Quantitative Static Timing Analysis

Denis Mazzucato, Marco Campion, and Caterina Urban

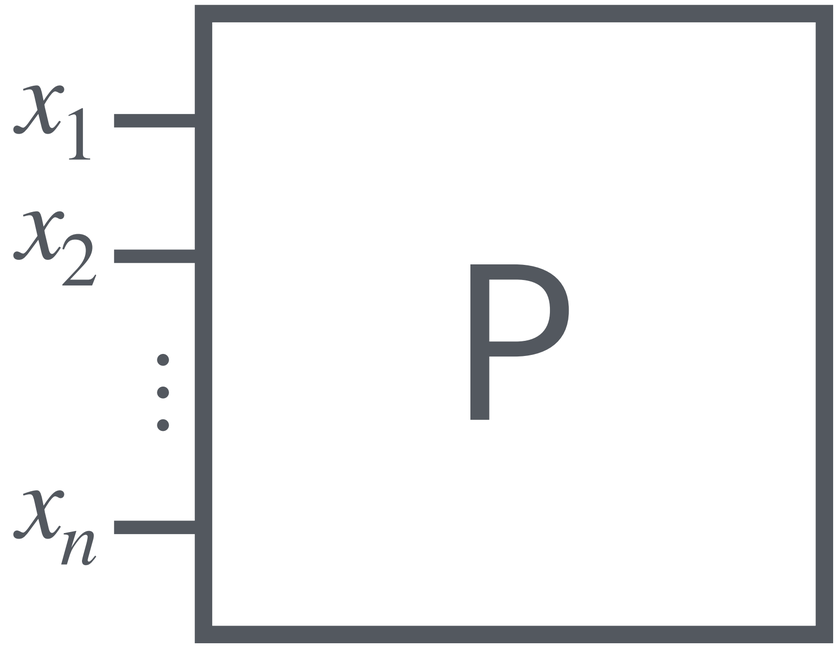
21th October 2024

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

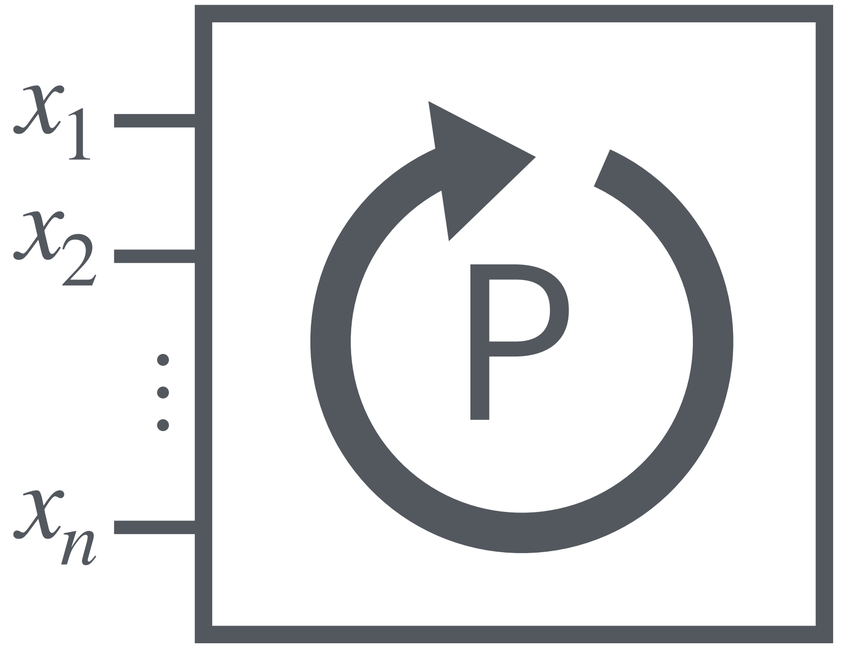
Quantifying the impact of input variables on the number of iterations of a program

Input variables



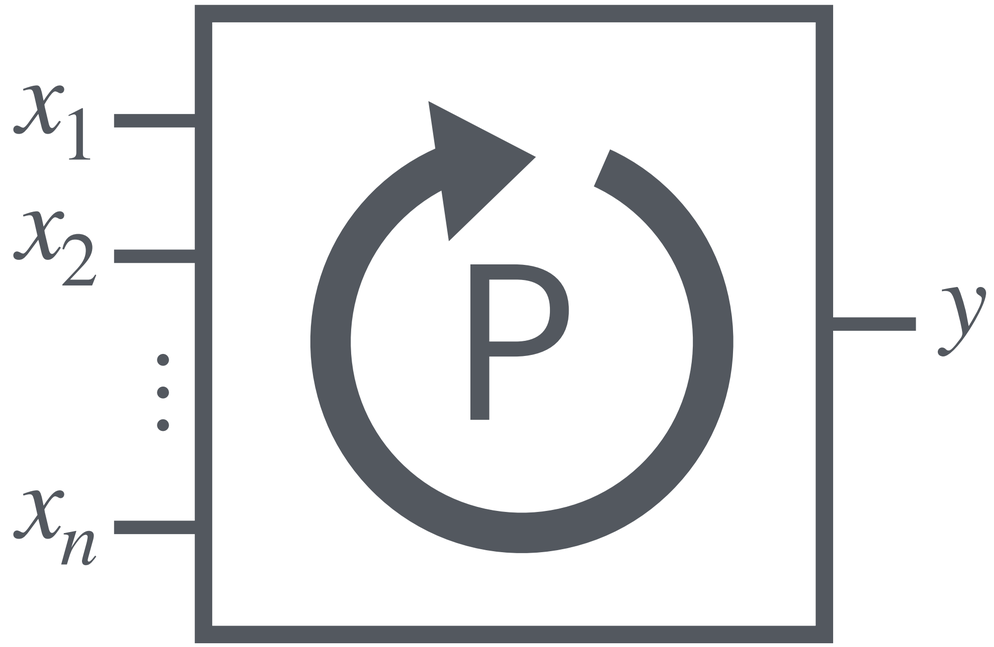
Quantifying the impact of input variables on the number of iterations of a program

Input variables



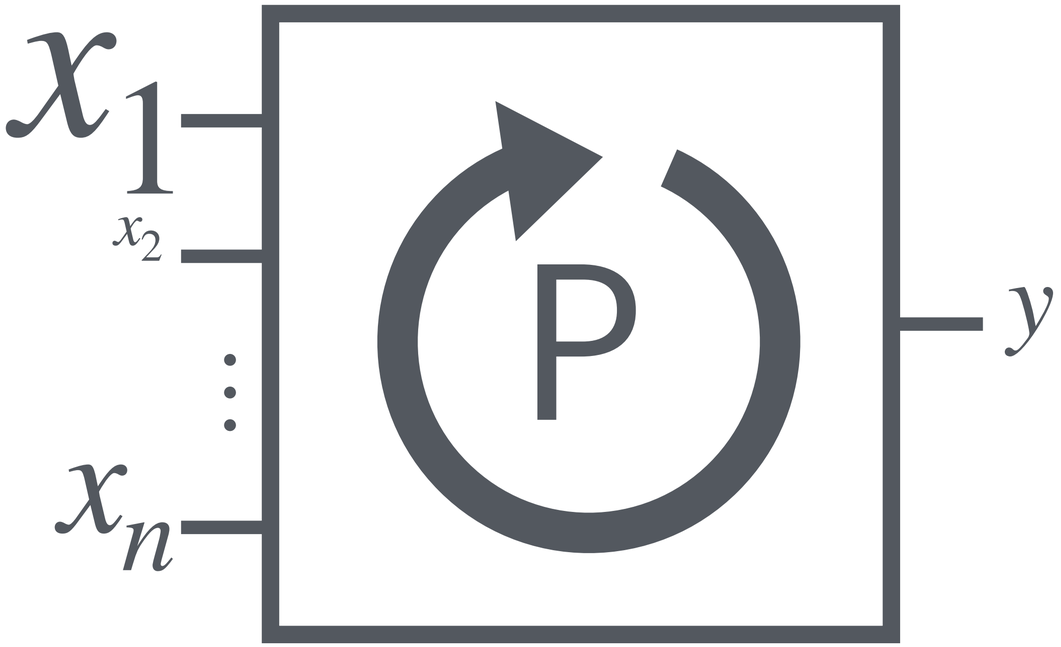
Quantifying the impact of input variables on the number of iterations of a program

Input variables



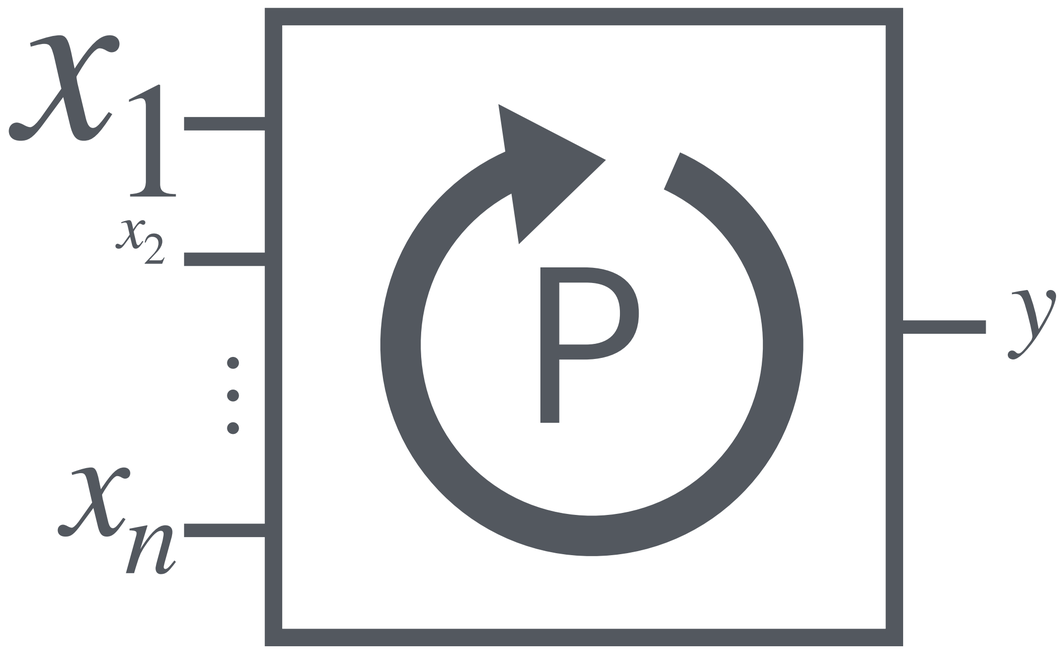
Quantifying the impact of input variables on the number of iterations of a program

How much impact on iterations?

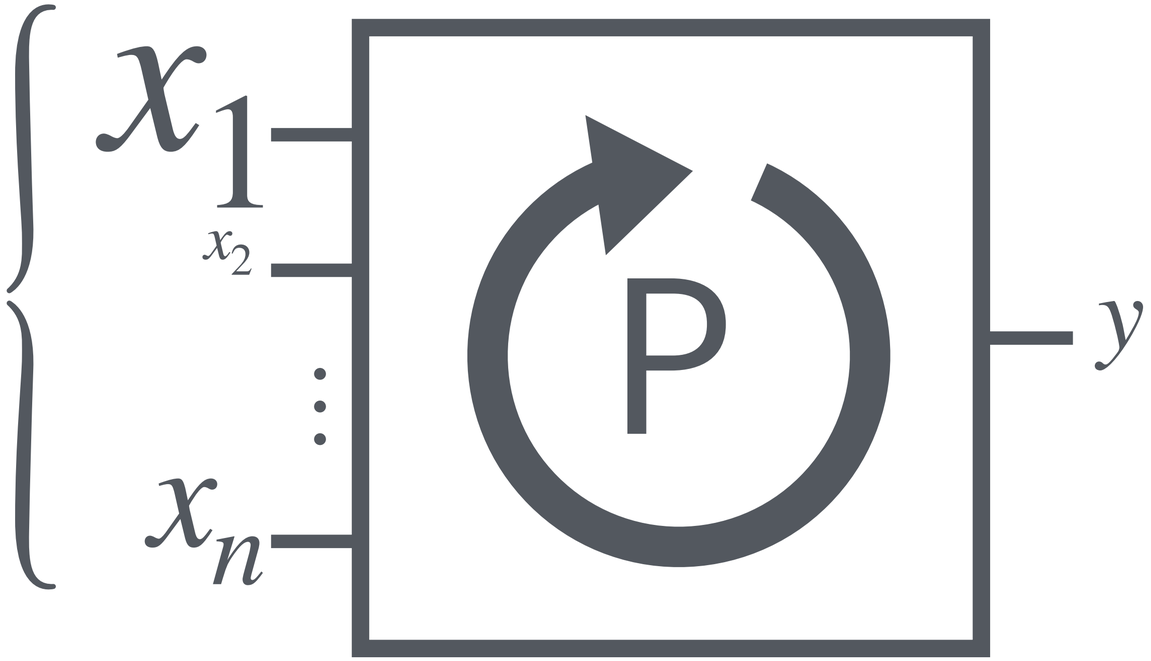


Quantifying the impact of input variables on the number of iterations of a program

$$\textsc{Impact}_{x_2}(P) \leq \cdots \leq \textsc{Impact}_{x_n}(P) \leq \textsc{Impact}_{x_1}(P)$$

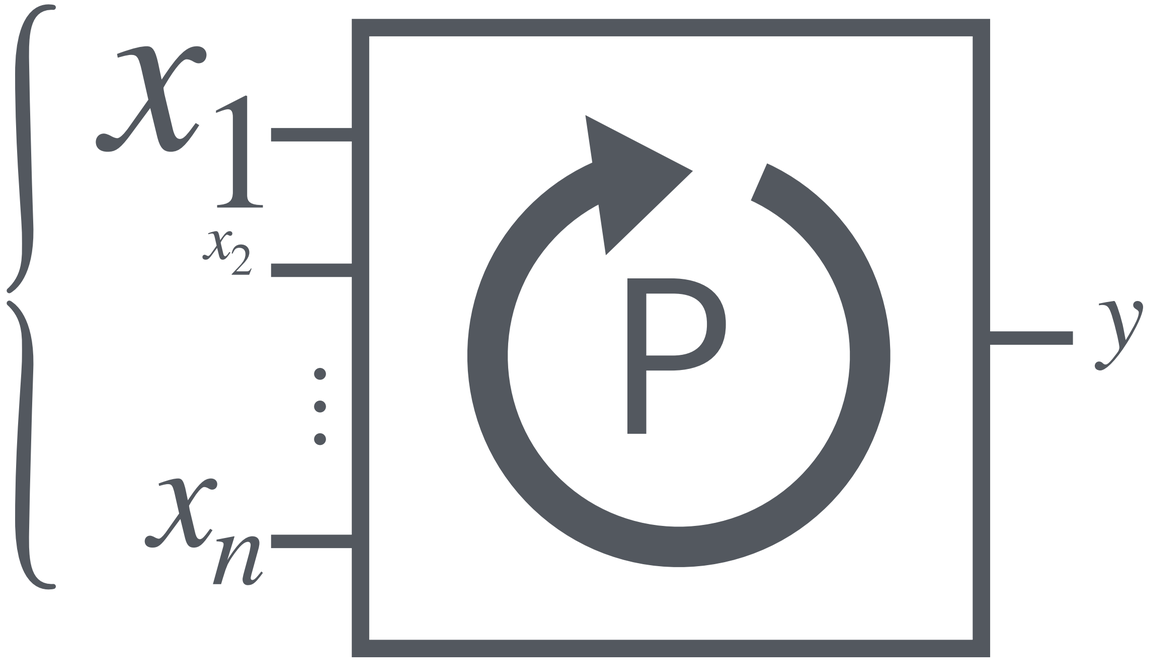


Why?



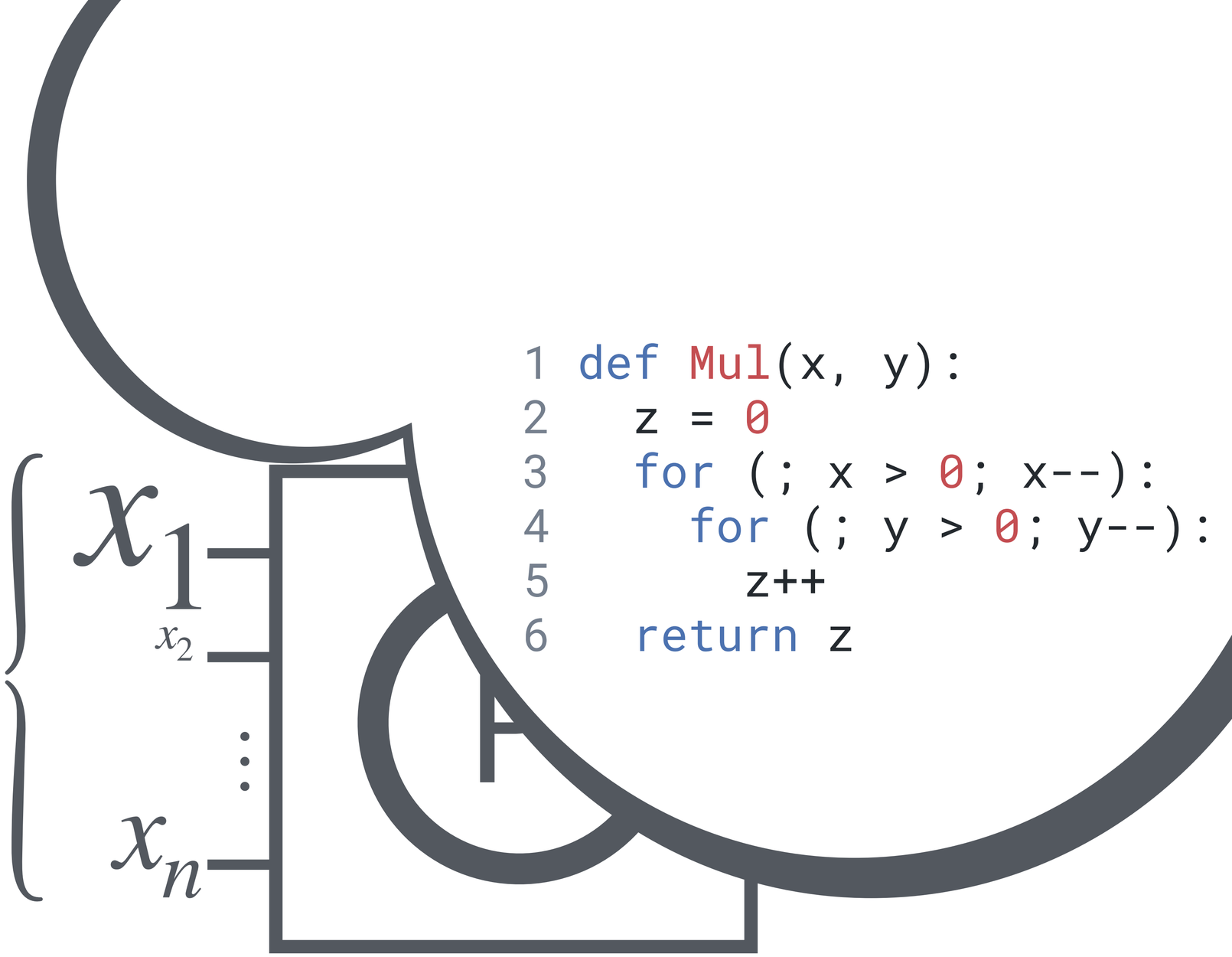
Why?

- Correctness



Why?

- Correctness

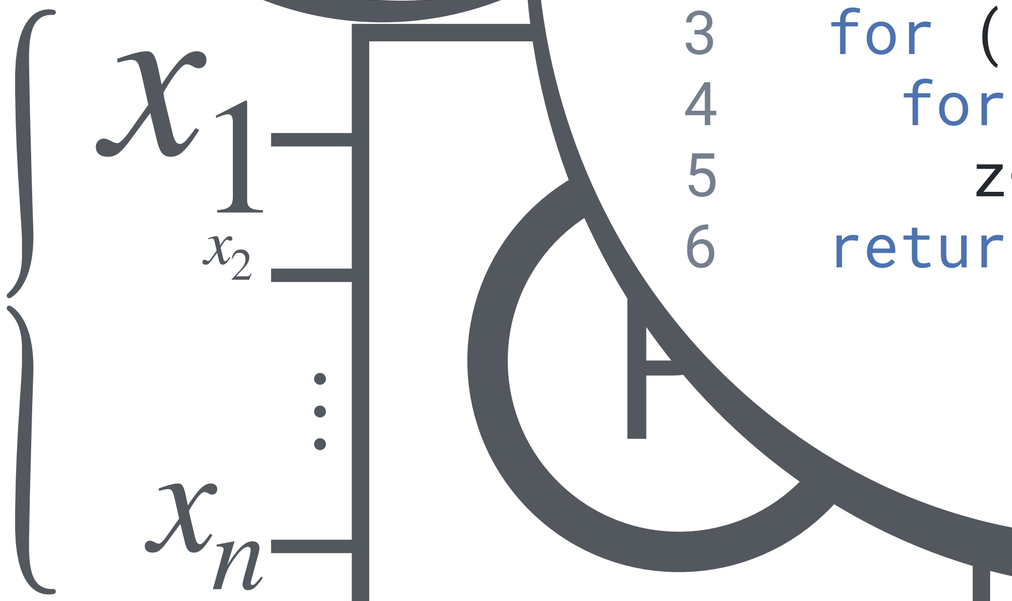



```
1 def Mul(x, y):
2   z = 0
3   for (; x > 0; x--):
4     for (; y > 0; y--):
5       z++
6   return z
```

Why?

- Correctness

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

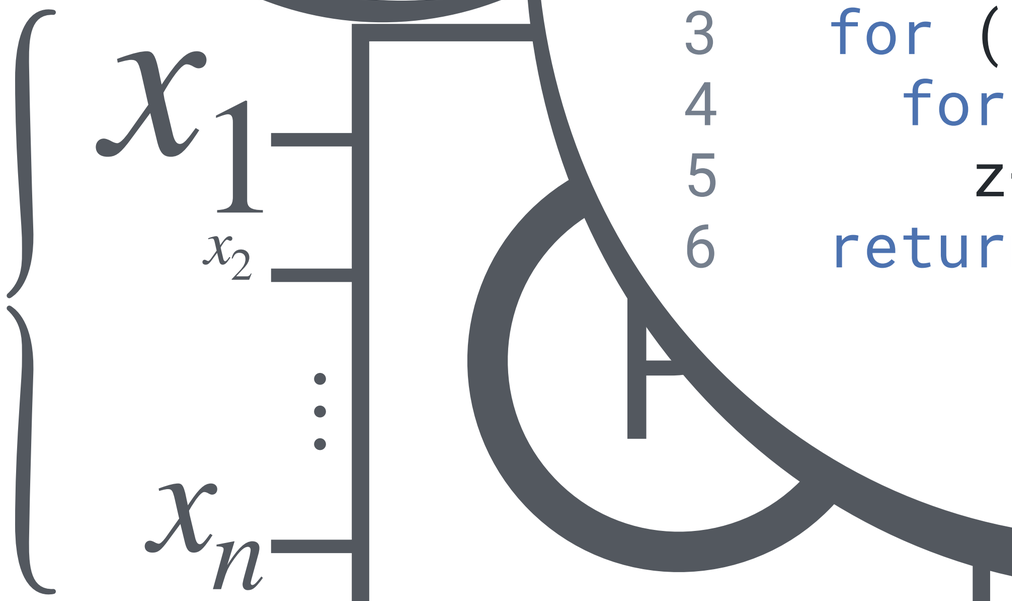



```
1 def Mul(x, y):
2   z = 0
3   for (; x > 0; x--):
4     for (; y > 0; y--):
5       z++
6   return z
```

Why?

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

- Correctness

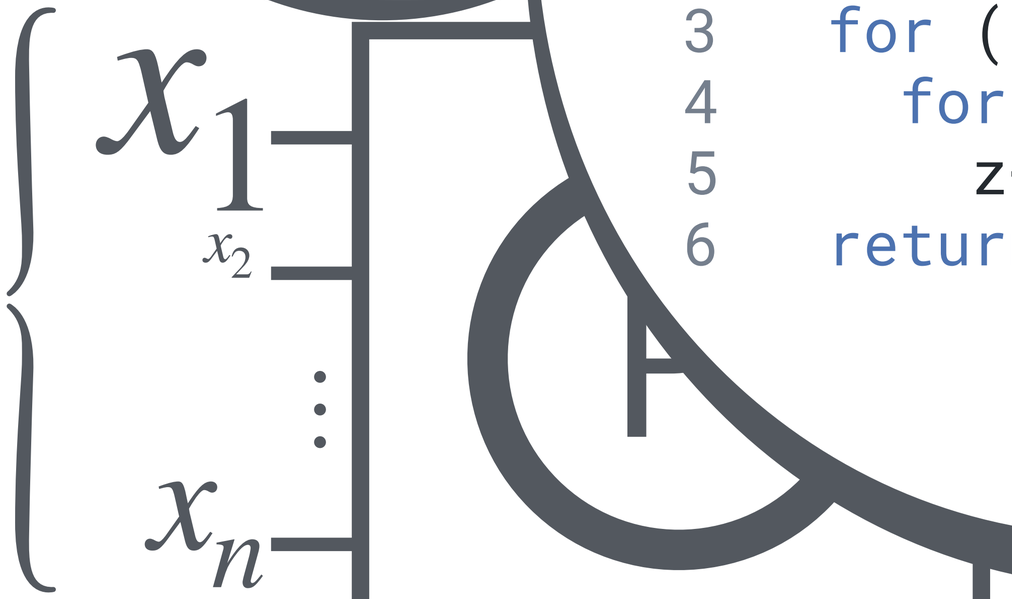



```
1 def Mul(x, y):
2   z = 0
3   for (; x > 0; x--):
4     for (; y > 0; y--):
5       z++
6   return z
```

Why?

- Correctness

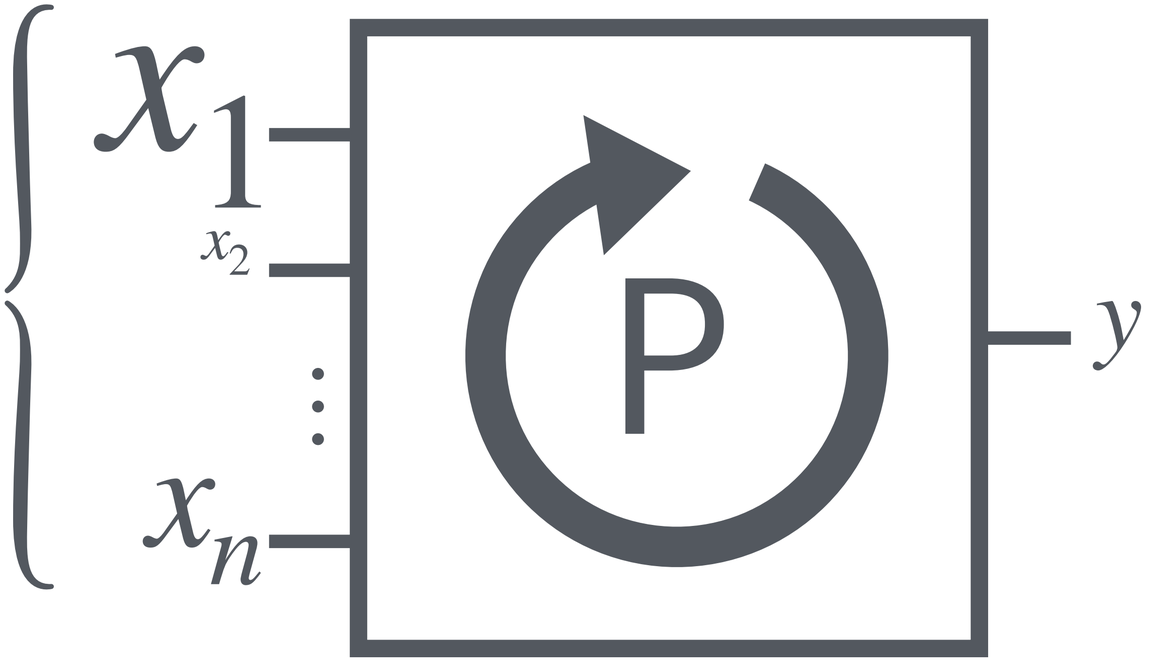
$$\text{IMPACT}_x(\texttt{Mul}) = 2 \cdot \text{IMPACT}_y(\texttt{Mul})$$




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

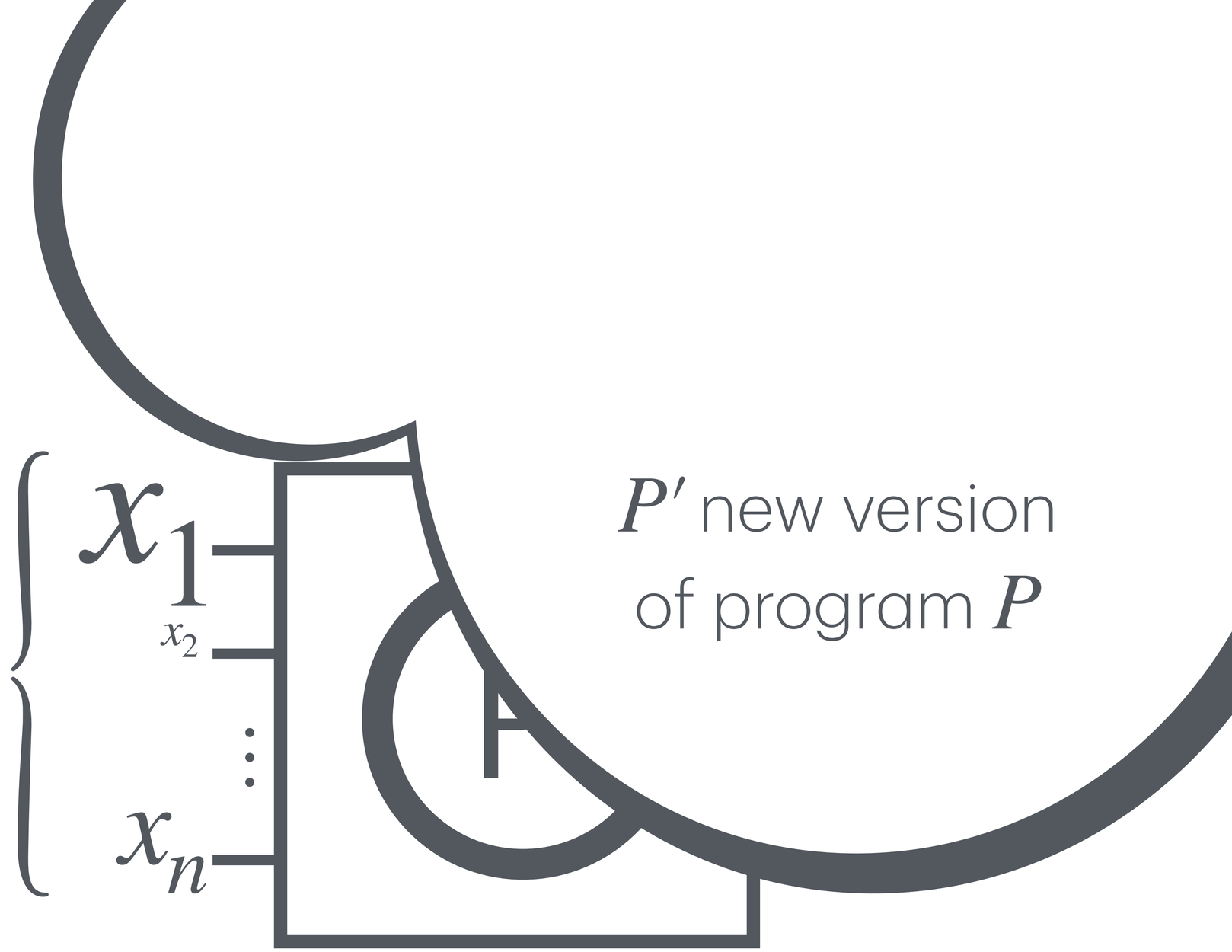
Why?

- Correctness
- Performance



Why?

- Correctness
- Performance

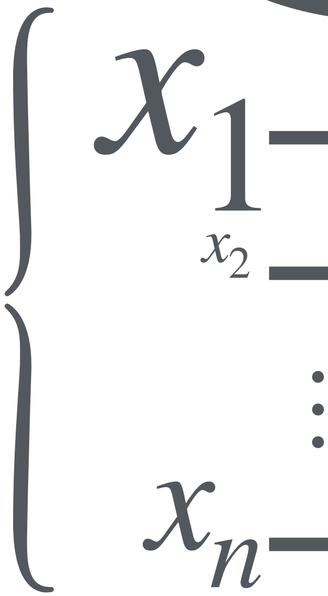


P' new version of program P

Why?

- Correctness
- Performance

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

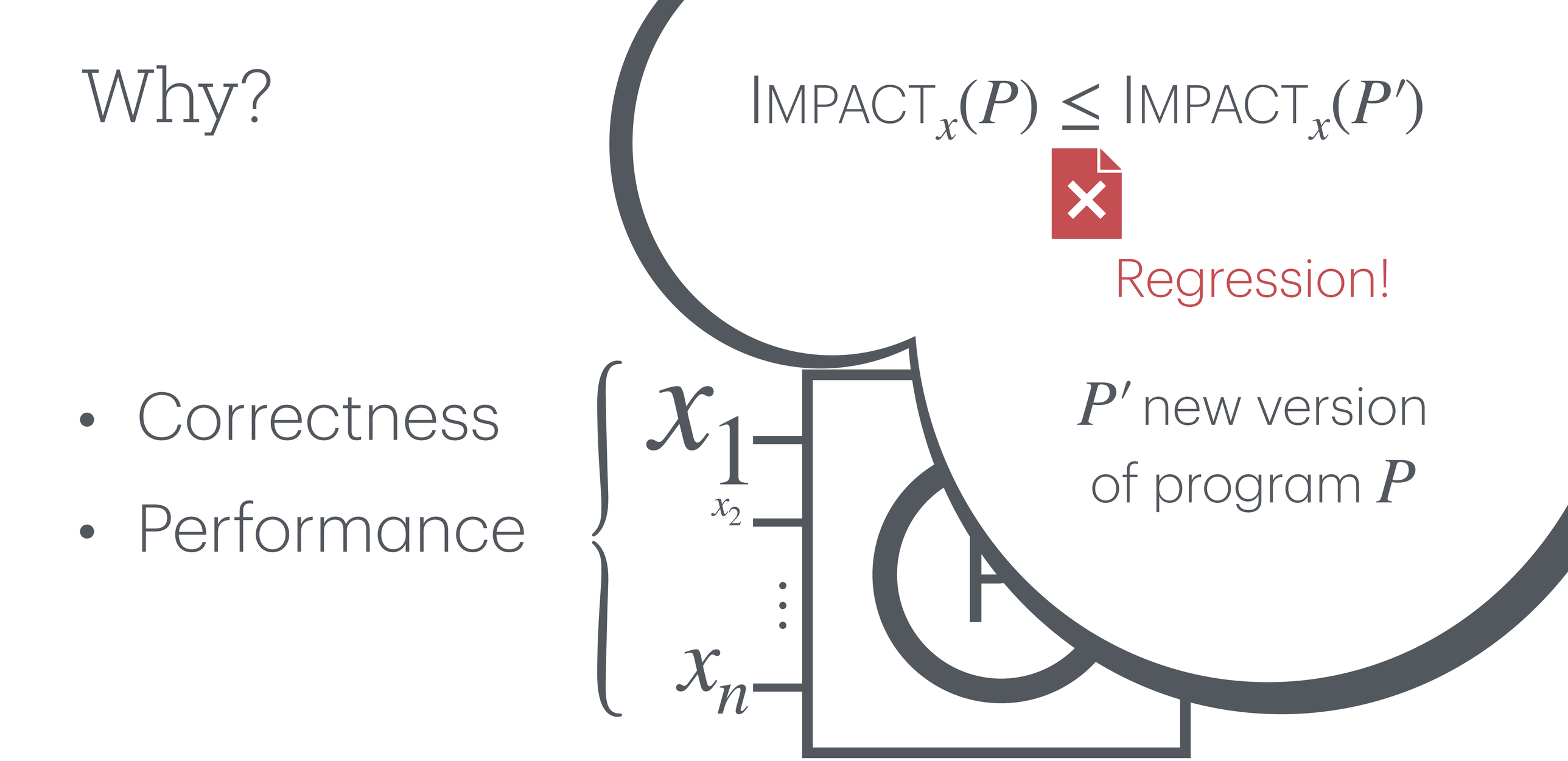


P' new version of program P

Why?

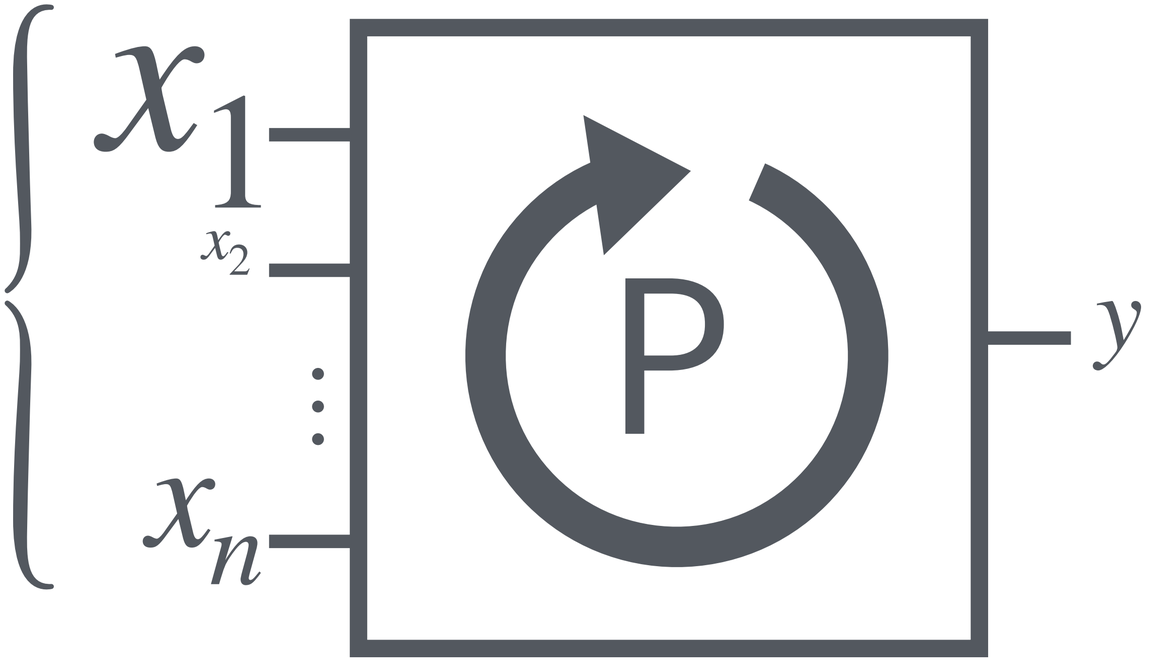
- Correctness
- Performance

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



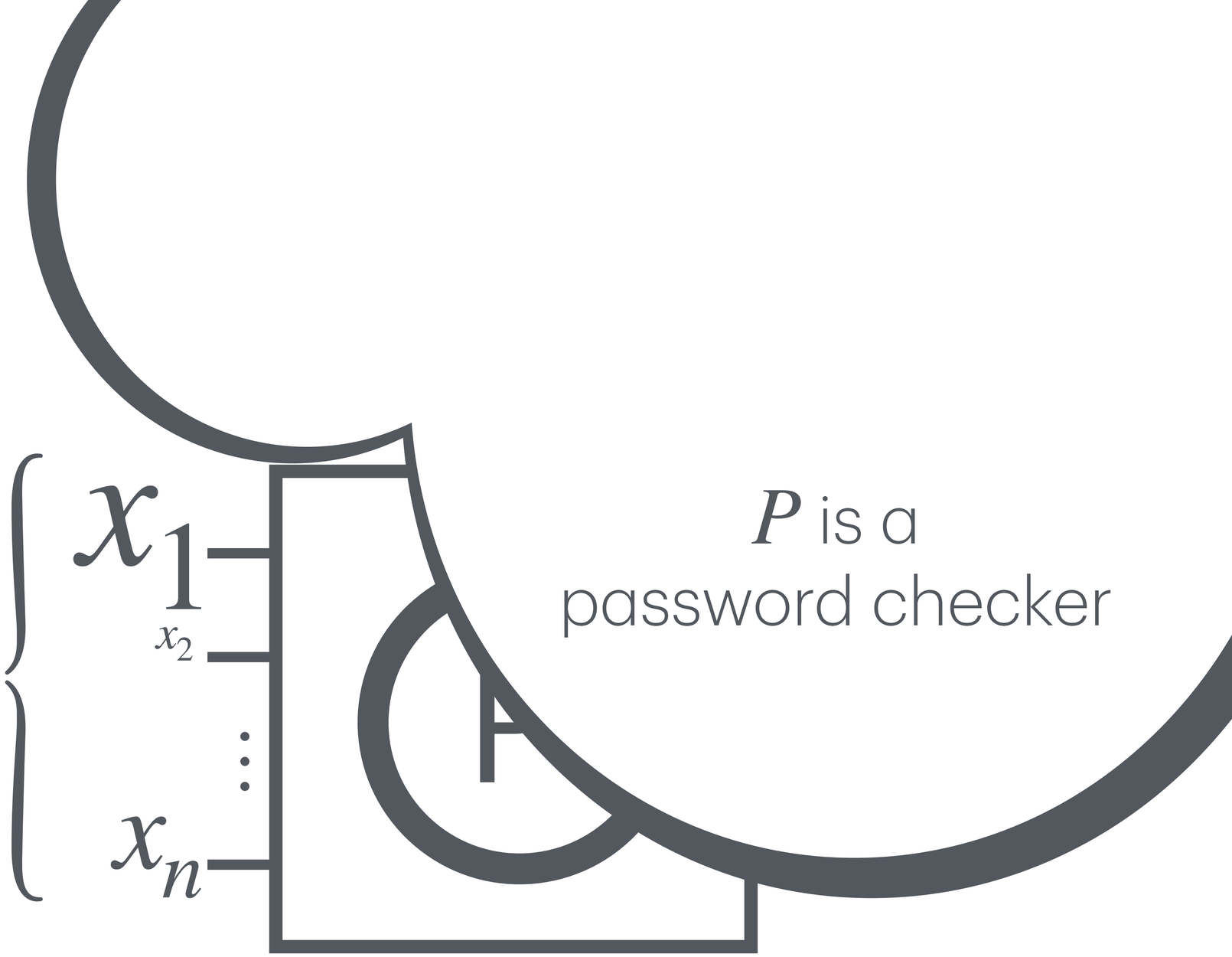
Why?

- Correctness
- Performance
- Security



Why?

- Correctness
- Performance
- Security

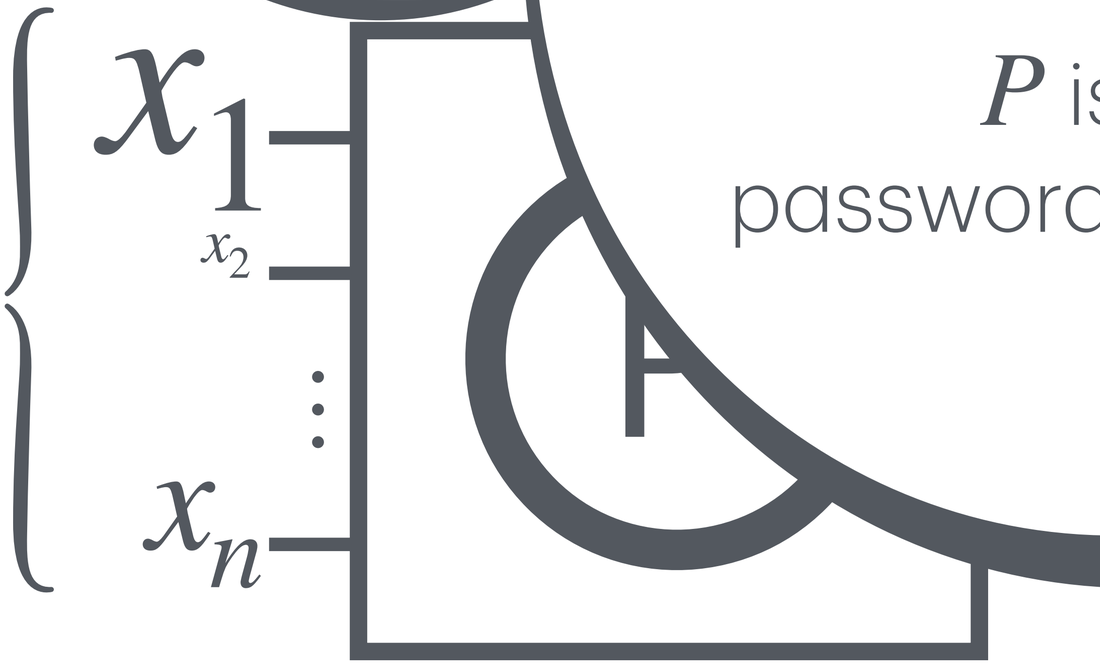


P is a
password checker

Why?

- Correctness
- Performance
- Security

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

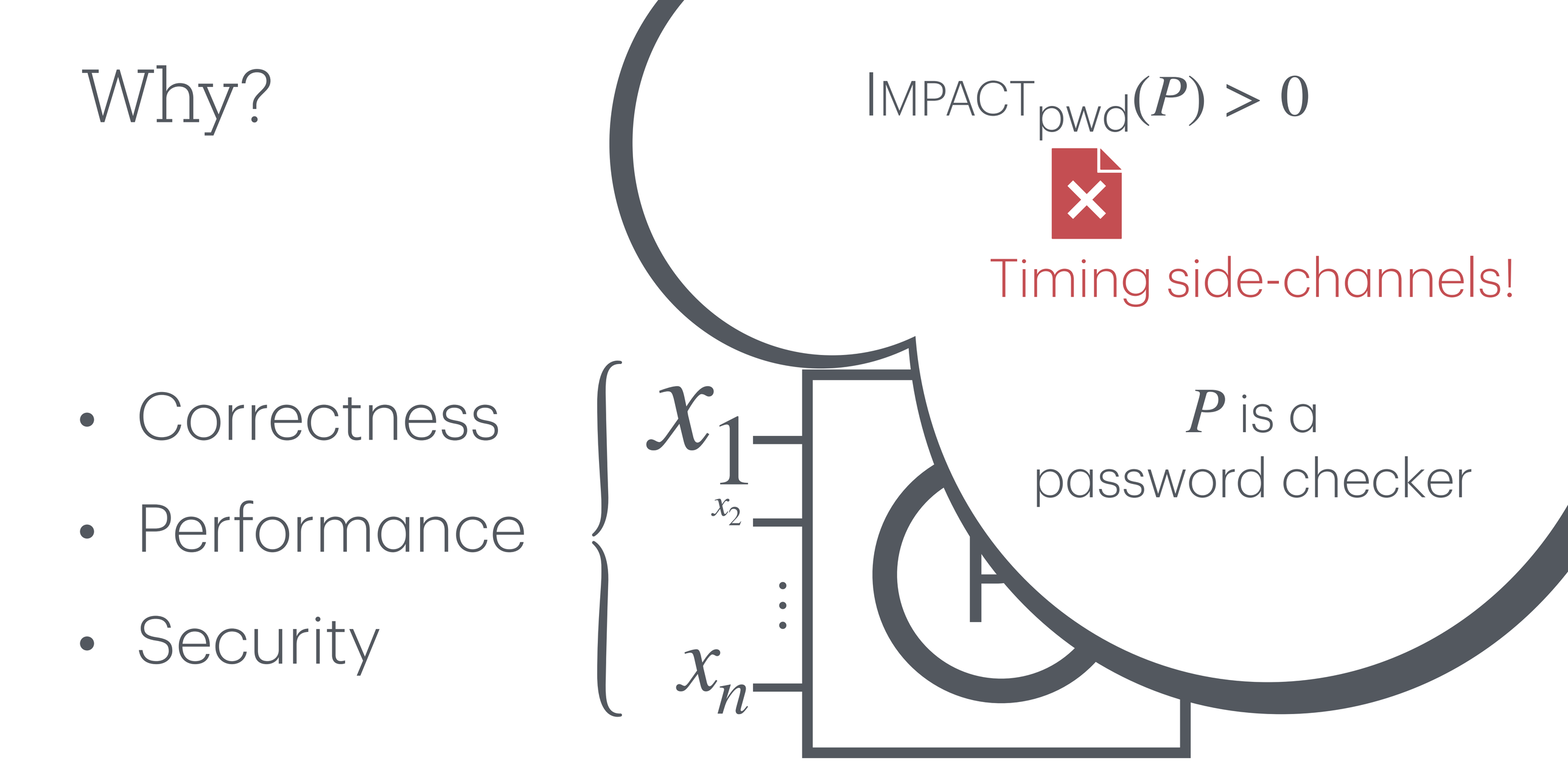


P is a
password checker

Why?

- Correctness
- Performance
- Security

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



Timing side-channels!

P is a password checker

Timing Side-Channels Attacks

Learning a secret based on observations of the execution time of a program

Timing Side-Channels Attacks

Learning a **secret** based on observations of the **execution time** of a program

```
1  def Check(pwd):
2    for (i = 0; i < len; i++):
3      if pwd[i] != secret[i]:
4        return false
5    return true
```

Timing Side-Channels Attacks

Learning a **secret** based on observations of the **execution time** of a program

```
1  def Check(pwd):                          secret=1234
2    for (i = 0; i < len; i++):
3      if pwd[i] != secret[i]:
4        return false
5    return true
```

Timing Side-Channels Attacks

Learning a **secret** based on observations of the **execution time** of a program

```
1  def Check(pwd):
2    for (i = 0; i < len; i++):
3      if pwd[i] != secret[i]:
4        return false
5    return true
```

```
pwd    secret=1234
0000
```

Timing Side-Channels Attacks

Learning a **secret** based on observations of the **execution time** of a program

```
1  def Check(pwd):
2    for (i = 0; i < len; i++):
3      if pwd[i] != secret[i]:
4        return false
5    return true
```

pwd secret=1234

0000 → 1 Iter.

Timing Side-Channels Attacks

Learning a **secret** based on observations of the **execution time** of a program

```
1  def Check(pwd):
2    for (i = 0; i < len; i++):
3      if pwd[i] != secret[i]:
4        return false
5    return true
```

```
pwd    secret=1234
0000 → 1 Iter. → false
```

Timing Side-Channels Attacks

Learning a **secret** based on observations of the **execution time** of a program

```
1  def Check(pwd):
2    for (i = 0; i < len; i++):
3      if pwd[i] != secret[i]:
4        return false
5    return true
```

```
pwd    secret=1234
0000 → 1 Iter. → false
1111 → 1 Iter. → 2 Iter. → false
```

Timing Side-Channels Attacks

Learning a **secret** based on observations of the **execution time** of a program

```
1  def Check(pwd):
2    for (i = 0; i < len; i++):
3      if pwd[i] != secret[i]:
4        return false
5    return true
```

```
pwd    secret=1234
0000 → 1 Iter. → false
1111 → 1 Iter. → 2 Iter. → false
1222 → 1 Iter. → 2 Iter. → 3 Iter. → false
```

Timing Side-Channels Attacks

Learning a **secret** based on observations of the **execution time** of a program

```
1  def Check(pwd):
2    for (i = 0; i < len; i++):
3      if pwd[i] != secret[i]:
4        return false
5    return true
```

pwd secret=1234

0000 → 1 Iter. → false

1111 → 1 Iter. → 2 Iter. → false

1222 → 1 Iter. → 2 Iter. → 3 Iter. → false

1233 → 1 Iter. → 2 Iter. → 3 Iter. → 4 Iter. → false

Timing Side-Channels Attacks

Learning a **secret** based on observations of the **execution time** of a program

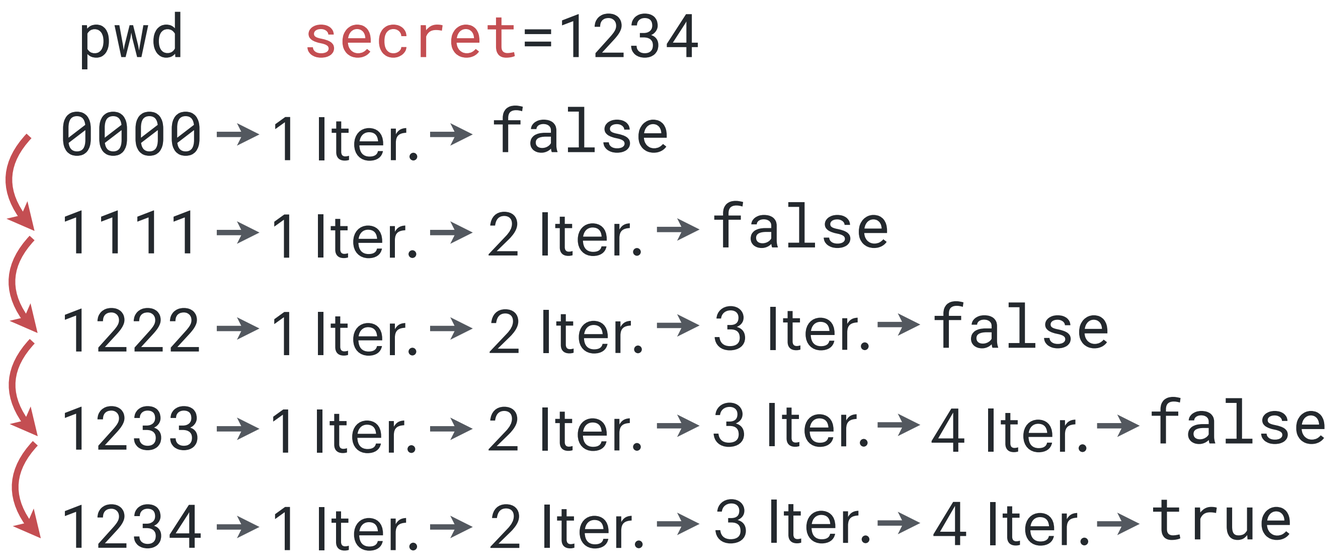
```
1  def Check(pwd):
2    for (i = 0; i < len; i++):
3      if pwd[i] != secret[i]:
4        return false
5    return true
```

```
pwd    secret=1234
0000 → 1 Iter. → false
1111 → 1 Iter. → 2 Iter. → false
1222 → 1 Iter. → 2 Iter. → 3 Iter. → false
1233 → 1 Iter. → 2 Iter. → 3 Iter. → 4 Iter. → false
1234 → 1 Iter. → 2 Iter. → 3 Iter. → 4 Iter. → true
```

Timing Side-Channels Attacks

Learning a **secret** based on observations of the **execution time** of a program

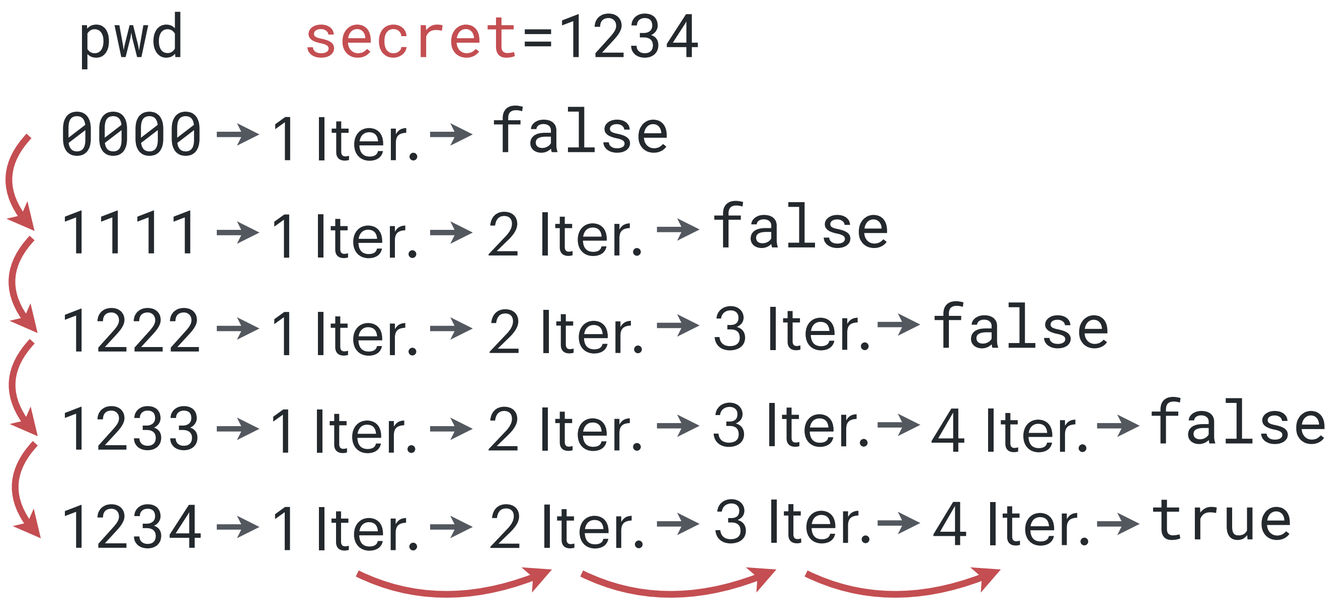
```
1  def Check(pwd):
2    for (i = 0; i < len; i++):
3      if pwd[i] != secret[i]:
4        return false
5    return true
```



Timing Side-Channels Attacks

Learning a **secret** based on observations of the **execution time** of a program

```
1  def Check(pwd):
2    for (i = 0; i < len; i++):
3      if pwd[i] != secret[i]:
4        return false
5    return true
```



Timing Side-Channels Attacks

Learning a **secret** based on observations of the **execution time** of a program

```
1  def Check(pwd):
2    for (i = 0; i < len; i++):
3      if pwd[i] != secret[i]:
4        return false
5    return true
```

if $\textsc{Impact}_{pwd}(\texttt{Check}) > 0$

then an attacker could retrieve the secret!

S2N-Bignum

S2N-Bignum

Add Function

S2N-Bignum

Add Function

3 8
  4

S2N-Bignum

Add Function

$$
\begin{array}{rrr}
 & 4 & 2 \quad = \\
\hline
 & 3 & 8 \quad + \\
 & & 4
\end{array}
$$

S2N-Bignum

Add Function

array

$$\frac{[0, 4, 2] =}{[3, 8] + [4]}$$

S2N-Bignum

Add Function

length	array
3	[0, 4, 2] =
2	[3, 8] +
1	[4]

S2N-Bignum

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
```

S2N-Bignum

Add Function

$$\mathbf{Add}\begin{pmatrix} 3, & z, \\ 2, & [3,8], \\ 1, & [4] \end{pmatrix}$$

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

S2N-Bignum

Add Function

$$\mathtt{Add}\begin{pmatrix} 3, & z, \\ 2, & [3, 8], \\ 1, & [4] \end{pmatrix}$$

size data

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

S2N-Bignum

Add Function

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

size data

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

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

How?

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

How?

abstract

concrete

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

How?

abstract

concrete

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

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

S2N-Bignum

Add Function

```
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
```

S2N-Bignum

Add Function

```
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
```

S2N-Bignum

Add Function

```
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
```

S2N-Bignum

Add Function

```
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
```

S2N-Bignum

Add Function

```
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
```

S2N-Bignum

Add Function

```
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
```

S2N-Bignum

Add Function

```
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
```

S2N-Bignum

Add Function

```
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
```

S2N-Bignum

Add Function

Syntactic Dependency Analysis

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

```
1  def Add(p, z, m, x, n, y):
2    r = min(p, m)
3    s = min(p, n)
4    if (r < s):
5      t = p - s
6      q = s - r
7      i = 0
8      a = 0
9      for (; r > 0; r--):
10       s = x[i]
11       w = y[i]
12       z[i] = s + w + a
13       i = i + 1
14       a = (w < a) ||
15         (s + w < s) ||
16         (s + w + a < s)
17     do:
18       r = y[i]
19       b = (r < a) ||
20         (r + a < r)
21       z[i] = r + a
22       i = i + 1
23       q--
24       a = b
25     while (q > 0)
```

```
26   else:
27     t = p - r
28     q = r - s
29     i = 0
30     b = 0
31     for (; s > 0; s--):
32       r = x[i]
33       w = y[i]
34       z[i] = r + w + b
35       i = i + 1
36       b = (w < b) ||
37         (r + w < r) ||
38         (r + w + b < r)
39     for (; q > 0; q--):
40       r = x[i]
41       z[i] = r + b
42       i = i + 1
43       b = (r < b) ||
44         (r + b < r)
45   if (t > 0):
46     z[i] = b
47     while (t > 0):
48       i = i + 1
49       t--
50       if (t > 0):
51         z[i] = 0
```

S2N-Bignum

Add Function

Syntactic Dependency Analysis

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

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

S2N-Bignum

Add Function

```
 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--
```

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

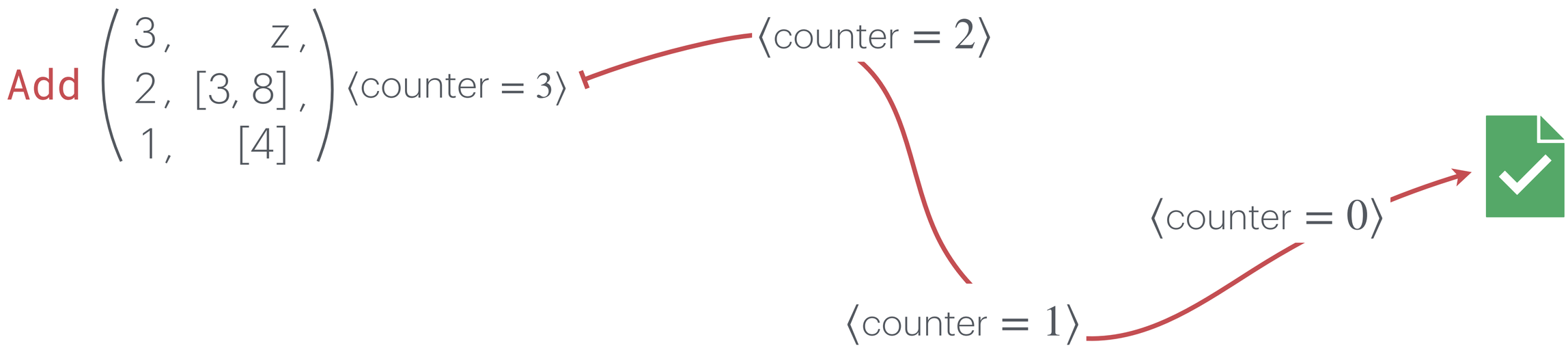
S2N-Bignum

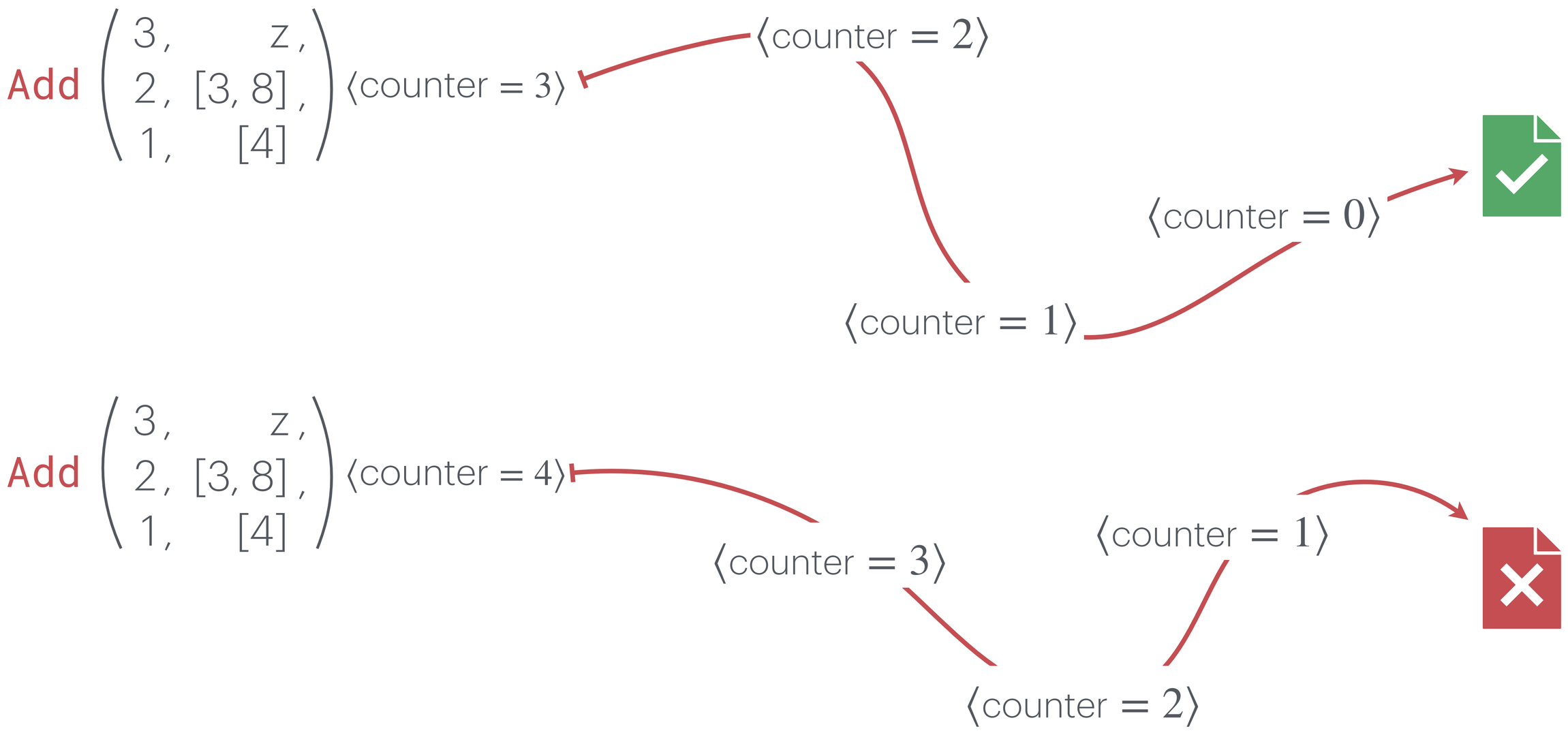
Add Function

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

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

Backwards starting from zero!


Add
3, z,
2, [3, 8],
1, [4]
⟨counter = 3⟩
⟨counter = 2⟩
⟨counter = 1⟩
⟨counter = 0⟩


Add (3, z, 2, [3, 8], 1, [4])
⟨counter = 3⟩
⟨counter = 2⟩
⟨counter = 1⟩
⟨counter = 0⟩
Add (3, z, 2, [3, 8], 1, [4])
⟨counter = 4⟩
⟨counter = 3⟩
⟨counter = 2⟩
⟨counter = 1⟩

S2N-Bignum

Add Function

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

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

Backwards starting from zero!

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

S2N-Bignum

Add Function

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

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

Backwards starting from zero!

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

Backward abstract analysis

S2N-Bignum

Add Function

Abstract invariant on the

input variables + counter

```
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
```

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

S2N-Bignum

Add Function

Abstract invariant on the

input variables + counter

```
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
```

Forward +

Backward abstract analysis

Augment each loop body with a counter for iterations

Backwards starting from zero!

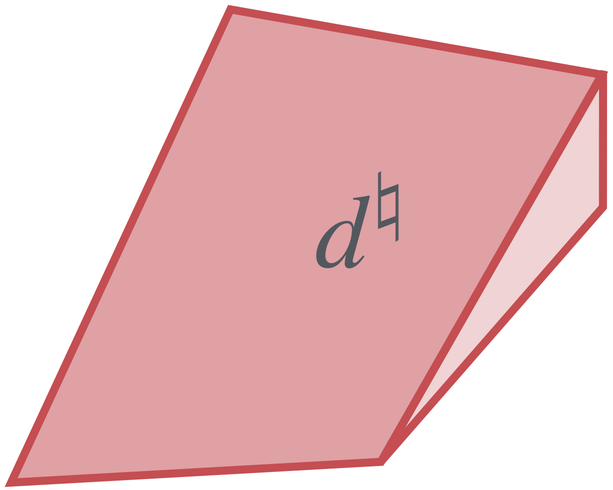
At the beginning, the counter yields the global number of iterations

Impact Quantification

Mixed-Integer Linear Programming

Abstract invariant on the

input variables + counter



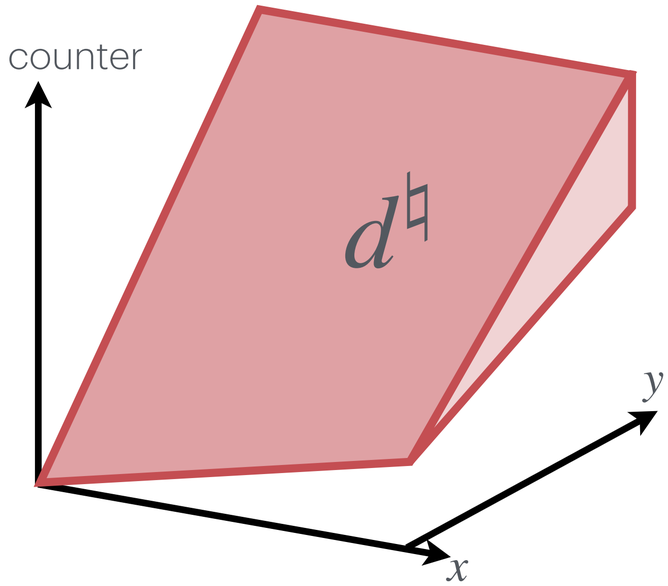
Impact Quantification

Mixed-Integer Linear Programming

Abstract invariant on the

input variables + counter

(i) Assume input variable of interest x



Impact Quantification

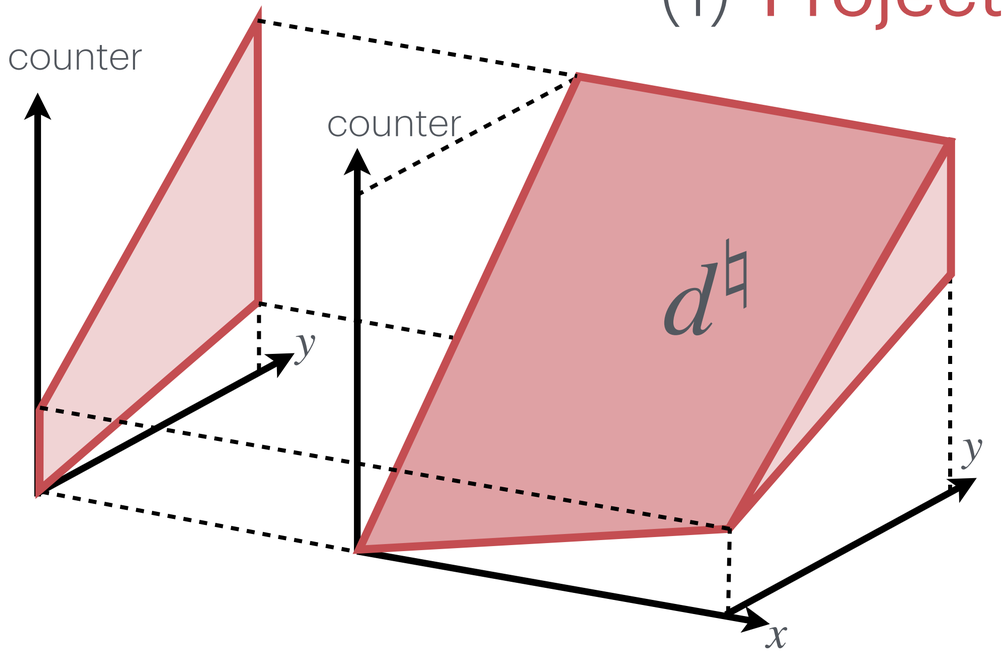
Mixed-Integer Linear Programming

Abstract invariant on the

input variables + counter

(i) Assume input variable of interest x

(1) Project away x

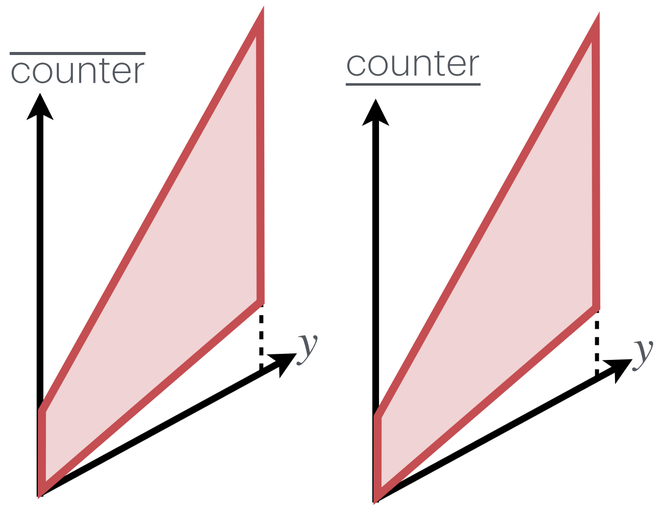


Impact Quantification

Mixed-Integer Linear Programming

Abstract invariant on the

input variables + counter



(i) Assume input variable of interest x

(1) Project away x

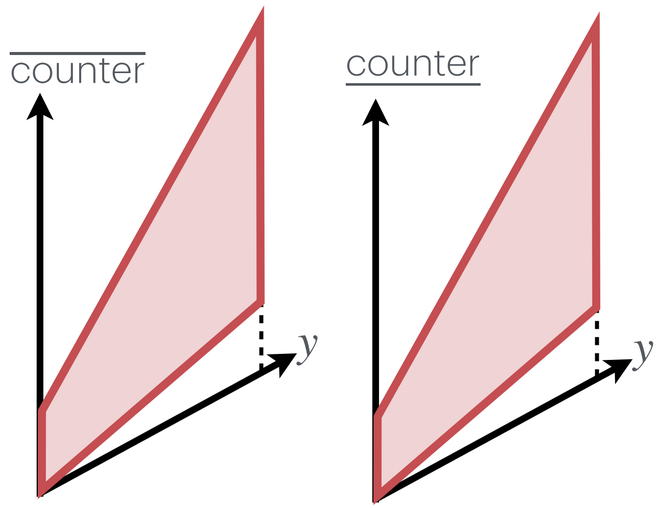
(2) Duplicate the invariant and substitute the counter with $\underline{\text{counter}}$ and $\overline{\text{counter}}$

Impact Quantification

Mixed-Integer Linear Programming

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 $\overline{\text{counter}}$

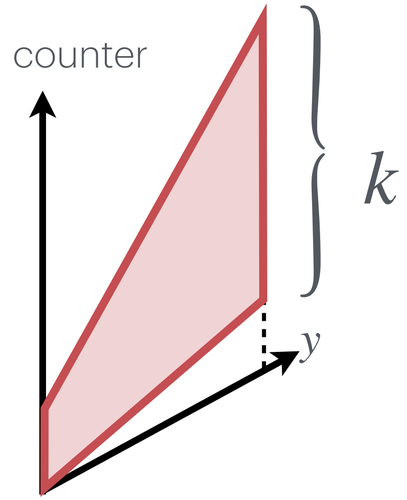
(3) Maximize the distance between the two

Impact Quantification

Mixed-Integer Linear Programming

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

Impact Quantification

Mixed-Integer Linear Programming

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

Impact Quantification

Mixed-Integer Linear Programming

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

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

Impact Quantification

Mixed-Integer Linear Programming

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

$$\mathrm{Project}_x(d^{\lightning}))$$

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

Impact Quantification

Mixed-Integer Linear Programming

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

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

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

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

abstract

concrete

Theorem 3 (Soundness of Range$^{\lightning}$)

$$\mathrm{Impact}^{\lightning}_{x}(\mathsf{P}) \leq k \implies$$

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

Lemma 3

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

S2N-Bignum

Add Function

```
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
```

$d^\natural$ is

$p = counter \wedge 0 \leq p \leq u$

Forward +

Backward abstract analysis

S2N-Bignum

Add Function

$d^\natural$ is

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

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

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

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

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

S2N-Bignum

Add Function

$d^{\natural}$ is

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

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

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

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

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

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

S2N-Bignum

Add Function

$d^{\natural}$ is

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

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

S2N-Bignum

Add Function

$d^\natural$ is

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

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

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

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

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

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

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

S2N-Bignum

Add Function

$d^\natural$ is

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

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

S2N-Bignum

Add Function

$d^{\natural}$ is

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

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

S2N-Bignum

Add Function

$d^{\natural}$ is

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

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

$$\left.\begin{array}{l} \max k \text{ subject to} \\ \quad \mathsf{p} = \overline{\mathsf{counter}} \wedge \\ \quad \mathsf{p} = \underline{\mathsf{counter}} \wedge \\ \quad 0 \leq \mathsf{p} \leq u \wedge \\ \quad 0 \leq k \leq \overline{\mathsf{counter}} - \underline{\mathsf{counter}} \end{array}\right\} 0$$

And to all the other input variables

S2N-Bignum

Add Function

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

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

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

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

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

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

```
1  def Add(p, z, m, x, n, y):
2      r = min(p, m)
3      s = min(p, n)
4      if (r < s):
5          t = p - s
6          q = s - r
9          for (; r > 0; r--):
--             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
```

$d^{\natural}$ is

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

S2N-Bignum

Add Function

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

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

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

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

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

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

```
1  def Add(p, z, m, x, n, y):
2      r = min(p, m)
3      s = min(p, n)
4      if (r < s):
5          t = p - s
6          q = s - r
9          for (; r > 0; r--):
--             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
```

$d^{\natural}$ is

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

S2N-Bignum

Add Function

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

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

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

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

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

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

```
1  def Add(p, z, m, x, n, y):
2      r = min(p, m)
3      s = min(p, n)
4      if (r < s):
5          t = p - s
6          q = s - r
9          for (; r > 0; r--):
--             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
```

$d^{\natural}$ is

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

The Add function is safe

from timing side-channels

on input variables

m, n, x, y, z

S2N-Bignum

Add Function

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

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

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

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

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

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

```
1  def Add(p, z, m, x, n, y):
2      r = min(p, m)
3      s = min(p, n)
4      if (r < s):
5          t = p - s
6          q = s - r
9          for (; r > 0; r--):
--             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
```

$d^{\natural}$ is

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

The Add function is safe

from timing side-channels

on input variables

m, n, x, y, z

data input variables

S2N-Bignum

Table 4: Input composition of the S2N-BIGNUM library. The variables SAFE $\Delta|_S$ are highlighted in green, while the variables NUMERICAL $\Delta|_N$ in red. No numerical variable should be MAYBE DANGEROUS.

PROGRAM	INPUT VARIABLES Δ SAFE $\Delta\|_S$	INPUT VARIABLES Δ NUMERICAL $\Delta\|_N$	MAYBE DANGEROUS	ZERO IMPACT
Add	s_1, s_3, s_5	n_2, n_4, n_6	s_1	s_3, s_5, n_2, n_4, n_6
Amontifier	s_1	n_2, n_3, n_4	s_1	n_2, n_3, n_4
Amontmul	s_1	n_2, n_3, n_4, n_5	s_1	n_2, n_3, n_4, n_5
Amontredc	s_1, s_3, s_6	n_2, n_4, n_5	s_1, s_3, s_6	n_2, n_4, n_5
Amontsqr	s_1	n_2, n_3, n_4	s_1	n_2, n_3, n_4
Bitfield	s_1	n_2, n_3, n_4, n_5	s_1	n_2, n_3, n_4, n_5
Bitsize	s_1	n_2	s_1	n_2
Cdiv	s_1, s_3	n_2, n_4, n_5	s_1, s_3	n_2, n_4, n_5
Cdiv_exact	s_1, s_3	n_2, n_4, n_5	s_1	n_2, s_3, n_4, n_5
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, n_6
Word_bytereverse		n_1		n_1
Word_clz		n_1		n_1
Word_ctz		n_1		n_1
Word_divstep59		n_1, n_2, n_3, n_4		n_1, n_2, n_3, n_4
Word_max		n_1, n_2		n_1, n_2
Word_min		n_1, n_2		n_1, n_2
Word_negmodinv		n_1		n_1
Word_recip		n_1		n_1
TOTAL VARIABLES:	93	179	85	187

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

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

Conclusion