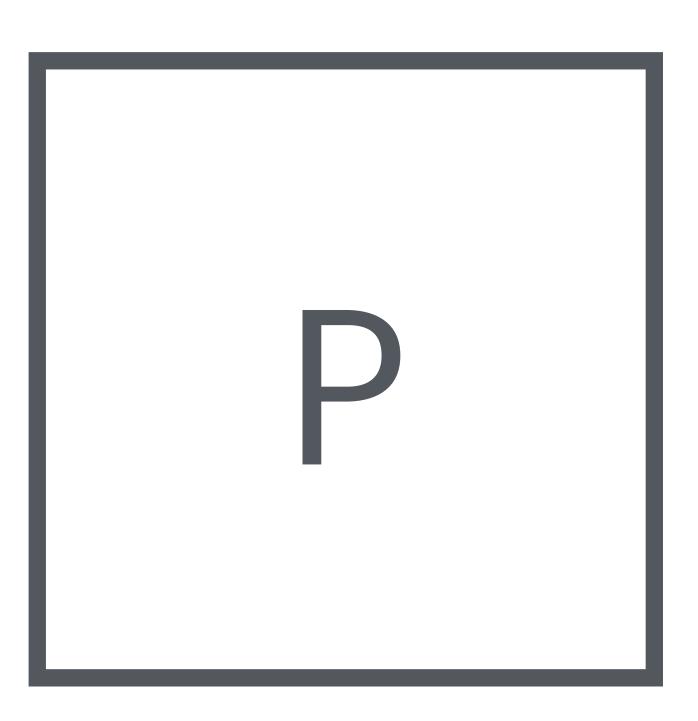
# Quantitative Static Timing Analysis

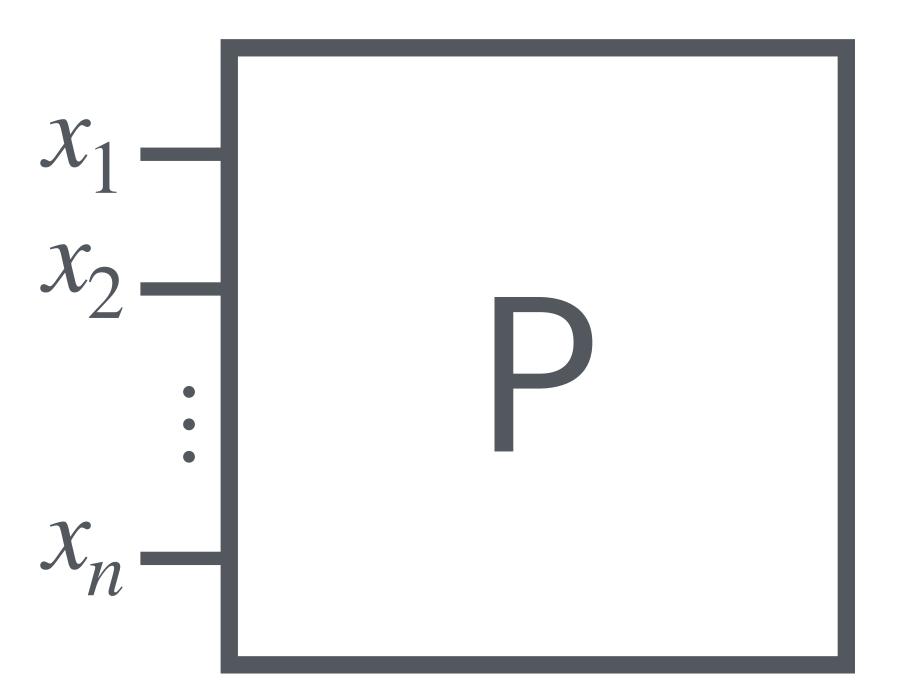
Denis Mazzucato, Marco Campion, and Caterina Urban



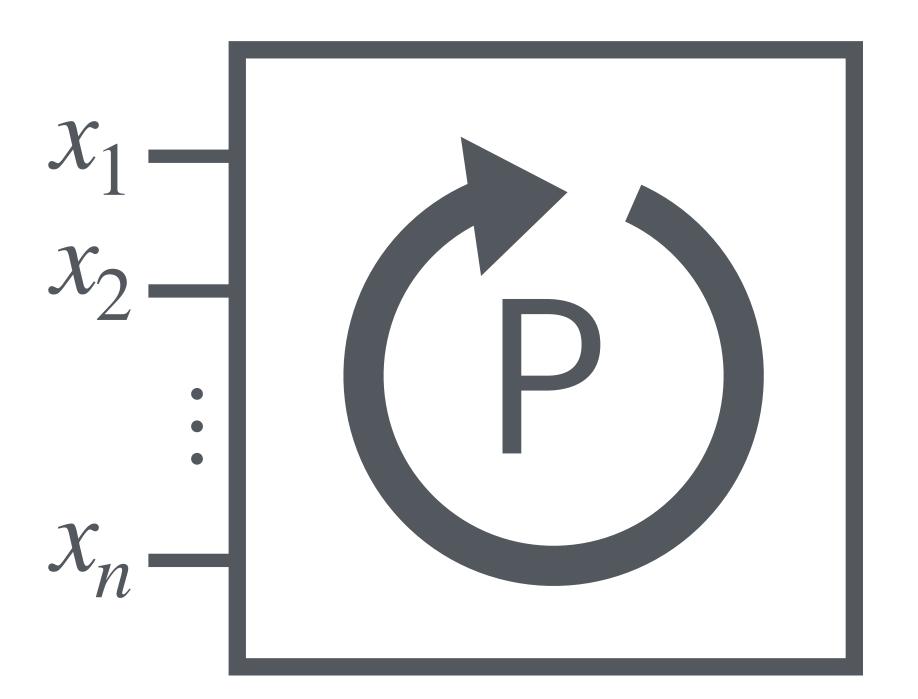




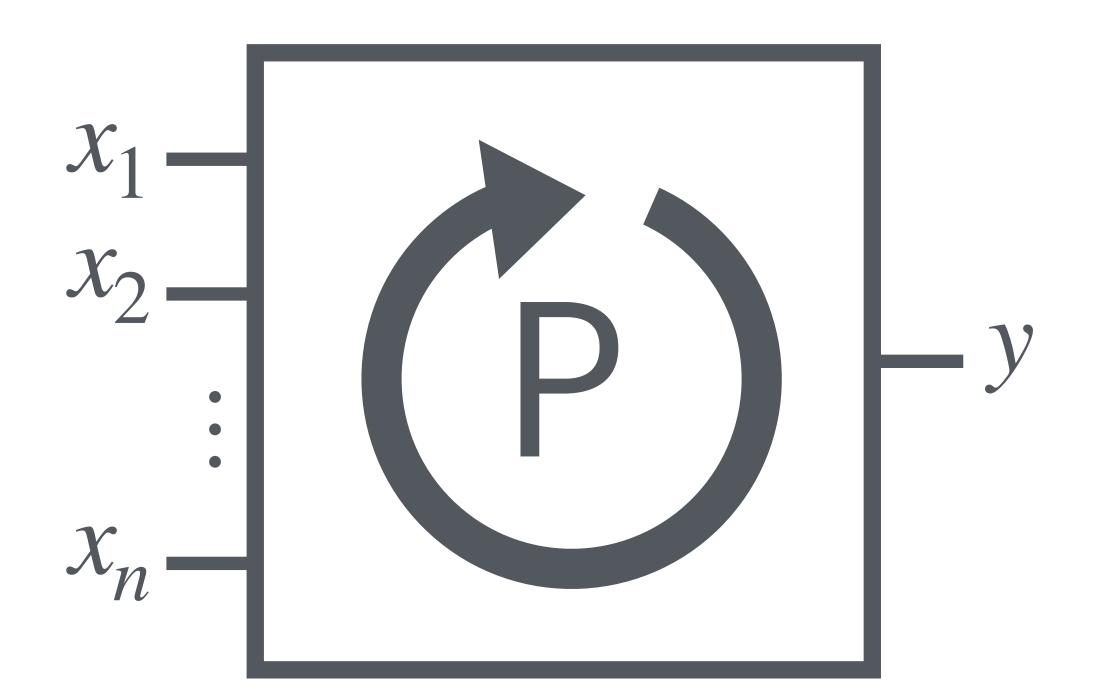
Input variables



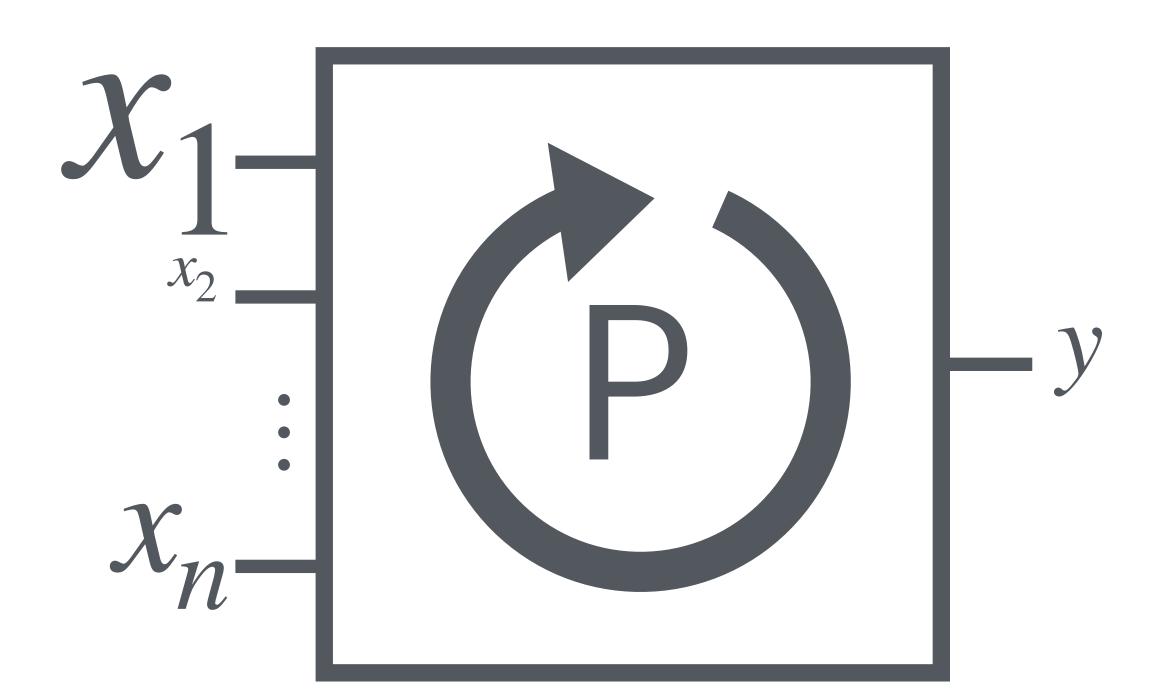
Input variables

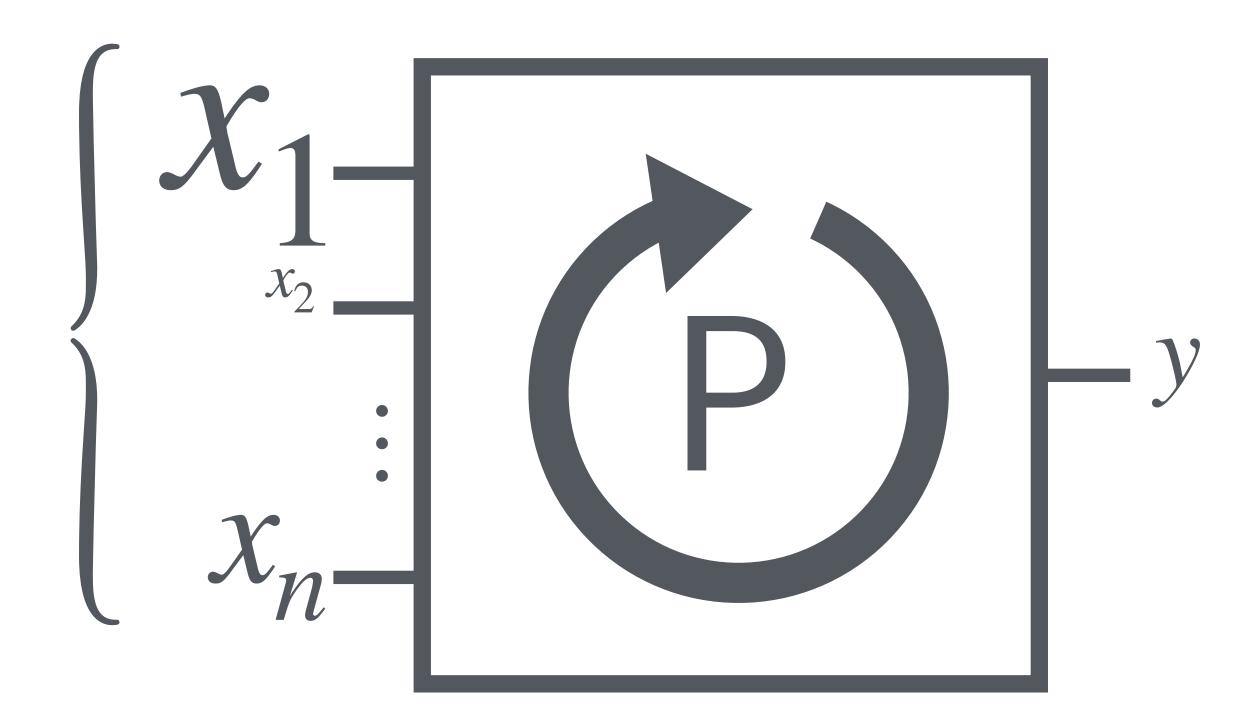


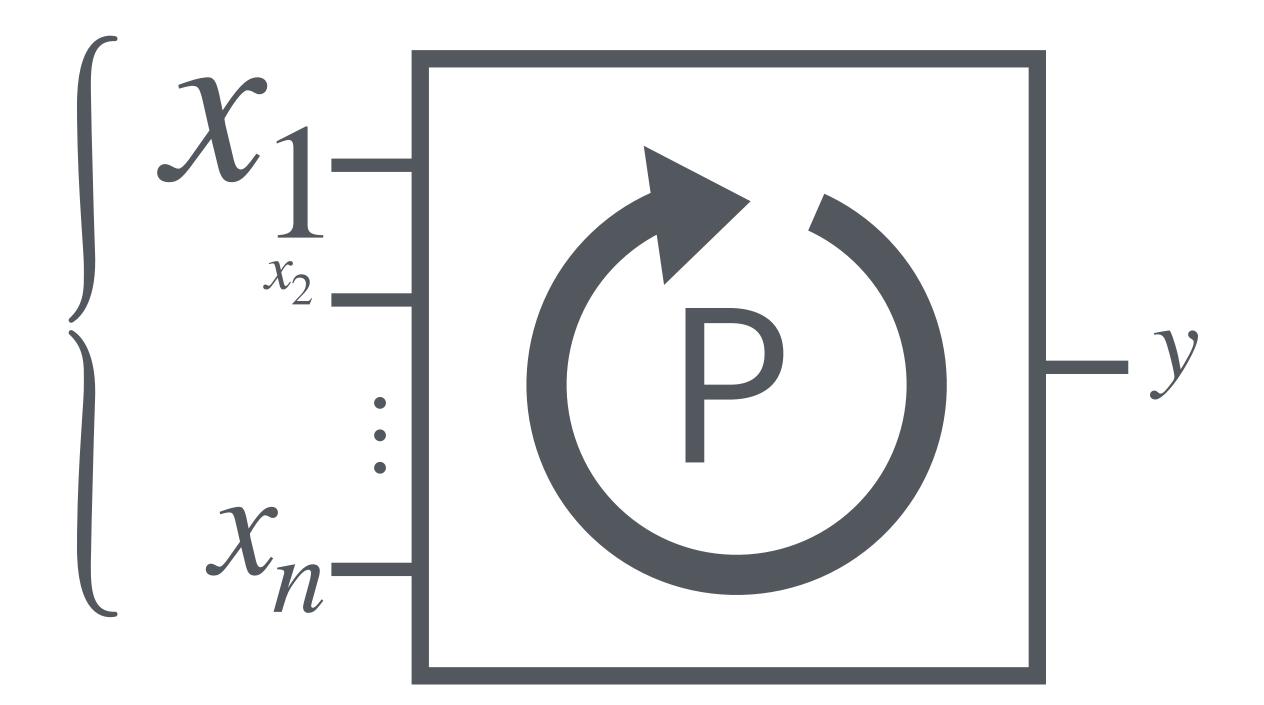
Input variables

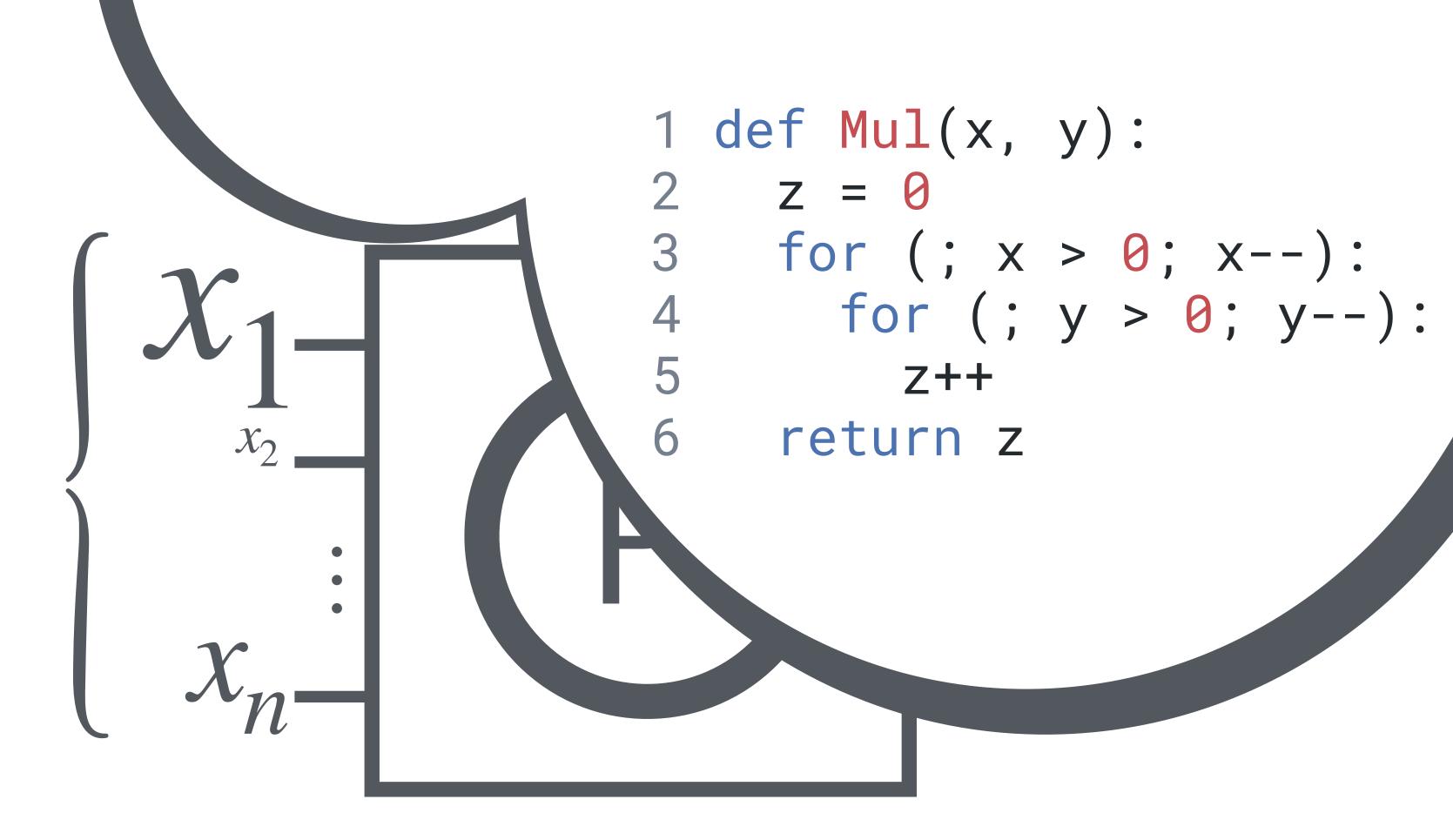


How much impact on loop iterations?

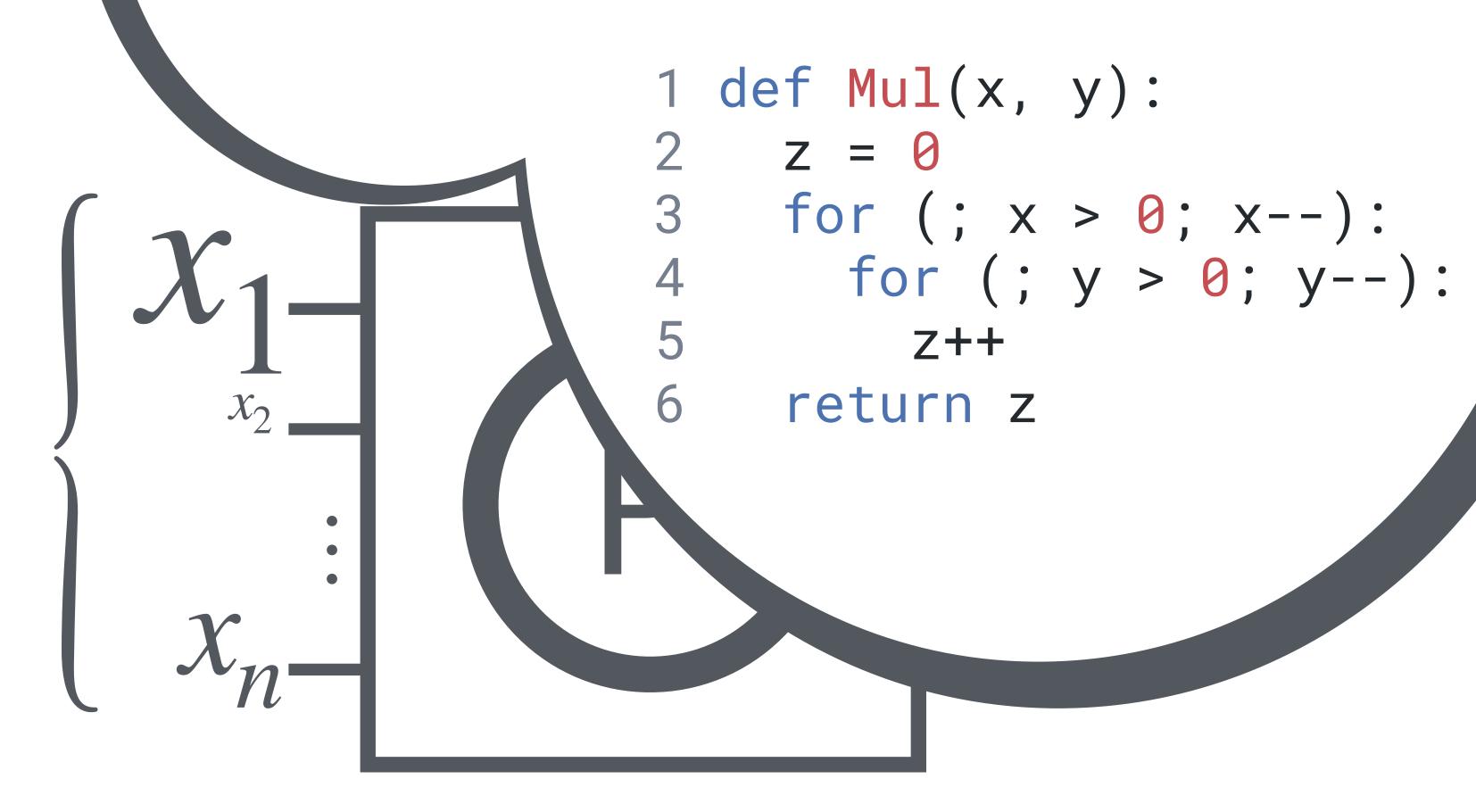




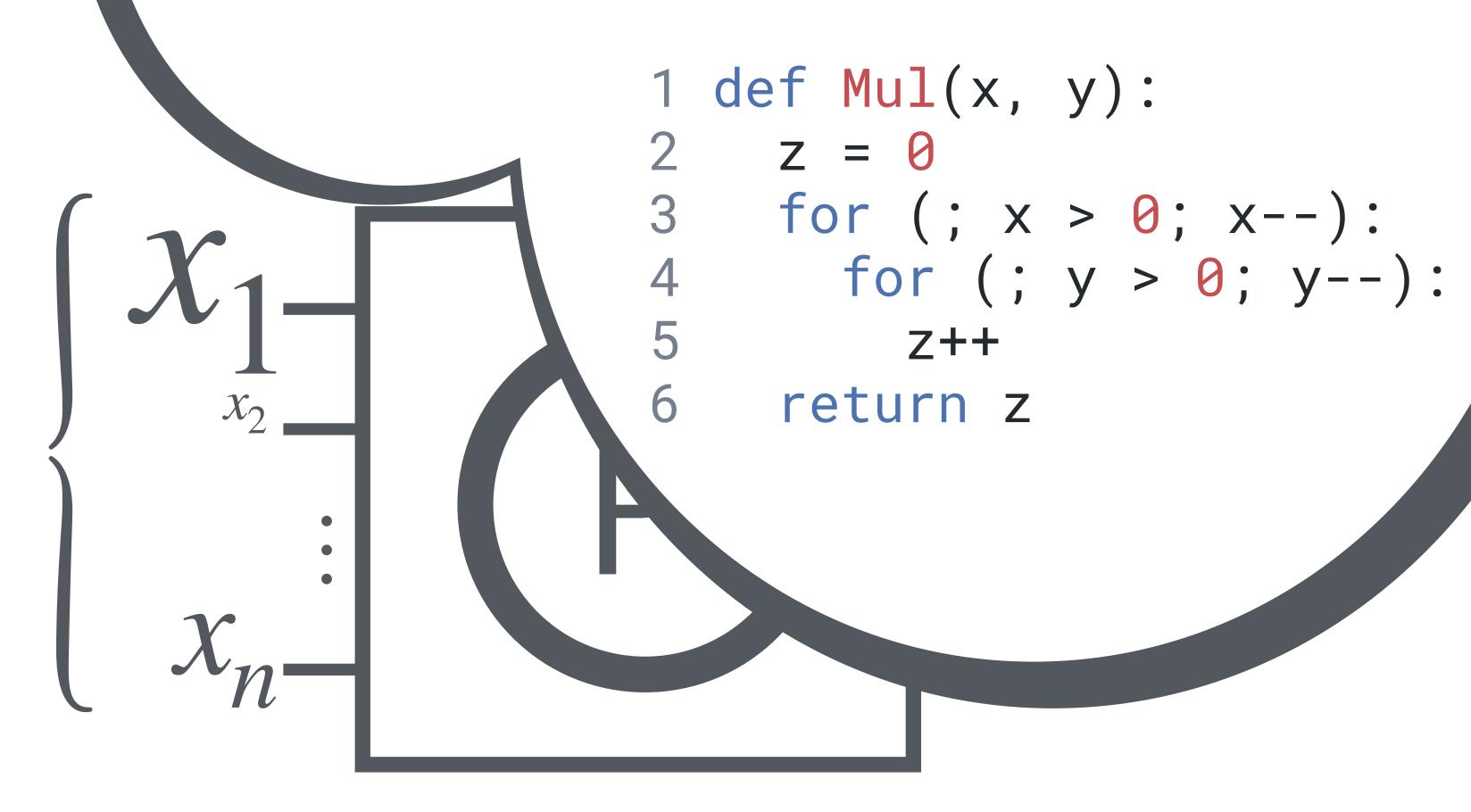


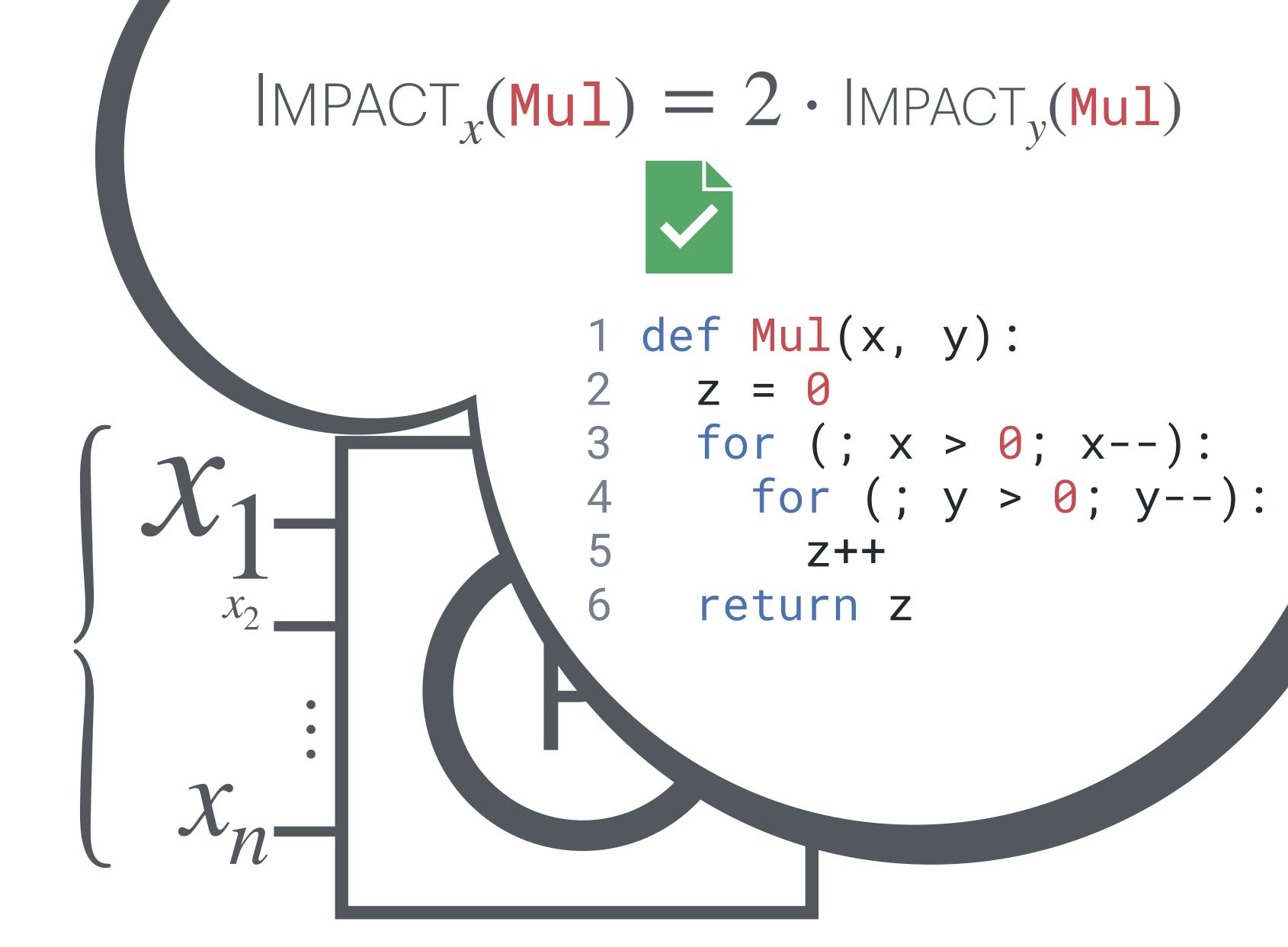


 $IMPACT_{x}(Mul)$ 

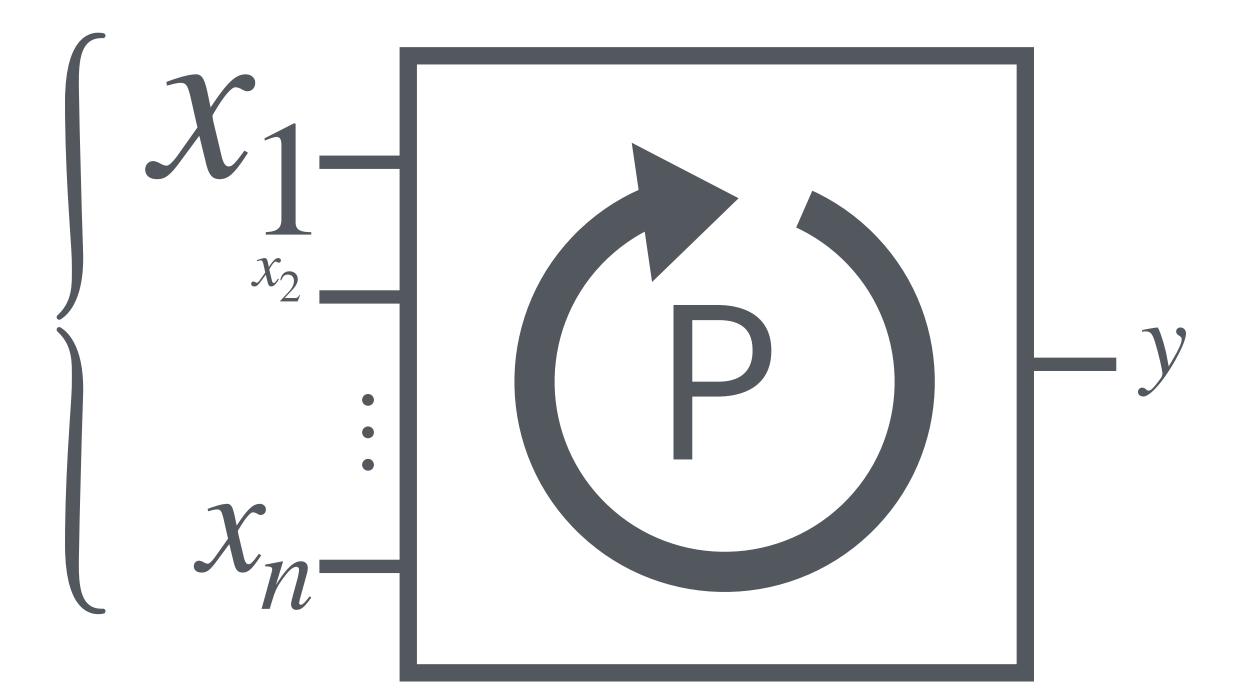


$$IMPACT_x(Mu1) = 2 \cdot IMPACT_y(Mu1)$$



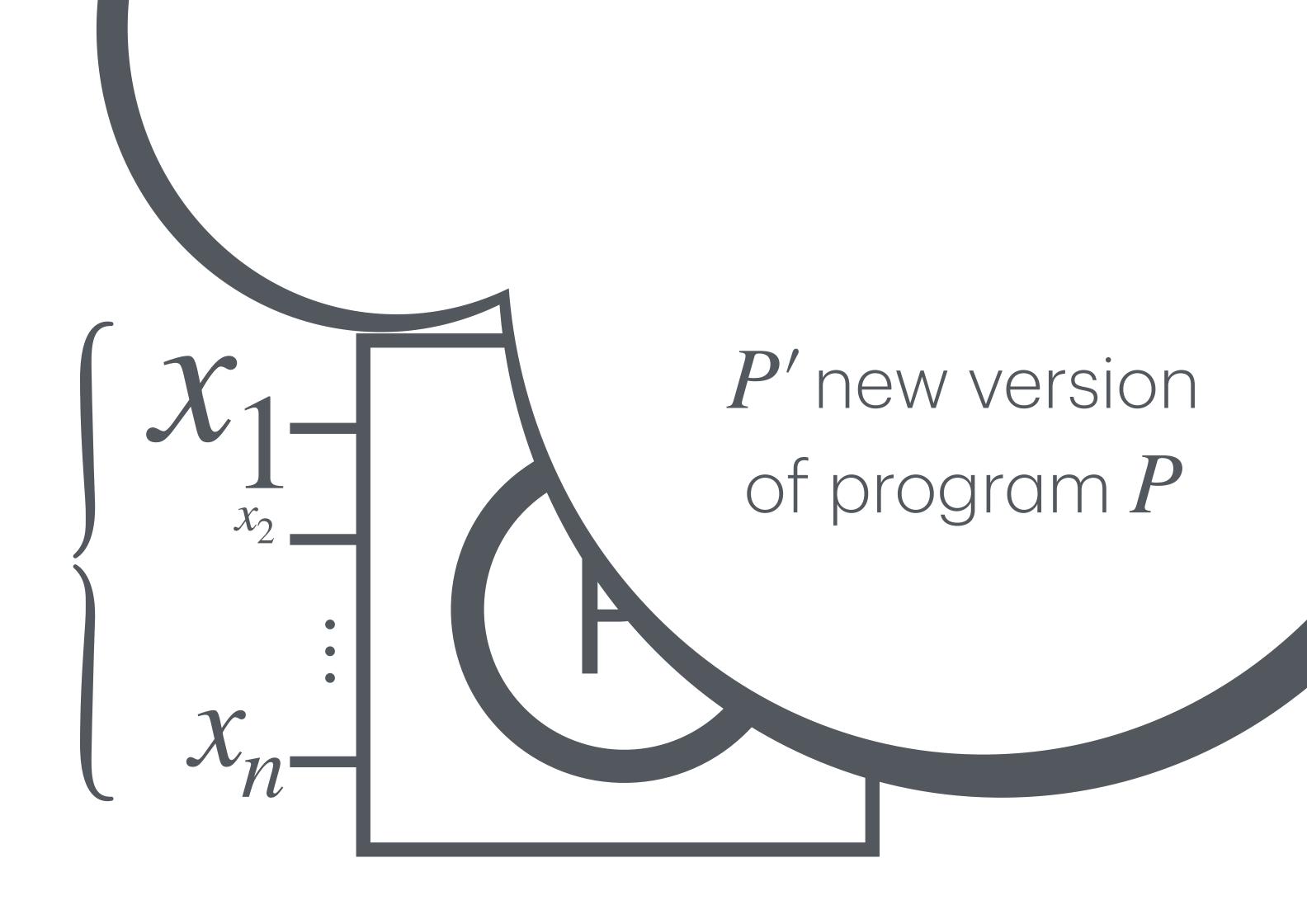


- Correct Loop
   Behaviour
- Performance



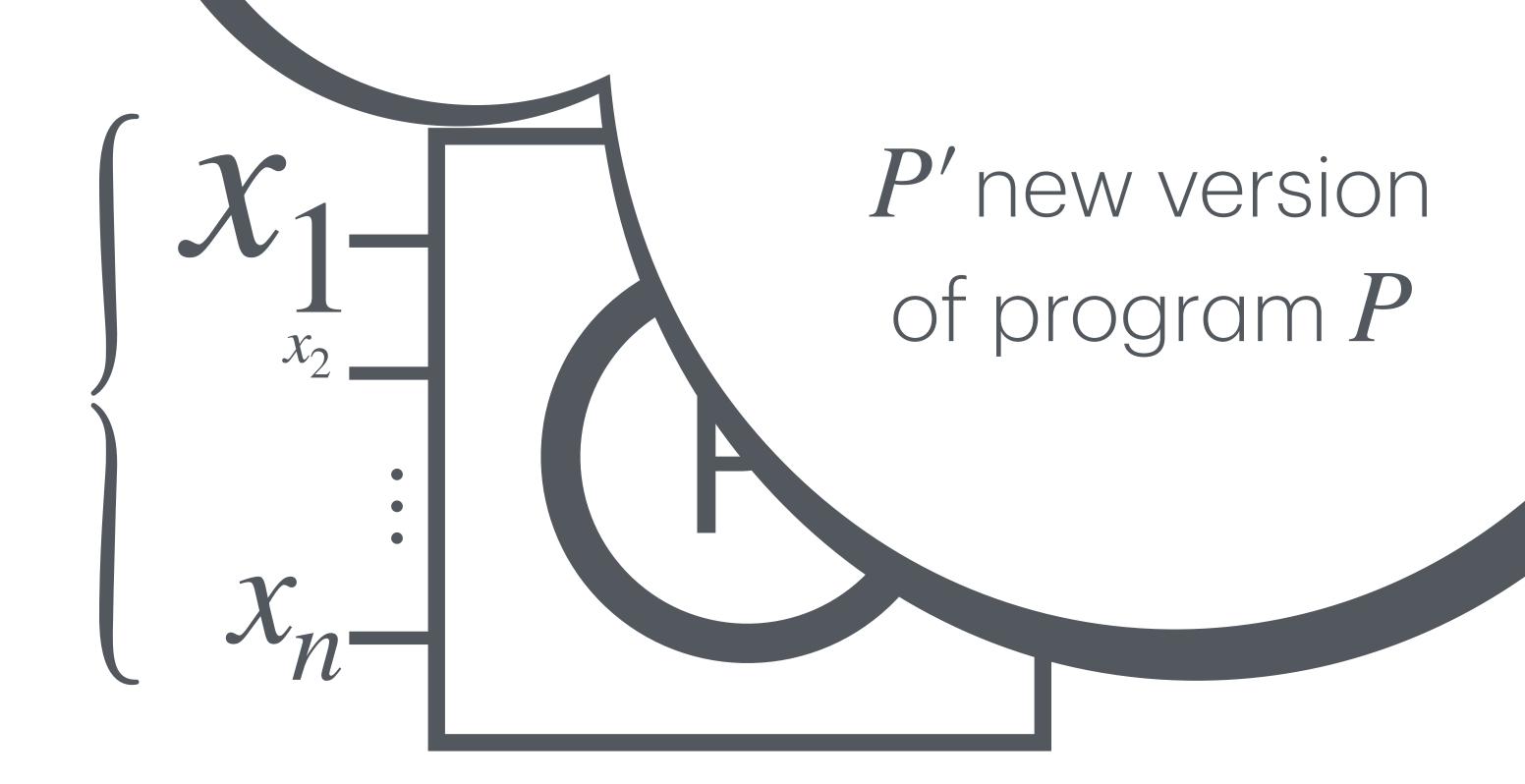
Correct Loop
 Behaviour

Performance

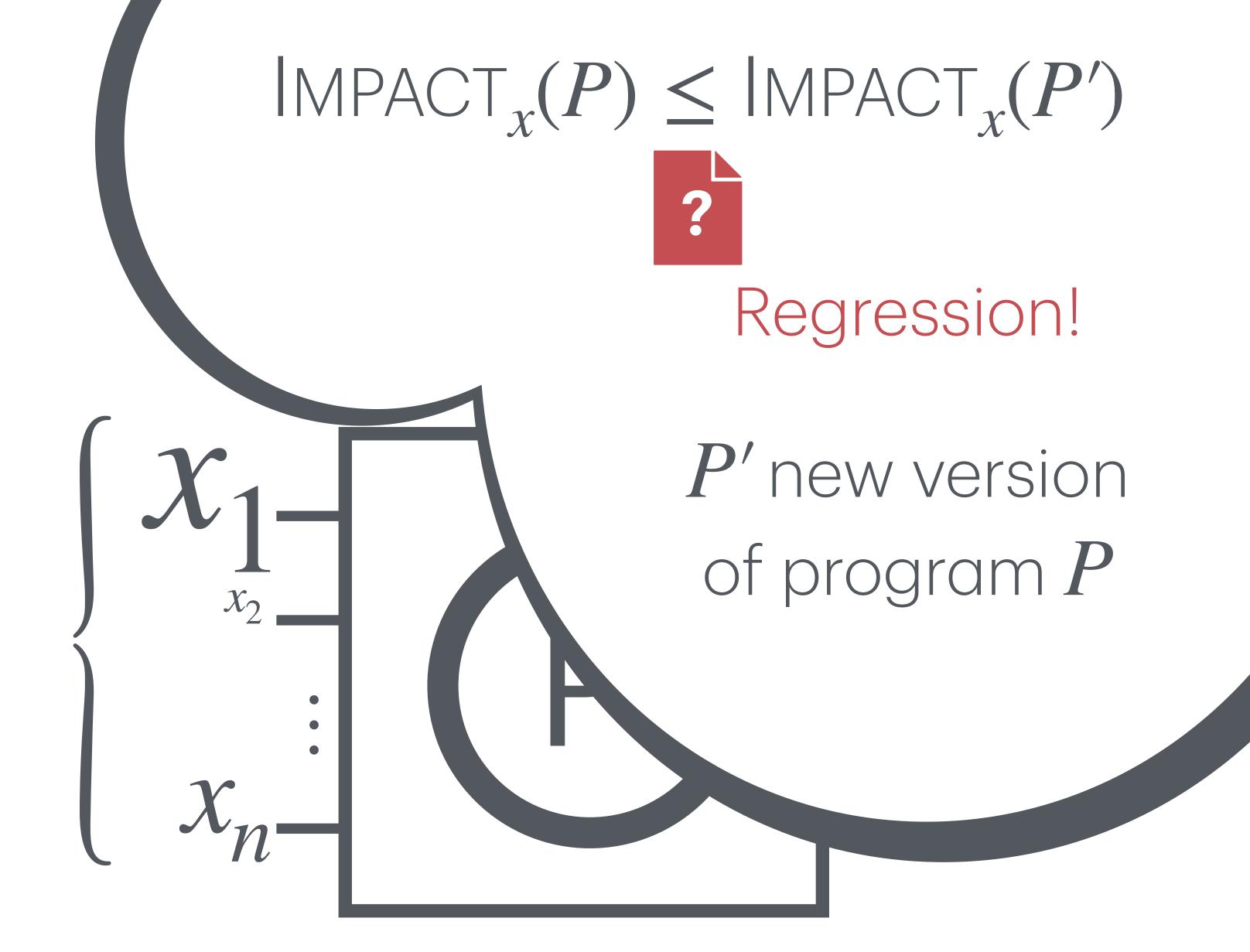


 $\mathsf{IMPACT}_{\mathcal{X}}(P) \leq \mathsf{IMPACT}_{\mathcal{X}}(P')$ 

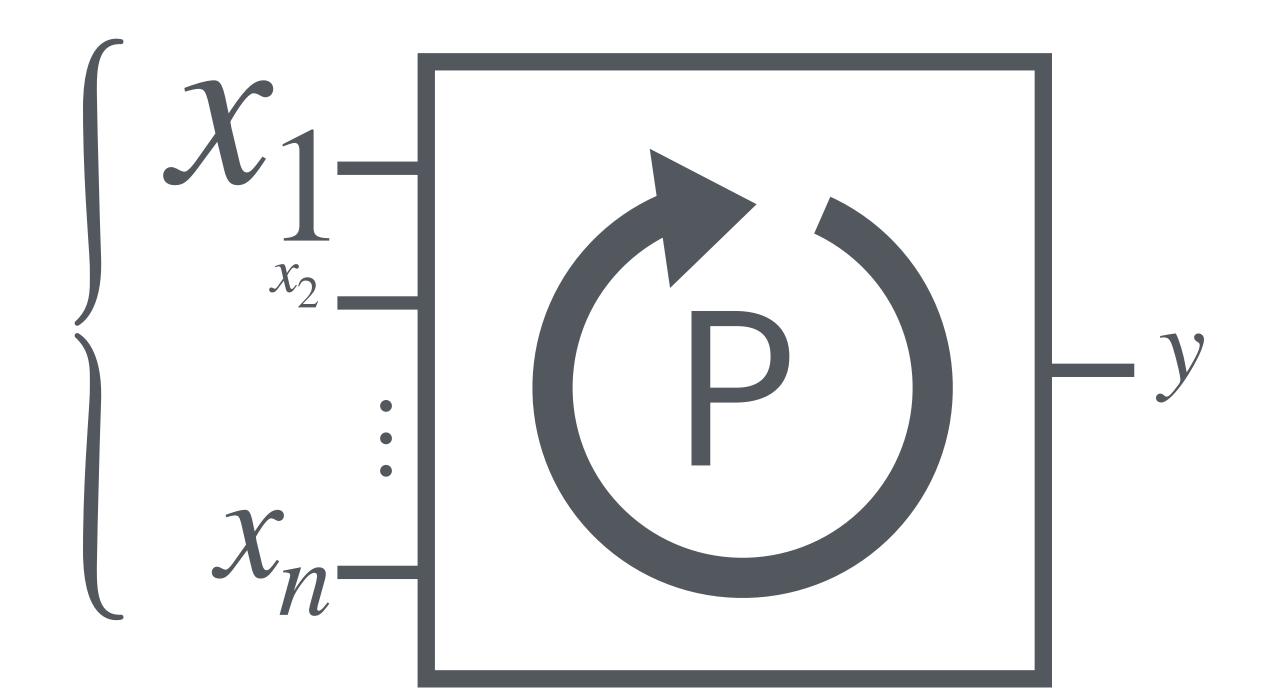
- Correct Loop
   Behaviour
- Performance



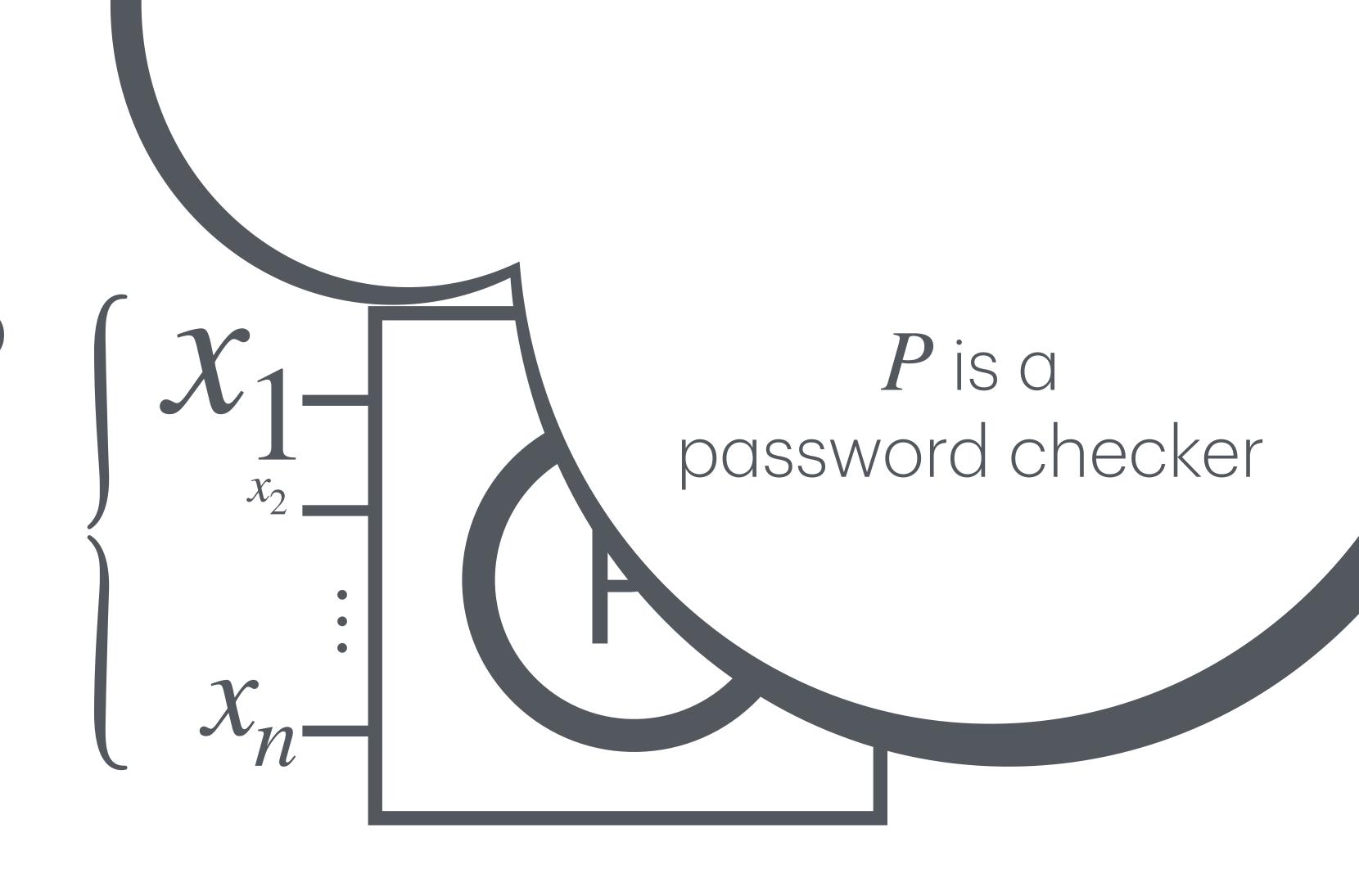
- Correct Loop
   Behaviour
- Performance



- Correct Loop
   Behaviour
- Performance
- Security

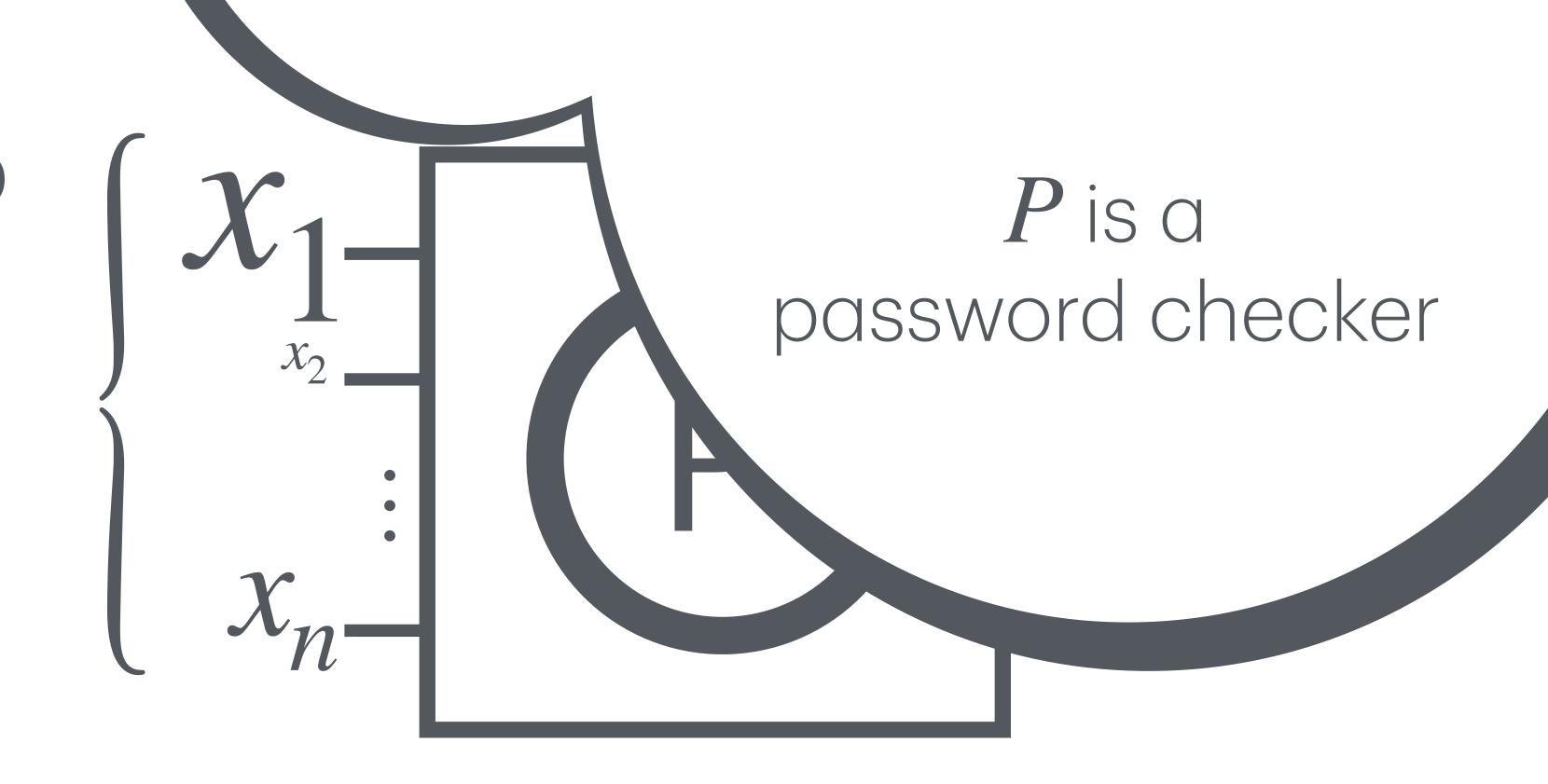


- Correct Loop
   Behaviour
- Performance
- Security

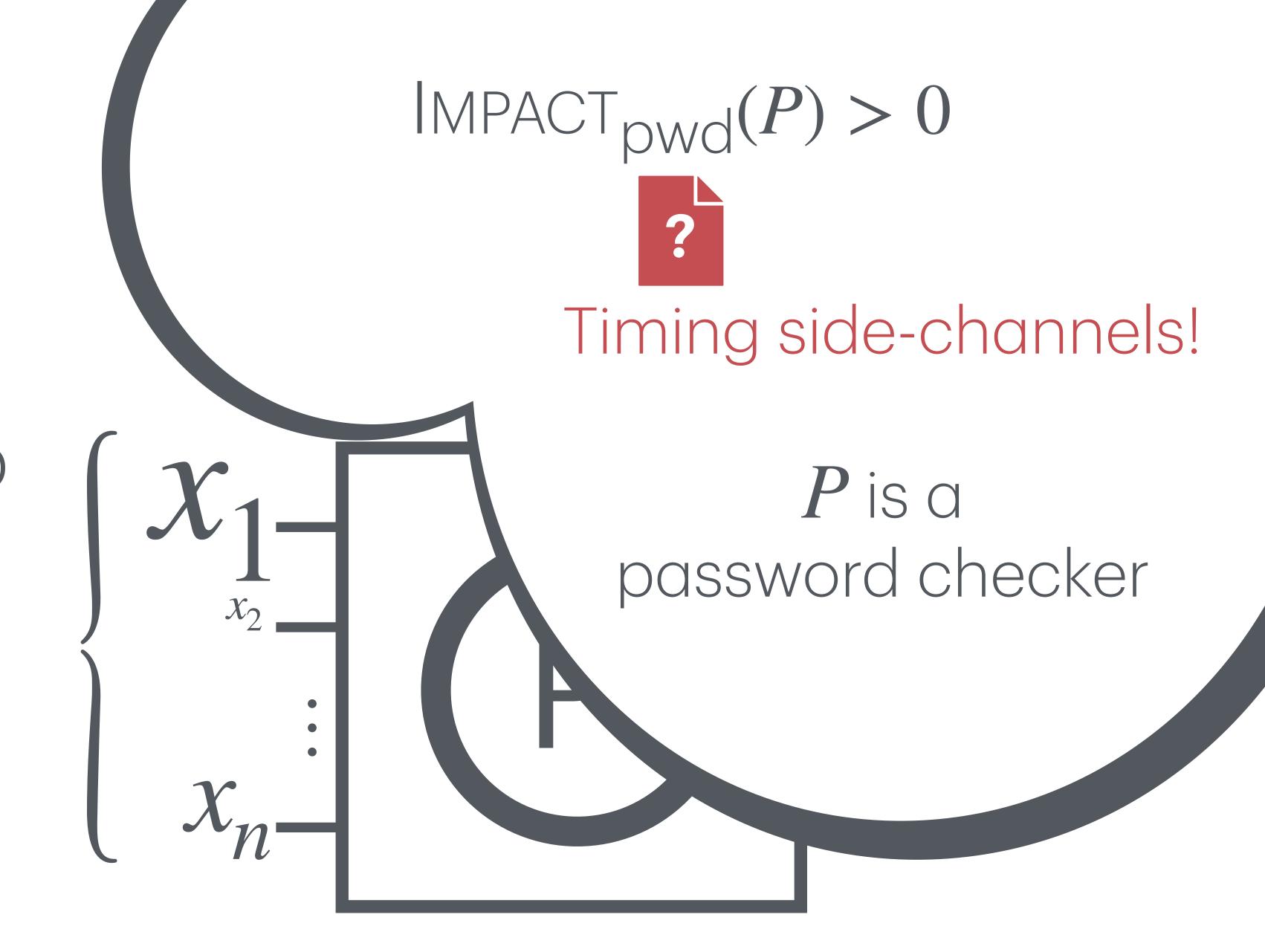


 $\mathsf{IMPACT}_{\mathsf{pWd}}(P) > 0$ 

- Performance
- Security



- Correct Loop
   Behaviour
- Performance
- Security



S2N-Bignum

3 8 4

$$42 = 38 + 4$$

array

$$[0, 4, 2] =$$
 $[3, 8] +$ 
 $[4]$ 

length array

$$3 [0, 4, 2] =$$
 $2 [3, 8] +$ 
 $1 [4]$ 

length array

```
3[0, 4, 2] =
```

18

19

20

21

23

24

25

```
1 \det Add(p, z, m, x, n, y): 26
                                  else:
   r = min(p, m)
                             27
                                    t = p - r
   s = min(p, n)
                             28
                                    q = r - s
   if (r < s):
                             29
                                    i = 0
     t = p - s
                             30
      q = s - r
                             31
                                    for (; s > 0; s--):
                             32
                                      r = x[i]
                                      w = y[i]
                             33
      for (; r > 0; r--):
                             34
                                      z[i] = r + w + b
      s = x[i]
                             35
                                      i = i + 1
       w = y[i]
                             36
                                      b = (w < b) | |
        z[i] = s + w + a
                                       (r + w < r) | |
                             37
        i = i + 1
                             38
                                        (r + w + b < r)
        a = (w < a) | |
                             39
                                    for (; q > 0; q--):
          (s + w < s) | |
                                     r = x[i]
          (s + w + a < s)
                                      z[i] = r + b
      do:
                             42
                                     | i = i + 1
                                      b = (r < b) | |
       r = y[i]
                             43
        b = (r < a) | |
                                      (r + b < r)
         (r + a < r)
                             45
                                  if (t > 0):
        z[i] = r + a
                             46
                                    z[i] = b
        i = i + 1
                                    while (t > 0):
                             48
                             49
     while (q > 0)
                                      if (t > 0):
                             50
                                      |z[i]| = 0
```

```
1 \det Add(p, z, m, x, n, y): 26
                                else:
   r = min(p, m)
                            27
                                  t = p - r
    s = min(p, n)
                            28
    if (r < s):
                            29
     t = p - s
                            30
                                   for (; s > 0; s--):
                                    r = x[i]
                            33
                                     w = y[i]
      for (; r > 0; r--):
                                     z[i] = r + w + b
       s = x[i]
                            35
                                   | i = i + 1
        w = y[i]
                                     b = (w < b) | |
       z[i] = s + w + a
                                     (r + w < r) | |
                            37
    | i = i + 1
                            38
                                      (r + w + b < r)
        a = (w < a) | |
                                   for (; q > 0; q--):
         (s + w < s) | |
                                    r = x[i]
                                    z[i] = r + b
          (s + w + a < s)
      do:
                                   | i = i + 1
                            42
                                   b = (r < b) | |
       r = y[i]
        b = (r < a) | |
                                    (r + b < r)
20
          (r + a < r)
                                 if (t > 0):
                            45
        z[i] = r + a
                                   while (t > 0):
                            48
24
                            49
      while (q > 0)
25
                                     if (t > 0):
                            50
                                     |z[i]| = 0
```

# $\begin{array}{c} \text{Add Function} \\ \text{Add} \\ \begin{pmatrix} 1, z, \\ 1, [4], \\ 1 \end{bmatrix} \end{array}$

$$\begin{array}{cccc} \text{Add} \left( \begin{array}{cccc} p \,, & Z \,, \\ m \,, & X \,, \\ n \,, & Y \,, \end{array} \right)$$

# 

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

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

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

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

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

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

2 iterations 
$$\longrightarrow$$
  $z = [4, 2]$ 

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

-- 3 iterations  $\longrightarrow$  z = [0, 0, 6]

# How we compute IMPACT<sub>p</sub>(Add)?

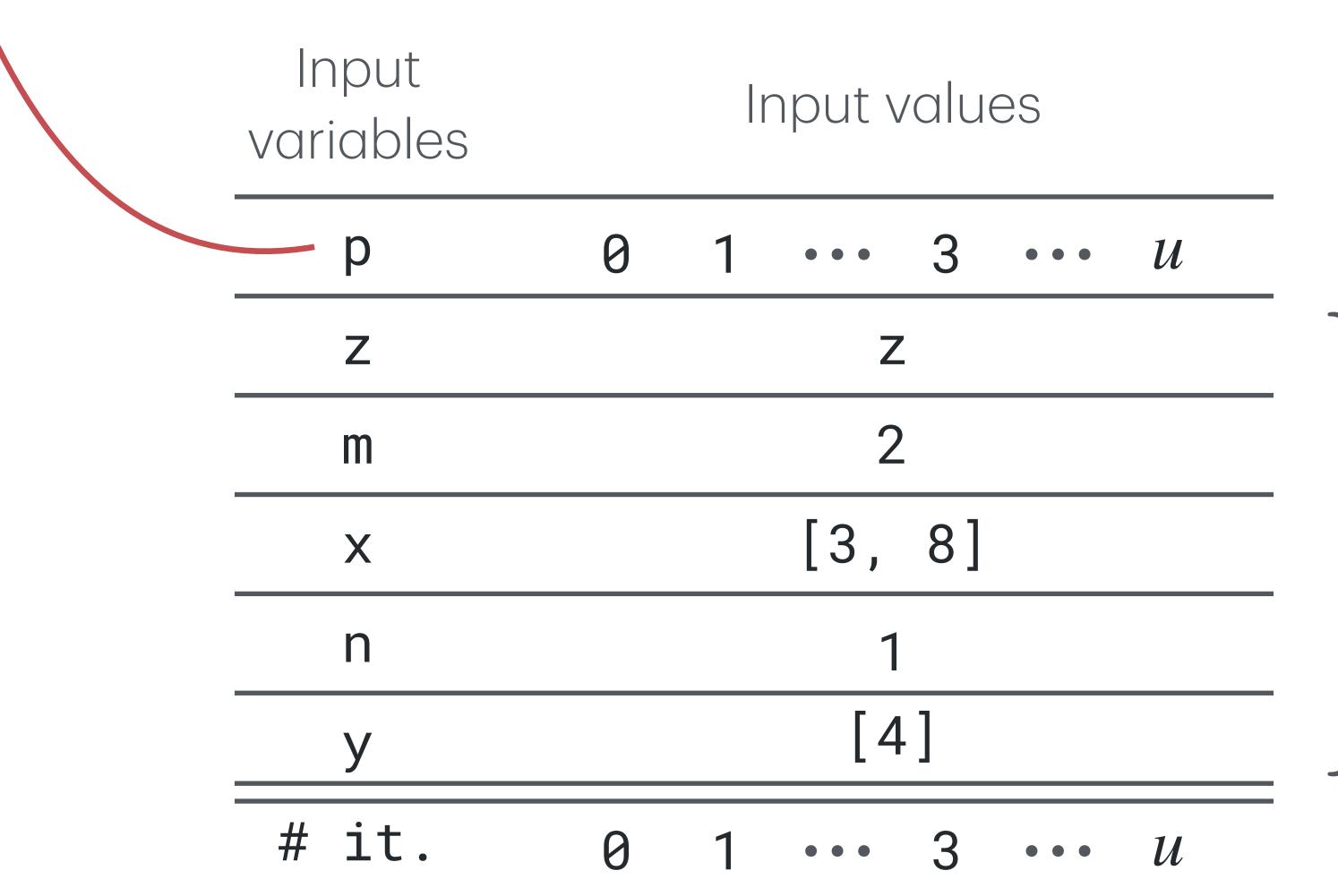
# How we compute IMPACT<sub>p</sub>(Add)? p input variable of interest

p input variable of interest Input Input values variables Z m [3, 8] X n

p input variable of interest Input Input values variables Z m [3, 8] X n [4]

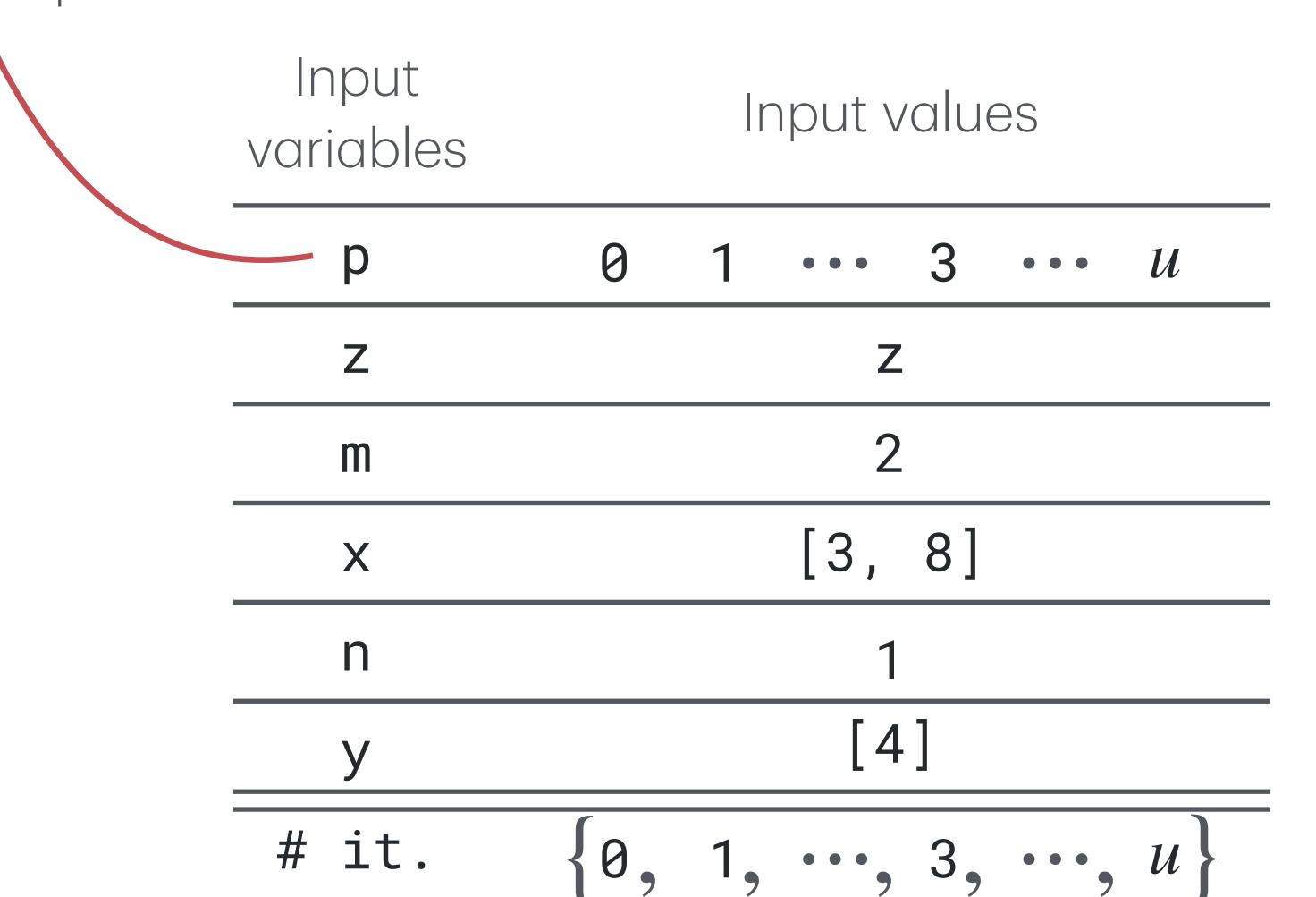
p input variable of interest Input Input values variables Z m [3, 8] X n [4] # it.

p input variable of interest



For all input values!

p input variable of interest



For all input values!

p input variable of interest

Input variables	Input values					
<u> </u>	0	1	• • •	3	• • •	U
Z	Z					
m	2					
X	[3, 8]					
n	1					
y	[4]					

For all input values!

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

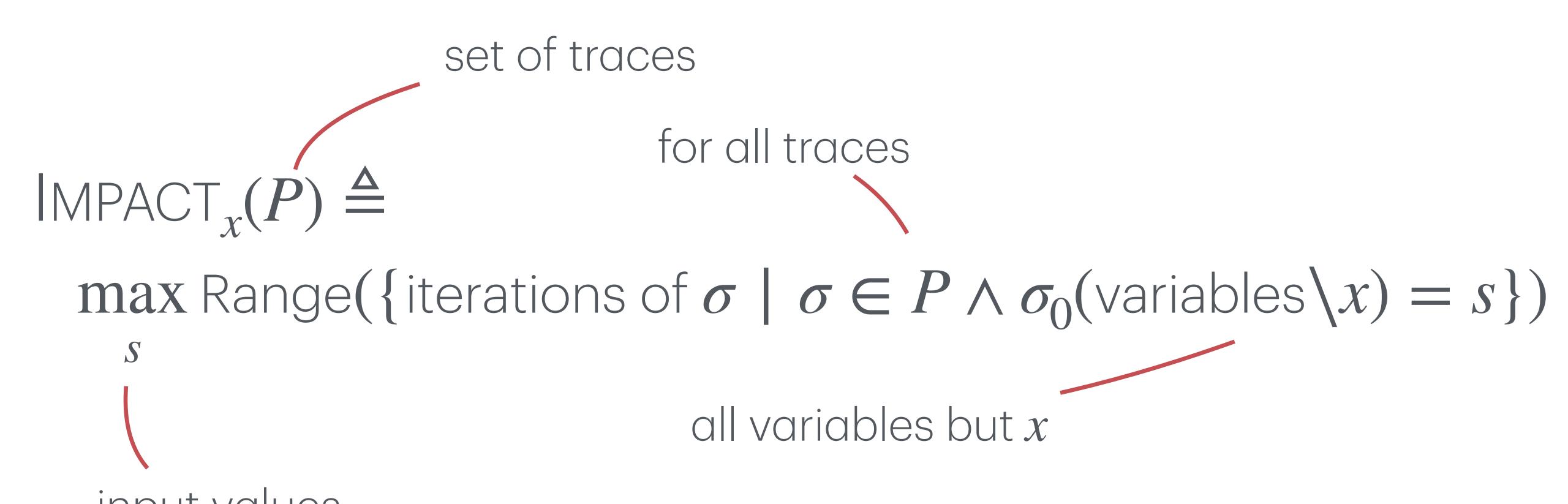
$$\max_{s} \operatorname{Range}(\{\text{iterations of } \sigma \mid \sigma \in P \land \sigma_0(\text{variables} \backslash x) = s\})$$

$$\mathsf{IMPACT}_{\mathcal{X}}(P) \triangleq$$

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

set of traces  $\mathsf{IMPACT}_{\mathcal{X}}(P) \triangleq$ max Range({iterations of  $\sigma \mid \sigma \in P \land \sigma_0(\text{variables} \setminus x) = s})$ input values

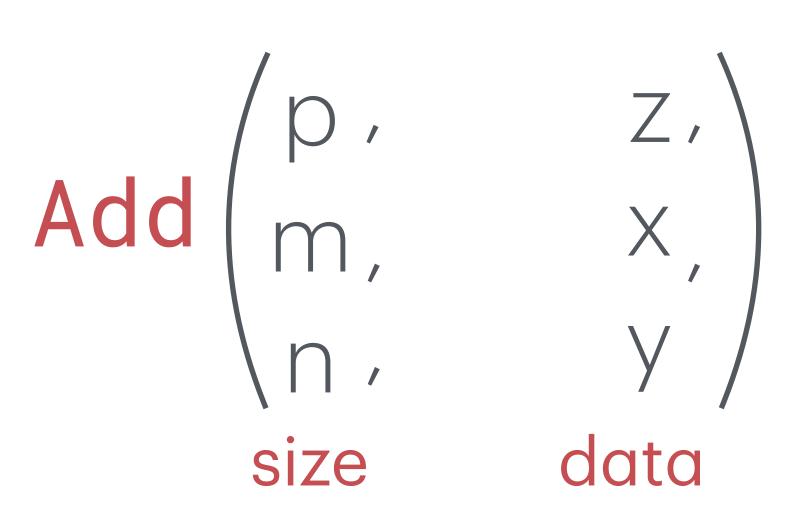
set of traces for all traces  $\mathsf{IMPACT}_{x}(P) \triangleq$ max Range({iterations of  $\sigma \mid \sigma \in P \land \sigma_0(\text{variables} \setminus x) = s})$ 



set of traces for all traces  $\mathsf{IMPACT}_{x}(P) \triangleq$ max Range({iterations of  $\sigma \mid \sigma \in P \land \sigma_0(\text{variables} \setminus x) = s})$ all variables but x

all perturbations of variable x

#### Add Function



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

#### Add Function

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

•  $IMPACT_{\{p,m,n\}}(Add) \ge 0$ 

- -

#### Add Function

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

- $IMPACT_{\{p,m,n\}}(Add) \ge 0$   $IMPACT_{\{Z,X,y\}}(Add) = 0$

#### How?

$$IMPACT_{\{Z,X,y\}}(Add)$$

How?

abstract

concrete

$$Impact^{\natural}_{\{Z,X,Y\}}(Add) \ge Impact_{\{Z,X,Y\}}(Add)$$

How?

abstract

concrete

$$\label{eq:local_problem} \text{Impact}^{\natural}_{\{\text{Z},\text{X},\text{Y}\}}(\text{Add}) = 0 \text{ then } \text{IMPACT}_{\{\text{Z},\text{X},\text{Y}\}}(\text{Add}) = 0$$

 $Impact^{\natural}_{\{Z,X,Y\}}(Add) \ge Impact_{\{Z,X,Y\}}(Add)$ 

In three steps:

(i) Remove irrelevant instructions

In three steps:

(i) Remove irrelevant instructions

Syntactic Dependency Analysis

In three steps:

(i) Remove irrelevant instructions

Syntactic Dependency Analysis

(ii) Abstract Interpretation

In three steps:

(i) Remove irrelevant instructions

Syntactic Dependency Analysis

(ii) Abstract Interpretation

Invariant on input variables + iteration counter

In three steps:

(i) Remove irrelevant instructions

Syntactic Dependency Analysis

- (ii) Abstract Interpretation Invariant on input variables + iteration counter
- (iii) Impact quantification

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

```
def Add(p, z, m, x, n, y): 26
                                    else:
    r = min(p, m)
                               27
                                      t = p - r
     s = min(p, n)
                               28
                                      q = r - s
     if (r < s):
                               29
      t = p - s
                               30
       q = s - r
                                      for (; s > 0; s--):
                               32
                                        r = x[i]
                               33
                                        w = y[i]
       for (; r > 0; r--):
                                        z[i] = r + w + b
                               34
        s = x[i]
                               35
                                        i = i + 1
         w = y[i]
                               36
                                        b = (w < b) | |
         z[i] = s + w + a
                                          (r + w < r) | |
                               37
13
         i = i + 1
                               38
                                           (r + w + b < r)
         a = (w < a) | |
14
                               39
                                      for (; q > 0; q--):
            (s + w < s) | |
                                        r = x[i]
16
            (s + w + a < s)
                                        z[i] = r + b
17
       do:
                               42
                                        i = i + 1
         r = y[i]
18
                                        b = (r < b) | |
                               43
19
         b = (r < a) | |
                                          (r + b < r)
           (r + a < r)
20
                               45
                                    if (t > 0):
21
         z[i] = r + a
                               46
                                      while (t > 0):
23
                               48
24
                               49
       while (q > 0)
25
                                        if (t > 0):
                               50
                                         |z[i]| = 0
```

```
def Add(p, z, m, x, n, y): 26
                                   else:
    r = min(p, m)
                              27
                                     t = p - r
     s = min(p, n)
                              28
                                     q = r - s
     if (r < s):
                              29
                                     i = 0
      t = p - s
                              30
       q = s - r
                                     for (; s > 0; s--):
                              32
                                       r = x[i]
                              33
                                       w = y[i]
       for (; r > 0; r--):
                              34
                                       z[i] = r + w + b
       s = x[i]
10
                              35
                                       i = i + 1
         w = y[i]
                              36
                                       b = (w < b) | |
         z[i] = s + w + a
                                         (r + w < r) | |
                              37
13
                              38
                                          (r + w + b < r)
         a = (w < a)
                              39
                                     for (; q > 0; q--):
           (s + w < s)
                                       r = x[i]
           (s + w + a < s)
16
                                       z[i] = r + b
17
       do:
                              42
                                       i = i + 1
         r = y[i]
18
                                       b = (r < b) | |
                              43
         b = (r < a) | |
19
                                         (r + b < r)
           (r + a < r)
20
                              45
                                    if (t > 0):
21
         z[i] = r + a
                              46
                                     while (t > 0):
23
                              48
24
                              49
       while (q > 0)
25
                                       if (t > 0):
                               50
                                        |z[i]| = 0
```

```
def Add(p, z, m, x, n, y): 26
                                     else:
     r = min(p, m)
                                27
                                       t = p - r
     s = min(p, n)
                                28
     if (r < s):
                                29
                                30
                                        for (; s > 0; s--):
                                          r = x[i]
                                33
                                          w = y[i]
                                          z[i] = r + w + b
                                35
                                          i = i + 1
                                          b = (w < b) | |
                                           (r + w < r) | |
                                37
                 s + w + a
13
                                38
                                            (r + w + b < r)
          a = (w < a) | |
                                        for (; q > 0; q--):
                                          r = x[i]
            (s + w + a < s)
16
                                          z[i] = r + b
                                42
                                          i = i + 1
18
                                          b = (r < b) | |
19
                                            (r + b < r)
20
             (r + a < r)
                                45
                                      if (t > 0):
21
                                        while (t > 0):
23
                                48
24
                                49
       while (q > 0)
                                          if (t > 0):
25
                                50
                                          z[i] = 0
```

```
def Add(p, z, m, x, n, y): 26
                                      else:
     r = min(p, m)
                                27
                                        t = p - r
     s = min(p, n)
                                28
     if (r < s):
                                29
                                        for (; s > 0; s--):
                                          r = x[i]
                                33
                                          w = y[i]
                                          z[i] = r + w + b
                                35
                                          i = i + 1
                                          b = (w < b) | |
                                            (r + w < r) | |
                                37
                                38
                                            (r + w + b < r)
          a = (w < a) | |
                                        for (; q > 0; q--):
                                          r = x[i]
16
            (s + w + a < s)
                                          z[i] = r + b
                                42
                                          i = i + 1
18
                                          b = (r < b) | |
19
                                            (r + b < r)
20
             (r + a < r)
                                45
                                      if (t > ∅):
21
                                        while (t > 0):
23
                                48
24
                                49
25
                                          if (t > 0):
                                50
                                            z[i] = 0
```

```
def Add(p, z, m, x, n, y): 26
                                   else:
    r = min(p, m)
                              27
                                     t = p - r
     s = min(p, n)
                              28
                                     q = r - s
     if (r < s):
                              29
                                     i = 0
      t = p - s
                              30
       q = s - r
                                     for (; s > 0; s--):
                              32
                                       r = x[i]
                              33
                                       w = y[i]
       for (; r > 0; r--):
                              34
                                       z[i] = r + w + b
       s = x[i]
10
                              35
                                       i = i + 1
         w = y[i]
                              36
                                       b = (w < b) | |
         z[i] = s + w + a
                                        (r + w < r) | |
                              37
13
                              38
                                         (r + w + b < r)
14
         a = (w < a)
                              39
                                     for (; q > 0; q--):
           (s + w < s)
                                      r = x[i]
           (s + w + a < s)
16
                                       z[i] = r + b
17
       do:
                              42
                                       i = i + 1
18
       | r = y[i]
                                       b = (r < b) | |
                              43
         b = (r < a) | |
19
                                        (r + b < r)
20
          (r + a < r)
                              45
                                   if (t > 0):
21
                              46
                                     while (t > 0):
23
24
                              49
       while (q > 0)
                                       if (t > 0):
                              50
                                       z[i] = 0
```

```
def Add(p, z, m, x, n, y): 26
                                   else:
     r = min(p, m)
                              27
                                     t = p - r
     s = min(p, n)
                              28
                                     q = r - s
     if (r < s):
                              29
       t = p - s
                              30
       q = s - r
                                     for (; s > 0; s--):
                              32
                                       r = x[i]
                              33
                                       w = y[i]
       for (; r > 0; r--):
                              34
                                       z[i] = r + w + b
       | s = x[i]
10
                              35
                                       i = i + 1
         W = y[i]
                              36
                                       b = (w < b) | |
         z[i] = s + w + a
                              37
                                         (r + w < r)
13
                              38
                                         (r + w + b < r)
14
         a = (w < a)
                              39
                                     for (; q > 0; q--):
                                       r = x[i]
           (s + w < s)
           (s + w + a < s)
16
                                       z[i] = r + b
17
       do:
                              42
18
        | r = y[i]
                              43
                                       b = (r < b) | |
19
         b = (r < a) | |
                                        (r + b < r)
20
          (r + a < r)
                              45
                                   if (t > 0):
21
                              46
                                     while (t > 0):
                                     Ti = i + 1
23
                              49
      while (q > 0)
                                       if (t > 0):
                              50
                                       z[i] = 0
```

```
def Add(p, z, m, x, n, y): 26
                                    else:
     r = min(p, m)
                              27
                                      t = p - r
     s = min(p, n)
                              28
                                      q = r - s
     if (r < s):
                               29
       t = p - s
                              30
       q = s - r
                                      for (; s > 0; s--):
                              32
                                       r = x[i]
                              33
                                        w = y[i]
       for (; r > 0; r--):
                              34
                                        z[i] = r + w + b
       | s = x[i]
10
                               35
                                       i = i + 1
         W = y[i]
                               36
                                        b = (w < b) | |
         z[i] = s + w + a
                              37
                                         (r + w < r)
13
                              38
                                          (r + w + b < r)
14
         a = (w < a) | |
                              39
                                      for (; q > 0; q--):
           (s + w < s)
                                       r = x[i]
           (s + w + a < s)
16
                                        z[i] = r + b
17
       do:
                               42
18
        | r = y[i]
                               43
                                        b = (r < b) | |
         b = (r < a) | |
19
                                        (r + b < r)
20
          (r + a < r)
                              45
                                    if (t > 0):
21
                              46
                                      while (t > 0):
23
                               48
24
                               49
       while (q > 0)
25
                               50
```

# Syntactic Dependency Analysis

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

19

20

21

```
def Add(p, z, m, x, n, y): 26
                                  else:
  r = min(p, m)
                            27
                                    t = p - r
  s = min(p, n)
                            28
                                    q = r - s
  if (r < s):
                            29
    t = p - s
    q = s - r
                                    for (; s > 0; s--):
                                      r = x[i]
                            33
                                      W = y[i]
    for (; r > 0; r--):
                                      z[i] = r + w + b
     | s = x[i]
                            35
                                      i = i + 1
      W = y | i |
                            36
                                      b = (w < b) | |
      z[i] = s + w + a
                            37
                                        (r + w < r) | |
                                        (r + w + b < r)
                            38
      a = (w < a) | |
                            39
                                    for (; q > 0; q--):
        (s + w < s)
                                     r = x[i]
        (s + w + a < s)
                                      z[i] = r + b
    do:
                            42
      r = y[i]
                                      b = (r < b) | |
      b = (r < a) | |
                                       (r + b < r)
        (r + a < r)
                                  if (t > 0):
                            45
                                    while (t > 0):
                                    Ti = i + 1
                            49
    while (q > 0)
                                      if (t > 0):
                                      z[i] = 0
```

# 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):
    r = min(p, m)
    s = min(p, n)
    if (r < s):
      t = p - s
      q = s - r
      for (; r > 0; r--):
       ⊤skip;
       do:
      I q--;
25
      while (q > 0)
     else:
      t = p - r
      for (; s > 0; s--):
       Tskip;
       for (; q > 0; q--):
39
       T skip;
     if (t > ∅):
47 while (t > 0):
```

#### (ii) Abstract Interpretation

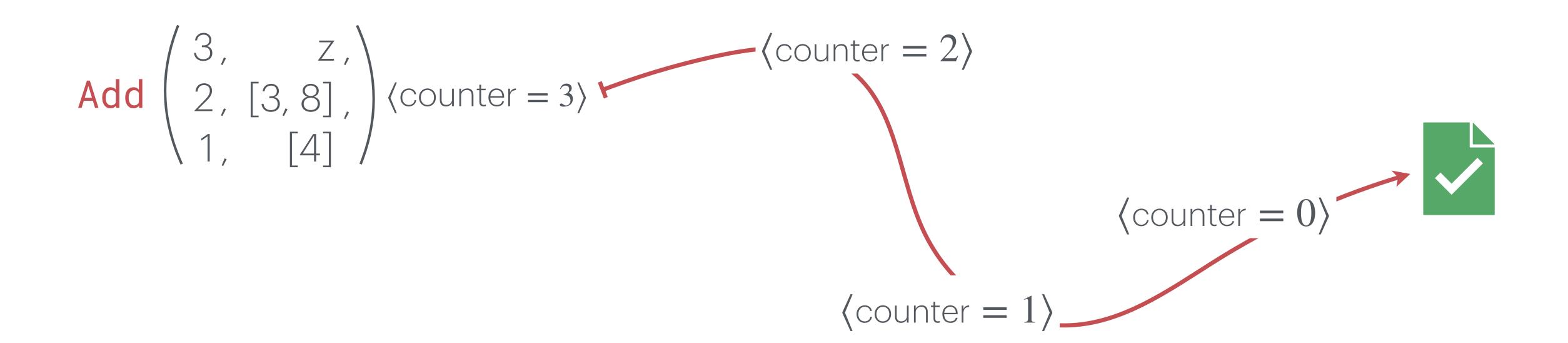
```
1 def Add(p, z, m, x, n, y):
2 r = min(p, m)
3 \mid s = min(p, n)
4 if (r < s):
5 t = p - s
6 q = s - r
9 for (; r >
    for (; r > 0; r--):
      T skip;
      do:
     I q--;
    while (q > 0)
25
26
    else:
    Tt = p - r
28 | q = r - s
31 for (; s > 0; s--):
-- | Tskip;
      for (; q > 0; q--):
39
      ⊤skip;
    if (t > 0):
```

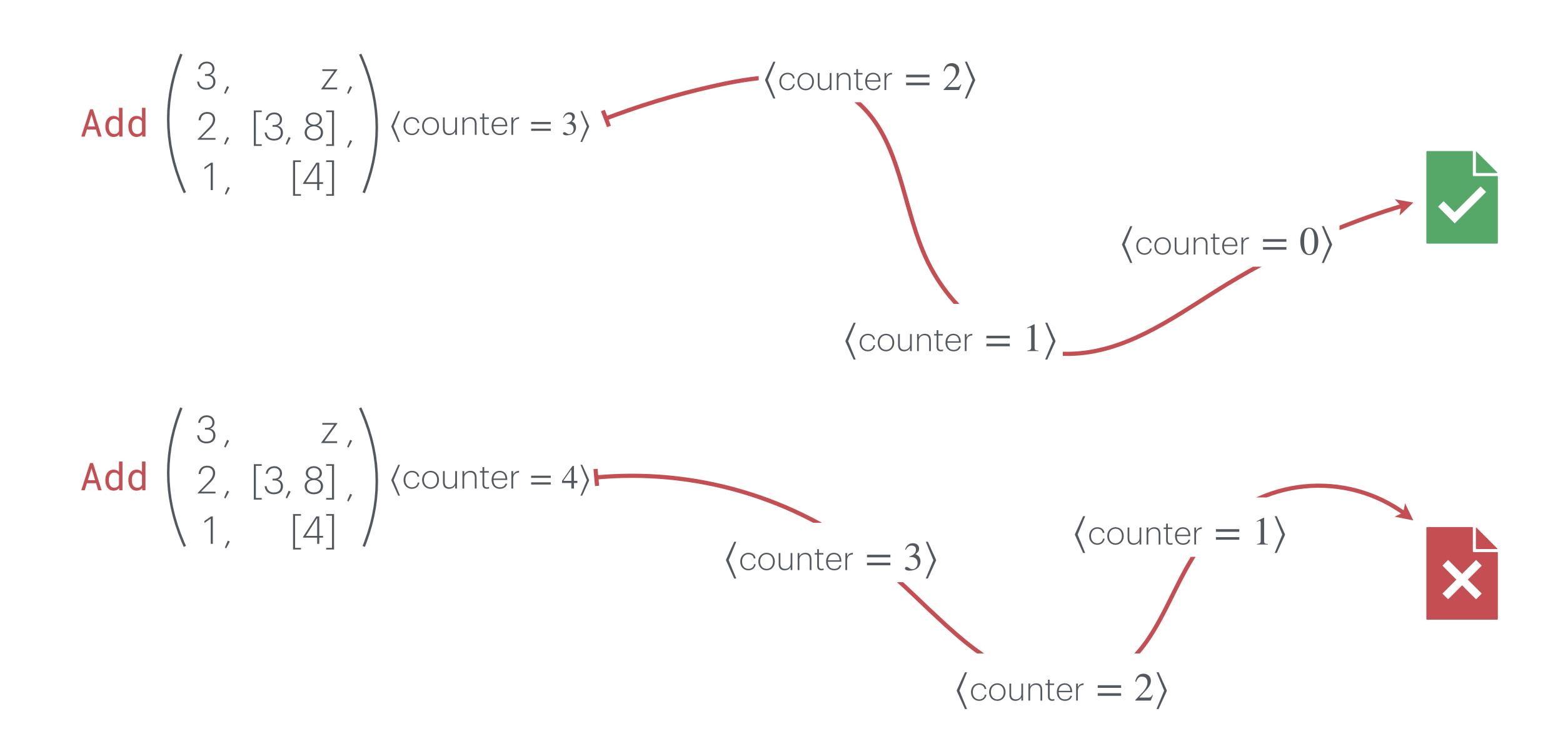
Intuitively:

#### (ii) Abstract Interpretation

```
1 def Add(p, z, m, x, n, y):
                                 Intuitively:
    r = min(p, m)
    s = min(p, n)
    if (r < s):
    t = p - s
6 \qquad q = s - r
    for (; r > 0; r--):
                              Augment each
      T skip; counter--
      do:
      I q--; counter-- loop body with a counter
      while (q > 0)
25
                               for iterations
26
     else:
     T t = p - r
28
      q = r - s
      for (; s > 0; s--):
      T skip; counter--
      for (; q > 0; q--):
39
      T skip; counter--
     if (t > ∅):
    Twhile (t > 0):
     Tt--; counter--
```

```
1 def Add(p, z, m, x, n, y):
                                 Intuitively:
    r = min(p, m)
    s = min(p, n)
    if (r < s):
    t = p - s
      q = s - r
      for (; r > 0; r--):
                              Augment each
      T skip; counter--
      do:
      Iq--; counter-- loop body with a counter
      while (q > 0)
25
                               for iterations
26
     else:
      t = p - r
28
      q = r - s
      for (; s > 0; s--):
      T skip; counter--
                           Backwards starting
      for (; q > 0; q--):
39
      T skip; counter--
                                 from zero!
     if (t > ∅):
      while (t > 0):
47
49
        t--, counter
     assert counter == 0
```





```
def Add(p, z \neq m, x, n, y):
                                  Intuitively:
     r = min(p/m)
 6
           (: r > 0; r--):
                              Augment each
       T skip; counter--
       do:
       I q--; counter-- loop body with a counter
       while (q > 0)
                               for iterations
26
28
31
       for (; s > 0; s--):
         skip; counter--
                            Backwards starting
          (; q > 0; q--):
39
        skip; counter--
                                 from zero!
     1f (t > 0):
      while (t > 0):
     T t--; counter--
49
```

assert counter == 0

Backward abstract analysis

### Without rewritings!

```
for (; r > 0; r--):
    Tskip; counter--
do:
    I q--; counter--
while (q > 0)
else:
    I t = p - r
```

#### Iteration counter is handled

# <sup>39</sup>semantically<sup>39</sup>

```
45
47
47
49
-- assert counter == 0
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

 $\mathsf{Filter}^{\natural} \llbracket b \rrbracket (\Lambda^{\natural} \llbracket stmt \rrbracket) \sqcup^{\natural} \mathsf{Filter}^{\natural} \llbracket \neg b \rrbracket (\Lambda^{\natural} \llbracket stmt' \rrbracket)$ 

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

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

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

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

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

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

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

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

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

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

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

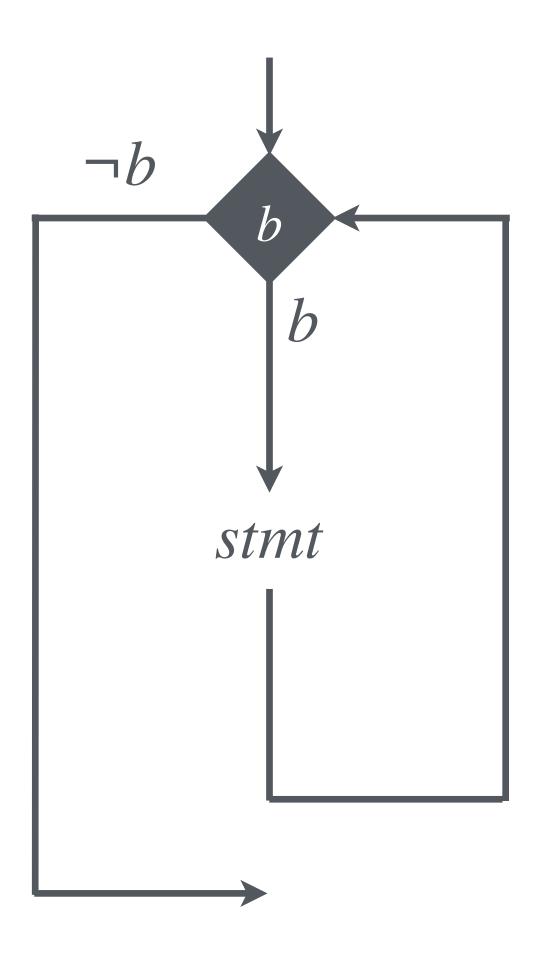
 $\Lambda^{\natural}$ [[if b then stmt else stmt']] $d^{\natural} \triangleq$ 

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

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

$$F(x^{\natural}) \triangleq \text{Filter}^{\natural} \llbracket \neg b \rrbracket d^{\natural} \sqcup^{\natural}$$

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



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

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

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

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

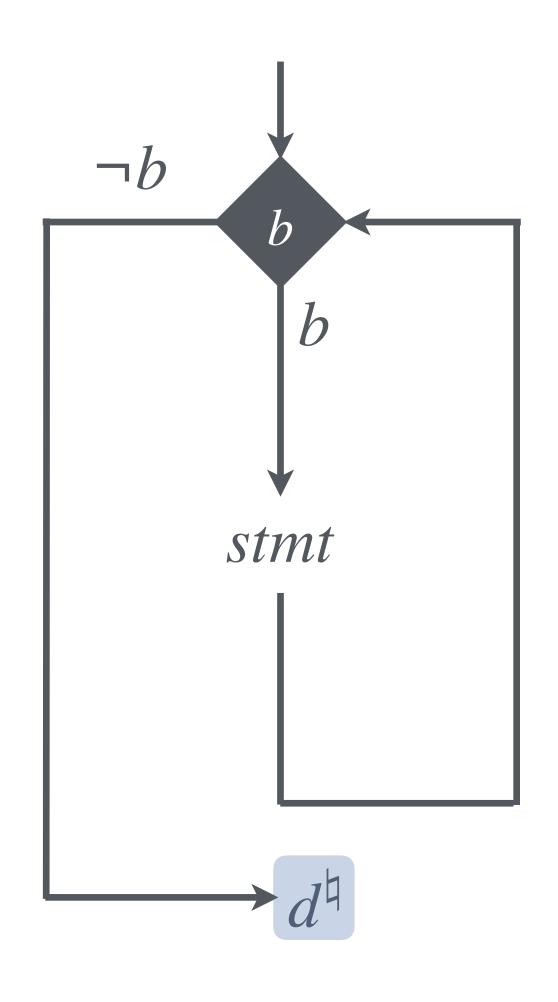
 $\Lambda^{\natural}$  [[if b then stmt else stmt']]  $d^{\natural} \triangleq$ 

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

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

$$F(x^{\natural}) \triangleq \text{Filter}^{\natural} \llbracket \neg b \rrbracket d^{\natural} \sqcup^{\natural}$$

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



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

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

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

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

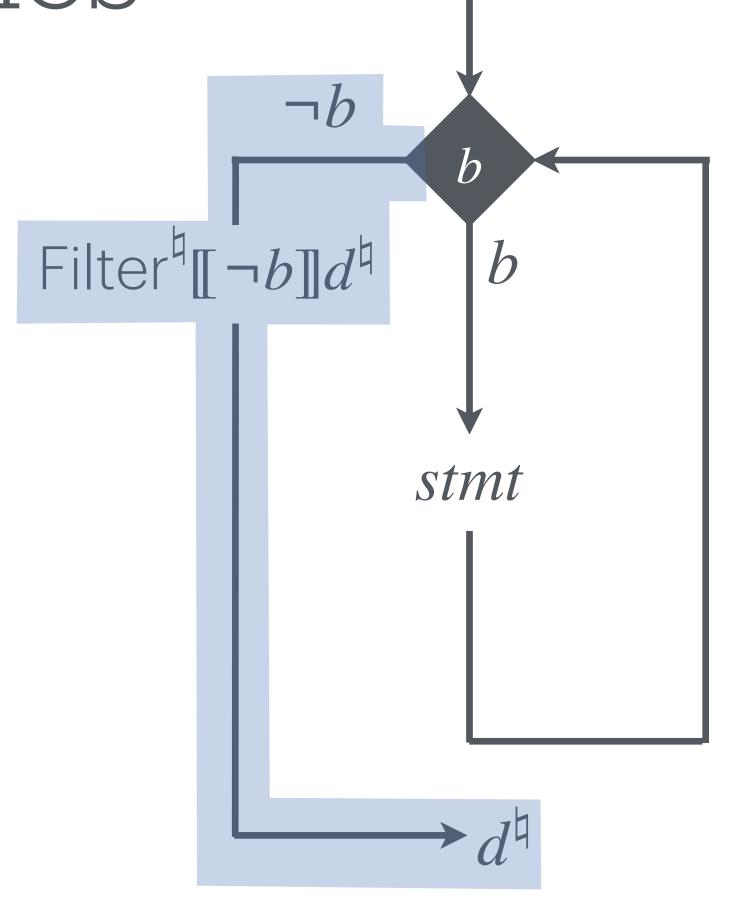
 $\Lambda^{\natural}$ [[if b then stmt else stmt']] $d^{\natural} \triangleq$ 

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

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

$$F(x^{\natural}) \triangleq \text{Filter}^{\natural} \llbracket \neg b \rrbracket d^{\natural} \sqcup^{\natural}$$

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



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

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

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

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

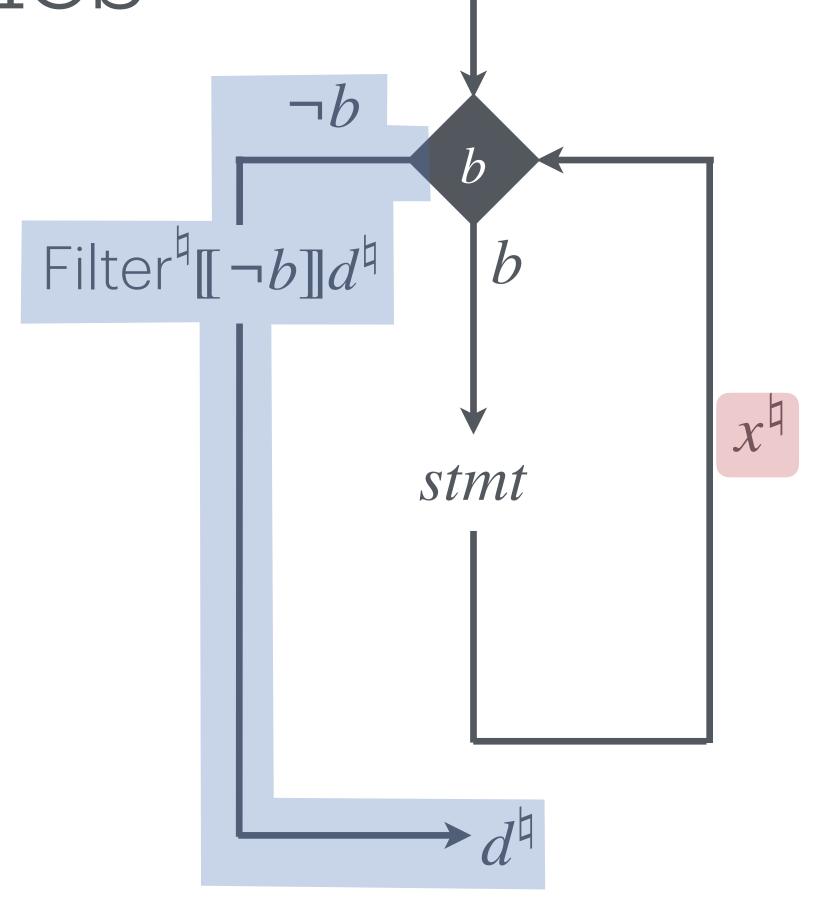
 $\Lambda^{\natural}$  [[if b then stmt else stmt']]  $d^{\natural} \triangleq$ 

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

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

$$F(x^{\natural}) \triangleq \text{Filter}^{\natural} \llbracket \neg b \rrbracket d^{\natural} \sqcup^{\natural}$$

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



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

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

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

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

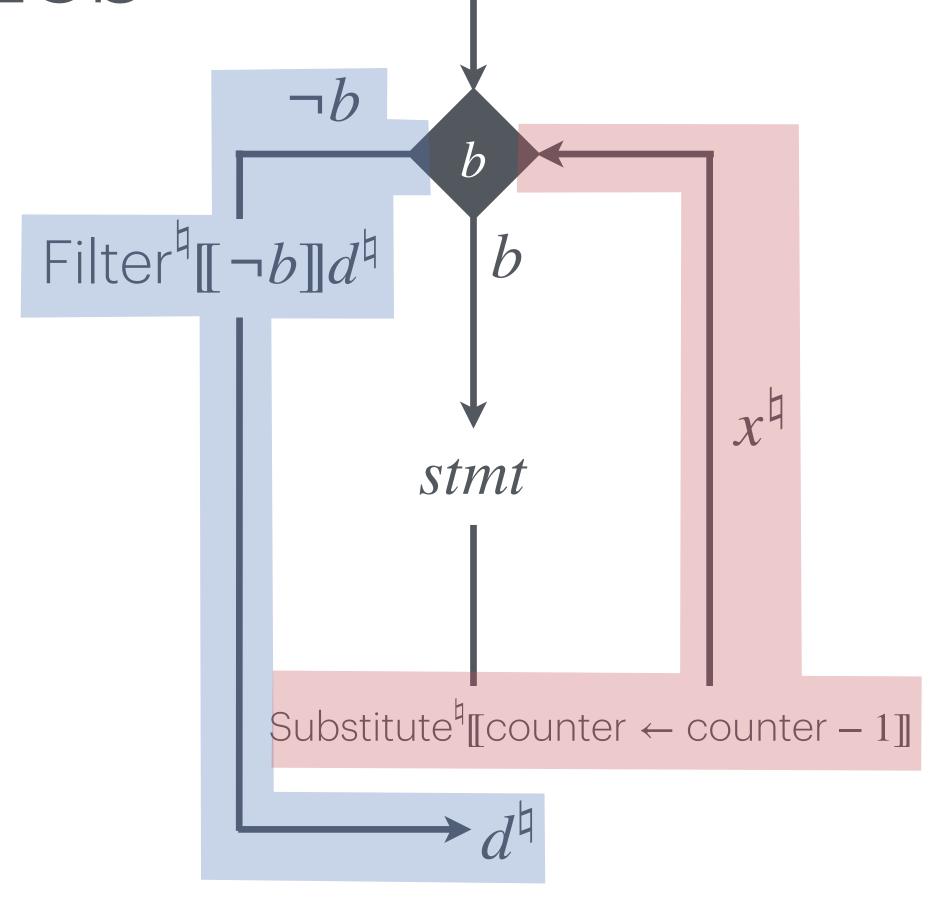
 $\Lambda^{\natural}$ [[if b then stmt else stmt']] $d^{\natural} \triangleq$ 

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

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

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

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



starts from

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

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

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

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

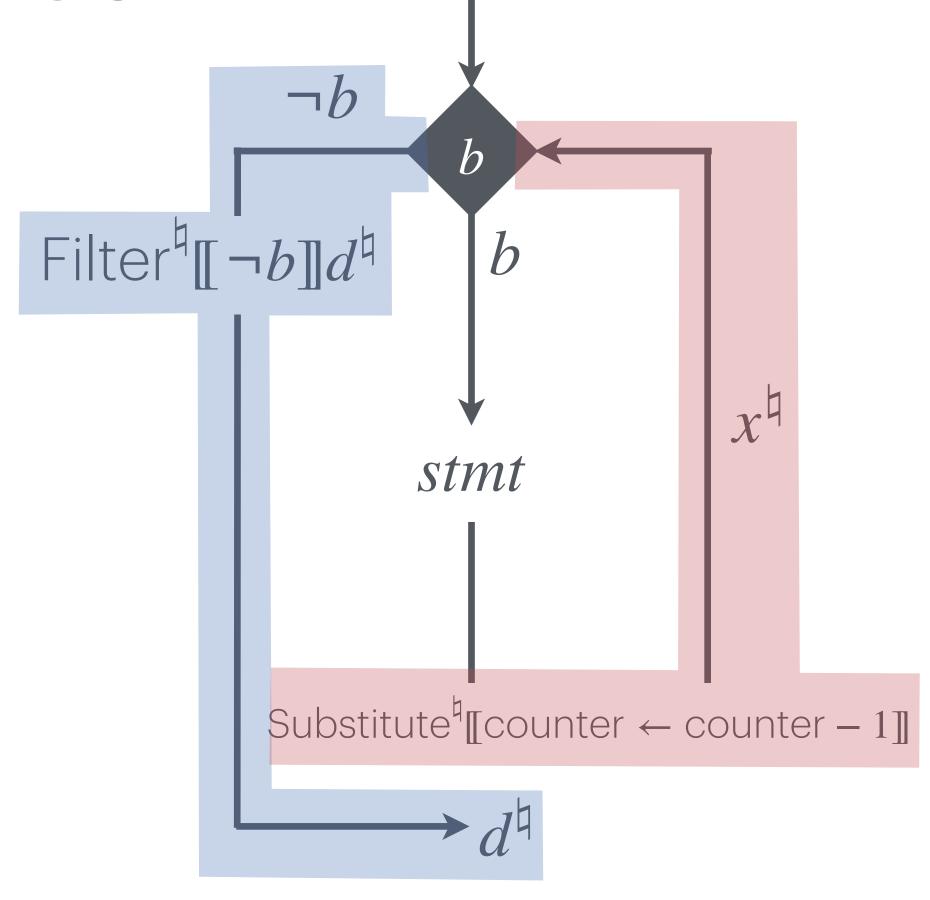
 $\Lambda^{\natural}$ [[if b then stmt else stmt']] $d^{\natural} \triangleq$ 

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

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

$$F(x^{\natural}) \triangleq \text{Filter}^{\natural} \llbracket \neg b \rrbracket d^{\natural} \sqcup^{\natural}$$

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



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

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

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

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

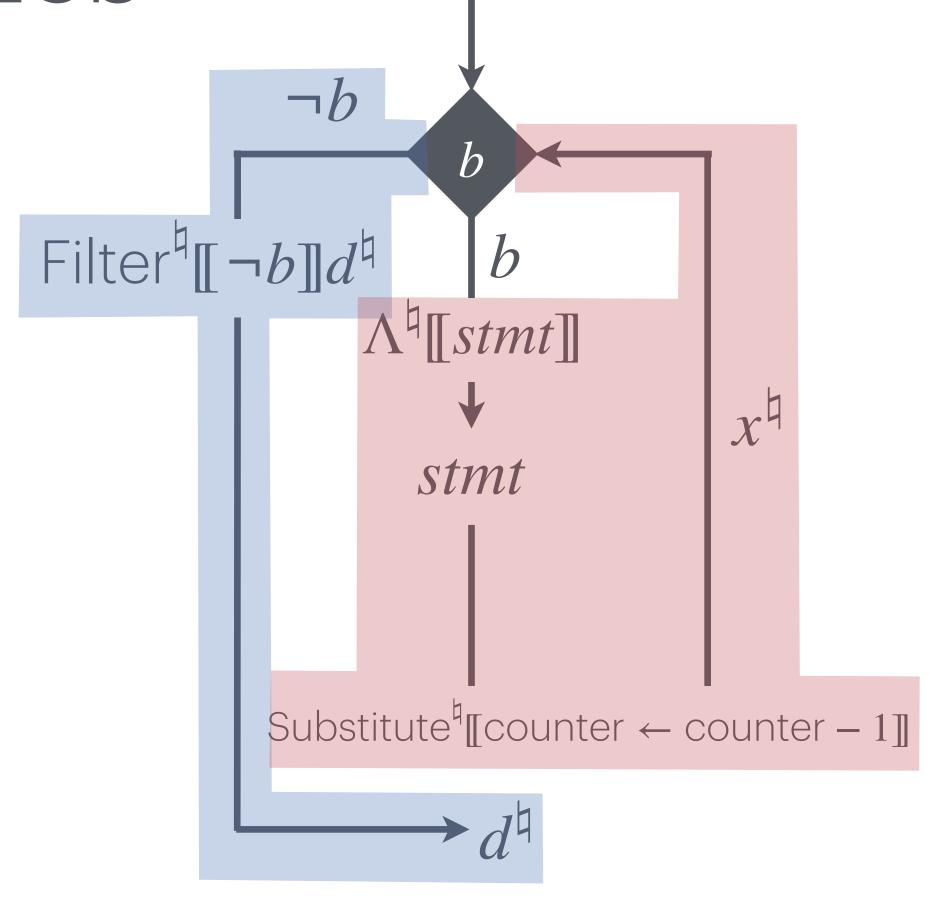
 $\Lambda^{\natural}$  [[if b then stmt else stmt']]  $d^{\natural} \triangleq$ 

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

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

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

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



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

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

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

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

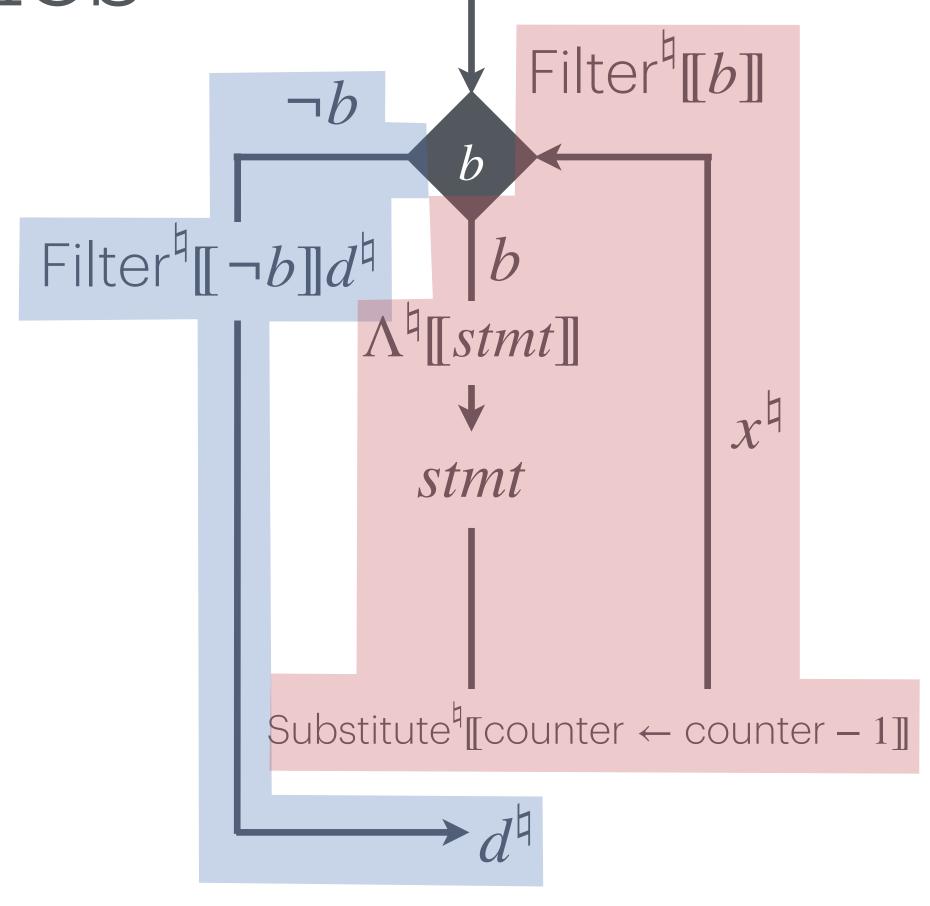
 $\Lambda^{\natural}$ [[if b then stmt else stmt']] $d^{\natural} \triangleq$ 

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

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

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

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



Abstract invariant on the input variables + counter

Backward abstract analysis

```
Augment each
  def Add(p, z \neq m, x, n, y):
     r = min(p/m)
                        loop body with a counter
                               for iterations
           ; r > 0; r--):
        skip; counter--
       do:
                            Backwards starting
        q--; counter--
      while \sqrt{q} > 0
                                 from zero!
26
28
       for (; s > 0; s--):
        skip; counter--
                           At the beginning, the
           (; q > 0; q--):
39
        'skip; counter--
                        counter yields the global
```

while (t > 0):

49

Tt--; counter--

assert counter == 0

number of iterations

Abstract invariant on the input variables + counter

```
Forward +

Backward abstract

analysis
```

```
Augment each
  def Add(p, z \neq m, x, n, y):
     r = min(p/m)
                        loop body with a counter
                               for iterations
           ; r > 0; r--):
        skip; counter--
       do:
                            Backwards starting
       q--; counter--
      while \sqrt{q} > 0
                                 from zero!
28
       for (; s > 0; s--):
        skip; counter--
                           At the beginning, the
          (; q > 0; q--):
39
        'skip; counter--
                        counter yields the global
```

while (t > 0):

49

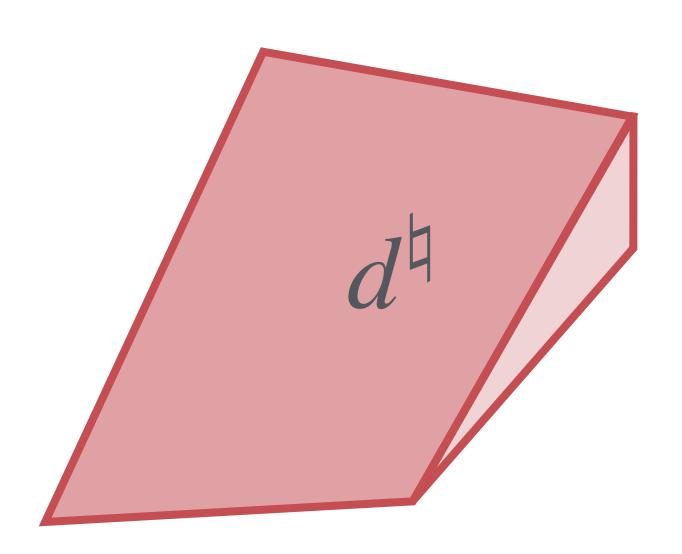
Tt--; counter--

assert counter == 0

number of iterations

Abstract invariant on the

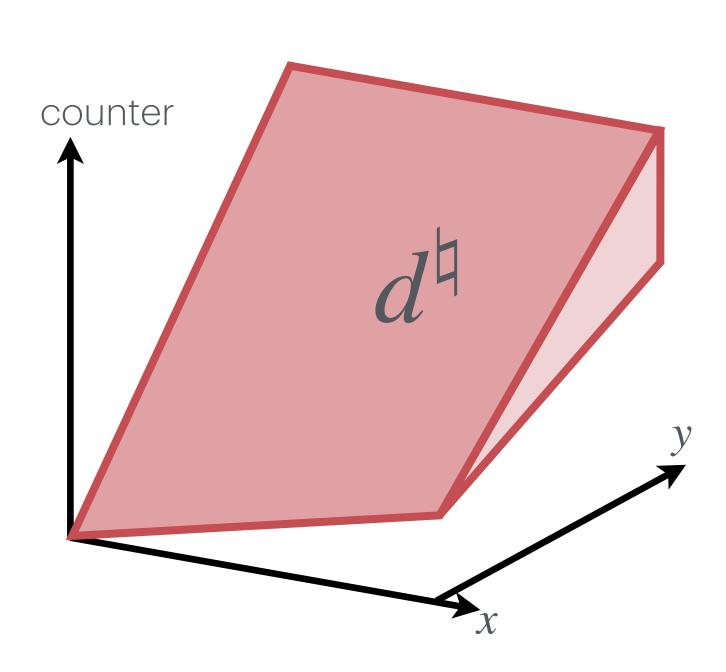
input variables + counter



Abstract invariant on the

input variables + counter

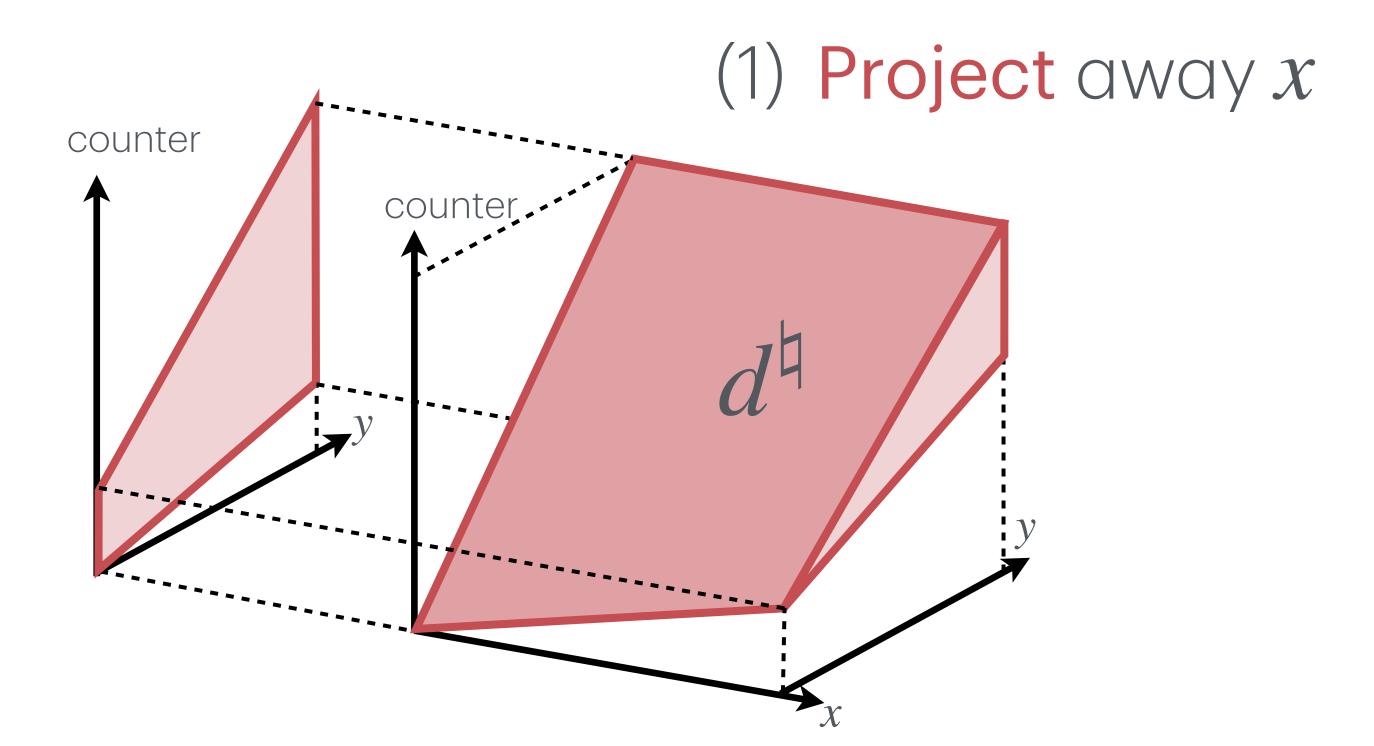
(i) Assume input variable of interest x



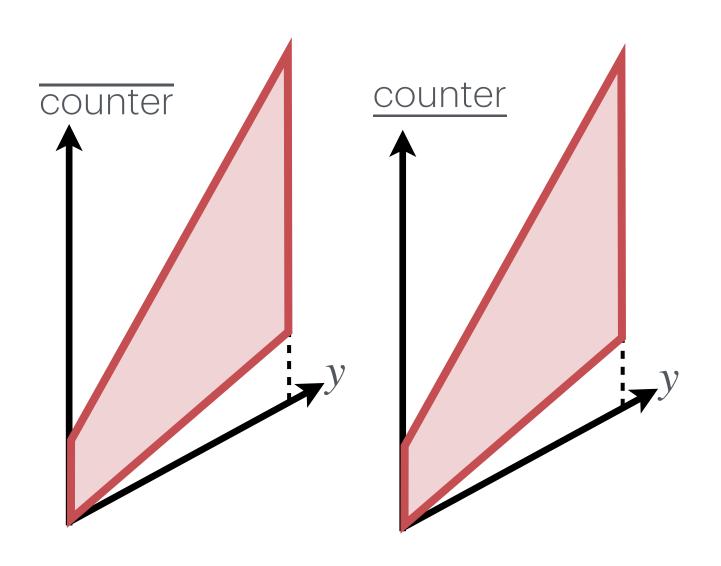
Abstract invariant on the

input variables + counter

(i) Assume input variable of interest x

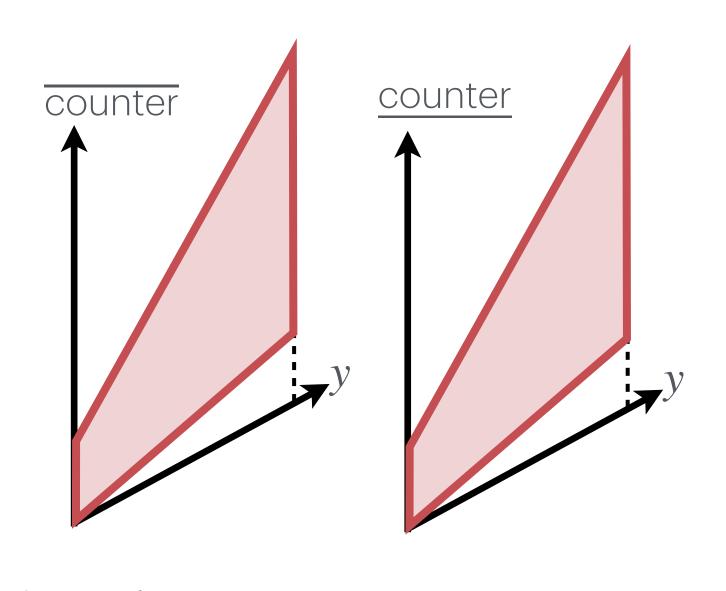


Abstract invariant on the input variables + counter



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

Abstract invariant on the input variables + counter

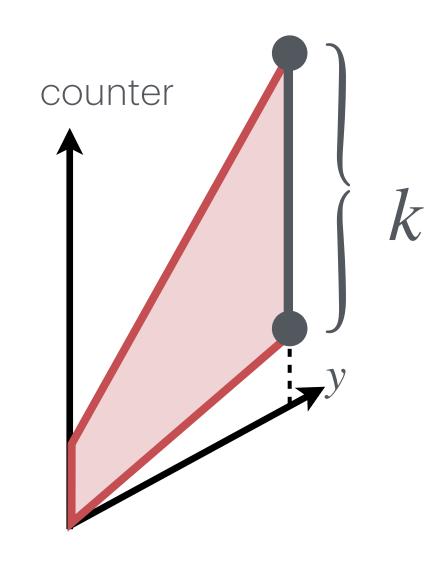


 $0 \le k \le \overline{\text{counter}} - \text{counter}$ 

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

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 counter and counter
- (3) Maximize the distance between the two

k is the impact of x

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

Impact<sub>x</sub><sup>$$\natural$$</sup> $(d^{\natural}) = \max k \text{ subject to}$ 

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

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

$$Project_{\chi}(d^{\natural})$$

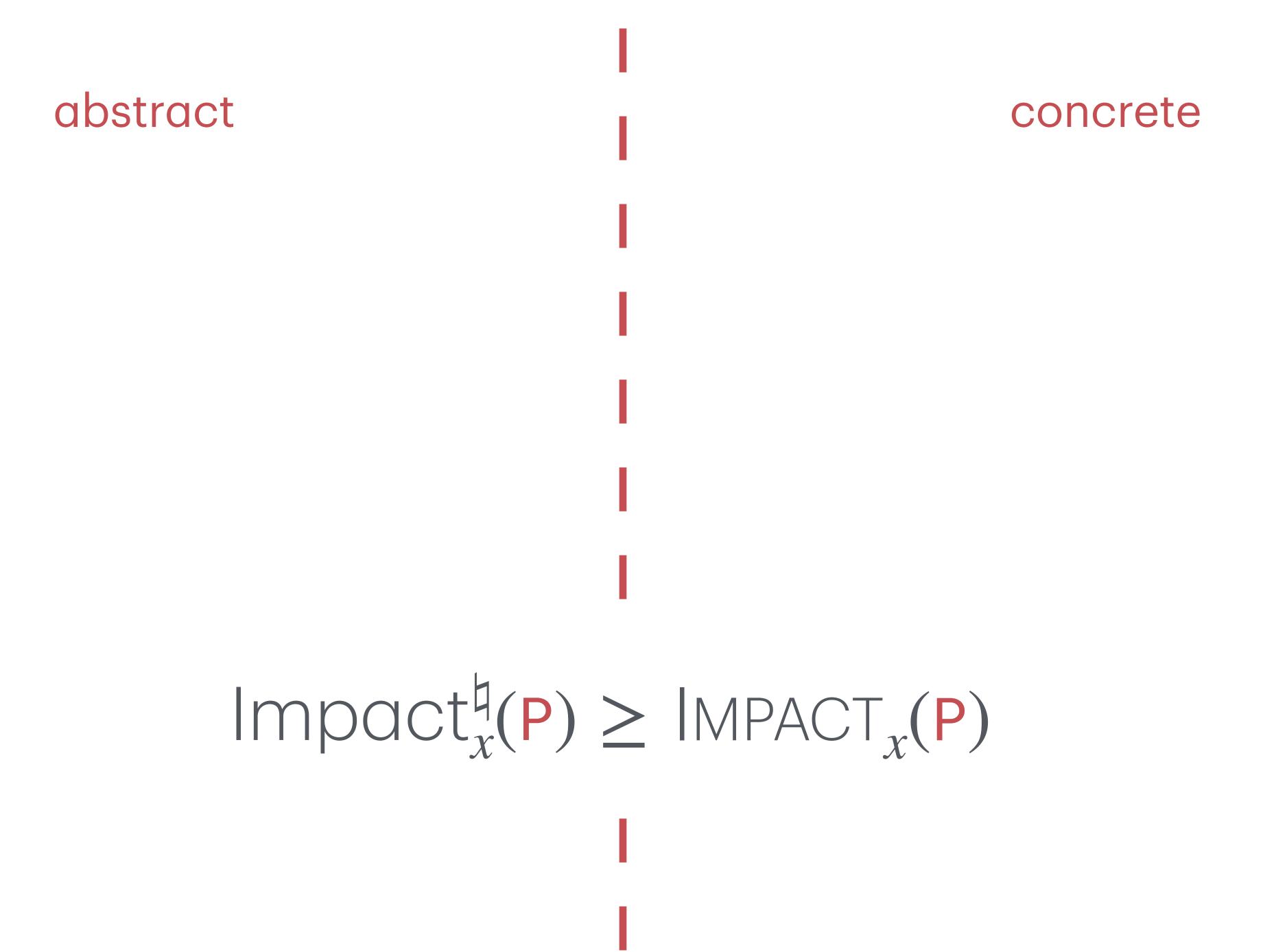
$$0 \le k \le \overline{\text{counter}} - \underline{\text{counter}}$$

Impact<sub>x</sub><sup>$$\dagger$$</sup> $(d^{\dagger}) = \max k \text{ subject to}$ 

Substitute[[counter  $\leftarrow$  counter]](Project<sub>x</sub>( $d^{\natural}$ ))  $\land$ 

Substitute[[counter  $\leftarrow$  counter]](Project<sub> $\chi$ </sub>( $d^{\dagger}$ ))  $\land$ 

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



Denis Mazzucato et al.

abstract concrete input variable x has an  $Impact_{x}^{\sharp}(P) \leq k$ impact below k on the iterations of the program P

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

### Add Function

```
def Add(p, z, m, x, n, y):
     r = min(p, m)
                           p = counter \land 0 \le p \le u
           (; r > 0; r--):
        skip; counter--
       do:
       _ q--; counter--
             (p > 0)
     else:
       for (; s > 0; s--):
         skip; counter--
39
           (; q > 0; q--):
         skip; counter--
       while (t > 0):
       Tt--; counter--
49
     assert counter = 0
```

Forward +

Backward abstract

analysis

### Add Function

$$d^{\natural} \text{ is}$$
 
$$p = \text{counter} \land 0 \le p \le u$$

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

Substitute[[counter  $\leftarrow$  counter]](Project<sub>x</sub>( $d^{\dagger}$ ))  $\land$ 

Substitute[[counter  $\leftarrow$  counter]](Project<sub>x</sub>( $d^{\natural}$ ))  $\land$ 

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

### Add Function

$$d^{\natural} \text{ is}$$

$$p = \text{counter} \land 0 \le p \le u$$

Impact 
$$p$$
 = counter  $\wedge 0 \le p \le u$  = max  $k$  subject to

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

$$0 \le counter \le u \land$$

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

$$d^{\natural} \text{ is}$$

$$p = \text{counter} \land 0 \le p \le u$$

Impact
$$_p^{\natural}(p = \text{counter} \land 0 \le p \le u) = \max k \text{ subject to}$$

$$0 \le \overline{\text{counter}} \le u \land$$

$$0 \le \text{counter} \le u \land$$

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

U

$$d^{\natural} \text{ is}$$

$$p = \text{counter} \land 0 \le p \le u$$

Impact 
$$h$$
 (p = counter  $\wedge$  0  $\leq$  p  $\leq$  u) = max  $k$  subject to

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

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

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

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

$$d^{\natural}$$
 is 
$$p = \text{counter} \land 0 \le p \le u$$

$$d^{\natural}$$
 is 
$$p = \text{counter} \land 0 \le p \le u$$

$$d^{\natural}$$
 is 
$$p = \text{counter} \land 0 \le p \le u$$

Impact<sub>n</sub><sup>$$\natural$$</sup>(p = counter  $\land$  0  $\leq$  p  $\leq$  u) = max k subject to

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

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

$$0 \le p \le u \land$$

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

0

And to all the other input variables

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

```
d is
 1 def Add(p, z, m, x, n, y):
    r = min(p, m)
                         p = counter \land 0 \le p \le u
    s = min(p, n)
    if (r < s):
    t = p - s

q = s - r
      for (; r > 0; r--):
      T skip; counter--
       do:
      while (q > 0)
26
     else:
     T t = p - r
28
      q = r - s
      for (; s > 0; s--):
      T skip; counter--
39
      for (; q > 0; q--):
      T skip; counter--
     if (t > 0):
      while (t > 0):
```

T t--; counter--

assert counter = 0

49

```
\operatorname{Impact}_{p}^{\natural}(d^{\natural}) = u
\operatorname{Impact}_{m}^{\sharp}(d^{\sharp}) = 0
\mathrm{Impact}_n^{\natural}(d^{\natural}) = 0
Impact^{\sharp}_{r}(d^{\sharp}) = 0
Impact<sup>\dagger</sup><sub>v</sub>(d^{\dagger}) = 0
Impact^{\sharp}(d^{\sharp}) = 0
```

```
d<sup>q</sup> is
 1 def Add(p, z, m, x, n, y):
     r = min(p, m)
                           p = counter \land 0 \le p \le u
     s = min(p, n)
     if (r < s):
      t = p - s
       q = s - r
       for (; r > 0; r--):
       T skip; counter--
       do:
       I q--; counter--
       while (q > 0)
     else:
      t = p - r
28
       q = r - s
       for (; s > 0; s--):
       T skip; counter--
39
       for (; q > 0; q--):
       T skip; counter--
     if (t > 0):
       while (t > 0):
     Tt--; counter--
-- assert counter = 0
```

```
\operatorname{Impact}_{p}^{\natural}(d^{\natural}) = u
Impact<sup>\dagger</sup><sub>m</sub>(d^{\dagger}) \neq 0
Impact<sup>\beta</sup><sub>n</sub>(d<sup>\beta</sup>) = 0
Impact_x^{\natural}(d^{\natural}) = 0
Impact^{\natural}(d^{\natural})
Impact^{\sharp}_{z}(d^{\sharp}) = 0

Assert counter = 0
```

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

```
\operatorname{Impact}_{p}^{\natural}(d^{\natural}) = u
\operatorname{Impact}_{m}^{\sharp}(d^{\sharp}) = 0
\operatorname{Impact}_{n}^{\natural}(d^{\natural}) = 0
Impact^{\natural}_{r}(d^{\natural}) = 0
Impact<sup>\dagger</sup><sub>v</sub>(d^{\dagger}) = 0
Impact^{\sharp}(d^{\sharp}) = 0
```

```
d^{\natural} is
 1 def Add(p, z, m, x, n, y):
     r = min(p, m)
                          p = counter \land 0 \le p \le u
     s = min(p, n)
     if (r < s):
       t = p - s
       q = s - r
       for (; r > 0; r--):
       T skip; counter--
       do:
       I q--; counter--
       while (q > 0)
                             The Add function is safe
     else:
       t = p - r
                           from timing side-channels
       q = r - s
       for (; s > 0; s--):
       T skip; counter--
                                on input variables
39
       for (; q > 0; q--):
       T skip; counter--
     if (t > 0):
                                 m, n, x, y, z
       while (t > 0):
     Tt--; counter--
-- assert counter = 0
```

```
\operatorname{Impact}_{p}^{\natural}(d^{\natural}) = u
\mathrm{Impact}_m^{\natural}(d^{\natural}) = 0
\mathrm{Impact}_n^{\natural}(d^{\natural}) = 0
Impact^{\natural}_{r}(d^{\natural}) = 0
Impact<sup>\dagger</sup><sub>v</sub>(d^{\dagger}) = 0
Impact d^{\dagger}(d^{\dagger}) = 0
```

```
1 def Add(p, z, m, x, n, y):
     r = min(p, m)
                           p = counter \land 0 \le p \le u
     s = min(p, n)
     if (r < s):
       t = p - s
       q = s - r
       for (; r > 0; r--):
       T skip; counter--
       do:
       I q--; counter--
       while (q > 0)
     else:
       t = p - r
       q = r - s
       for (; s > 0; s--):
       T skip; counter--
39
       for (; q > 0; q--):
       T skip; counter--
     if (t > 0):
       while (t > 0):
      Tt--; counter--
     assert counter = 0
```

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

 $d^{\natural}$  is

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 Safe $\Delta _{\mathrm{S}}$	Variables $\Delta$ Numerical $\Delta _{_{ m N}}$	Maybe Dangerous	$_{ m Zero}$
Add	$s_1, s_3, s_5$	$n_2, n_4, n_6$	$s_1$	$s_3, s_5, n_2, n_4, n_4$
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 Bitfield	81	$n_2, n_3, n_4$	81	$n_2, n_3, n_4$
Bitsize	$s_1$ $s_1$	$n_2, n_3, n_4, n_5 \ n_2$	S <sub>1</sub> S <sub>1</sub>	$n_2, n_3, n_4, n_5$ $n_2$
Cdiv	$s_1, s_3$	$n_2, n_4, n_5$	$s_1, s_3$	$n_2, n_4, n_5$
Cdiv_exact	$s_1, s_3$	$n_2, n_4, n_5$	81	$n_2, s_3, n_4, n_5$
Cld	$s_1$	$n_2$	$s_1$	$n_2$
Clz	$s_1$	$n_2$	81	$n_2$
Cmadd Cmnegadd	81,84	$n_2, n_3, n_5$	$s_1, s_4$	$n_2, n_3, n_5$
Cmod	$s_1, s_4$ $s_1$	$n_2,n_3,n_5 \ n_2,n_3$	$s_1, s_4$ $s_1$	$n_2, n_3, n_5 \ n_2, n_3$
Cmul	$s_1, s_4$	$n_2, n_3, n_5$	$s_1, s_4$	$n_2, n_3, n_5$
Coprime	$s_1, s_3$	$n_2,n_4,n_5$	$s_1, s_3$	$n_2, n_4, n_5$
Сору	$s_1, s_3$	$n_2, n_4$	$s_1, s_3$	$n_2, n_4$
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 Copy_row_from_table_32_neon	83 83	$n_1, n_2, n_4$	83 83	$n_1, n_2, n_4$
Copy_row_from_table_8n_neon	83,84	$n_1, n_2, n_4 \ n_1, n_2, n_5$	83,84	$n_1, n_2, n_4 \ n_1, n_2, n_5$
Ctd	81	$n_2$	81	$n_2$
Ctz	81	$n_2$	81	$n_2$
Demont	$s_1$	$n_2,n_3,n_4$	$s_1$	$n_2,n_3,n_4$
Digit	81	$n_2, n_3$	81	$n_2, n_3$
Digitsize Divmod10	$s_1$ $s_1$	$n_2$	81	$n_2$
Emontredc	81	$n_2 \ n_2, n_3, n_4$	$s_1 \\ s_1$	$n_2, n_3, n_4$
Eq	$s_1, s_3$	$n_2, n_4$	$s_1, s_3$	$n_2, n_4$
Even	$s_1$	$n_2$	1	$s_1, n_2$
Ge	$s_1, s_3$	$n_2, n_4$	$s_1, s_3$	$n_2, n_4$
Gt Iszero	$s_1, s_3$	$n_2, n_4$	$s_1, s_3$	$n_2, n_4$
Le	$s_1, s_3$	$n_2 \ n_2, n_4$	$s_1$ $s_1, s_3$	$n_2 \atop n_2, n_4$
Lt	81,83	$n_2, n_4$	81,83	$n_2, n_4$
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 Modinv	$s_1$ $s_1$	$n_2, n_3, n_4$	8 <sub>1</sub> 8 <sub>1</sub>	$n_2, n_3, n_4$
Modoptneg	81	$n_2, n_3, n_4, n_5 \ n_2, n_3, n_4, n_5$	$s_1$	$n_2, n_3, n_4, n_5 \ n_2, n_3, n_4, n_5$
Modsub	$s_1$	$n_2, n_3, n_4, n_5$	$s_1$	$n_2, n_3, n_4, n_5$
Montifier	$s_1$	$n_2, n_3, n_4$	81	$n_2, n_3, n_4$
Montmul	$s_1$	$n_2, n_3, n_4, n_5$	$s_1$	$n_2, n_3, n_4, n_5$
Montredc Montsqr	$s_1, s_3, s_6$	$n_2, n_4, n_5$	$s_1, s_3, s_6$	$n_2, n_4, n_5$
Mul	$s_1, s_3, s_5$	$n_2, n_3, n_4 \ n_2, n_4, n_6$	$s_1, s_3, s_5$	$n_2, n_3, n_4 \ n_2, n_4, n_6$
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	$s_1$	$n_2,n_3,n_4$	$s_1$	$n_2, n_3, n_4$
Wegmodinv Wonzero	81	$n_2, n_3$	81	$n_2, n_3$
Nonzero Normalize	$s_1$ $s_1$	$n_2 \\ n_2$	$s_1 \\ s_1$	$n_2 \\ n_2$
Odd	$s_1$	$n_2$	01	$s_1, n_2$
Of_word	$s_1$	$n_2, n_3$	$s_1$	$n_2, n_3$
Optadd	81	$n_2, n_3, n_4, n_5$	81	$n_2, n_3, n_4, n_5$
Optneg	$s_1$	$n_2, n_3, n_4$	81	$n_2, n_3, n_4$
Dptsub Optsubadd	$s_1$ $s_1$	$n_2, n_3, n_4, n_5 \ n_2, n_3, n_4, n_5$	8 <sub>1</sub> 8 <sub>1</sub>	$n_2, n_3, n_4, n_5 \ n_2, n_3, n_4, n_5$
Pow2	$s_1$	$n_2, n_3, n_4, n_5$	$s_1$	$n_2, n_3, n_4, n_5$
Shl_small	$s_1, s_3$	$n_2, n_4, n_5$	$s_1, s_3$	$n_2, n_4, n_5$
Shr_small	$s_1, s_3$	$n_2, n_4, n_5$	81	$s_3, n_2, n_4, n_5$
Sqr	$s_1, s_3$	$n_2,n_4$	$s_1, s_3$	$n_2, n_4$
Sub	$s_1, s_3, s_5$	$n_2, n_4, n_6$	$s_1$	$s_3, s_5, n_2, n_4, n_4, n_5$
Nord_bytereverse Nord_clz		$n_1 \\ n_1$		$n_1 \\ n_1$
Word_ctz		$n_1$		$\stackrel{n_1}{n_1}$
Nord_divstep59		$n_1, n_2, n_3, n_4$		$n_1, n_2, n_3, n_4$
Nord_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
	00	-10		101

#### Conclusion