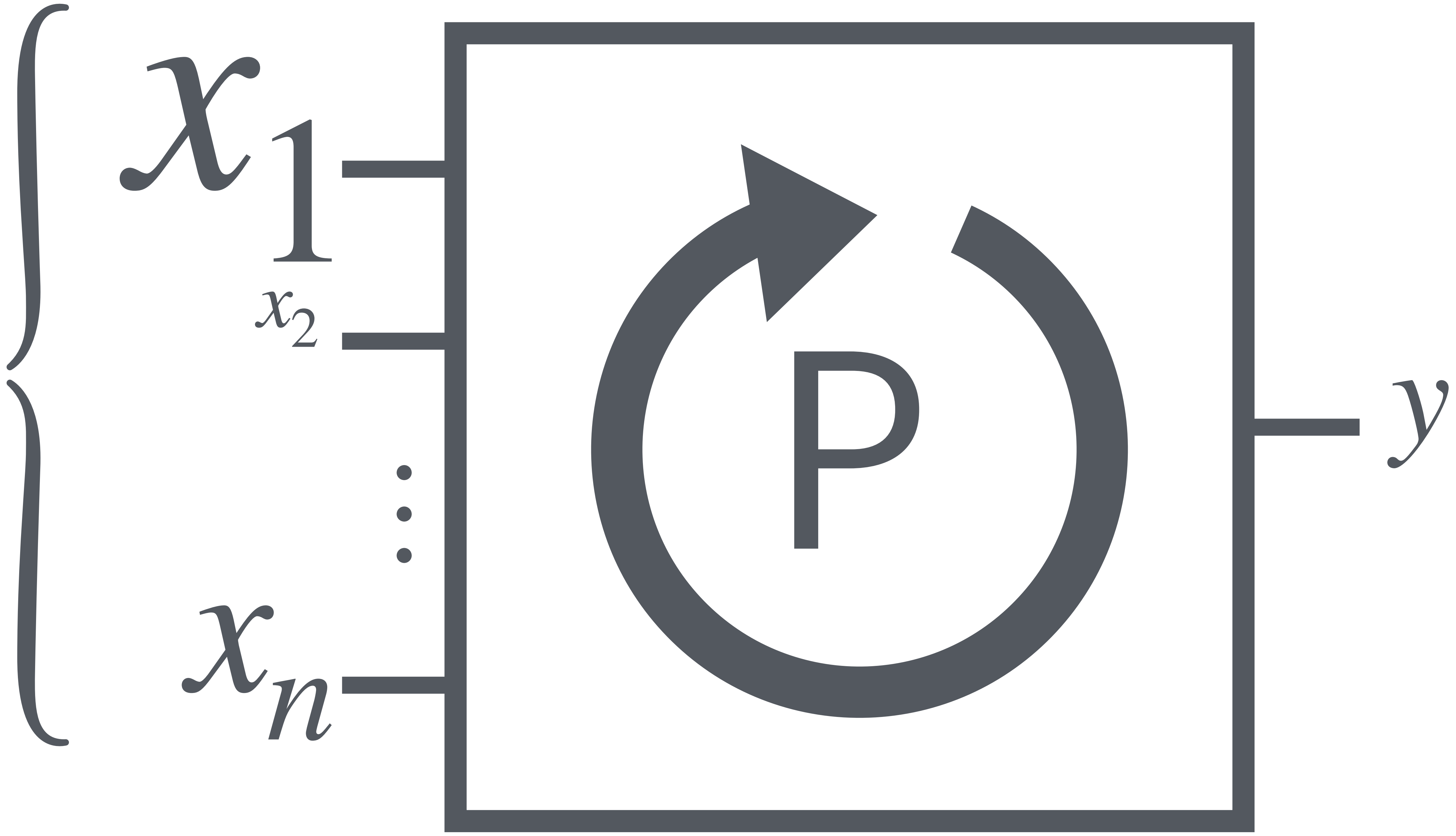


Why?



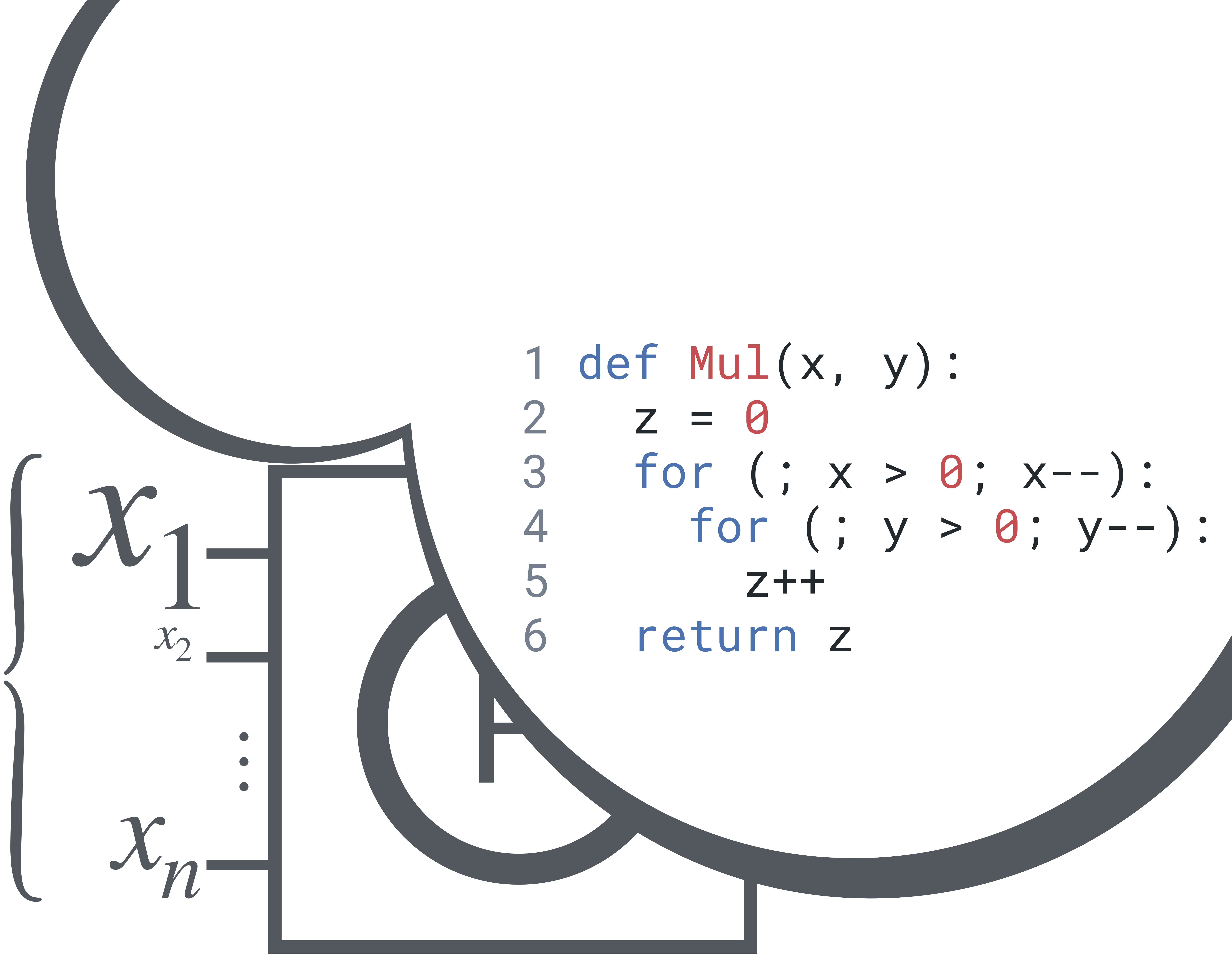
Why?

- Correct Loop Behaviour



Why?

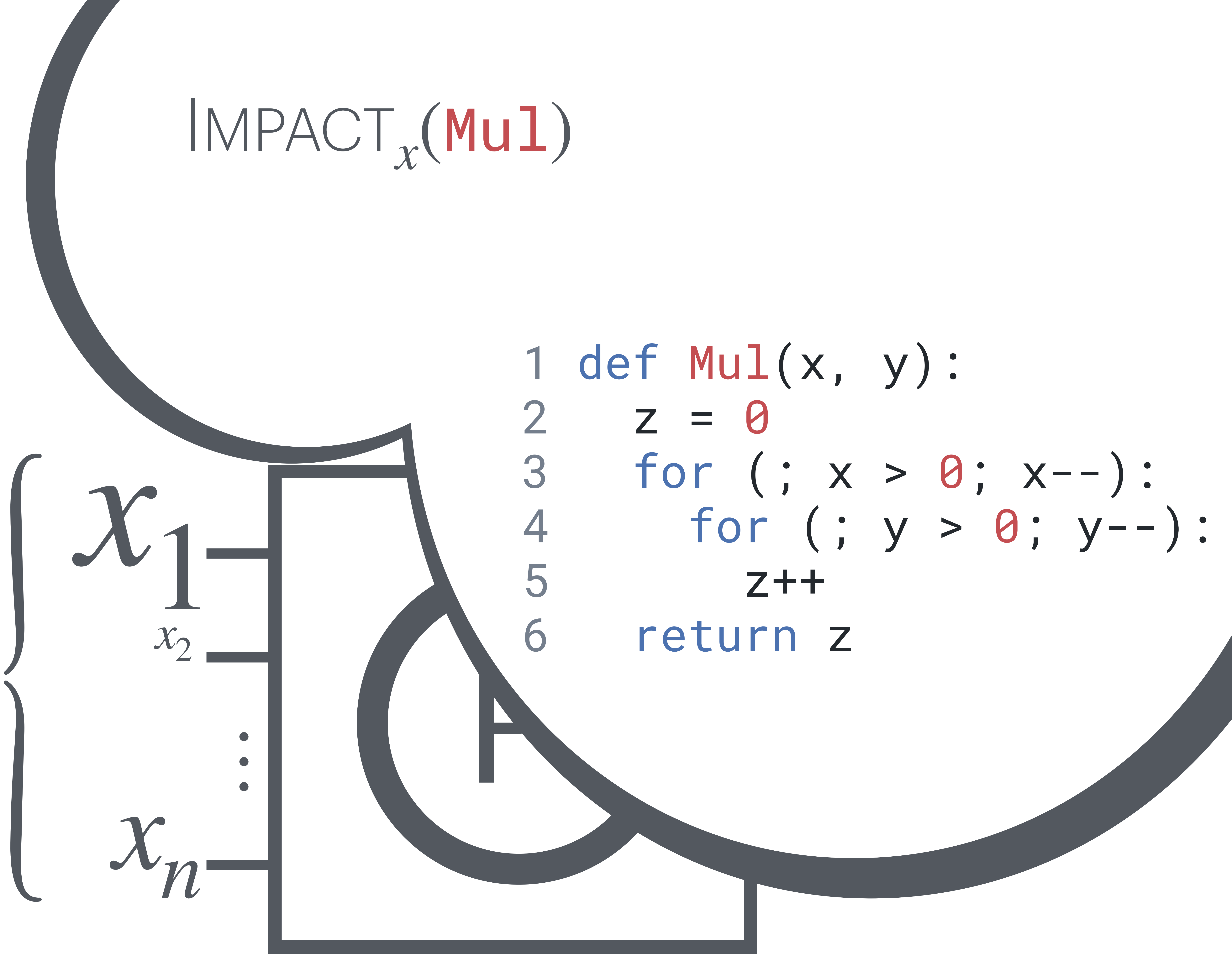
- Correct Loop Behaviour



```
def Mul(x, y):
  z = 0
  for (; x > 0; x--):
    for (; y > 0; y--):
      z++
  return z
```

Why?

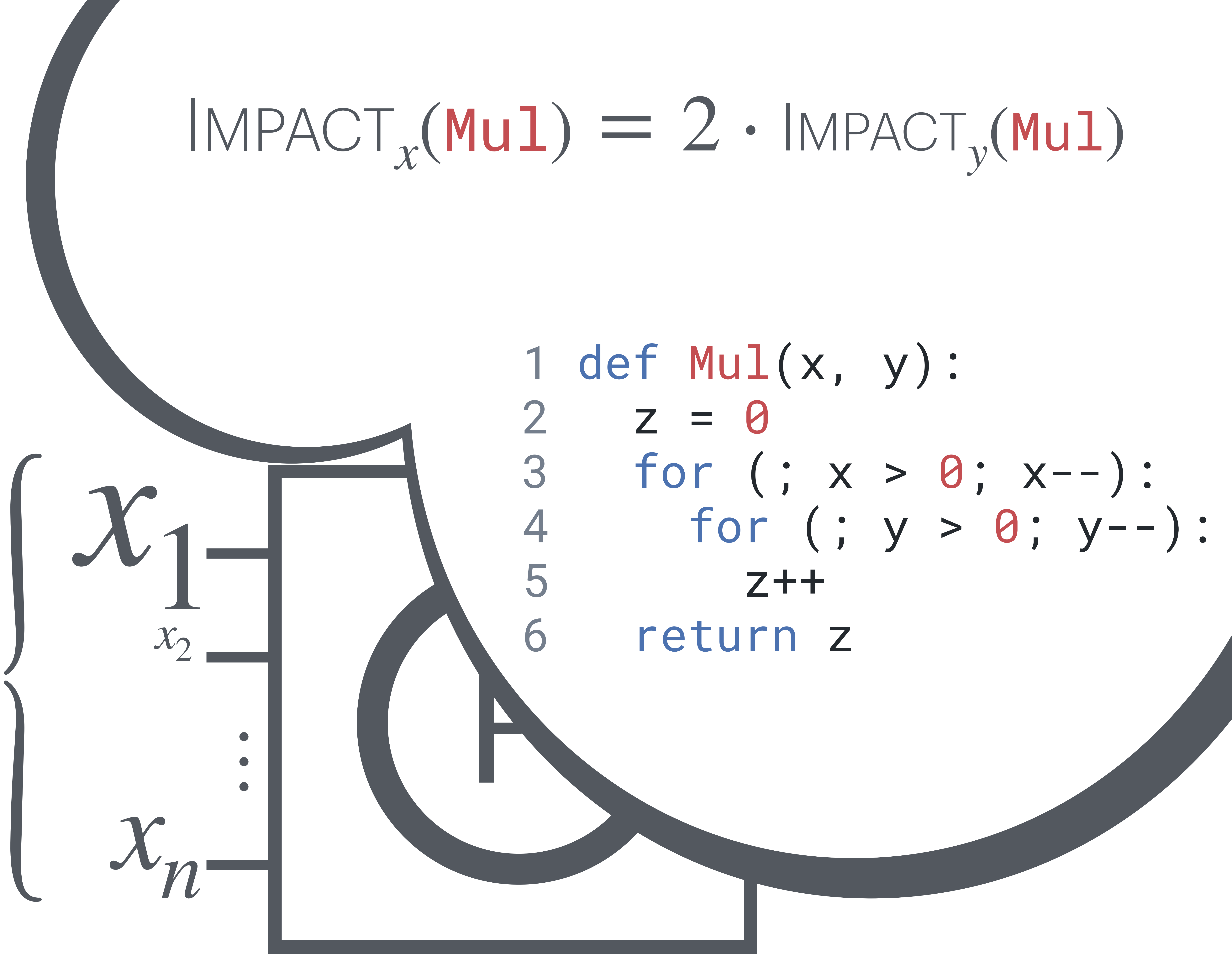
- Correct Loop Behaviour



```
def Mul(x, y):
  z = 0
  for (; x > 0; x--):
    for (; y > 0; y--):
      z++
  return z
```

Why?

- Correct Loop Behaviour

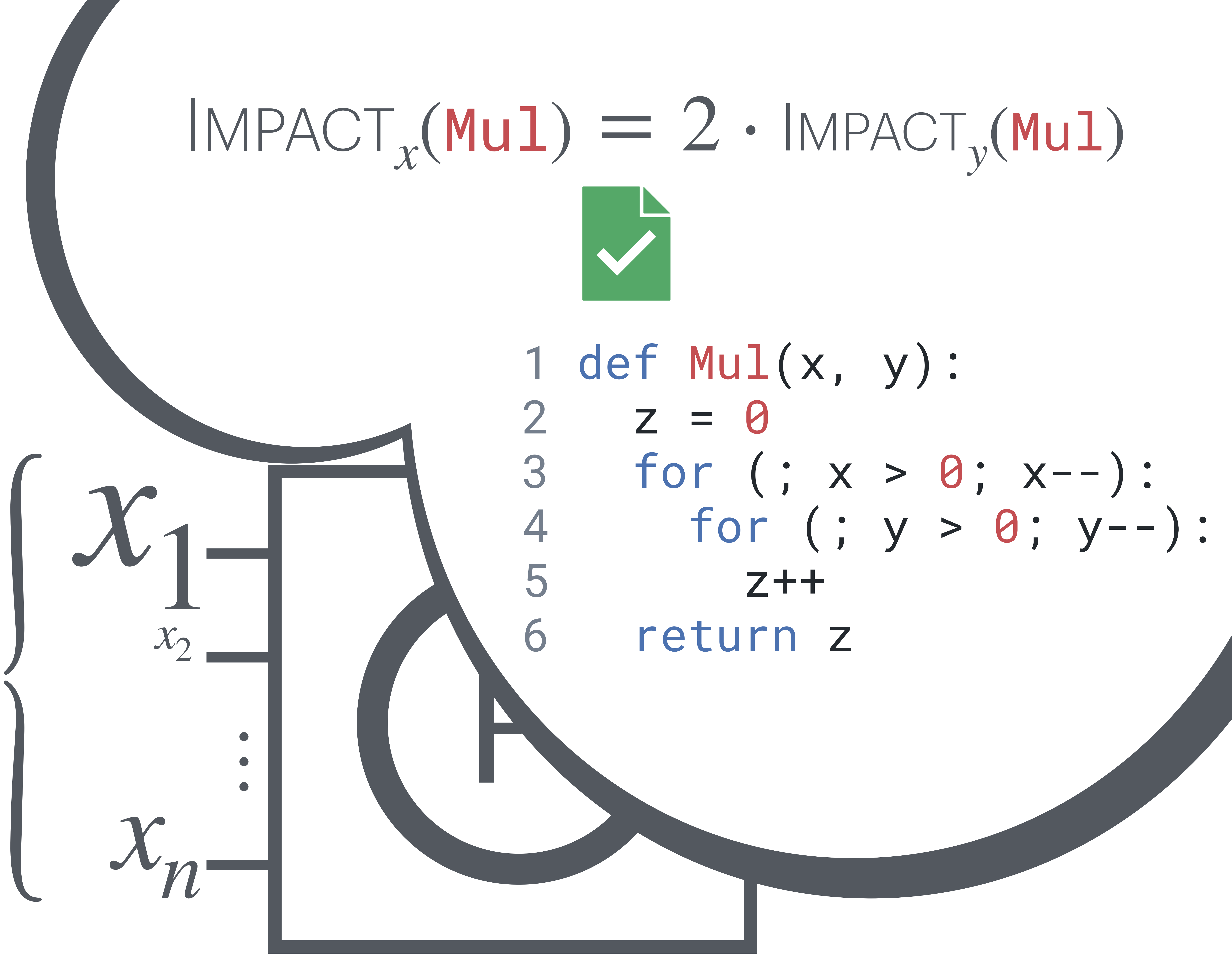


$$\textsc{Impact}_x(\texttt{Mul}) = 2 \cdot \textsc{Impact}_y(\texttt{Mul})$$

```
def Mul(x, y):
  z = 0
  for (; x > 0; x--):
    for (; y > 0; y--):
      z++
  return z
```

Why?

- Correct Loop Behaviour



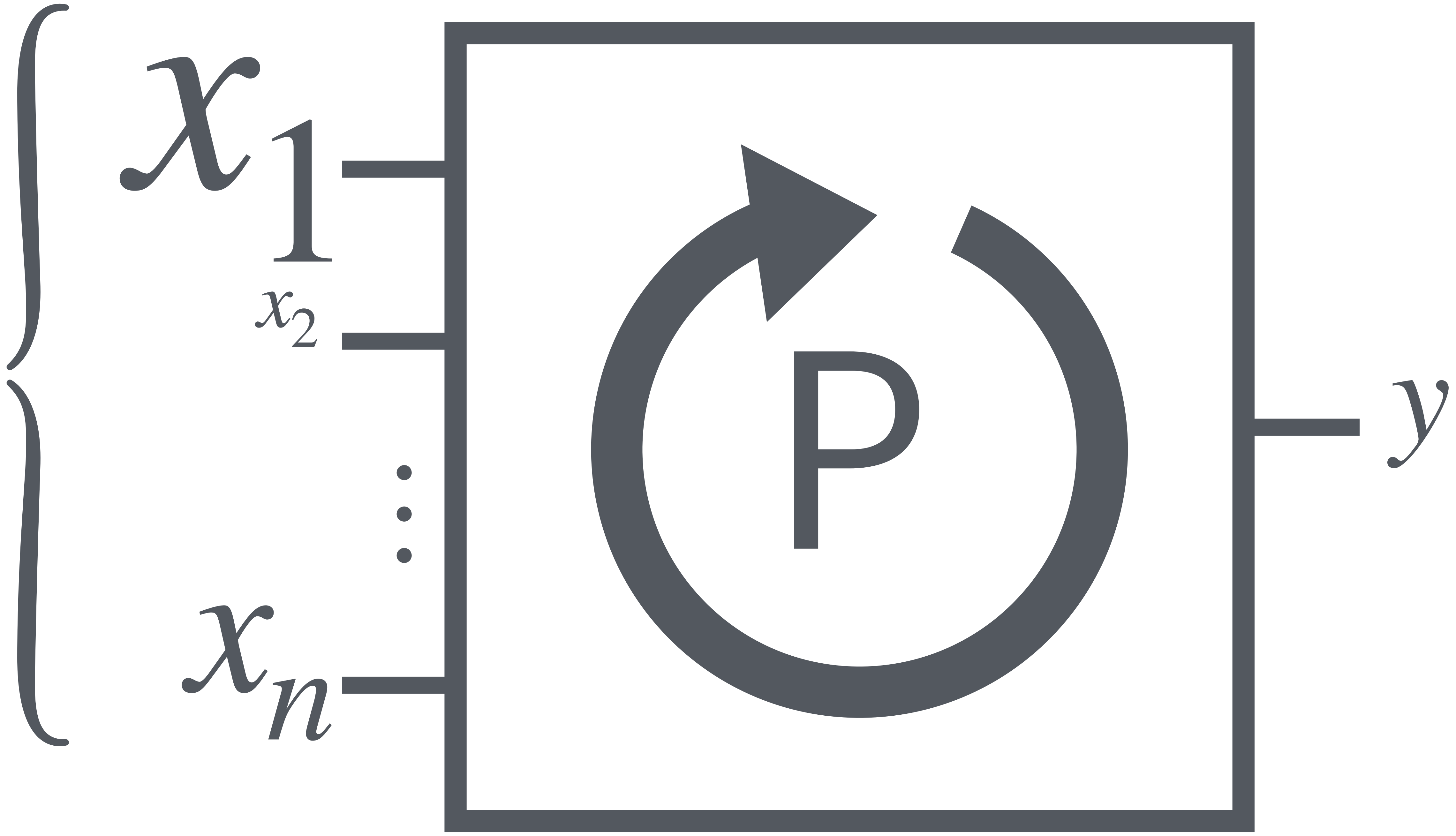
$$\textsc{Impact}_x(\texttt{Mul}) = 2 \cdot \textsc{Impact}_y(\texttt{Mul})$$

✓

```
def Mul(x, y):
  z = 0
  for (; x > 0; x--):
    for (; y > 0; y--):
      z++
  return z
```

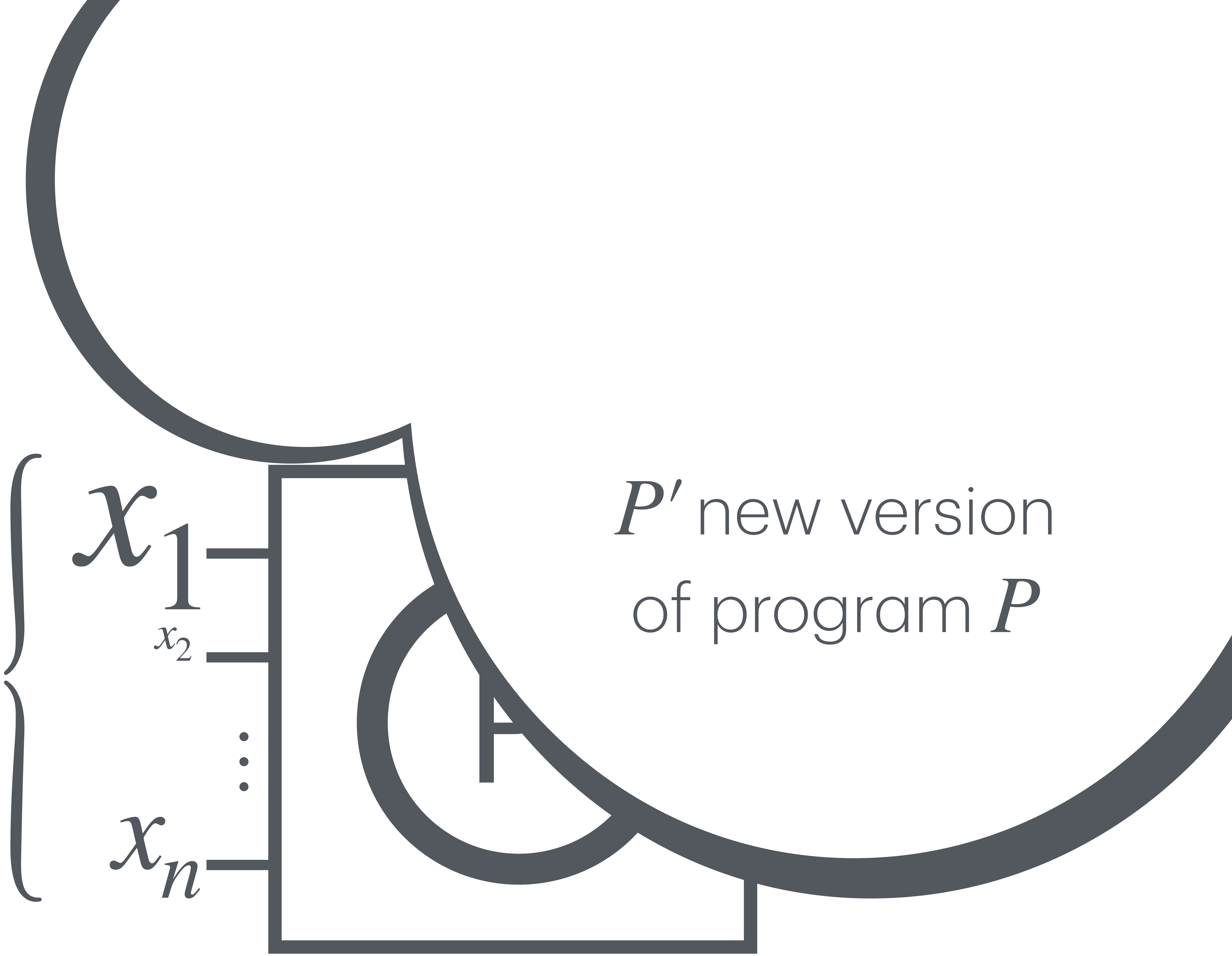
Why?

- Correct Loop Behaviour
- Performance



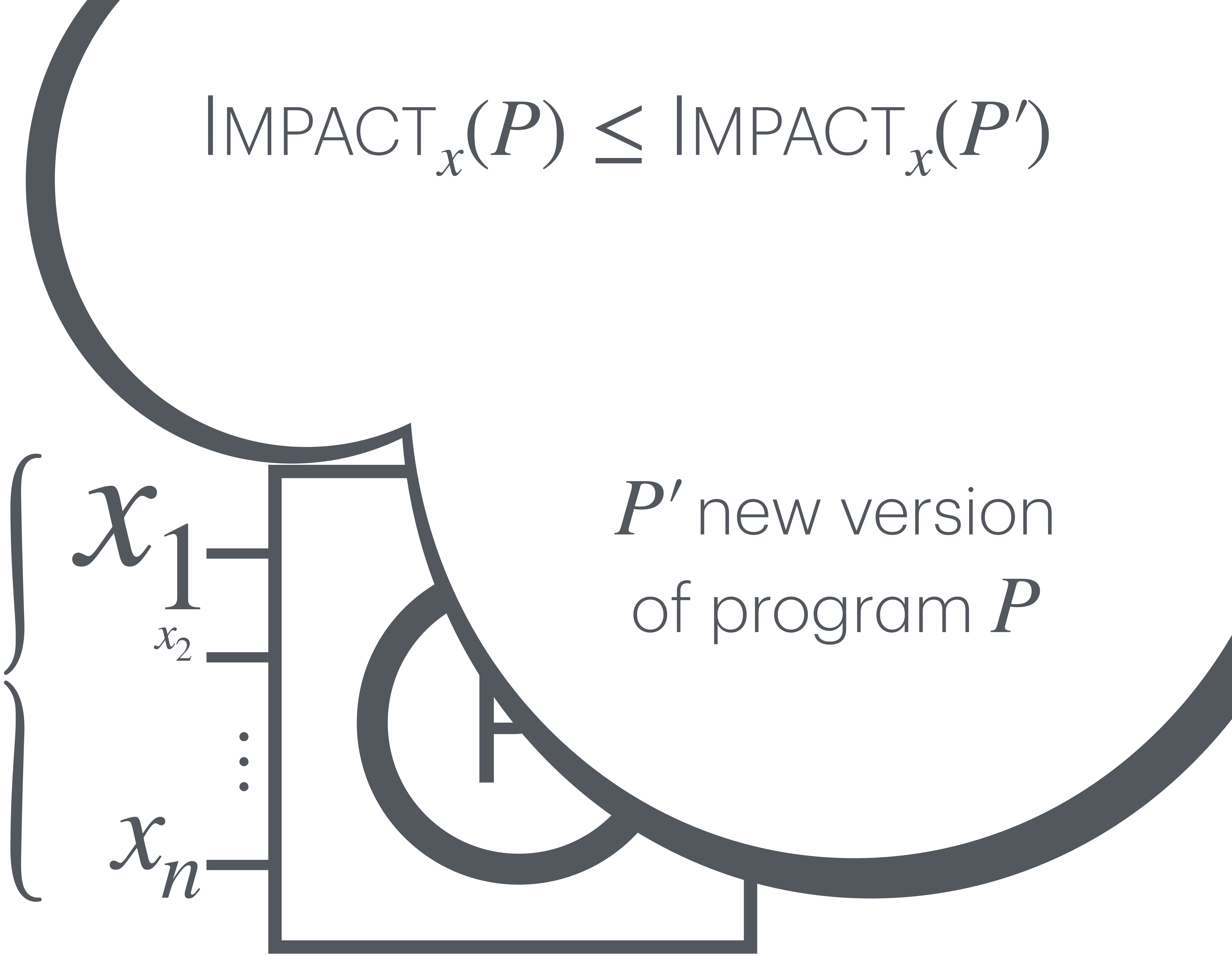
Why?

- Correct Loop Behaviour
- Performance



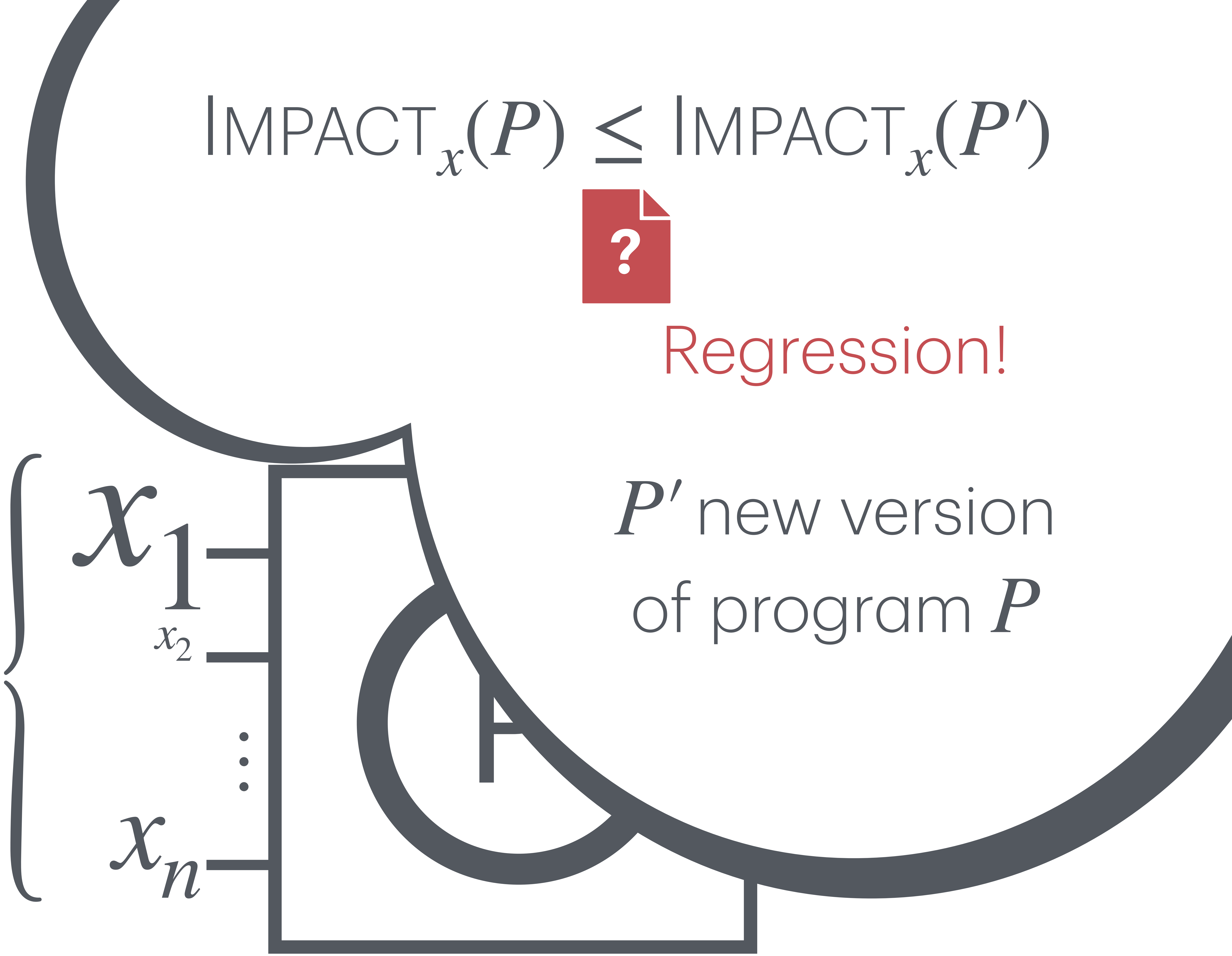
Why?

- Correct Loop Behaviour
- Performance


$\textsc{Impact}_x(P) \leq \textsc{Impact}_x(P')$
x_1
x_2
$\vdots$
x_n
P
P' new version of program P

Why?

- Correct Loop Behaviour
- Performance



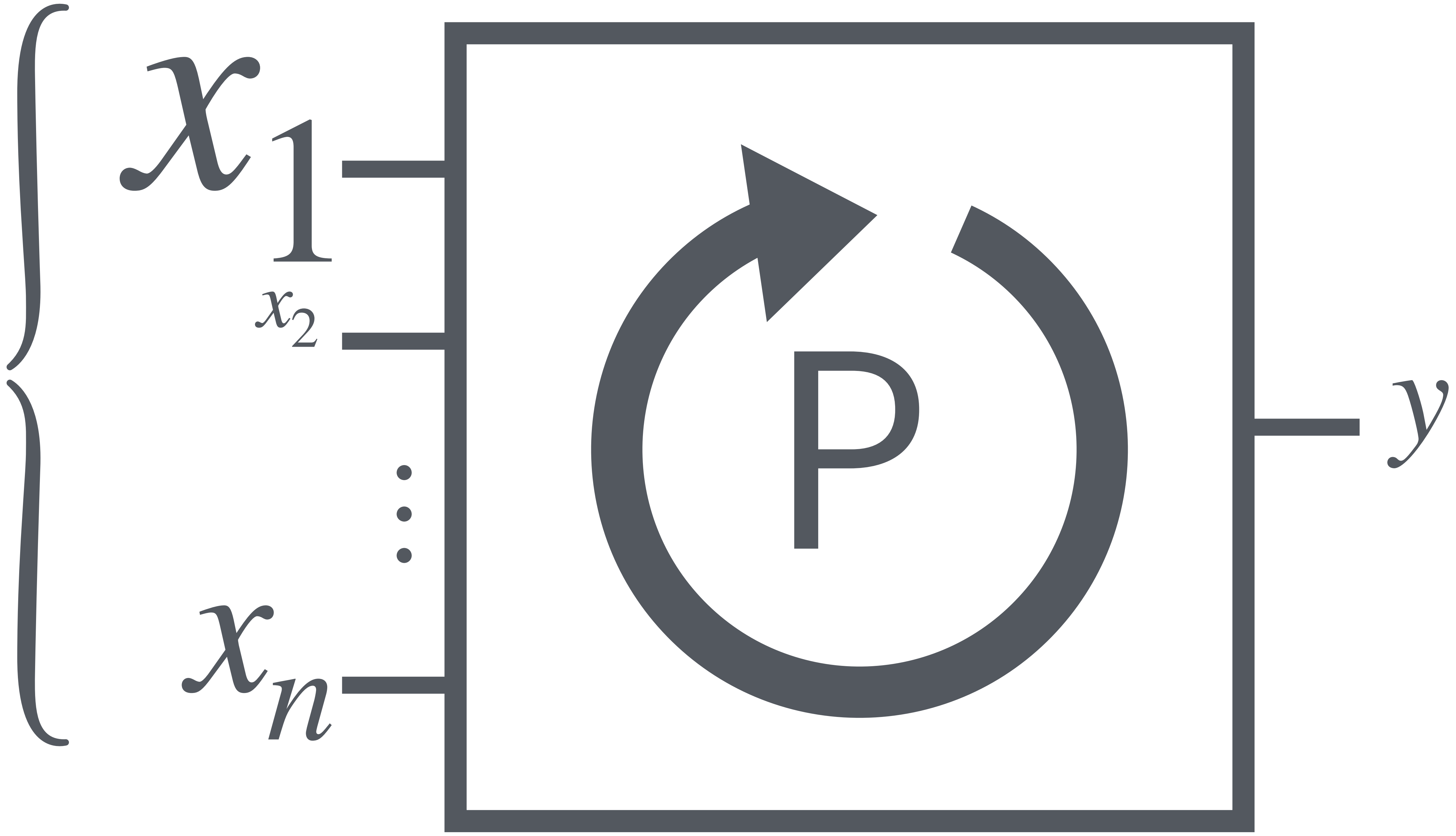
$$\textsc{Impact}_x(P) \leq \textsc{Impact}_x(P')$$

Regression!

P' new version of program P

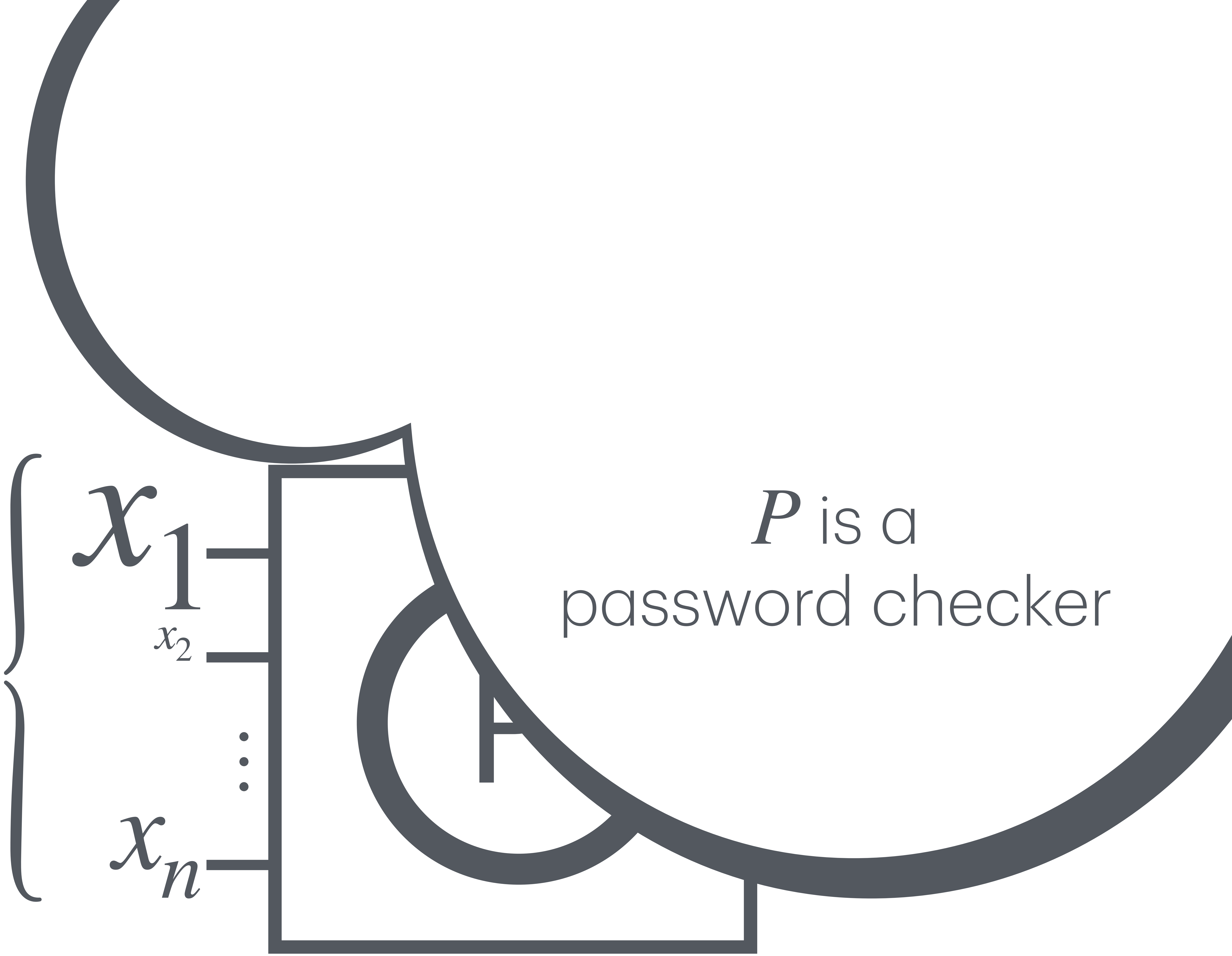
Why?

- Correct Loop Behaviour
- Performance
- Security



Why?

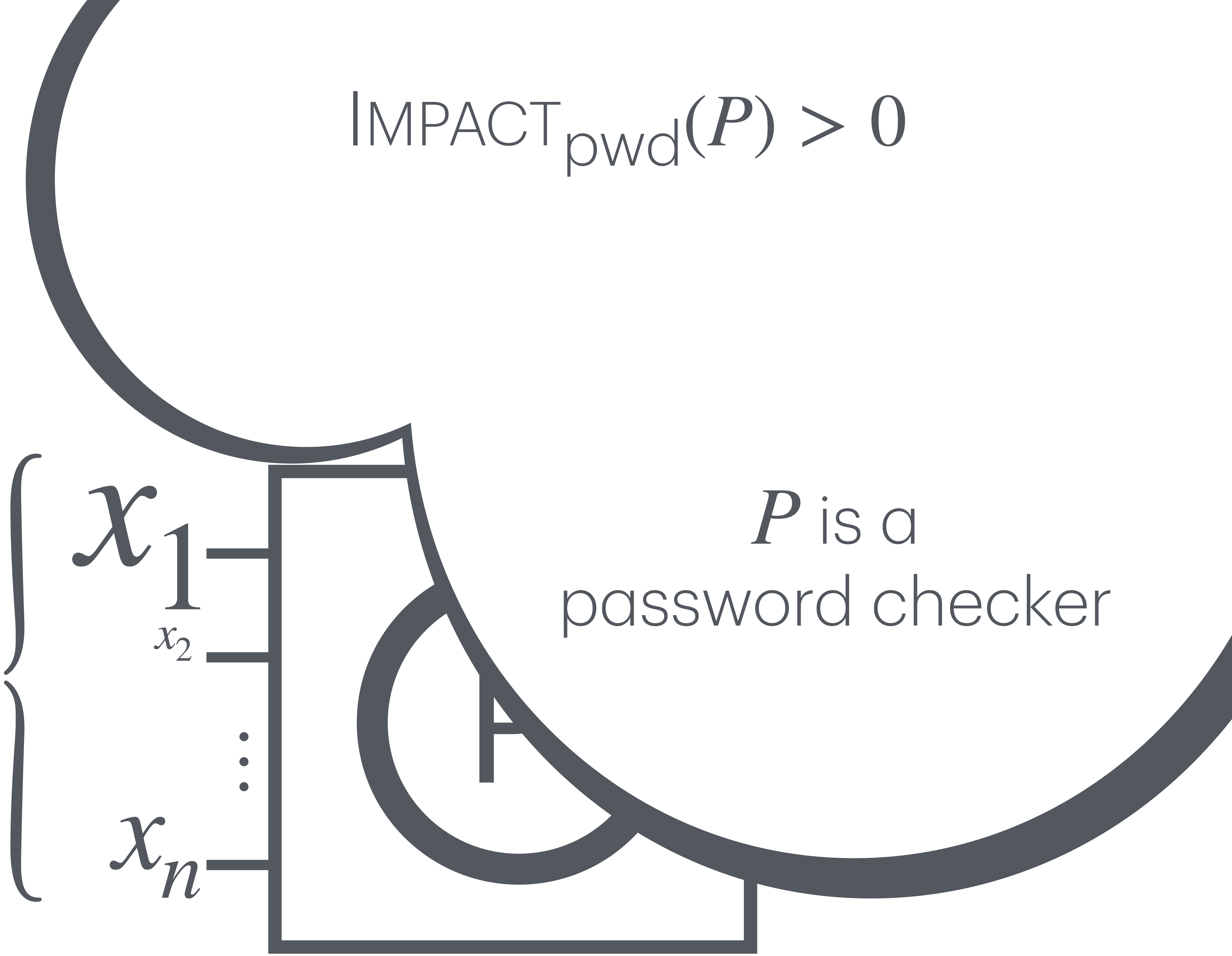
- Correct Loop Behaviour
- Performance
- Security



P is a password checker

Why?

- Correct Loop Behaviour
- Performance
- Security

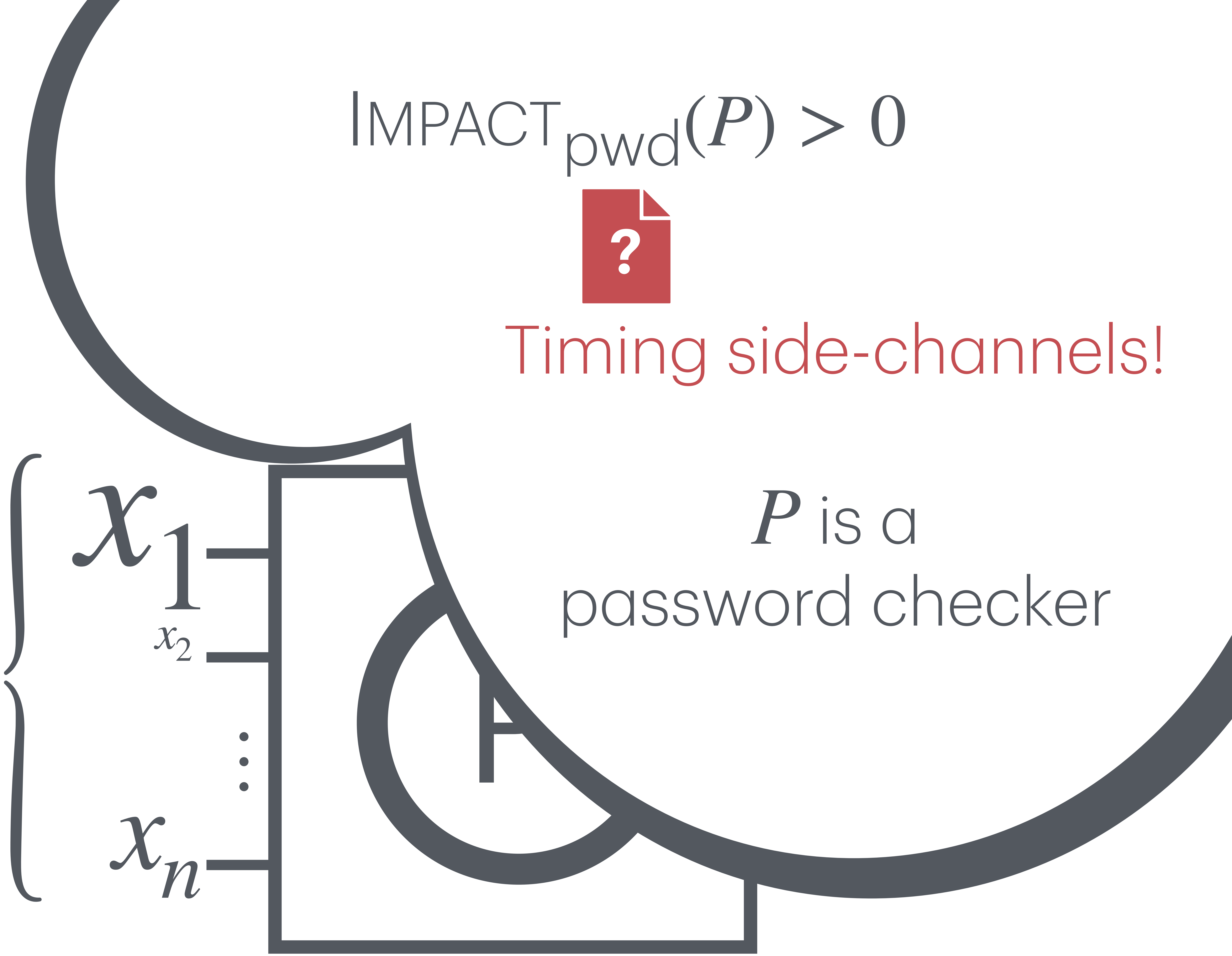


$\textsc{Impact}_{\text{pwd}}(P) > 0$

P is a password checker

Why?

- Correct Loop Behaviour
- Performance
- Security



$\textsc{Impact}_{\mathrm{pwd}}(P) > 0$

Timing side-channels!

P is a password checker

Add Function

S2N-Bignum

$$\begin{array}{cc} 3 & 8 \\ & 4 \end{array}$$

Add Function

4 2 =

3 8 +

4

Add Function

array

[0, 4, 2] =

[3, 8] +

[4]

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]

```
def Add(p, z, m, x, n, y):
  r = min(p, m)
  s = min(p, n)
  if (r < s):
    t = p - s
    q = s - r
    i = 0
    a = 0
    for (; r > 0; r--):
      s = x[i]
      w = y[i]
      z[i] = s + w + a
      i = i + 1
      a = (w < a) ||
        (s + w < s) ||
        (s + w + a < s)
    do:
      r = y[i]
      b = (r < a) ||
        (r + a < r)
      z[i] = r + a
      i = i + 1
      q--
      a = b
    while (q > 0)
  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

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

```
def Add(p, z, m, x, n, y):
    r = min(p, m)
    s = min(p, n)
    if (r < s):
        t = p - s
        q = s - r
        i = 0
        a = 0
        for (; r > 0; r--):
            s = x[i]
            w = y[i]
            z[i] = s + w + a
            i = i + 1
            a = (w < a) ||
                (s + w < s) ||
                (s + w + a < s)
        do:
            r = y[i]
            b = (r < a) ||
                (r + a < r)
            z[i] = r + a
            i = i + 1
            q--
            a = b
        while (q > 0)
    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

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

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

Add Function

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

1 iteration

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

Add Function

$$\texttt{Add}\begin{pmatrix}1, z,\\ 1, [4],\\ 1, [2]\end{pmatrix} \xrightarrow{\text{1 iteration}} z = [6]$$

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

Add Function

$$\textcolor{red}{\mathtt{Add}}\begin{pmatrix}1, z,\\ 1, [4],\\ 1, [2]\end{pmatrix} \xrightarrow{\text{1 iteration}} z = [6]$$

$$\textcolor{red}{\mathtt{Add}}\begin{pmatrix}2, z,\\ 1, [2],\\ 2, [1, 2]\end{pmatrix} \xrightarrow{\text{2 iterations}} z = [1, 4]$$

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

Add Function

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

$$\mathrm{Add}\begin{pmatrix} 1, z, \\ 1, [4], \\ 1, [2] \end{pmatrix} \xrightarrow{\text{1 iteration}} z = [6]$$

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

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

Add Function

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

$$\text{Add}\begin{pmatrix} 1, z, \\ 1, [4], \\ 1, [2] \end{pmatrix} \dashrightarrow \text{1 iteration} \dashrightarrow z = [6]$$

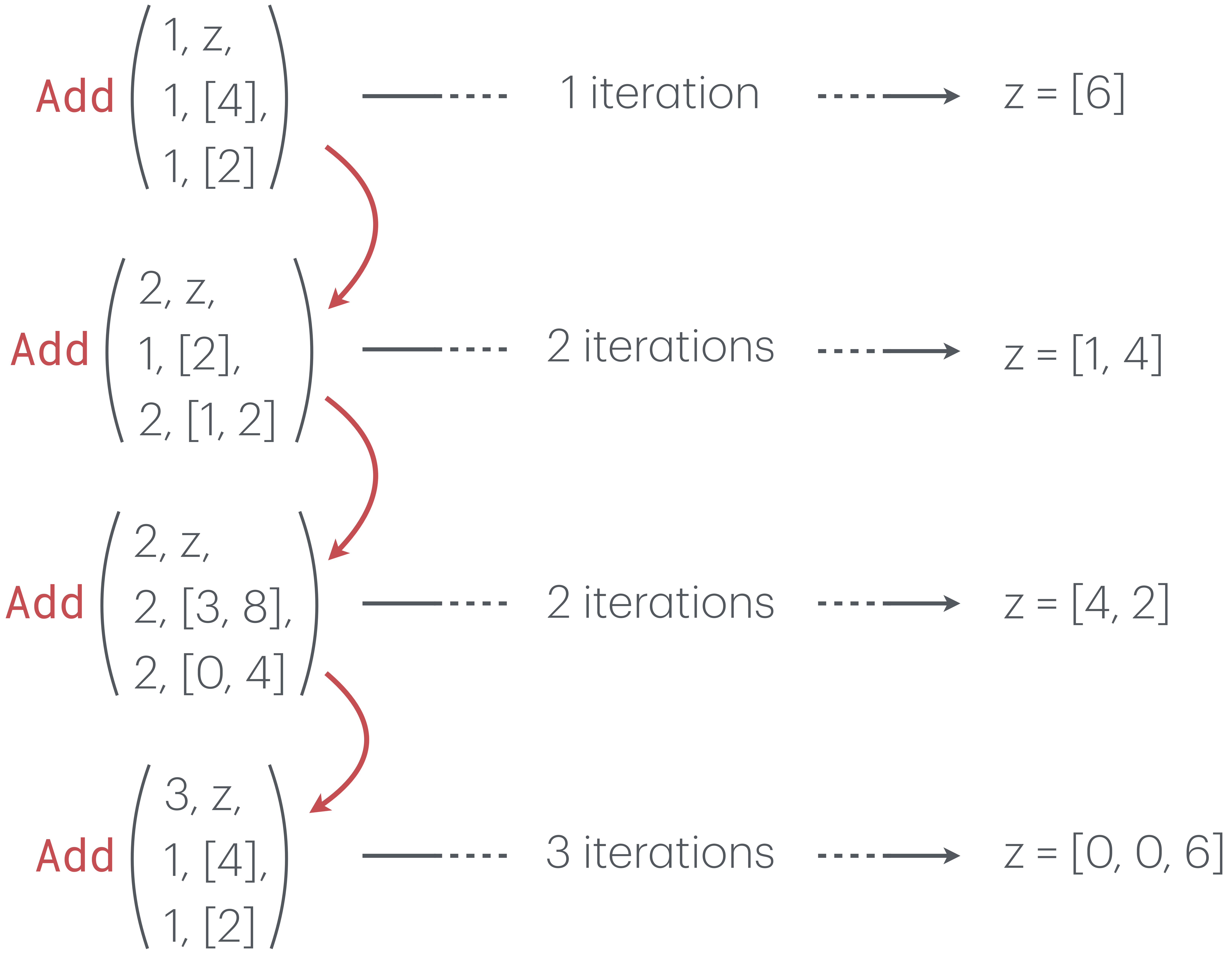
$$\text{Add}\begin{pmatrix} 2, z, \\ 1, [2], \\ 2, [1, 2] \end{pmatrix} \dashrightarrow \text{2 iterations} \dashrightarrow z = [1, 4]$$

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

$$\text{Add}\begin{pmatrix} 3, z, \\ 1, [4], \\ 1, [2] \end{pmatrix} \dashrightarrow \text{3 iterations} \dashrightarrow z = [0, 0, 6]$$

Add Function

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



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

1 iteration

$z = [6]$

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

2 iterations

$z = [1, 4]$

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

2 iterations

$z = [4, 2]$

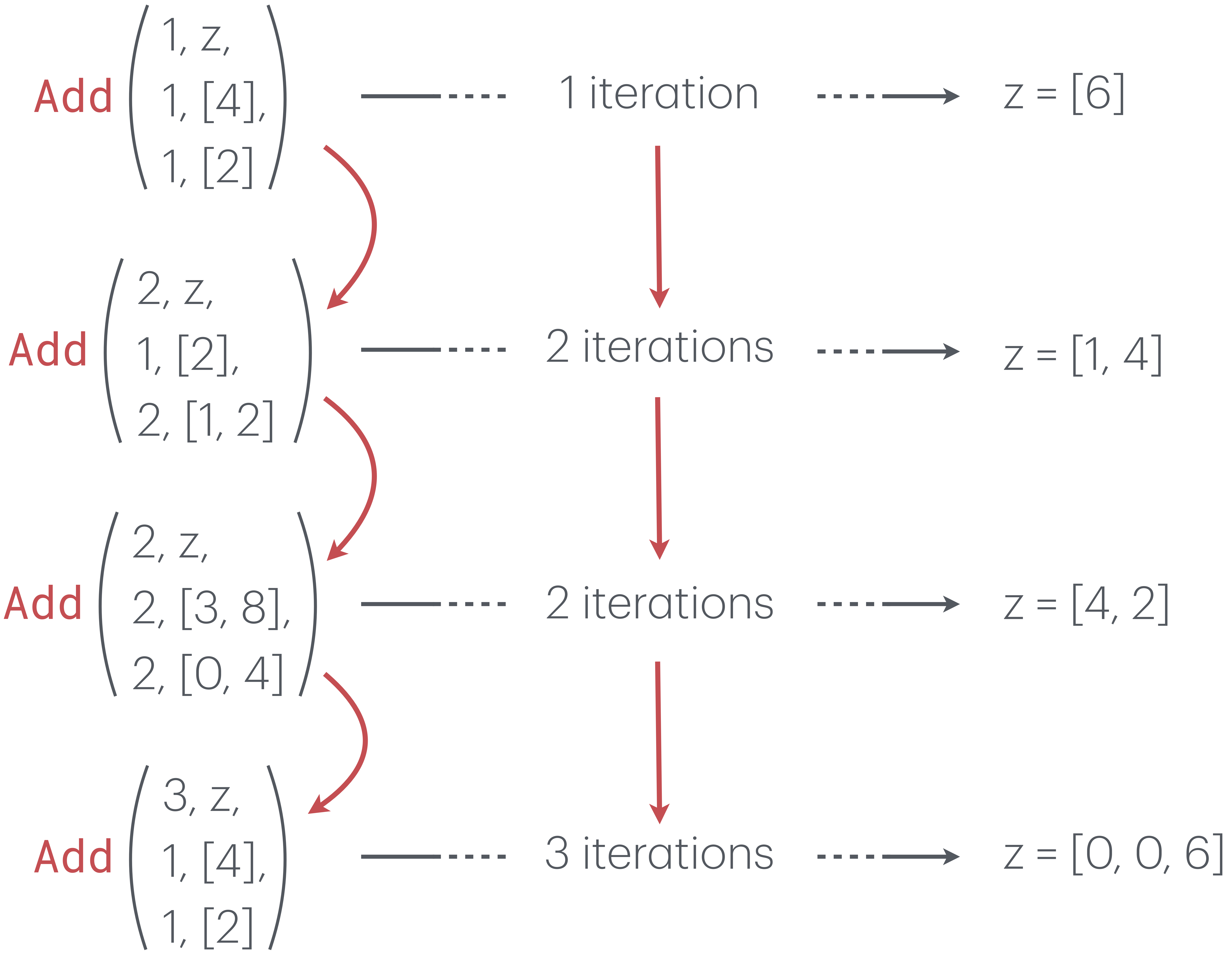
$$\text{Add}\begin{pmatrix} 3, z, \\ 1, [4], \\ 1, [2] \end{pmatrix}$$

3 iterations

$z = [0, 0, 6]$

Add Function

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



$\mathrm{Add}\begin{pmatrix} 1, z, \\ 1, [4], \\ 1, [2] \end{pmatrix}$ — 1 iteration → z = [6]

$\mathrm{Add}\begin{pmatrix} 2, z, \\ 1, [2], \\ 2, [1, 2] \end{pmatrix}$ — 2 iterations → z = [1, 4]

$\mathrm{Add}\begin{pmatrix} 2, z, \\ 2, [3, 8], \\ 2, [0, 4] \end{pmatrix}$ — 2 iterations → z = [4, 2]

$\mathrm{Add}\begin{pmatrix} 3, z, \\ 1, [4], \\ 1, [2] \end{pmatrix}$ — 3 iterations → z = [0, 0, 6]

How we compute $\textsc{Impact}_p$(`Add`)?

How we compute $\textsc{Impact}_{p}(\texttt{Add})$?

p input variable of interest

How we compute $\textsc{Impact}_{p}(\texttt{Add})$?

p input variable of interest

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

How we compute $\textsc{Impact}_{p}(\texttt{Add})$?

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]

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

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 IMPACT_{p}(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.	$\{0, 1, \cdots, 3, \cdots, u\}$

For all input values!

How we compute $\text{IMPACT}_{\mathtt{p}}(\texttt{Add})$?

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!

$$\mathrm{Range}(\{0, 1, \cdots, 3, \cdots, u\}) = u$$

How we compute $\textsc{Impact}_{\text{p}}(\texttt{Add})$?

$$\textsc{Impact}_x(P) \triangleq$$

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

How we compute $\textsc{Impact}_{\mathrm{p}}(\texttt{Add})$?

set of traces

$$\textsc{Impact}_{x}(P) \triangleq$$

$$\max_{s} \mathrm{Range}(\{\text{iterations of } \sigma \mid \sigma \in P \wedge \sigma_0(\text{variables} \setminus x) = s\})$$

How we compute $\textsc{Impact}_{\mathrm{p}}(\texttt{Add})$?

set of traces

$$\textsc{Impact}_x(P) \triangleq$$

$$\max_{s} \mathrm{Range}(\{\text{iterations of } \sigma \mid \sigma \in P \wedge \sigma_0(\text{variables} \setminus x) = s\})$$

input values

How we compute $\textsc{Impact}_p($`Add`$)$?

set of traces

for all traces

$$\textsc{Impact}_x(P) \triangleq \max_{s} \text{Range}(\{\text{iterations of } \boldsymbol{\sigma} \mid \boldsymbol{\sigma} \in P \wedge \boldsymbol{\sigma}_0(\text{variables} \setminus x) = s\})$$

input values

How we compute $\textsc{Impact}_p(\texttt{Add})$?

set of traces

for all traces

$$\textsc{Impact}_x(P) \triangleq \max_{s} \text{Range}(\{\text{iterations of } \sigma \mid \sigma \in P \land \sigma_0(\text{variables} \setminus x) = s\})$$

all variables but x

input values

How we compute $\textsc{Impact}_\text{p}(\texttt{Add})$?

set of traces

for all traces

$$\textsc{Impact}_x(P) \triangleq$$

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

all variables but x

input values

all perturbations of variable x

Add Function

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

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

Add Function

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

size data

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

- $\textsc{Impact}_{\{p,m,n\}}(\mathtt{Add}) \geq 0$

Add Function

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

$$\begin{array}{cc} \text{size} & \text{data} \end{array}$$

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

- $\textsc{Impact}_{\{p,m,n\}}(\texttt{Add}) \geq 0$
- $\textsc{Impact}_{\{z,x,y\}}(\texttt{Add}) = 0$

How?

$$\textsc{Impact}_{\{z,x,y\}}(\texttt{Add})$$

How?

abstract

concrete

$$\mathrm{Impact}^{\natural}_{\{z,x,y\}}(\texttt{Add}) \geq \textsc{Impact}_{\{z,x,y\}}(\texttt{Add})$$

How?

abstract

concrete

$$\mathrm{Impact}^{\natural}_{\{z,x,y\}}(\texttt{Add}) = 0 \text{ then } \textsc{Impact}_{\{z,x,y\}}(\texttt{Add}) = 0$$

$$\mathrm{Impact}^{\natural}_{\{z,x,y\}}(\texttt{Add}) \geq \textsc{Impact}_{\{z,x,y\}}(\texttt{Add})$$

Computing Impact$_{x}^{\natural}(P)$

In three steps:

(i) Remove irrelevant instructions

Computing Impact$^{\natural}_{x}(P)$

In three steps:

(i) Remove irrelevant instructions

Syntactic Dependency Analysis

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

In three steps:

(i) Remove irrelevant instructions

Syntactic Dependency Analysis

(ii) Abstract Interpretation

Computing $\mathrm{Impact}^{\natural}_{x}(P)$

In three steps:

(i) Remove irrelevant instructions

Syntactic Dependency Analysis

(ii) Abstract Interpretation

Invariant on input variables + iteration counter

Computing $\mathrm{Impact}^{\natural}_{x}(P)$

In three steps:

(i) Remove irrelevant instructions

Syntactic Dependency Analysis

(ii) Abstract Interpretation

Invariant on input variables + iteration counter

(iii) Impact quantification

Computing $\text{Impact}^{\natural}_{x}(P)$

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

```
def Add(p, z, m, x, n, y):
    r = min(p, m)
    s = min(p, n)
    if (r < s):
        t = p - s
        q = s - r
        i = 0
        a = 0
        for (; r > 0; r--):
            s = x[i]
            w = y[i]
            z[i] = s + w + a
            i = i + 1
            a = (w < a) ||
                (s + w < s) ||
                (s + w + a < s)
        do:
            r = y[i]
            b = (r < a) ||
                (r + a < r)
            z[i] = r + a
            i = i + 1
            q--
            a = b
        while (q > 0)
    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

```
def Add(p, z, m, x, n, y):
    r = min(p, m)
    s = min(p, n)
    if (r < s):
        t = p - s
        q = s - r
        i = 0
        a = 0
        for (; r > 0; r--):
            s = x[i]
            w = y[i]
            z[i] = s + w + a
            i = i + 1
            a = (w < a) ||
                (s + w < s) ||
                (s + w + a < s)
        do:
            r = y[i]
            b = (r < a) ||
                (r + a < r)
            z[i] = r + a
            i = i + 1
            q--
            a = b
        while (q > 0)
    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

```
def Add(p, z, m, x, n, y):
    r = min(p, m)
    s = min(p, n)
    if (r < s):
        t = p - s
        q = s - r
        i = 0
        a = 0
        for (; r > 0; r--):
            s = x[i]
            w = y[i]
            z[i] = s + w + a
            i = i + 1
            a = (w < a) ||
                (s + w < s) ||
                (s + w + a < s)
        do:
            r = y[i]
            b = (r < a) ||
                (r + a < r)
            z[i] = r + a
            i = i + 1
            q--
            a = b
        while (q > 0)
    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

```
def Add(p, z, m, x, n, y):
    r = min(p, m)
    s = min(p, n)
    if (r < s):
        t = p - s
        q = s - r
        i = 0
        a = 0
        for (; r > 0; r--):
            s = x[i]
            w = y[i]
            z[i] = s + w + a
            i = i + 1
            a = (w < a) ||
                (s + w < s) ||
                (s + w + a < s)
        do:
            r = y[i]
            b = (r < a) ||
                (r + a < r)
            z[i] = r + a
            i = i + 1
            q--
            a = b
        while (q > 0)
    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

```
def Add(p, z, m, x, n, y):
  r = min(p, m)
  s = min(p, n)
  if (r < s):
    t = p - s
    q = s - r
    i = 0
    a = 0
    for (; r > 0; r--):
      s = x[i]
      w = y[i]
      z[i] = s + w + a
      i = i + 1
      a = (w < a) ||
        (s + w < s) ||
        (s + w + a < s)
    do:
      r = y[i]
      b = (r < a) ||
        (r + a < r)
      z[i] = r + a
      i = i + 1
      q--
      a = b
    while (q > 0)
  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

```
def Add(p, z, m, x, n, y):
    r = min(p, m)
    s = min(p, n)
    if (r < s):
        t = p - s
        q = s - r
        i = 0
        a = 0
        for (; r > 0; r--):
            s = x[i]
            w = y[i]
            z[i] = s + w + a
            i = i + 1
            a = (w < a) ||
                (s + w < s) ||
                (s + w + a < s)
        do:
            r = y[i]
            b = (r < a) ||
                (r + a < r)
            z[i] = r + a
            i = i + 1
            q--
            a = b
        while (q > 0)
    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

```
def Add(p, z, m, x, n, y):
    r = min(p, m)
    s = min(p, n)
    if (r < s):
        t = p - s
        q = s - r
        i = 0
        a = 0
        for (; r > 0; r--):
            s = x[i]
            w = y[i]
            z[i] = s + w + a
            i = i + 1
            a = (w < a) ||
                (s + w < s) ||
                (s + w + a < s)
        do:
            r = y[i]
            b = (r < a) ||
                (r + a < r)
            z[i] = r + a
            i = i + 1
            q--
            a = b
        while (q > 0)
    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

```
def Add(p, z, m, x, n, y):
    r = min(p, m)
    s = min(p, n)
    if (r < s):
        t = p - s
        q = s - r
        i = 0
        a = 0
        for (; r > 0; r--):
            s = x[i]
            w = y[i]
            z[i] = s + w + a
            i = i + 1
            a = (w < a) ||
                (s + w < s) ||
                (s + w + a < s)
        do:
            r = y[i]
            b = (r < a) ||
                (r + a < r)
            z[i] = r + a
            i = i + 1
            q--
            a = b
        while (q > 0)
    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

```
def Add(p, z, m, x, n, y):
    r = min(p, m)
    s = min(p, n)
    if (r < s):
        t = p - s
        q = s - r
        for (; r > 0; r--):
            skip;
        do:
            q--;
        while (q > 0)
    else:
        t = p - r
        q = r - s
        for (; s > 0; s--):
            skip;
        for (; q > 0; q--):
            skip;
    if (t > 0):
        while (t > 0):
            t--;
```

(ii) Abstract Interpretation

```
def Add(p, z, m, x, n, y):
    r = min(p, m)
    s = min(p, n)
    if (r < s):
        t = p - s
        q = s - r
        for (; r > 0; r--):
            skip;
        do:
            q--;
        while (q > 0)
    else:
        t = p - r
        q = r - s
        for (; s > 0; s--):
            skip;
        for (; q > 0; q--):
            skip;
    if (t > 0):
        while (t > 0):
            t--;
```

Intuitively:

(ii) Abstract Interpretation

```
def Add(p, z, m, x, n, y):
    r = min(p, m)
    s = min(p, n)
    if (r < s):
        t = p - s
        q = s - r
        for (; r > 0; r--):
            skip; counter--
        do:
            q--; counter--
        while (q > 0)
    else:
        t = p - r
        q = r - s
        for (; s > 0; s--):
            skip; counter--
        for (; q > 0; q--):
            skip; counter--
    if (t > 0):
        while (t > 0):
            t--; counter--
```

Intuitively:

Augment each loop body with a counter for iterations

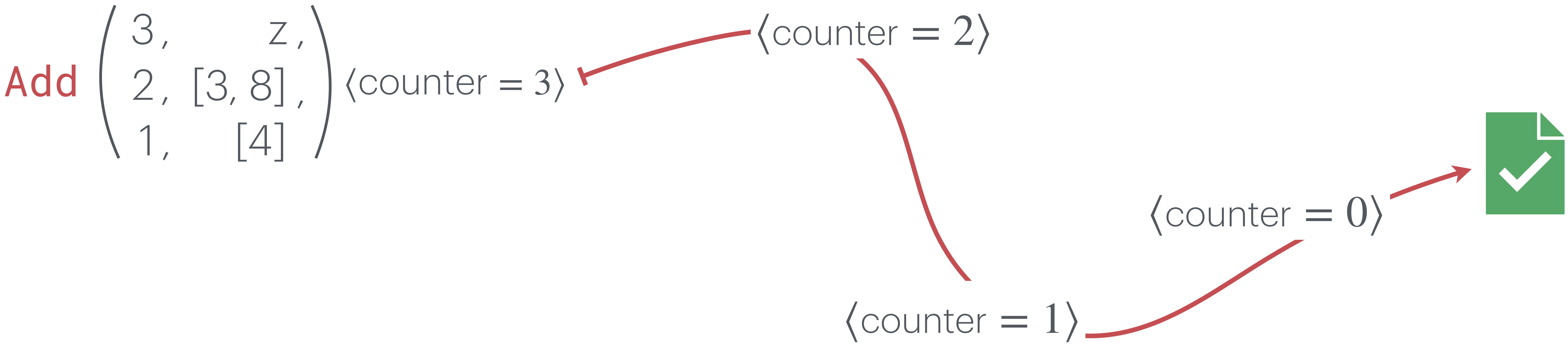
(ii) Abstract Interpretation

```
 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--):
--        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**

Backwards starting from zero!

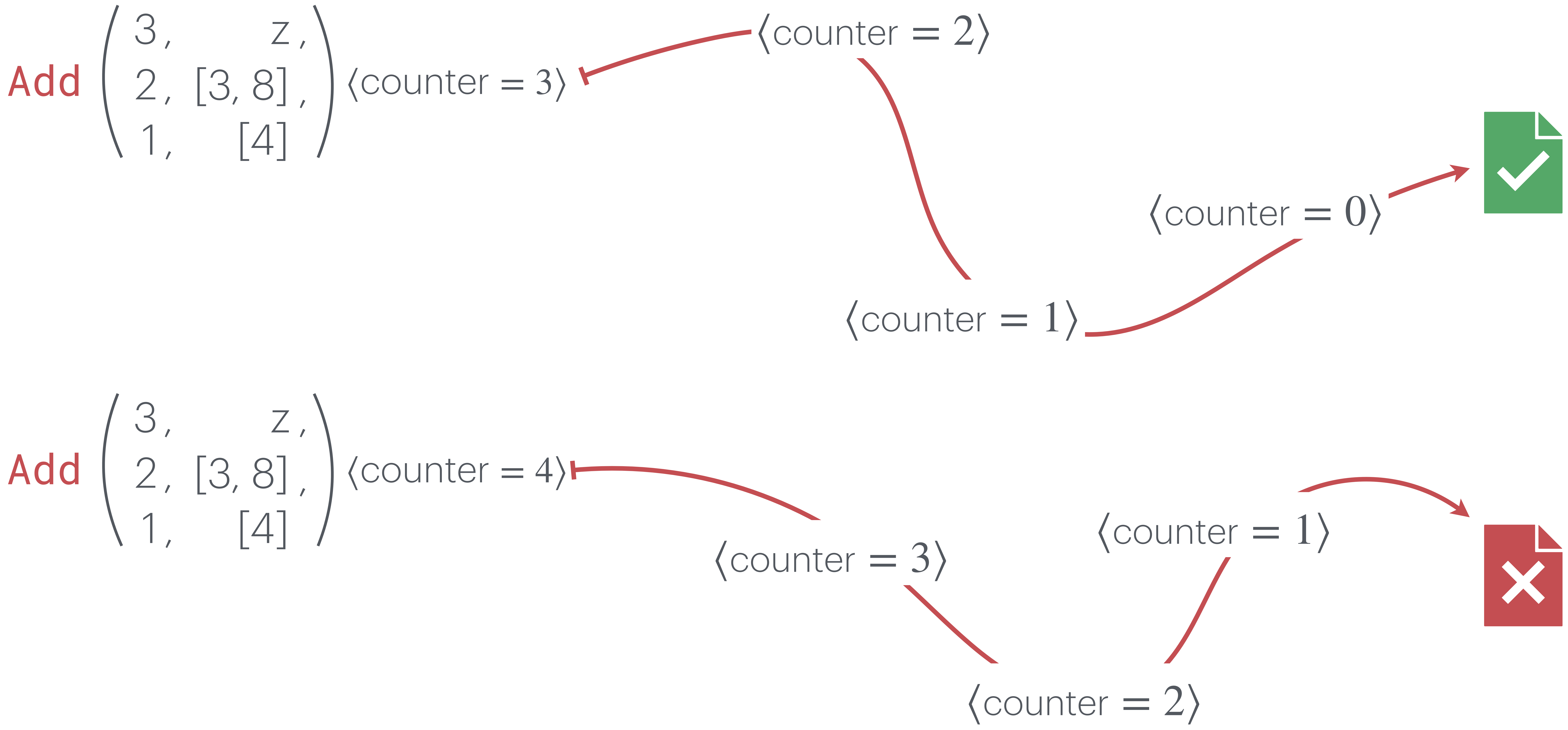


Add $\begin{pmatrix} 3, & z, \\ 2, & [3, 8], \\ 1, & [4] \end{pmatrix}$ $\langle \mathsf{counter} = 3 \rangle$

$\langle \mathsf{counter} = 2 \rangle$

$\langle \mathsf{counter} = 1 \rangle$

$\langle \mathsf{counter} = 0 \rangle$



$\texttt{Add}\begin{pmatrix}3, & z,\\ 2, & [3,8],\\ 1, & [4]\end{pmatrix}\langle \text{counter} = 3\rangle$

$\langle \text{counter} = 2\rangle$

$\langle \text{counter} = 1\rangle$

$\langle \text{counter} = 0\rangle$

✓

$\texttt{Add}\begin{pmatrix}3, & z,\\ 2, & [3,8],\\ 1, & [4]\end{pmatrix}\langle \text{counter} = 4\rangle$

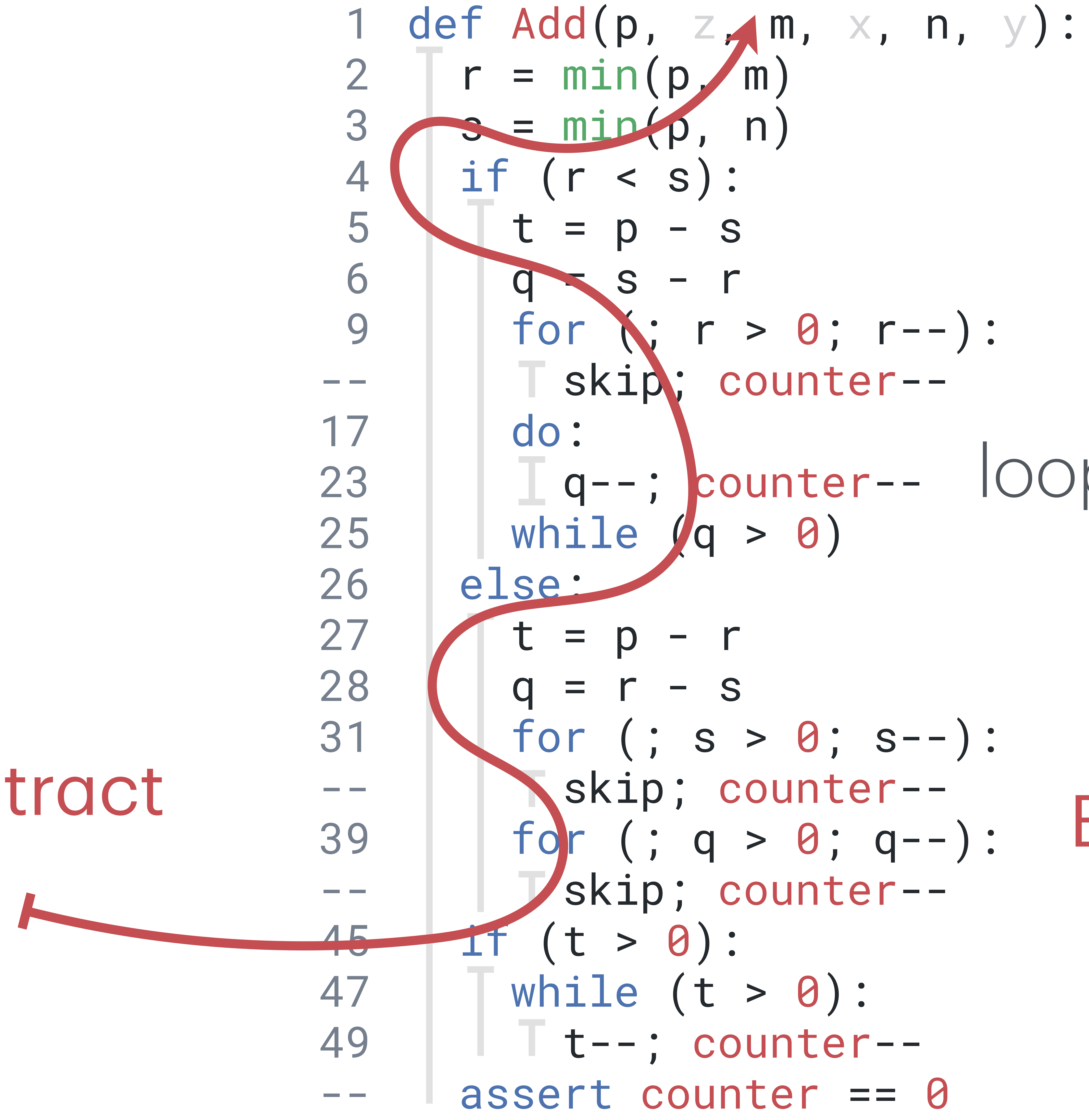
$\langle \text{counter} = 3\rangle$

$\langle \text{counter} = 2\rangle$

$\langle \text{counter} = 1\rangle$

✗

(ii) Abstract Interpretation



```
def Add(p, z, m, x, n, y):
  r = min(p, m)
  s = min(p, n)
  if (r < s):
    t = p - s
    q = s - r
    for (; r > 0; r--):
      skip; counter--
    do:
      q--; counter--
    while (q > 0)
  else:
    t = p - r
    q = r - s
    for (; s > 0; s--):
      skip; counter--
    for (; q > 0; q--):
      skip; counter--
  if (t > 0):
    while (t > 0):
      t--; counter--
  assert counter == 0
```

Backward abstract analysis

Intuitively:

Augment each loop body with a counter for iterations

Backwards starting from zero!

(ii) Abstract Interpretation

```
def Add(p, z, m, x, n, y):
    r = min(p, m)
    s = min(p, n)
    if (r < s):
        for (; r > 0; r--):
            skip; counter--
        do:
            q--; counter--
        while (q > 0)
    else:
        t = p - r
        q = r - s
            counter--
        for (; q > 0; q--):
            skip; counter--
        if (t > 0):
            while (t > 0):
                t--; counter--
    assert counter == 0
```

Without rewritings!

Iteration counter is handled

semantically

Backward Abstract Semantics

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

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

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

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

Backward Abstract Semantics

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

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

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

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

Backward Abstract Semantics

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

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

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

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

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

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

Backward Abstract Semantics

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

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

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

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

$$\Lambda^\natural[\![\mathsf{if}\ b\ \mathsf{then}\ \mathit{stmt}\ \mathsf{else}\ \mathit{stmt}']\!]d^\natural \triangleq$$

$$\mathsf{Filter}^\natural[\![b]\!](\Lambda^\natural[\![\mathit{stmt}]\!]) \sqcup^\natural \mathsf{Filter}^\natural[\![\neg b]\!](\Lambda^\natural[\![\mathit{stmt}']\!])$$

$$\Lambda^\natural[\![\mathsf{while}\ b\ \mathsf{do}\ \mathit{stmt}]\!]d^\natural \triangleq \lim_n F^n$$

Backward Abstract Semantics

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

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

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

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

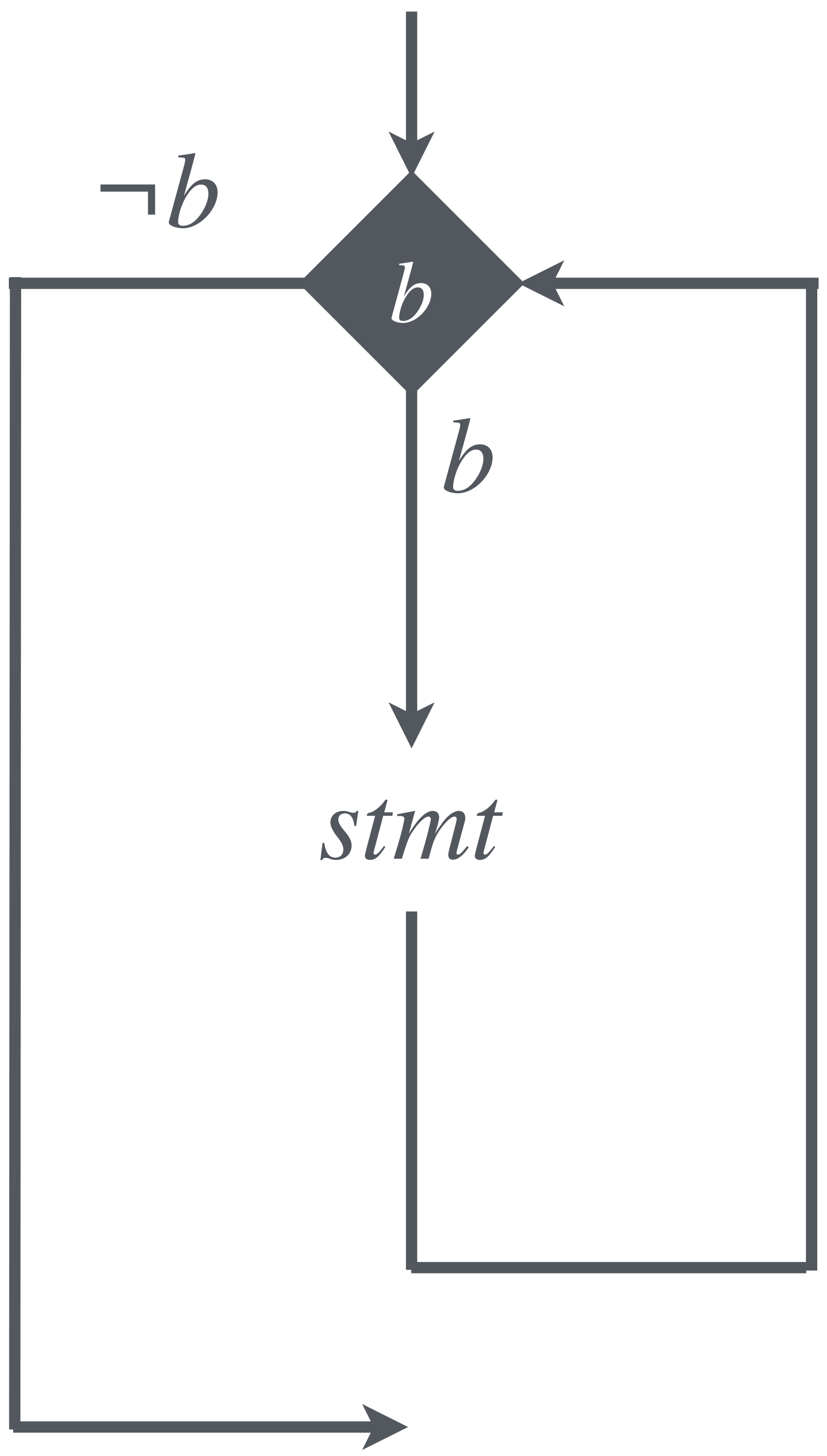
$$\Lambda^\natural[\![\text{if } b \text{ then } stmt \text{ else } stmt']\!]d^\natural \triangleq$$

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

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

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

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



Backward Abstract Semantics

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

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

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

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

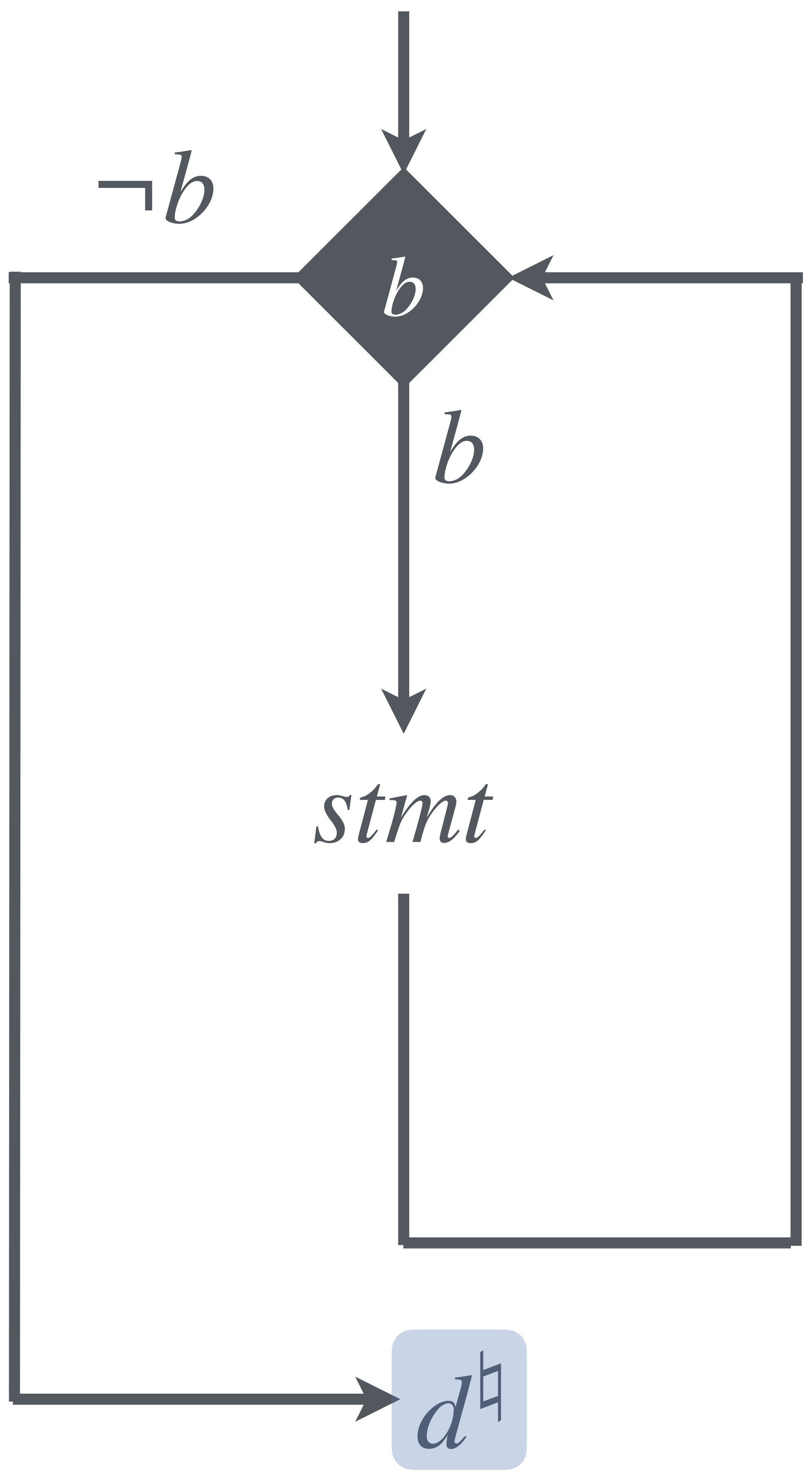
$$\Lambda^\natural[\![\text{if } b \text{ then } stmt \text{ else } stmt']\!]d^\natural \triangleq$$

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

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

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

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



Backward Abstract Semantics

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

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

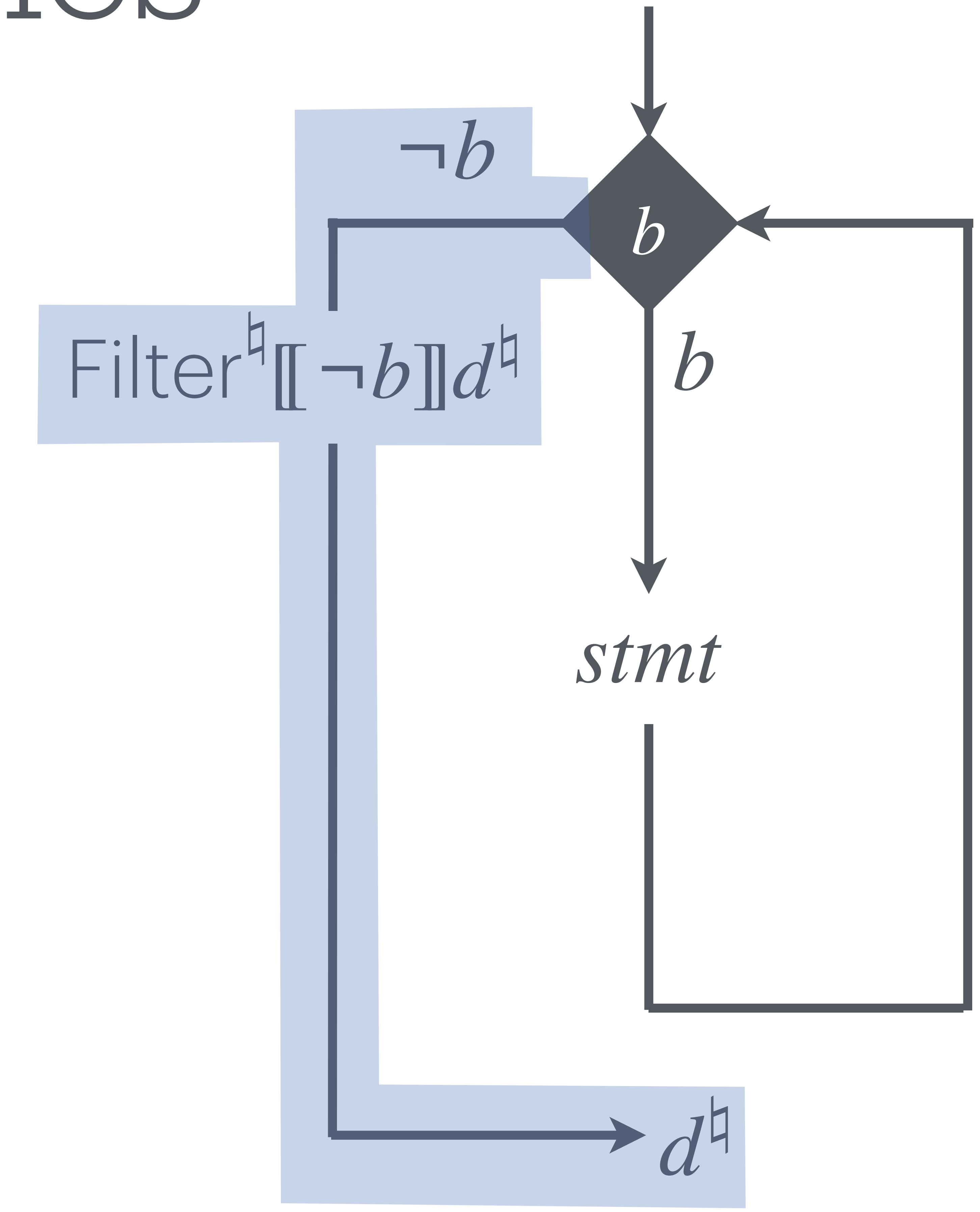
$$\Lambda^\natural[\![\text{skip}]\!]d^\natural \triangleq d^\natural$$

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

$$\Lambda^\natural[\![\text{if } b \text{ then } stmt \text{ else } stmt']\!]d^\natural \triangleq$$
$$\text{Filter}^\natural[\![b]\!](\Lambda^\natural[\![stmt]\!]) \sqcup^\natural \text{Filter}^\natural[\![\neg b]\!](\Lambda^\natural[\![stmt']\!])$$

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

$$F(x^\natural) \triangleq \boxed{\text{Filter}^\natural[\![\neg b]\!]d^\natural} \sqcup^\natural$$
$$\text{Filter}^\natural[\![b]\!](\Lambda^\natural[\![stmt]\!](\text{Substitute}^\natural[\![\text{counter} \leftarrow \text{counter} - 1]\!]x^\natural))$$



Backward Abstract Semantics

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

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

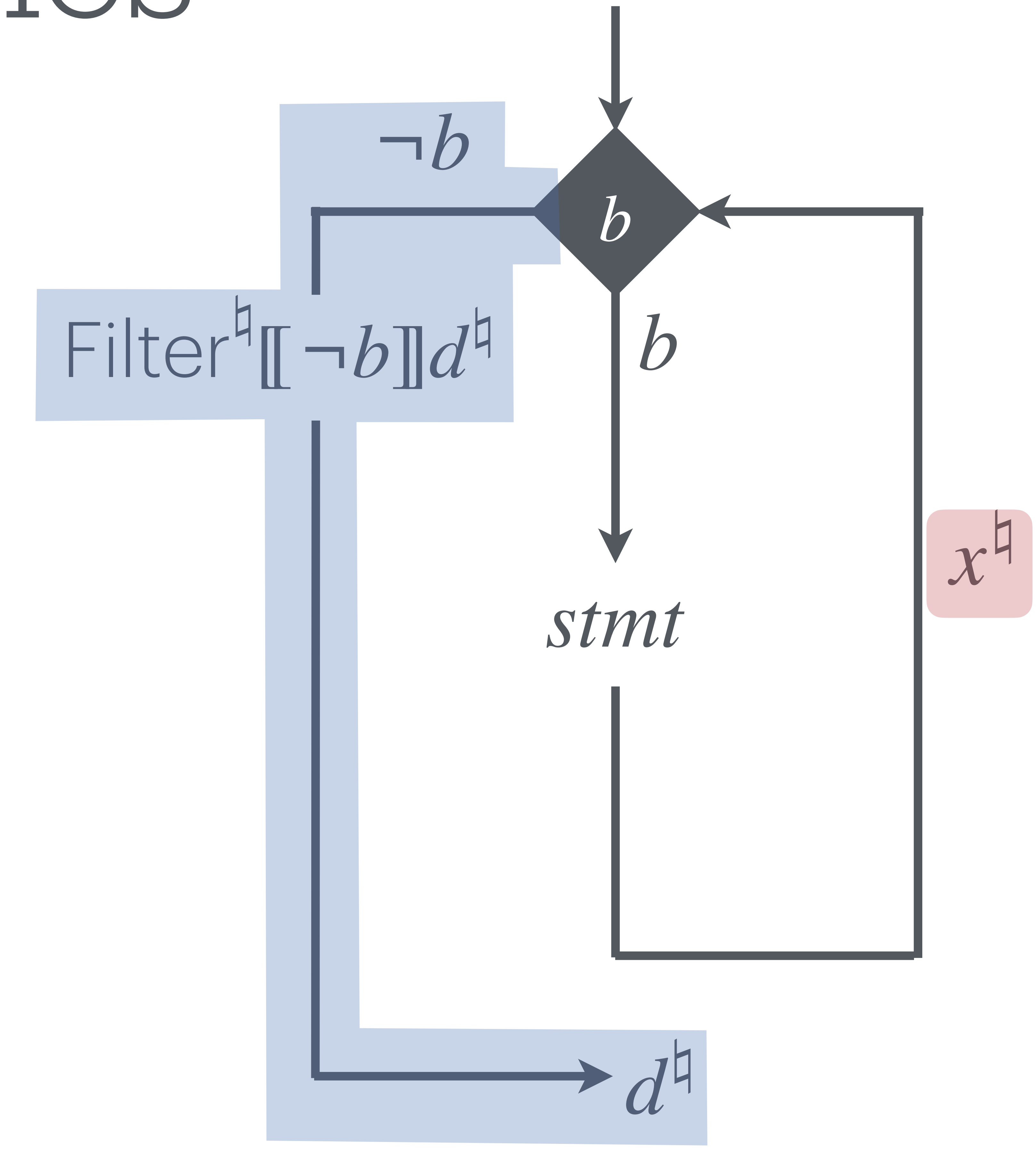
$$\Lambda^\natural[\![\text{skip}]\!]d^\natural \triangleq d^\natural$$

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

$$\Lambda^\natural[\![\text{if } b \text{ then } stmt \text{ else } stmt']\!]d^\natural \triangleq$$
$$\text{Filter}^\natural[\![b]\!](\Lambda^\natural[\![stmt]\!]) \sqcup^\natural \text{Filter}^\natural[\![\neg b]\!](\Lambda^\natural[\![stmt']\!])$$

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

$$F(x^\natural) \triangleq \text{Filter}^\natural[\![\neg b]\!]d^\natural \sqcup^\natural$$
$$\text{Filter}^\natural[\![b]\!](\Lambda^\natural[\![stmt]\!](\text{Substitute}^\natural[\![\text{counter} \leftarrow \text{counter} - 1]\!]x^\natural))$$



Backward Abstract Semantics

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

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

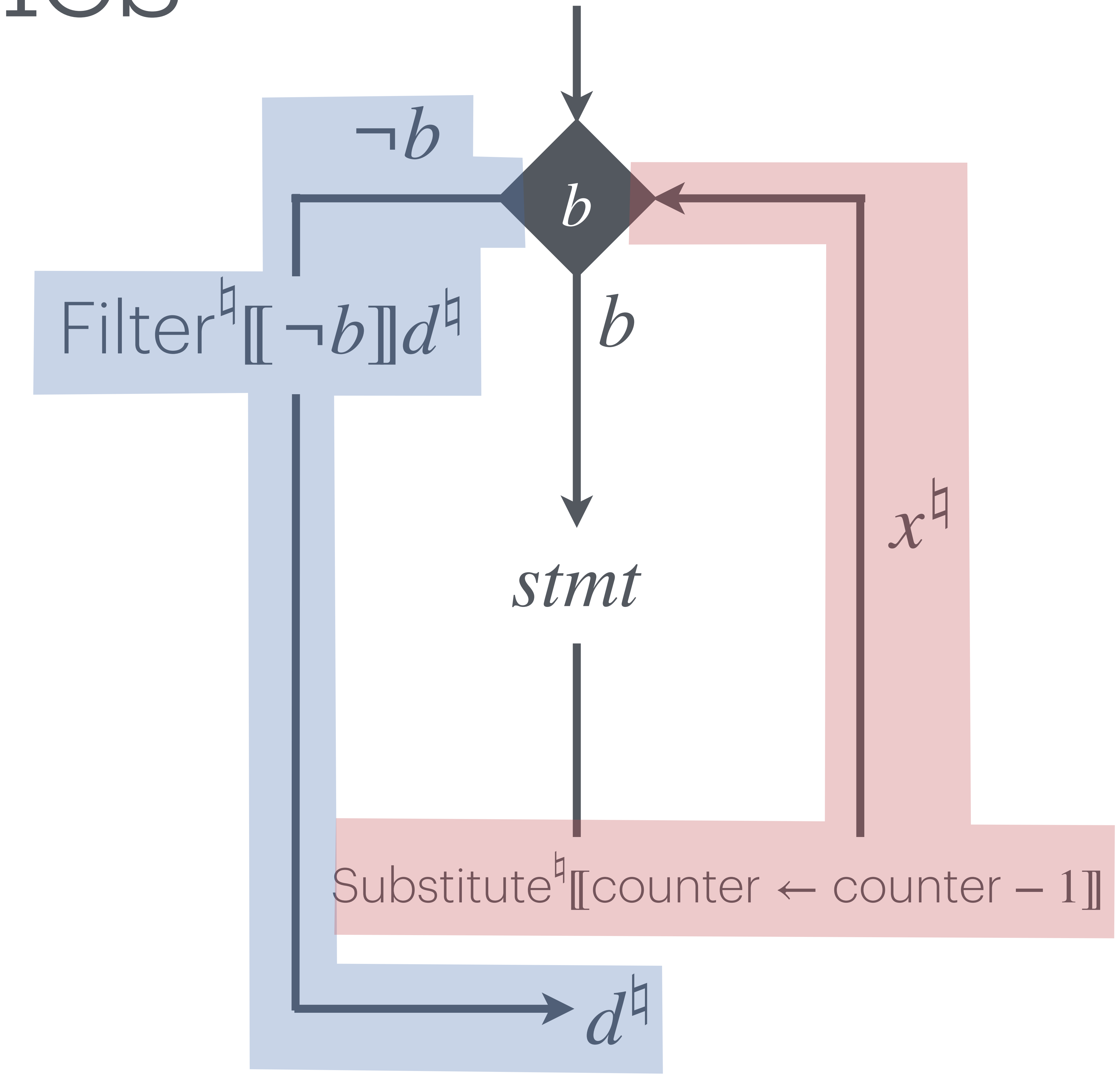
$$\Lambda^\natural[\![\text{skip}]\!]d^\natural \triangleq d^\natural$$

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

$$\Lambda^\natural[\![\text{if } b \text{ then } \mathit{stmt} \text{ else } \mathit{stmt}']\!]d^\natural \triangleq$$
$$\text{Filter}^\natural[\![b]\!](\Lambda^\natural[\![\mathit{stmt}]\!]) \sqcup^\natural \text{Filter}^\natural[\![\neg b]\!](\Lambda^\natural[\![\mathit{stmt}']\!])$$

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

$$F(x^\natural) \triangleq \text{Filter}^\natural[\![\neg b]\!]d^\natural \sqcup^\natural$$
$$\text{Filter}^\natural[\![b]\!](\Lambda^\natural[\![\mathit{stmt}]\!](\text{Substitute}^\natural[\![\text{counter} \leftarrow \text{counter} - 1]\!]x^\natural))$$


$\neg b$
b
$\text{Filter}^\natural[\![\neg b]\!]d^\natural$
b
stmt
$x^\natural$
$\text{Substitute}^\natural[\![\text{counter} \leftarrow \text{counter} - 1]\!]$
$d^\natural$

Backward Abstract Semantics

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

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

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

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

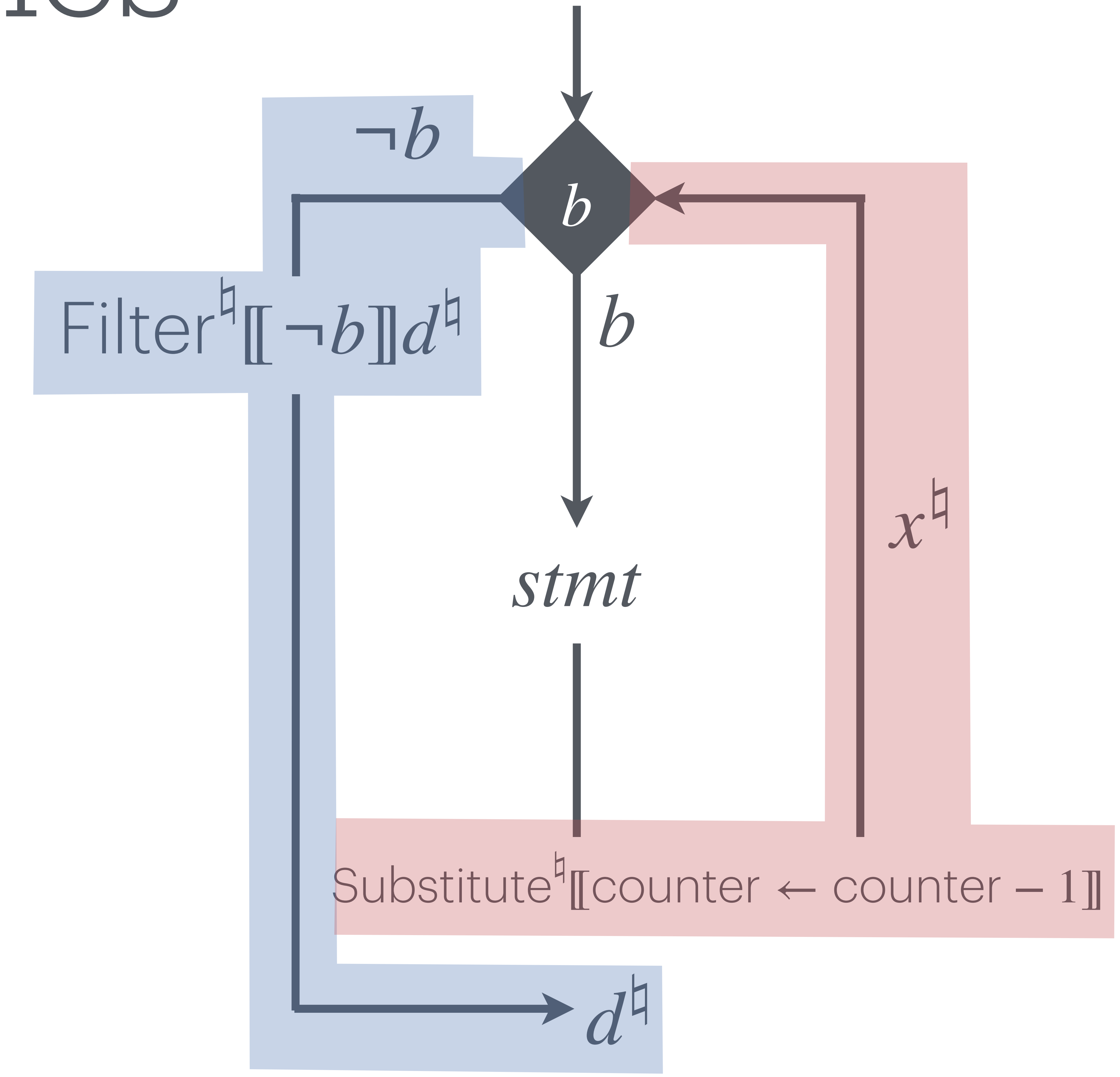
$$\Lambda^\natural[\![\text{if } b \text{ then } stmt \text{ else } stmt']\!]d^\natural \triangleq$$

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

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

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

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


$\neg b$
b
$\text{Filter}^\natural[\![\neg b]\!]d^\natural$
b
$stmt$
$x^\natural$
$\text{Substitute}^\natural[\![\text{counter} \leftarrow \text{counter} - 1]\!]$
$d^\natural$

Backward Abstract Semantics

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

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

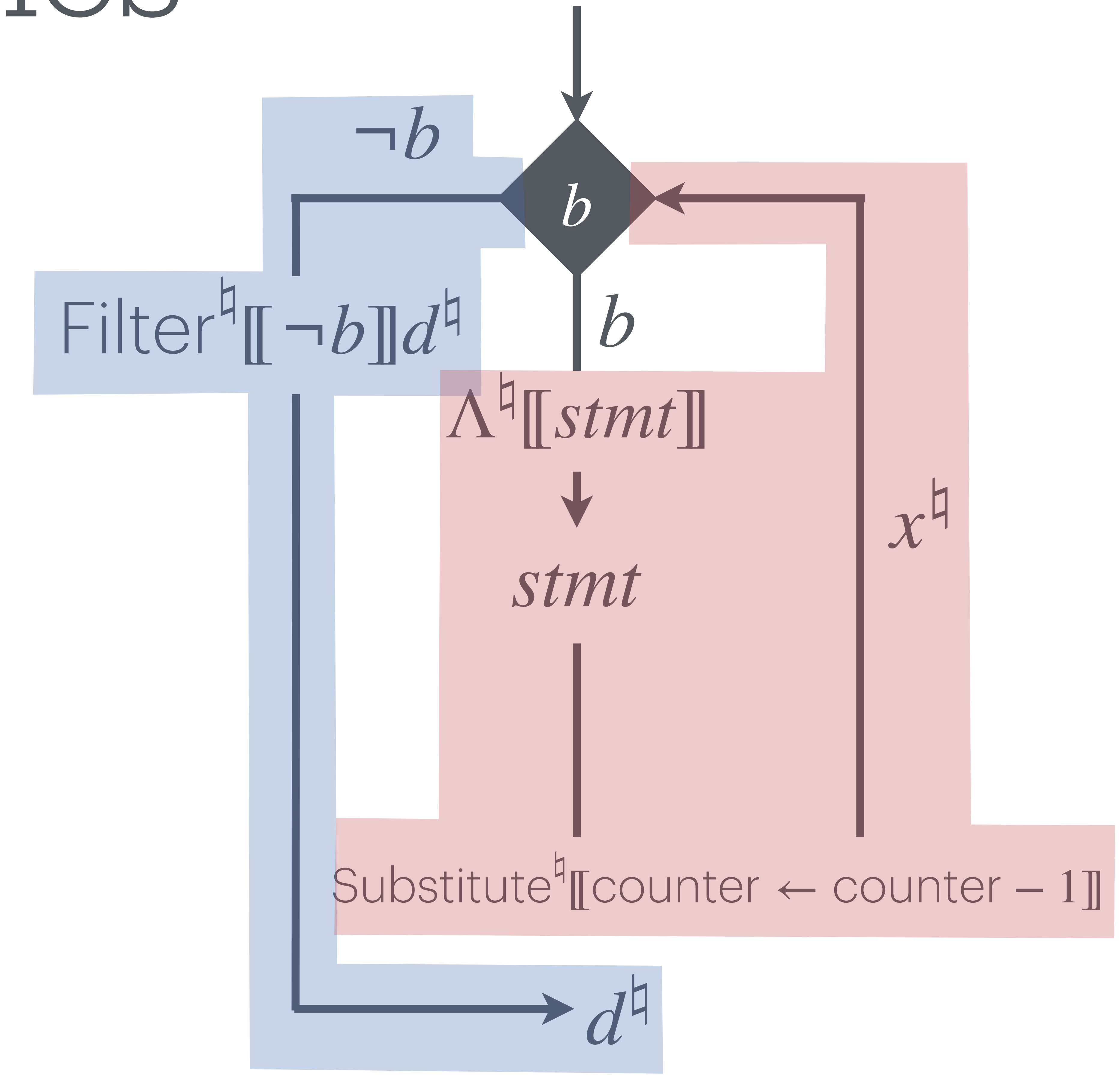
$$\Lambda^\natural[\![\text{skip}]\!]d^\natural \triangleq d^\natural$$

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

$$\Lambda^\natural[\![\text{if } b \text{ then } \mathit{stmt} \text{ else } \mathit{stmt}']\!]d^\natural \triangleq$$
$$\text{Filter}^\natural[\![b]\!](\Lambda^\natural[\![\mathit{stmt}]\!]) \sqcup^\natural \text{Filter}^\natural[\![\neg b]\!](\Lambda^\natural[\![\mathit{stmt}']\!])$$

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

$$F(x^\natural) \triangleq \text{Filter}^\natural[\![\neg b]\!]d^\natural \sqcup^\natural$$
$$\text{Filter}^\natural[\![b]\!](\Lambda^\natural[\![\mathit{stmt}]\!](\text{Substitute}^\natural[\![\text{counter} \leftarrow \text{counter} - 1]\!]x^\natural))$$



$\neg b$
b
$\text{Filter}^\natural[\![\neg b]\!]d^\natural$
b
$\Lambda^\natural[\![\mathit{stmt}]\!]$
stmt
$x^\natural$
$\text{Substitute}^\natural[\![\text{counter} \leftarrow \text{counter} - 1]\!]$
$d^\natural$

Backward Abstract Semantics

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

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

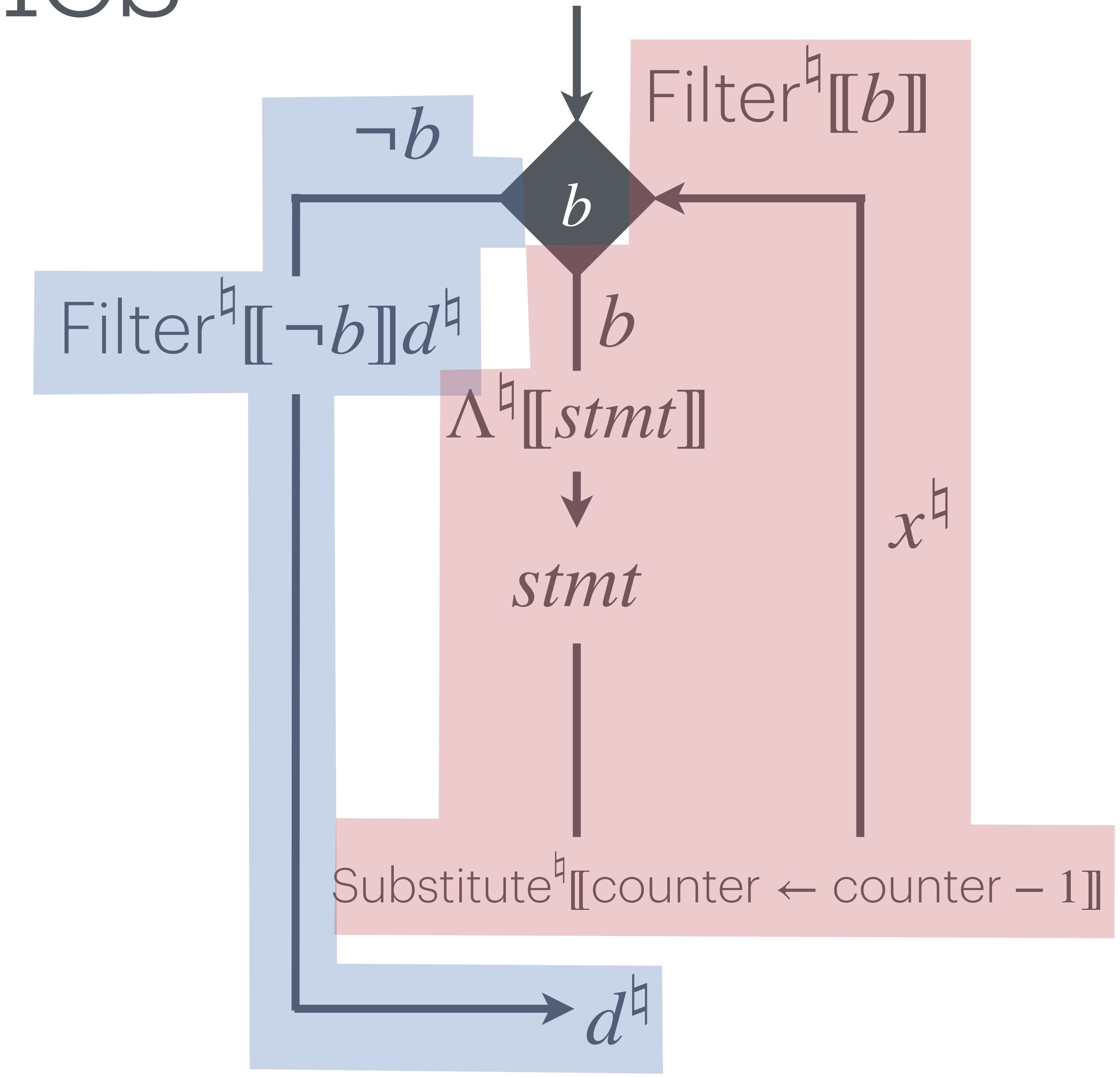
$$\Lambda^\natural[\![\text{skip}]\!]d^\natural \triangleq d^\natural$$

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

$$\Lambda^\natural[\![\text{if } b \text{ then } \mathit{stmt} \text{ else } \mathit{stmt}']\!]d^\natural \triangleq$$
$$\text{Filter}^\natural[\![b]\!](\Lambda^\natural[\![\mathit{stmt}]\!]) \sqcup^\natural \text{Filter}^\natural[\![\neg b]\!](\Lambda^\natural[\![\mathit{stmt}']\!])$$

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

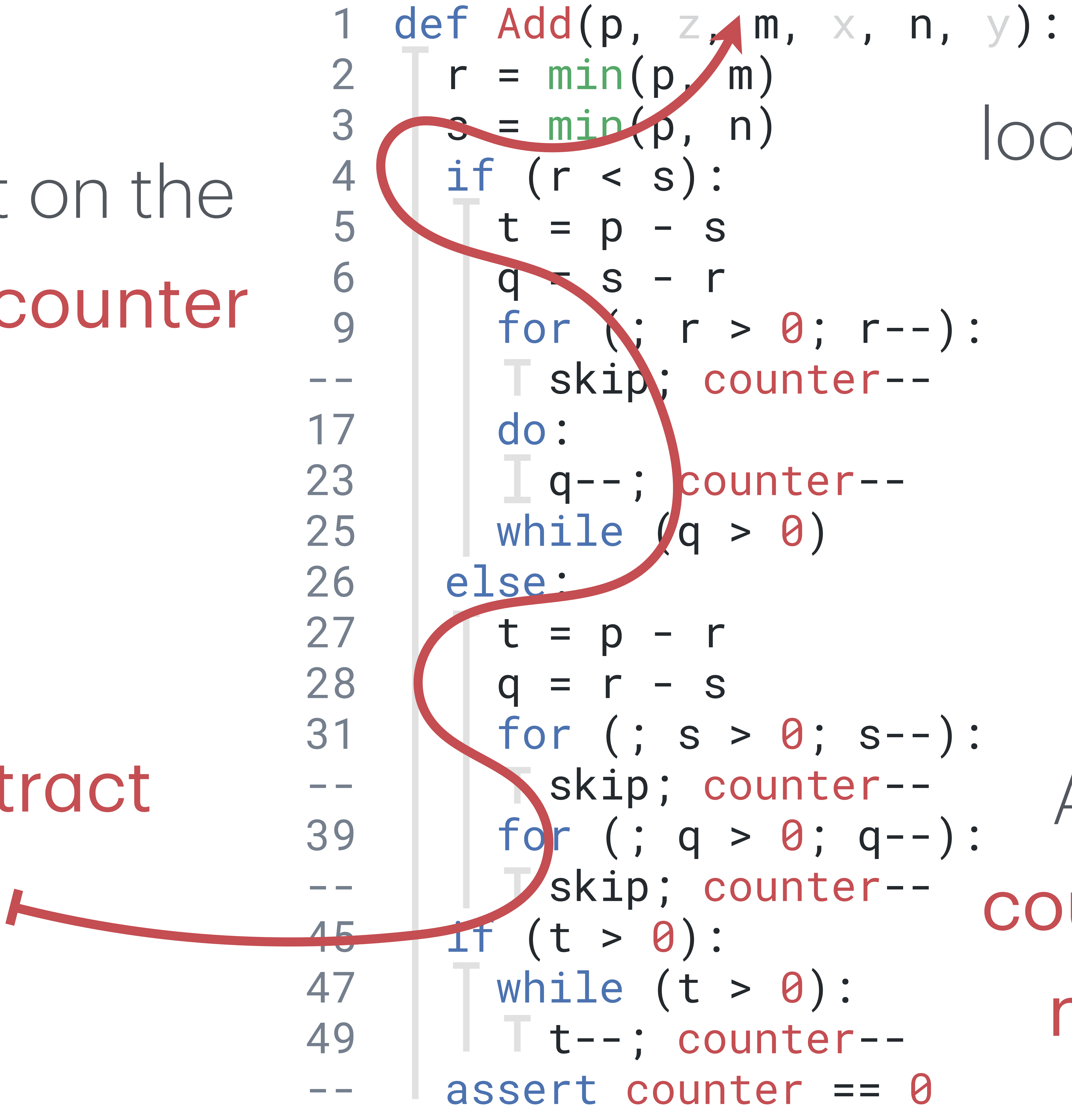
$$F(x^\natural) \triangleq \text{Filter}^\natural[\![\neg b]\!]d^\natural \sqcup^\natural$$
$$\text{Filter}^\natural[\![b]\!](\Lambda^\natural[\![\mathit{stmt}]\!](\text{Substitute}^\natural[\![\text{counter} \leftarrow \text{counter} - 1]\!]x^\natural))$$



$\text{Filter}^\natural[\![b]\!]$
$\neg b$
b
$\text{Filter}^\natural[\![\neg b]\!]d^\natural$
b
$\Lambda^\natural[\![\mathit{stmt}]\!]$
$x^\natural$
stmt
$\text{Substitute}^\natural[\![\text{counter} \leftarrow \text{counter} - 1]\!]$
$d^\natural$

(ii) Abstract Interpretation

Abstract invariant on the **input variables** + **counter**



```
def Add(p, z, m, x, n, y):
    r = min(p, m)
    s = min(p, n)
    if (r < s):
        t = p - s
        q = s - r
        for (; r > 0; r--):
            skip; counter--
        do:
            q--; counter--
        while (q > 0)
    else:
        t = p - r
        q = r - s
        for (; s > 0; s--):
            skip; counter--
        for (; q > 0; q--):
            skip; counter--
    if (t > 0):
        while (t > 0):
            t--; counter--
    assert counter == 0
```

Backward abstract analysis

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

```
def Add(p, z, m, x, n, y):
    r = min(p, m)
    s = min(p, n)
    if (r < s):
        t = p - s
        q = s - r
        for (; r > 0; r--):
            skip; counter--
        do:
            q--; counter--
        while (q > 0)
    else:
        t = p - r
        q = r - s
        for (; s > 0; s--):
            skip; counter--
        for (; q > 0; q--):
            skip; counter--
    if (t > 0):
        while (t > 0):
            t--; counter--
    assert counter == 0
```

Augment each loop body with a counter for iterations

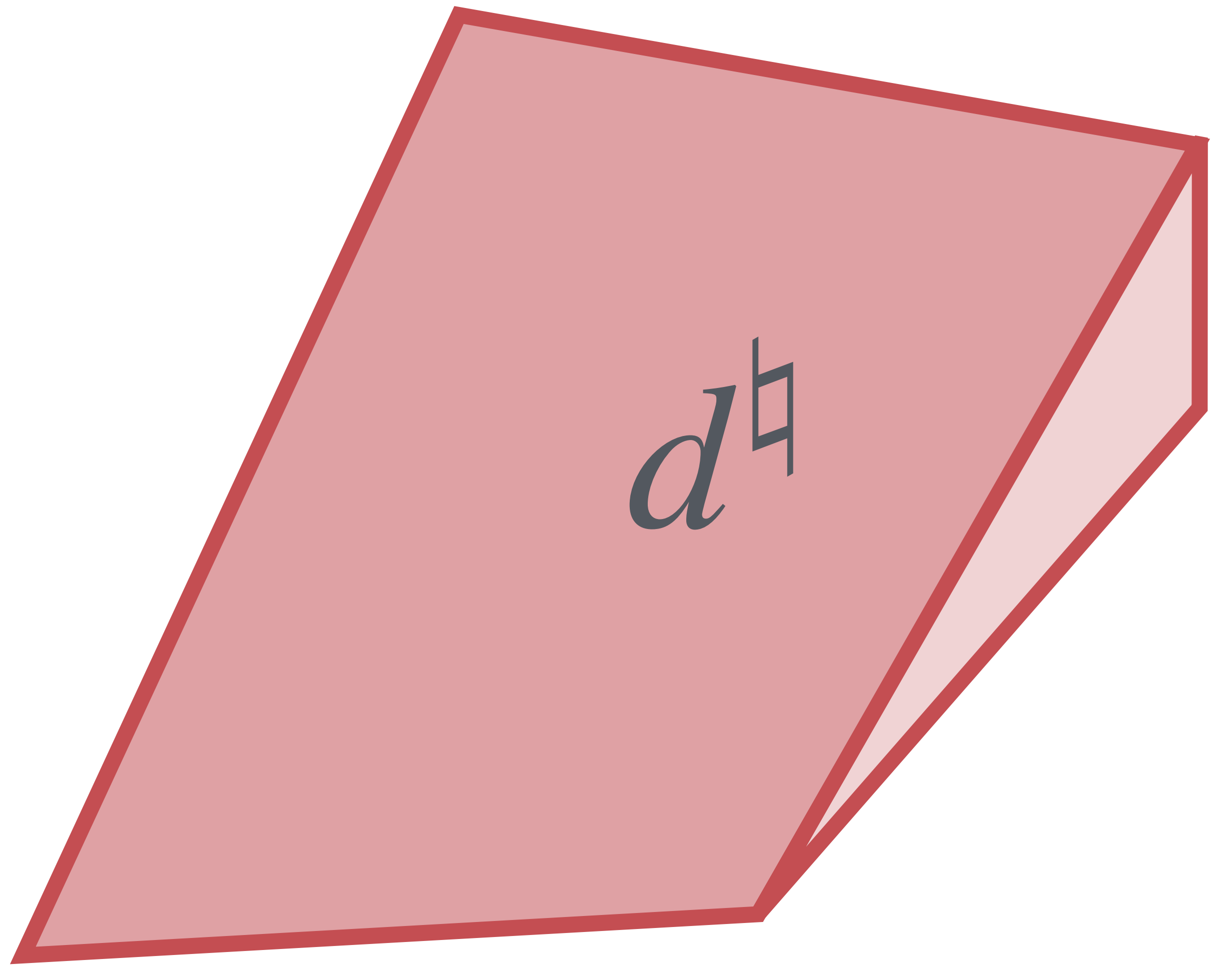
Backwards starting from zero!

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

Forward + Backward abstract analysis

(iii) Impact Quantification

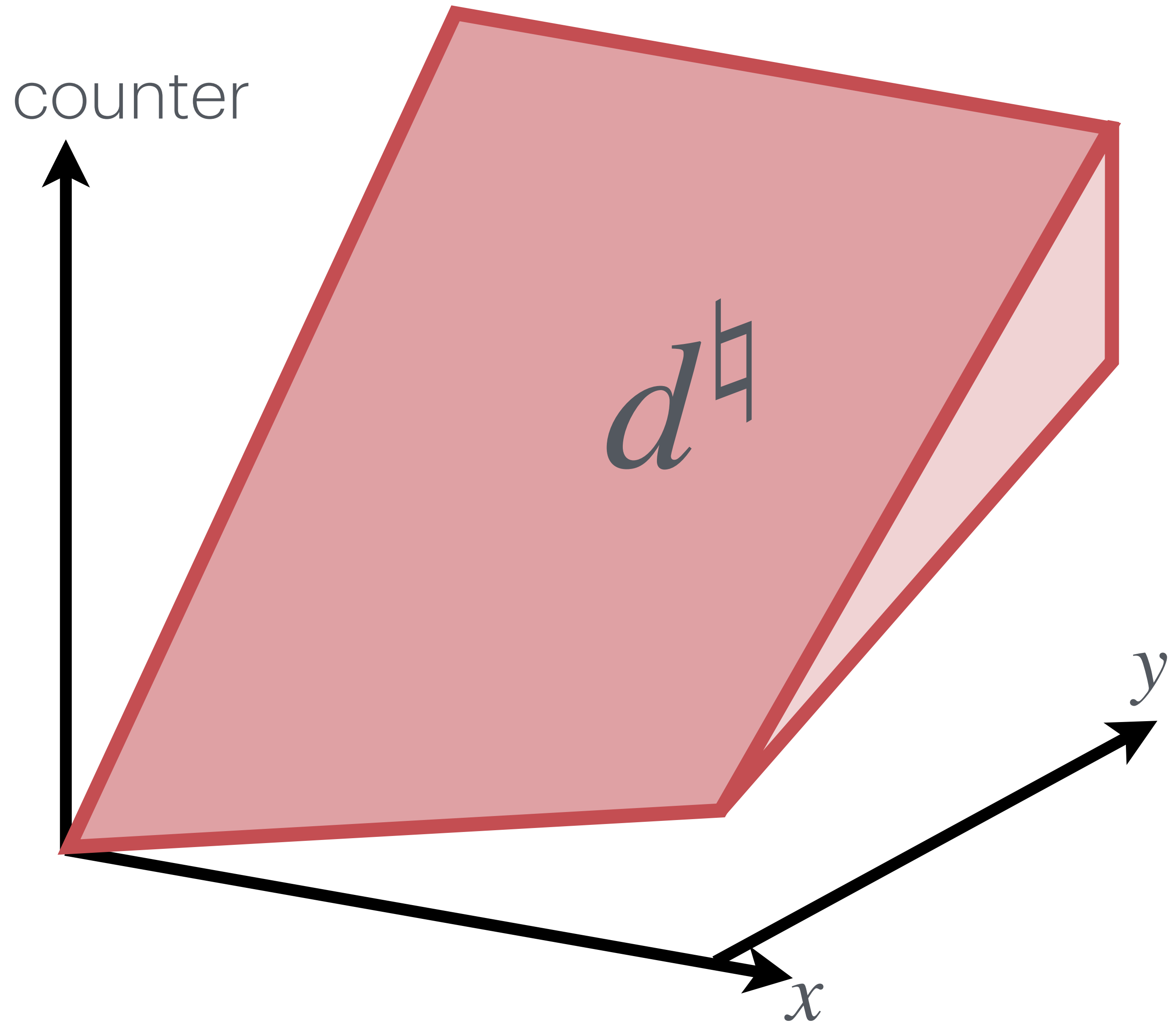
Abstract invariant on the
input variables + counter



(iii) Impact Quantification

Abstract invariant on the
input variables + counter

(i) Assume input variable of interest x



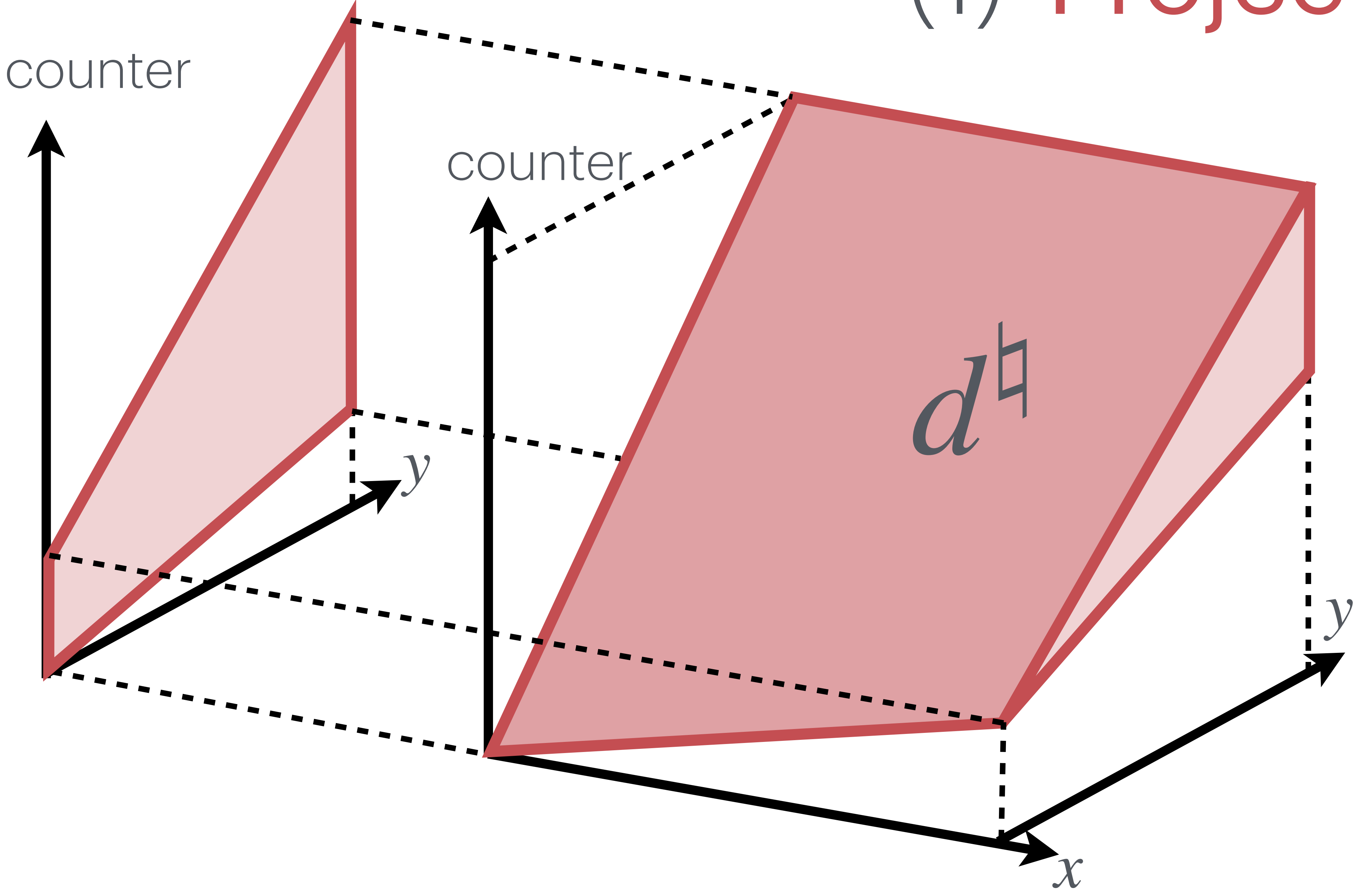
(iii) Impact Quantification

Abstract invariant on the

input variables + counter

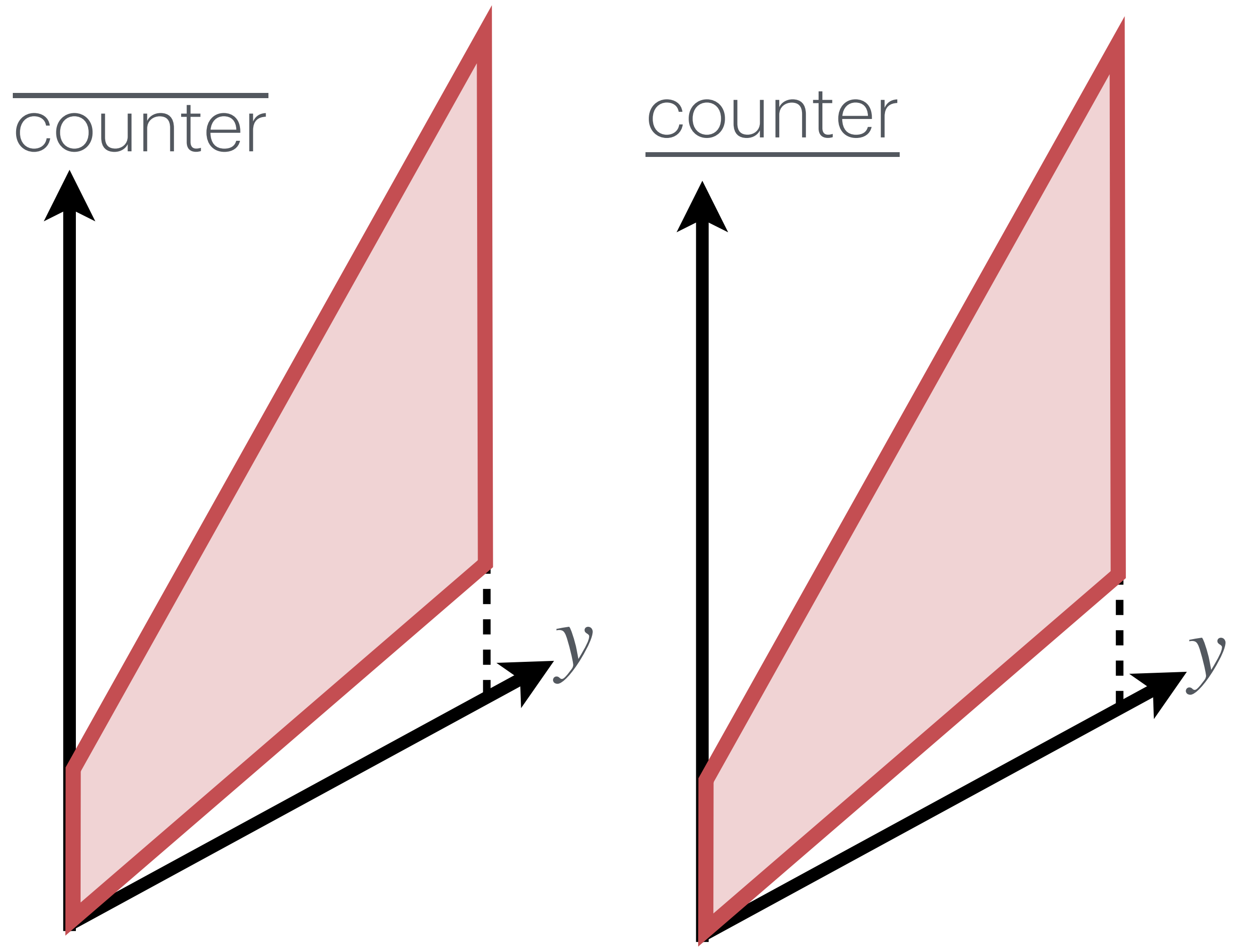
(i) Assume input variable of interest x

(1) Project away x



(iii) Impact Quantification

Abstract invariant on the
input variables + **counter**



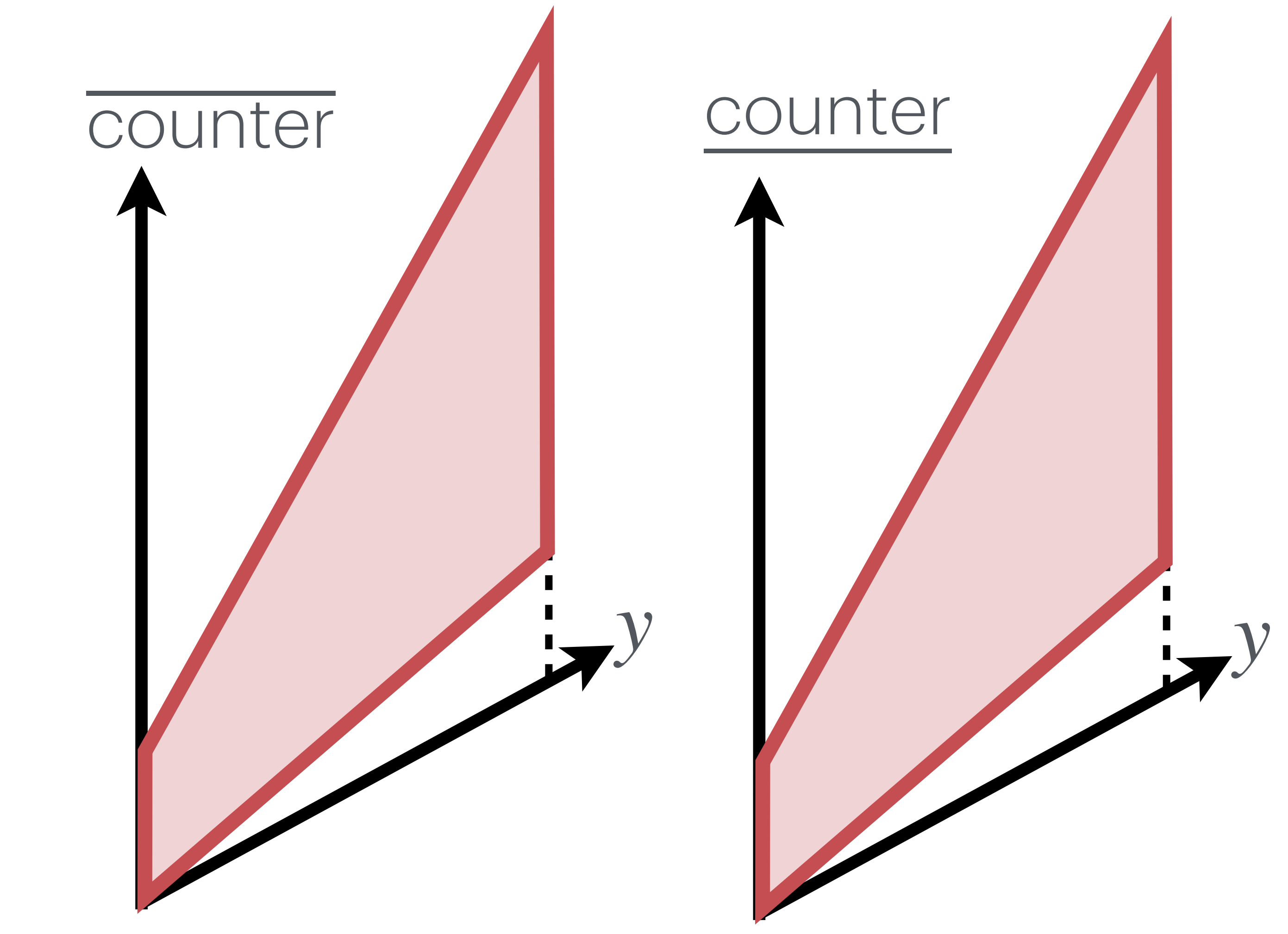
(i) Assume **input variable** of interest x

(1) **Project** away x

(2) **Duplicate** the invariant and **substitute** the counter with $\underline{\text{counter}}$ and $\overline{\text{counter}}$

(iii) Impact Quantification

Abstract invariant on the
input variables + counter



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

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

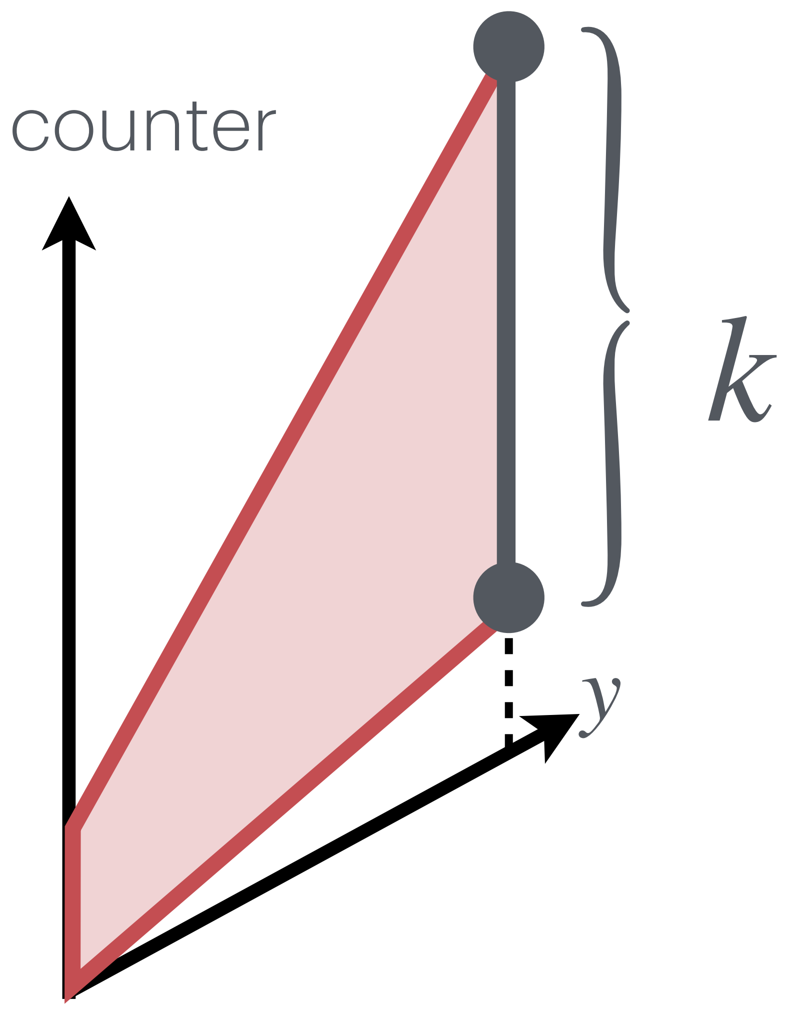
(1) **Project** away x

(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 $\underline{\text{counter}}$ and $\overline{\text{counter}}$

(3) **Maximize** the distance between the two

k is the impact of x

(iii) Impact Quantification

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

(iii) Impact Quantification

$$\mathsf{Impact}^{\natural}_{x}(d^{\natural}) = \max k \text{ subject to}$$

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

(iii) Impact Quantification

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

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

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

(iii) Impact Quantification

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

$$\text{Substitute}[\![\text{counter} \leftarrow \overline{\text{counter}}]\!](\text{Project}_x(d^\natural)) \land$$

$$\text{Substitute}[\![\text{counter} \leftarrow \underline{\text{counter}}]\!](\text{Project}_x(d^\natural)) \land$$

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

abstract

concrete

$$\mathrm{Impact}^{\natural}_{x}(\mathsf{P}) \geq \textsc{Impact}_{x}(\mathsf{P})$$

abstract

concrete

$$\mathrm{Impact}_x^\natural(\mathsf{P}) \leq k \quad \Longrightarrow$$

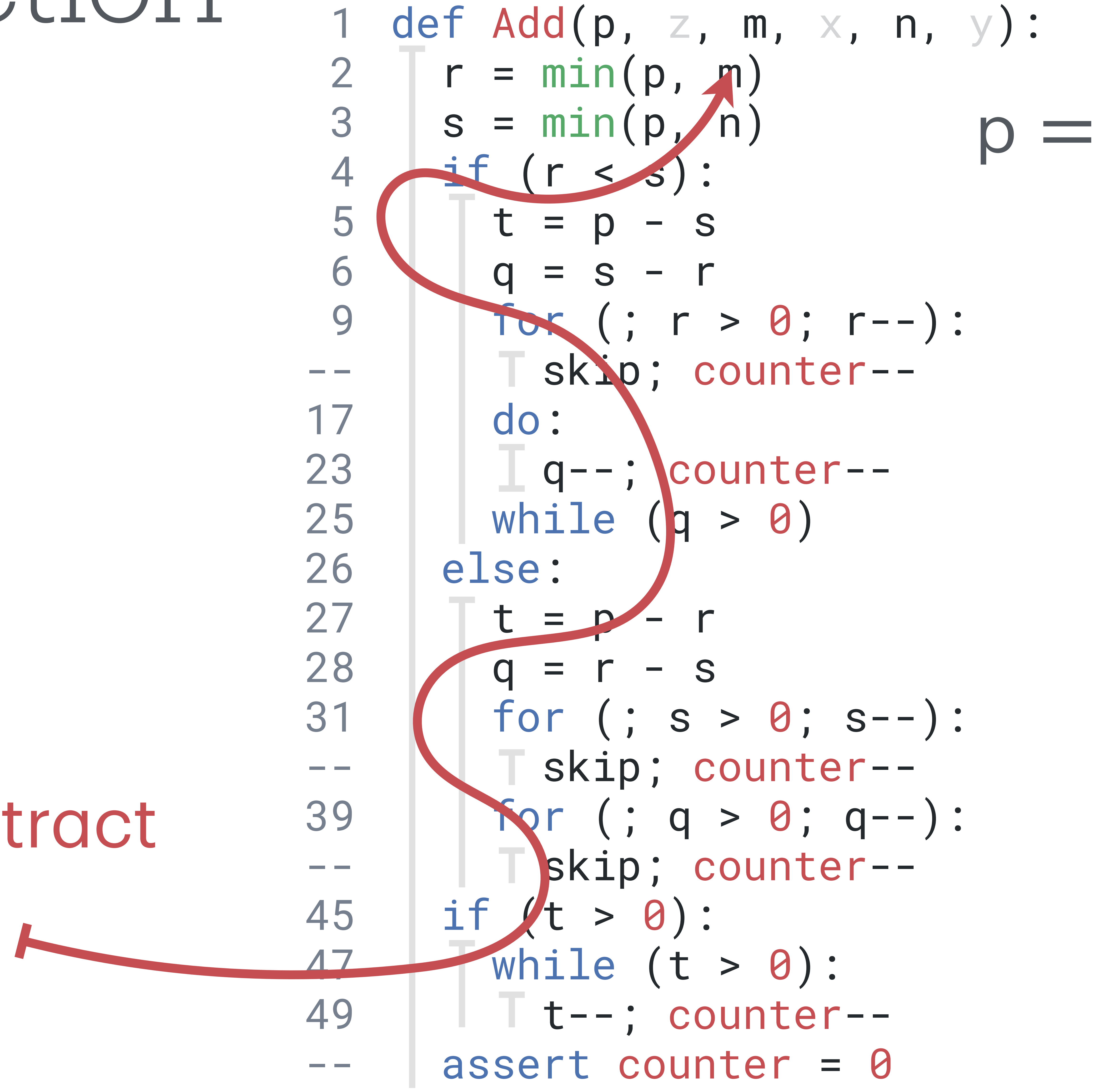
input variable x has an impact below k on the iterations of the program P

$$\mathrm{Impact}_x^\natural(\mathsf{P}) \geq \textsc{Impact}_x(\mathsf{P})$$

Add Function

$d^\natural$ is

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



```
def Add(p, z, m, x, n, y):
    r = min(p, m)
    s = min(p, n)
    if (r < s):
        t = p - s
        q = s - r
        for (; r > 0; r--):
            skip; counter--
        do:
            q--; counter--
        while (q > 0)
    else:
        t = p - r
        q = r - s
        for (; s > 0; s--):
            skip; counter--
        for (; q > 0; q--):
            skip; counter--
    if (t > 0):
        while (t > 0):
            t--; counter--
    assert counter = 0
```

Forward +
Backward abstract
analysis

Add Function

$d^\natural$ is

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

$$\mathsf{Impact}^\natural_x(d^\natural) = \max k \text{ subject to}$$

$$\mathsf{Substitute}[\![\mathsf{counter} \leftarrow \overline{\mathsf{counter}}]\!](\mathsf{Project}_x(d^\natural)) \wedge$$

$$\mathsf{Substitute}[\![\mathsf{counter} \leftarrow \underline{\mathsf{counter}}]\!](\mathsf{Project}_x(d^\natural)) \wedge$$

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

Add Function

$d^\natural$ is

$\mathsf{p} = \mathsf{counter} \land 0 \leq \mathsf{p} \leq u$

$$\mathrm{Impact}^\natural_{p}(\mathsf{p} = \mathsf{counter} \land 0 \leq \mathsf{p} \leq u) =$$

$$\max k \text{ subject to}$$

$$0 \leq \overline{\mathsf{counter}} \leq u \land$$

$$0 \leq \underline{\mathsf{counter}} \leq u \land$$

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

Add Function

$d^\natural$ is

$\mathsf{p} = \mathsf{counter} \land 0 \leq \mathsf{p} \leq u$

$$\left.\begin{aligned}&\mathrm{Impact}^\natural_p(\mathsf{p} = \mathsf{counter} \land 0 \leq \mathsf{p} \leq u) = \\ &\qquad \max k \text{ subject to} \\ &\qquad\qquad 0 \leq \overline{\mathsf{counter}} \leq u \land \\ &\qquad\qquad 0 \leq \underline{\mathsf{counter}} \leq u \land \\ &\qquad\qquad 0 \leq k \leq \overline{\mathsf{counter}} - \underline{\mathsf{counter}}\end{aligned}\right\} \quad u$$

Add Function

$d^\natural$ is

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

$$\mathsf{Impact}^\natural_n(\mathsf{p} = \mathsf{counter} \wedge 0 \leq \mathsf{p} \leq u) =$$

$$\max k \text{ subject to}$$

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

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

$$0 \leq \mathsf{p} \leq u \wedge$$

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

Add Function

$d^\natural$ is

$\mathsf{p} = \mathsf{counter} \land 0 \leq \mathsf{p} \leq u$

$$\left.\begin{array}{l} \mathsf{Impact}^\natural_n(\mathsf{p} = \mathsf{counter} \land 0 \leq \mathsf{p} \leq u) = \\ \quad \max k \text{ subject to} \\ \qquad \mathsf{p} = \overline{\mathsf{counter}} \land \\ \qquad \mathsf{p} = \underline{\mathsf{counter}} \land \\ \qquad 0 \leq \mathsf{p} \leq u \land \\ \qquad 0 \leq k \leq \overline{\mathsf{counter}} - \underline{\mathsf{counter}} \end{array}\right\} 0$$

Add Function

$d^\natural$ is

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

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

Add Function

$d^\natural$ is

$\mathsf{p} = \mathsf{counter} \land 0 \leq \mathsf{p} \leq u$

$$\left.\begin{aligned}
&\mathsf{Impact}^\natural_n(\mathsf{p} = \mathsf{counter} \land 0 \leq \mathsf{p} \leq u) = \\
&\quad \max k \text{ subject to} \\
&\qquad \mathsf{p} = \overline{\mathsf{counter}} \land \\
&\qquad \mathsf{p} = \underline{\mathsf{counter}} \land \\
&\qquad 0 \leq \mathsf{p} \leq u \land \\
&\qquad 0 \leq k \leq \overline{\mathsf{counter}} - \underline{\mathsf{counter}}
\end{aligned}\right\} 0$$

And to all the other input variables

Add Function

$$\text{Impact}^{\natural}_{p}(d^{\natural}) = u$$

$d^{\natural}$ is

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

```
def Add(p, z, m, x, n, y):
    r = min(p, m)
    s = min(p, n)
    if (r < s):
        t = p - s
        q = s - r
        for (; r > 0; r--):
            skip; counter--
        do:
            q--; counter--
        while (q > 0)
    else:
        t = p - r
        q = r - s
        for (; s > 0; s--):
            skip; counter--
        for (; q > 0; q--):
            skip; counter--
    if (t > 0):
        while (t > 0):
            t--; counter--
    assert counter = 0
```

Add Function

$$\text{Impact}_{p}^{\natural}(d^{\natural}) = u$$

$$\text{Impact}_{m}^{\natural}(d^{\natural}) = 0$$

$$\text{Impact}_{n}^{\natural}(d^{\natural}) = 0$$

$$\text{Impact}_{x}^{\natural}(d^{\natural}) = 0$$

$$\text{Impact}_{y}^{\natural}(d^{\natural}) = 0$$

$$\text{Impact}_{z}^{\natural}(d^{\natural}) = 0$$

```
def Add(p, z, m, x, n, y):
    r = min(p, m)
    s = min(p, n)
    if (r < s):
        t = p - s
        q = s - r
        for (; r > 0; r--):
            skip; counter--
        do:
            q--; counter--
        while (q > 0)
    else:
        t = p - r
        q = r - s
        for (; s > 0; s--):
            skip; counter--
        for (; q > 0; q--):
            skip; counter--
    if (t > 0):
        while (t > 0):
            t--; counter--
    assert counter = 0
```

$d^{\natural}$ is

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

Add Function

$$\mathsf{Impact}^{\natural}_{p}(d^{\natural}) = u$$

$$\mathsf{Impact}^{\natural}_{m}(d^{\natural}) = 0$$

$$\mathsf{Impact}^{\natural}_{n}(d^{\natural}) = 0$$

$$\mathsf{Impact}^{\natural}_{x}(d^{\natural}) = 0$$

$$\mathsf{Impact}^{\natural}_{y}(d^{\natural}) = 0$$

$$\mathsf{Impact}^{\natural}_{z}(d^{\natural}) = 0$$

```
def Add(p, z, m, x, n, y):
  r = min(p, m)
  s = min(p, n)
  if (r < s):
    t = p - s
    q = s - r
    for (; r > 0; r--):
      skip; counter--
    do:
      q--; counter--
    while (q > 0)
  else:
    t = p - r
    q = r - s
    for (; s > 0; s--):
      skip; counter--
    for (; q > 0; q--):
      skip; counter--
  if (t > 0):
    while (t > 0):
      t--; counter--
  assert counter = 0
```

$d^{\natural}$ is

$$\mathsf{p} = \mathsf{counter} \land 0 \leq \mathsf{p} \leq u$$

Add Function

$$\text{Impact}^{\natural}_{p}(d^{\natural}) = u$$

$$\text{Impact}^{\natural}_{m}(d^{\natural}) = 0$$

$$\text{Impact}^{\natural}_{n}(d^{\natural}) = 0$$

$$\text{Impact}^{\natural}_{x}(d^{\natural}) = 0$$

$$\text{Impact}^{\natural}_{y}(d^{\natural}) = 0$$

$$\text{Impact}^{\natural}_{z}(d^{\natural}) = 0$$

```
def Add(p, z, m, x, n, y):
  r = min(p, m)
  s = min(p, n)
  if (r < s):
    t = p - s
    q = s - r
    for (; r > 0; r--):
      skip; counter--
    do:
      q--; counter--
    while (q > 0)
  else:
    t = p - r
    q = r - s
    for (; s > 0; s--):
      skip; counter--
    for (; q > 0; q--):
      skip; counter--
  if (t > 0):
    while (t > 0):
      t--; counter--
  assert counter = 0
```

$d^{\natural}$ is

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

The Add function is safe

from timing side-channels

on input variables

m, n, x, y, z

Add Function

$$\mathrm{Impact}^{\natural}_{p}(d^{\natural}) = u$$

$$\mathrm{Impact}^{\natural}_{m}(d^{\natural}) = 0$$

$$\mathrm{Impact}^{\natural}_{n}(d^{\natural}) = 0$$

$$\mathrm{Impact}^{\natural}_{x}(d^{\natural}) = 0$$

$$\mathrm{Impact}^{\natural}_{y}(d^{\natural}) = 0$$

$$\mathrm{Impact}^{\natural}_{z}(d^{\natural}) = 0$$

```
def Add(p, z, m, x, n, y):
  r = min(p, m)
  s = min(p, n)
  if (r < s):
    t = p - s
    q = s - r
    for (; r > 0; r--):
      skip; counter--
    do:
      q--; counter--
    while (q > 0)
  else:
    t = p - r
    q = r - s
    for (; s > 0; s--):
      skip; counter--
    for (; q > 0; q--):
      skip; counter--
  if (t > 0):
    while (t > 0):
      t--; counter--
  assert counter = 0
```

$d^{\natural}$ is

$$\mathrm{p} = \mathrm{counter} \land 0 \leq \mathrm{p} \leq u$$

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/awslabs/s2n-bignum

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

github.com/denismazzucato/timesec

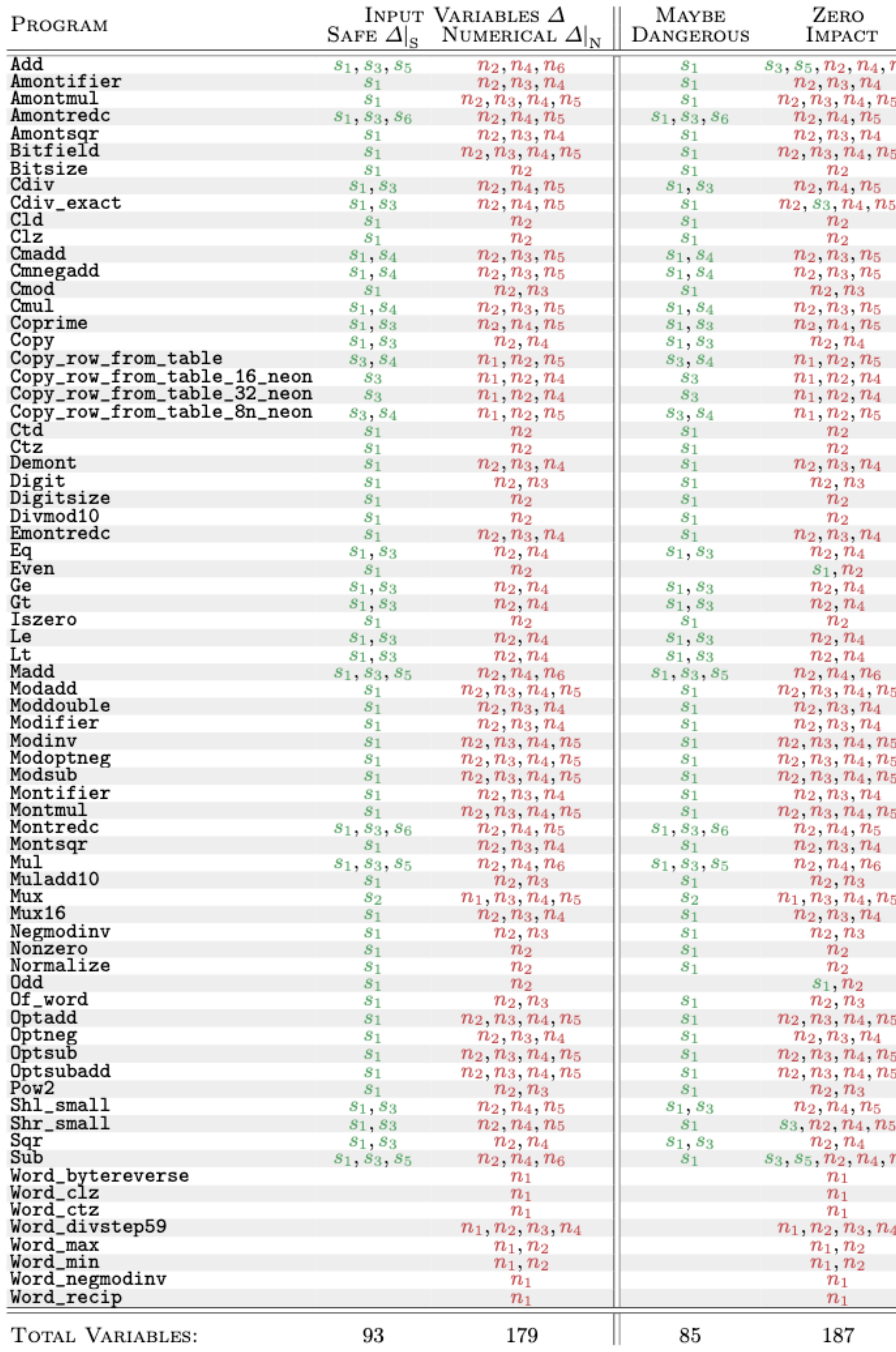
Python + Abstract Domain Library Apron

S2N-Bignum

https://github.com/awslabs/s2n-bignum

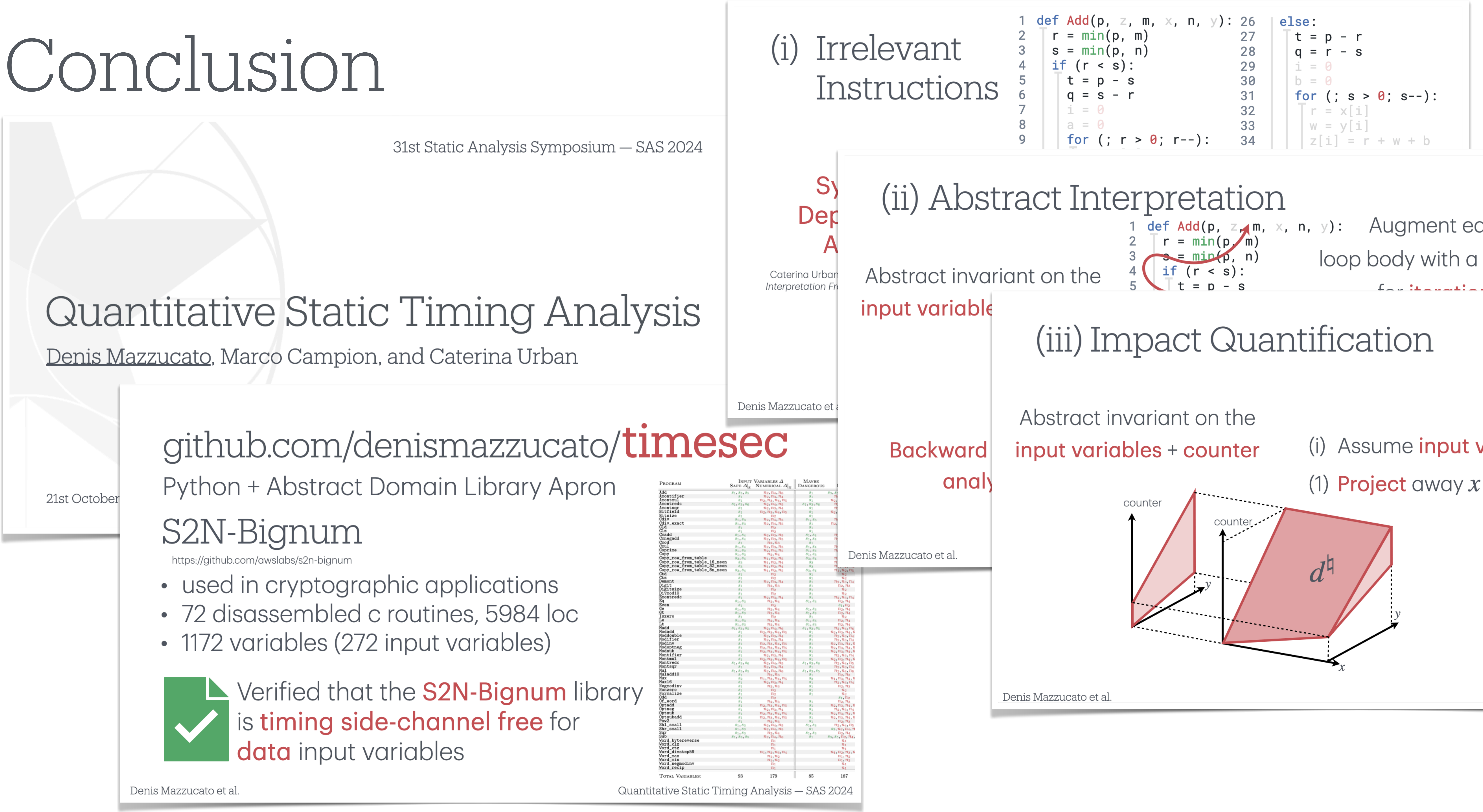
- used in cryptographic applications
- 72 disassembled c routines, 5984 loc
- 1172 variables (272 input variables)

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



Program	Input Variables Δ		Maybe	Zero
	Safe $\Delta\vert_S$	Numerical $\Delta\vert_N$	Dangerous	Impact
Add	s_1, s_3, s_5	n_2, n_4, n_6	s_1	$s_3, s_5, n_2, n_4,$
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
Cld	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
Le	s_1, s_3	n_2, n_4	s_1, s_3	n_2, n_4
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	$s_3, s_5, n_2, n_4,$
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

Conclusion



31st Static Analysis Symposium — SAS 2024

Quantitative Static Timing Analysis

Denis Mazzucato, Marco Campion, and Caterina Urban

21st October

(i) Irrelevant Instructions

```
def Add(p, z, m, x, n, y):
  r = min(p, m)
  s = min(p, n)
  if (r < s):
    t = p - s
    q = s - r
    i = 0
    a = 0
    for (; r > 0; r--):
```

```
  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
```

Sy
Dep
A

Caterina Urban
Interpretation Fr

Denis Mazzucato et a

(ii) Abstract Interpretation

Abstract invariant on the
input variable

```
def Add(p, z, m, x, n, y):
  r = min(p, m)
  s = min(p, n)
  if (r < s):
    t = p - s
```

Augment ec
loop body with a
for iteratio

Backward
analy

Denis Mazzucato et al.

(iii) Impact Quantification

Abstract invariant on the
input variables + counter

(i) Assume input v

(1) Project away x

Denis Mazzucato et al.

github.com/denismazzucato/timesec

Python + Abstract Domain Library Apron

S2N-Bignum

https://github.com/awslabs/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 Variables Δ		Maybe	
	Safe $\Delta\|_S$	Numerical $\Delta\|_N$	Dangerous	
Add	s_1, s_3, s_5	n_2, n_4, n_6	s_1	s_3, s_5
Amontifier	s_1	n_2, n_3, n_4	s_1	n
Amontmul	s_1	n_2, n_3, n_4, n_5	s_1	$n_2,$
Amontredc	s_1, s_3, s_6	n_2, n_4, n_5	s_1, s_3, s_6	n
Amontsqr	s_1	n_2, n_3, n_4	s_1	n
Bitfield	s_1	n_2, n_3, n_4, n_5	s_1	$n_2,$
Bitsize	s_1	n_2	s_1	
Cdiv	s_1, s_3	n_2, n_4, n_5	s_1, s_3	n
Cdiv_exact	s_1, s_3	n_2, n_4, n_5	s_1	$n_2,$
Cld	s_1	n_2	s_1	
Clz	s_1	n_2	s_1	
Cmadd	s_1, s_4	n_2, n_3, n_5	s_1, s_4	n
Cmnegadd	s_1, s_4	n_2, n_3, n_5	s_1, s_4	n
Cmod	s_1	n_2, n_3	s_1	
Cmul	s_1, s_4	n_2, n_3, n_5	s_1, s_4	n
Coprime	s_1, s_3	n_2, n_4, n_5	s_1, s_3	n
Copy	s_1, s_3	n_2, n_4	s_1, s_3	
Copy_row_from_table	s_3, s_4	n_1, n_2, n_5	s_3, s_4	n
Copy_row_from_table_16_neon	s_3	n_1, n_2, n_4	s_3	n
Copy_row_from_table_32_neon	s_3	n_1, n_2, n_4	s_3	n
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
Le	s_1, s_3	n_2, n_4	s_1, s_3	n_2, n_4
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
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
Modoptneg	s_1	n_2, n_3, n_4, n_5	s_1	n_2, n_3, n_4, n
Modsub	s_1	n_2, n_3, n_4, n_5	s_1	n_2, n_3, n_4, n
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
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
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
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
Optsubadd	s_1	n_2, n_3, n_4, n_5	s_1	n_2, n_3, n_4, n
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
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	$s_3, s_5, n_2, n_4,$
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
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

Denis Mazzucato et al.

Quantitative Static Timing Analysis — SAS 2024