Expanding the Reach of Fuzz Testing

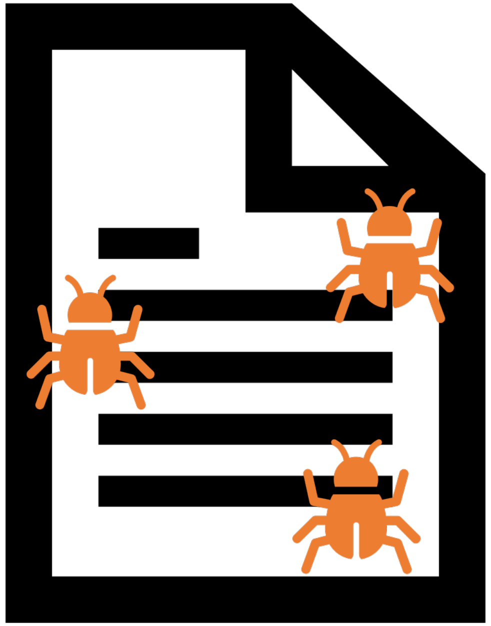
Caroline Lemieux

The University of British Columbia

CSER New Faculty Talk

June 7, 2023

Software Has Bugs



Bugs Have Increasing Consequences

Bugs Have Increasing Consequences

Badlock Cloudbleed Dirty COW GHOST Heartbleed StageFright ShellShock

Bugs Have Increasing Consequences

The Cost of Poor Software Quality in the US: A 2020 Report

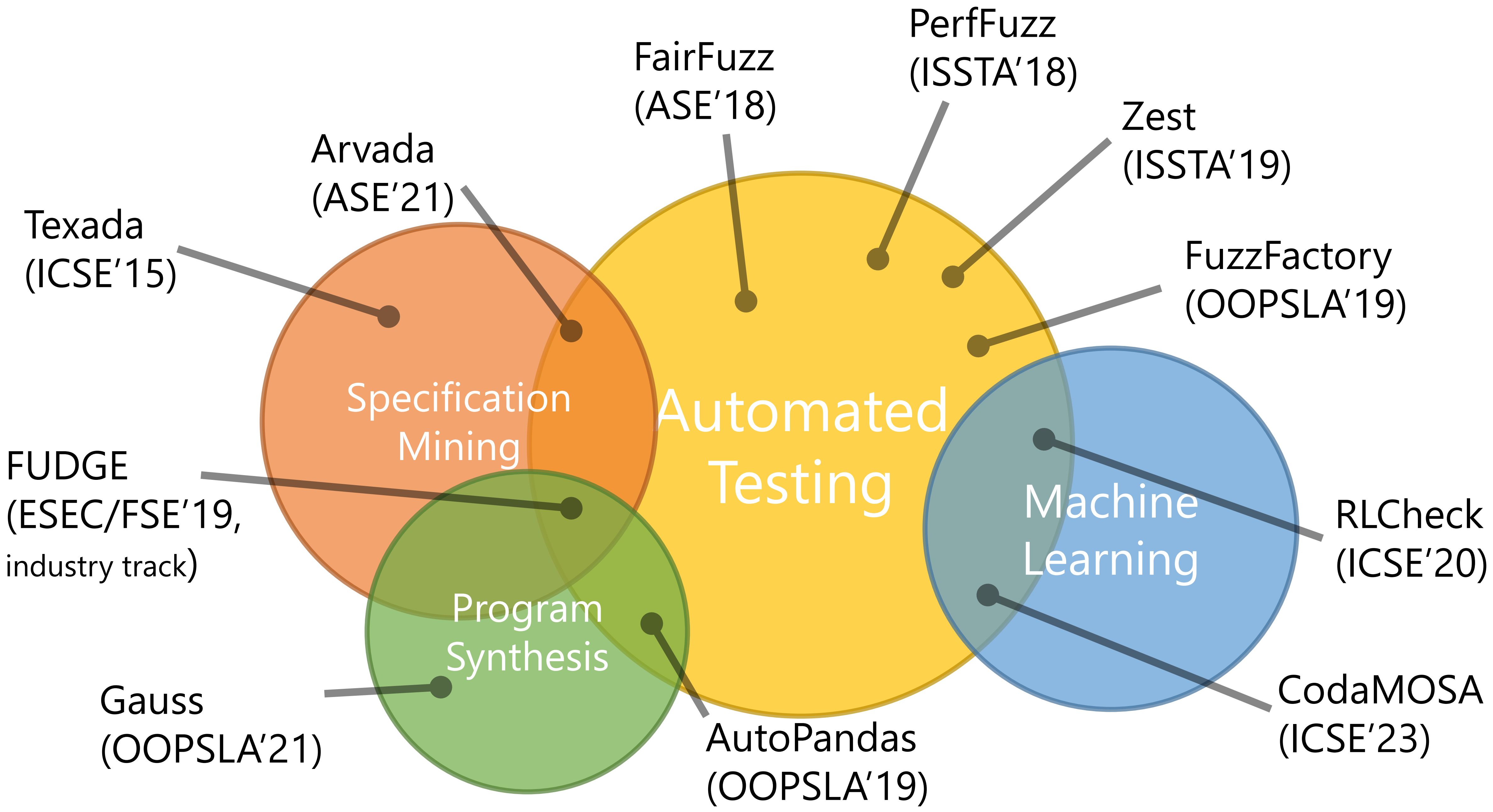
Herb Krasner
Member, Advisory Board
Consortium for Information & Software Quality™ (CISQ™)
www.it-cisq.org
hkrasner@utexas.edu
Date: January 1, 2021

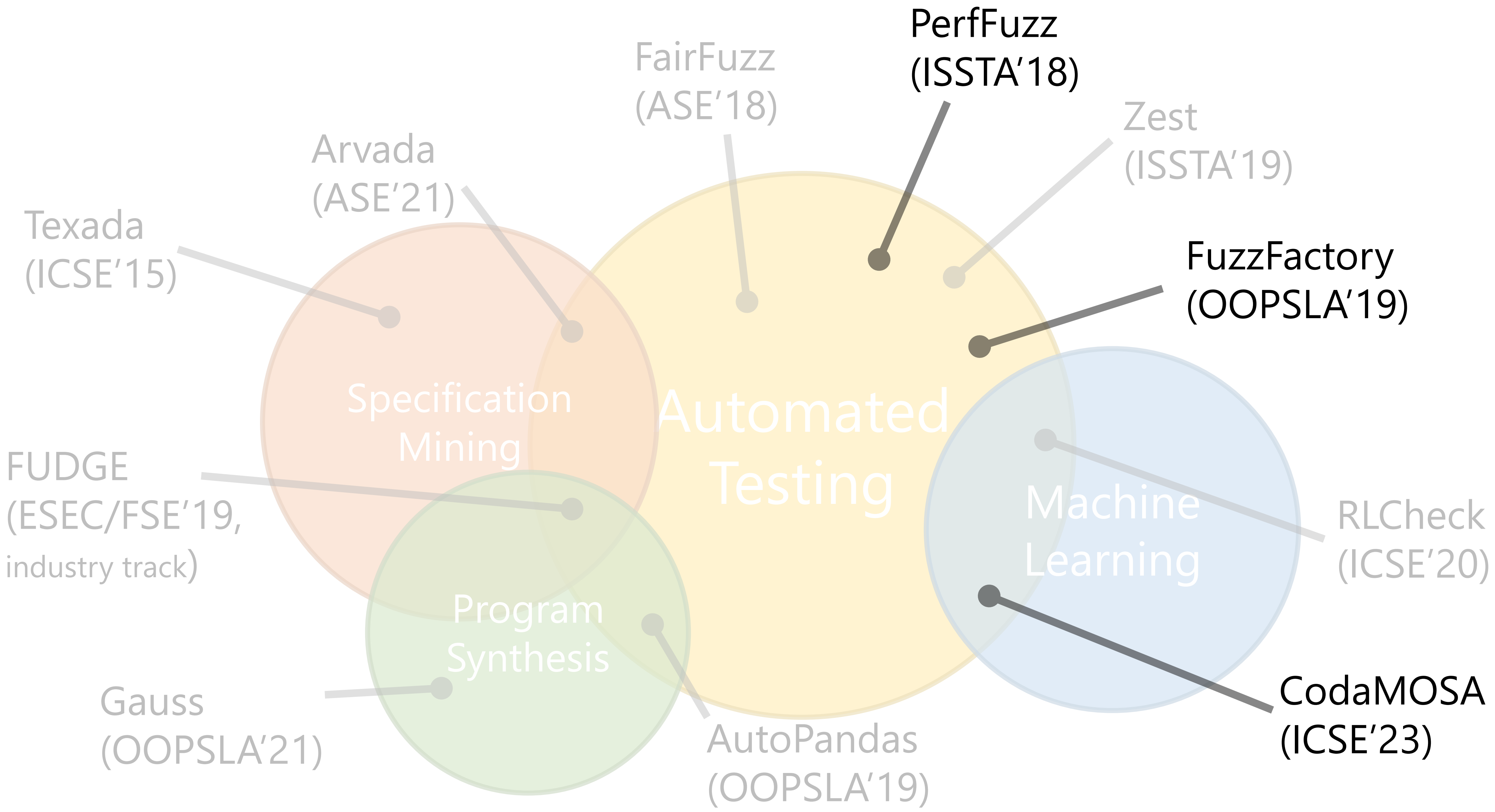
CISQ
Consortium for Information & Software Quality ™

the cost of
poor quality software is
$2.08 trillion
in the US in 2020

My Work:

build tools to help developers improve **correctness**, **security** and **performance** of software


PerfFuzz
(ISSTA'18)
FairFuzz
(ASE'18)
Zest
(ISSTA'19)
Arvada
(ASE'21)
Texada
(ICSE'15)
FuzzFactory
(OOPSLA'19)
Specification
Mining
Automated
Testing
FUDGE
(ESEC/FSE'19,
industry track)
Machine
Learning
RLCheck
(ICSE'20)
Program
Synthesis
Gauss
(OOPSLA'21)
CodaMOSA
(ICSE'23)
AutoPandas
(OOPSLA'19)



PerfFuzz
(ISSTA’18)

FairFuzz
(ASE’18)

Zest
(ISSTA’19)

Arvada
(ASE’21)

Texada
(ICSE’15)

FuzzFactory
(OOPSLA’19)

Specification
Mining

Automated
Testing

Machine
Learning

FUDGE
(ESEC/FSE’19,
industry track)

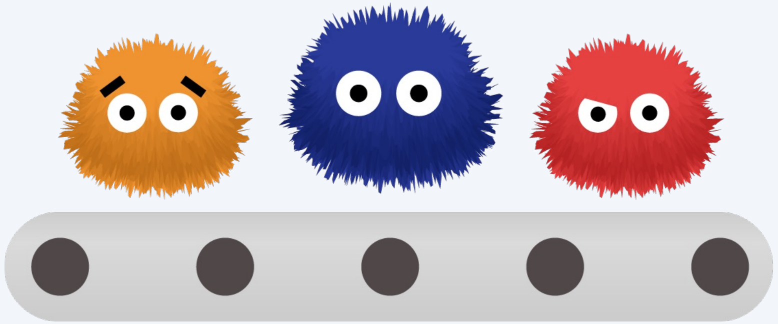
RLCheck
(ICSE’20)

Program
Synthesis

Gauss
(OOPSLA’21)

AutoPandas
(OOPSLA’19)

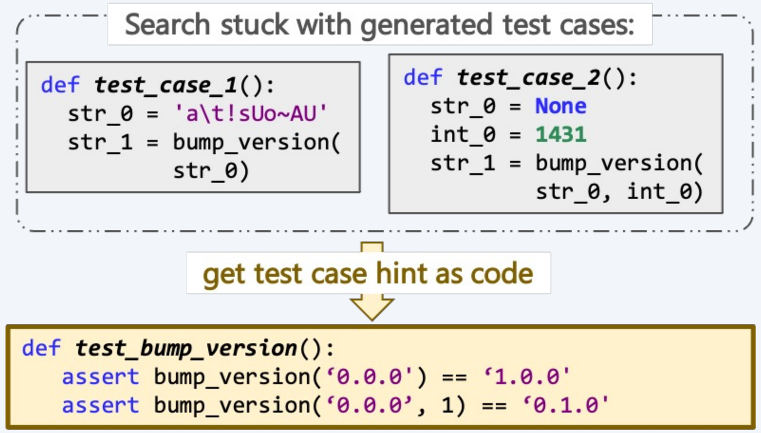
CodaMOSA
(ICSE’23)



Using ***generalized feedback maps*** to expand ***bugs findable by fuzz testing***

PerfFuzz (ISSTA’18)

FuzzFactory (OOPSLA’19)



Search stuck with generated test cases:

```
def test_case_1():
  str_0 = 'a\t!sUo~AU'
  str_1 = bump_version(
            str_0)
```

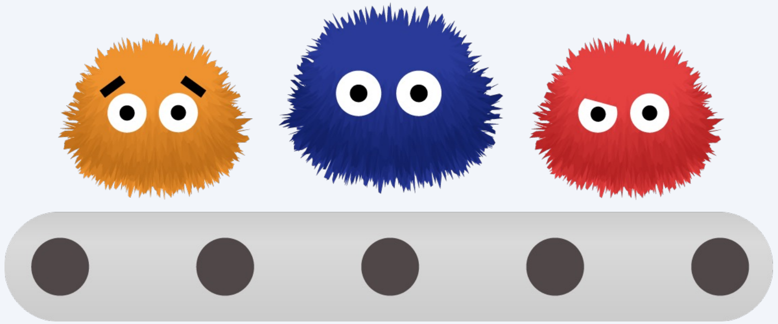
```
def test_case_2():
  str_0 = None
  int_0 = 1431
  str_1 = bump_version(
            str_0, int_0)
```

get test case hint as code

```
def test_bump_version():
   assert bump_version(‘0.0.0') == ‘1.0.0'
   assert bump_version(‘0.0.0’, 1) == ‘0.1.0'
```

Using ***large language models*** to improve ***automated test suite generation***

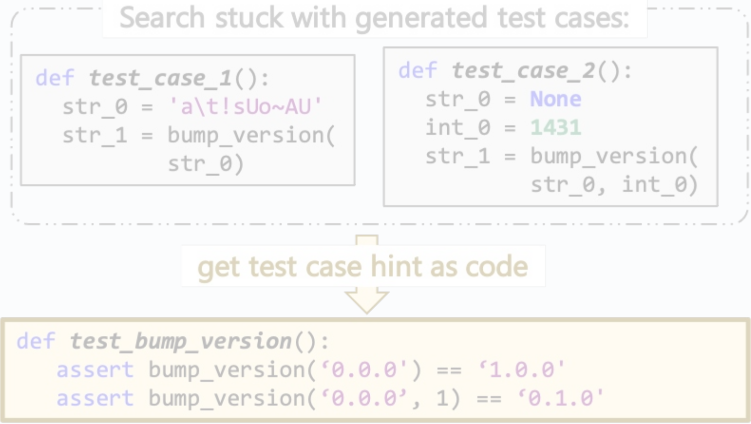
CodaMOSA (ICSE’23)



Using ***generalized feedback maps*** to expand ***bugs findable by fuzz testing***

PerfFuzz (ISSTA’18)

FuzzFactory (OOPSLA’19)



Search stuck with generated test cases:

```
def test_case_1():
  str_0 = 'a\t!sUo~AU'
  str_1 = bump_version(
            str_0)
```

```
def test_case_2():
  str_0 = None
  int_0 = 1431
  str_1 = bump_version(
            str_0, int_0)
```

get test case hint as code

```
def test_bump_version():
   assert bump_version(‘0.0.0') == ‘1.0.0'
   assert bump_version(‘0.0.0’, 1) == ‘0.1.0'
```

Using ***large language models*** to improve ***automated test suite generation***

CodaMOSA (ICSE’23)

Performance Bugs

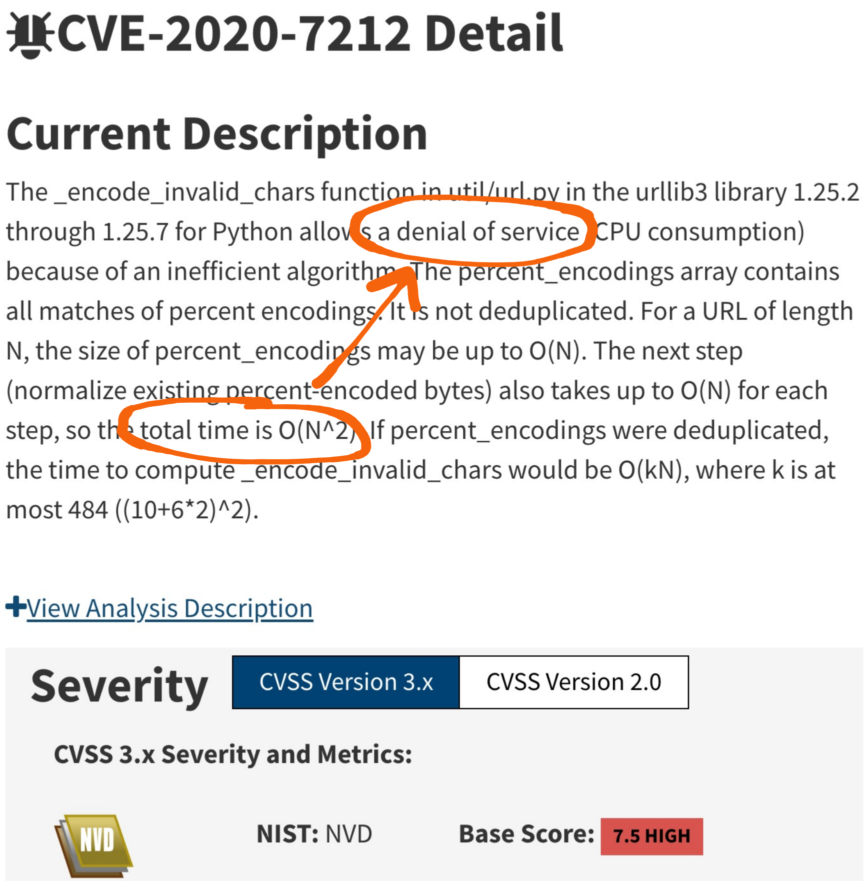
Performance Bugs

Performance Bugs

Takes **1+ minute** to report syntax error:

```
(((((((((((((((((((((((((((((((((((((((((((((((((((((e foo = 1; => 1;
```

Performance Bugs



CVE-2020-7212 Detail

Current Description

The _encode_invalid_chars function in util/url.py in the urllib3 library 1.25.2 through 1.25.7 for Python allows a denial of service (CPU consumption) because of an inefficient algorithm. The percent_encodings array contains all matches of percent encodings. It is not deduplicated. For a URL of length N, the size of percent_encodings may be up to O(N). The next step (normalize existing percent-encoded bytes) also takes up to O(N) for each step, so the total time is O(N^2). If percent_encodings were deduplicated, the time to compute _encode_invalid_chars would be O(kN), where k is at most 484 ((10+6*2)^2).

+View Analysis Description

Severity CVSS Version 3.x | CVSS Version 2.0

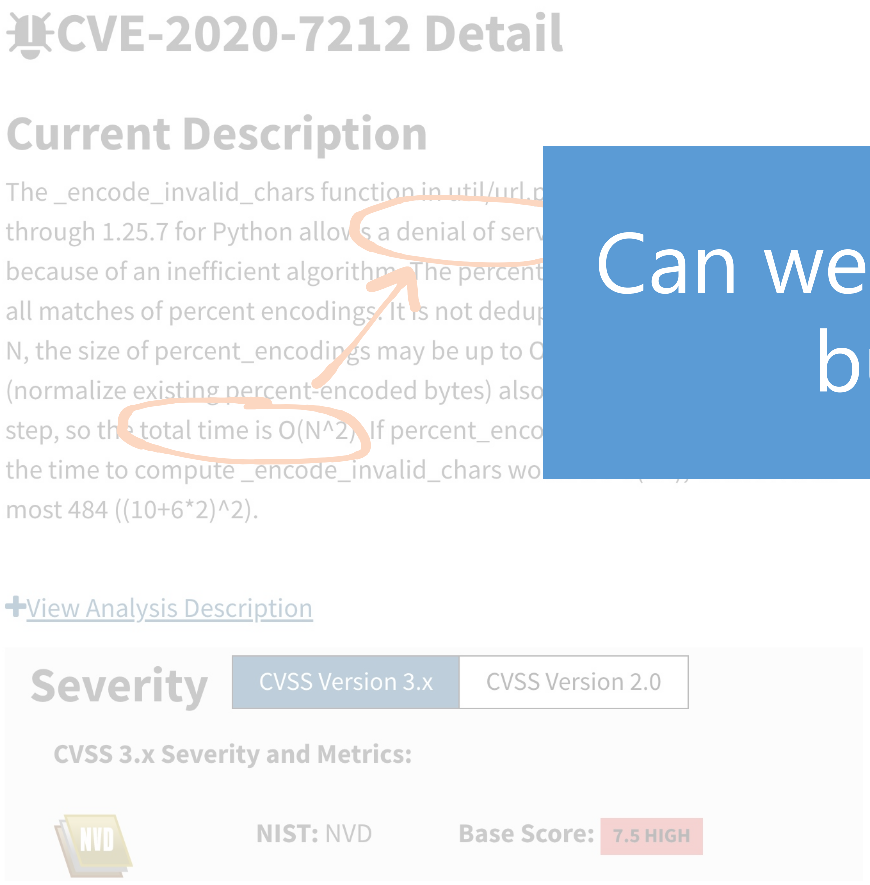
CVSS 3.x Severity and Metrics:

NVD **NIST:** NVD **Base Score:** 7.5 HIGH

Takes **1+ minute** to report syntax error:

```
(((((((((((((((((((((((((((((((((((((((((((((((((((e foo = 1; => 1;
```

Performance Bugs



CVE-2020-7212 Detail

Current Description

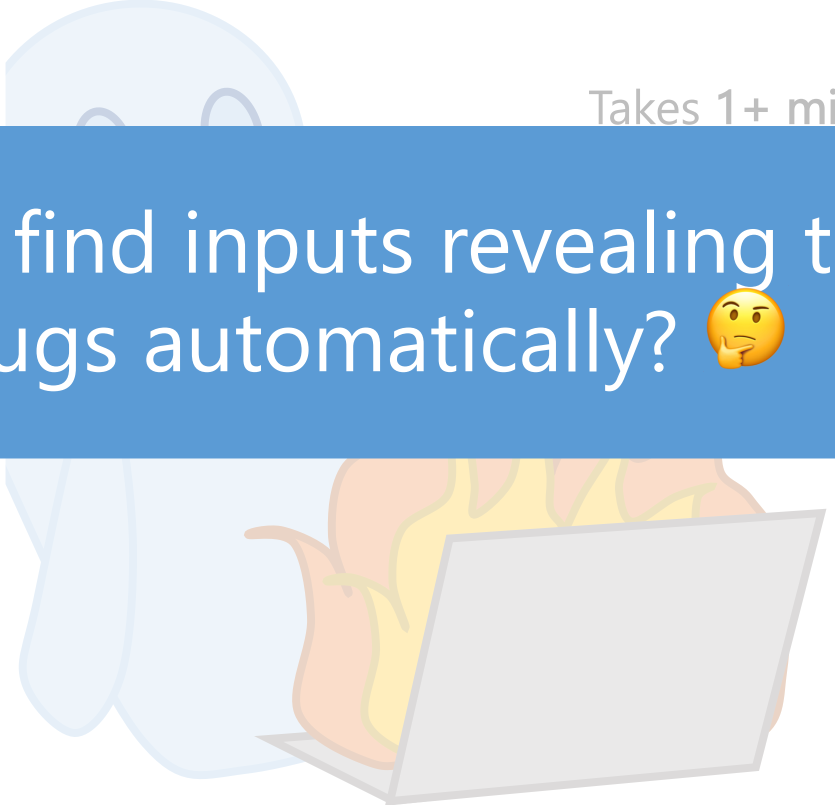
The _encode_invalid_chars function in util/url.p
through 1.25.7 for Python allows a denial of serv
because of an inefficient algorithm. The percent
all matches of percent encodings. It is not dedup
N, the size of percent_encodings may be up to O
(normalize existing percent-encoded bytes) also
step, so the total time is O(N^2). If percent_enco
the time to compute _encode_invalid_chars wo
most 484 ((10+6*2)^2).

+View Analysis Description

Severity CVSS Version 3.x CVSS Version 2.0

CVSS 3.x Severity and Metrics:

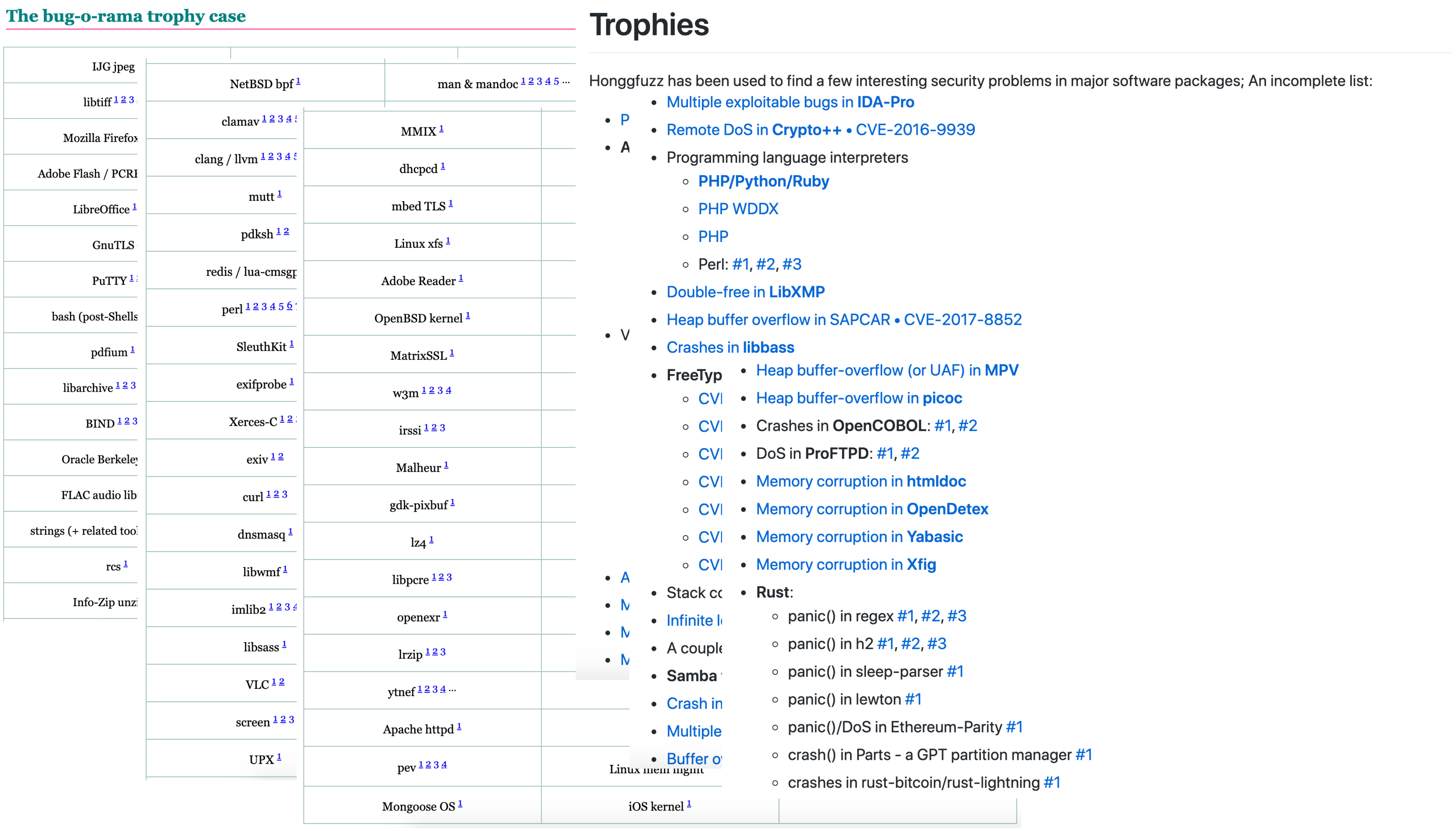
NVD **NIST:** NVD **Base Score:** 7.5 HIGH



Takes **1+ minute** to report syntax error:

```
(((((((((((((((e foo = 1; => 1;
```

Can we find inputs revealing these bugs automatically? 🤔



The bug-o-rama trophy case

IJG jpeg
libtiff 1 2 3
Mozilla Firefox
Adobe Flash / PCRE
LibreOffice 1
GnuTLS
PuTTY 1
bash (post-Shells
pdfium 1
libarchive 1 2 3
BIND 1 2 3
Oracle Berkeley
FLAC audio lib
strings (+ related tool
rcs 1
Info-Zip unzi

NetBSD bpf 1
clamav 1 2 3 4
clang / llvm 1 2 3 4 5
mutt 1
pdksh 1 2
redis / lua-cmsgp
perl 1 2 3 4 5 6
SleuthKit 1
exifprobe 1
Xerces-C 1 2
exiv 1 2
curl 1 2 3
dnsmasq 1
libwmf 1
imlib2 1 2 3 4
libsass 1
VLC 1 2
screen 1 2 3
UPX 1

man & mandoc 1 2 3 4 5 ...
MMIX 1
dhcpcd 1
mbed TLS 1
Linux xfs 1
Adobe Reader 1
OpenBSD kernel 1
MatrixSSL 1
w3m 1 2 3 4
irssi 1 2 3
Malheur 1
gdk-pixbuf 1
lz4 1
libpcre 1 2 3
openexr 1
lrzip 1 2 3
ytnef 1 2 3 4 ...
Apache httpd 1
pev 1 2 3 4
Mongoose OS 1

Linux mem mgmt
iOS kernel 1

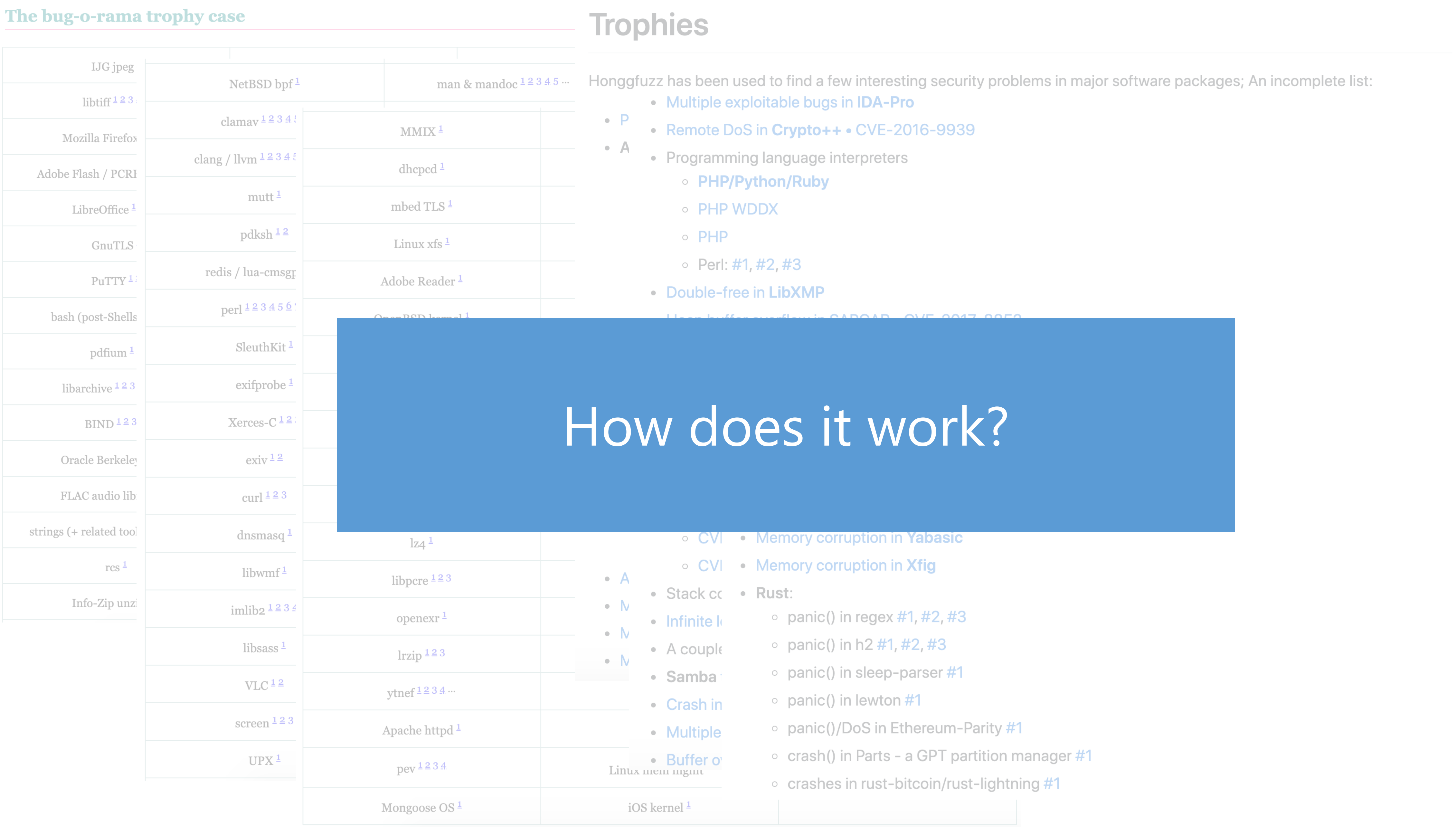
Trophies

Honggfuzz has been used to find a few interesting security problems in major software packages; An incomplete list:

- P
- A
- V
- A
- M
- M
- M

- Multiple exploitable bugs in **IDA-Pro**
- Remote DoS in **Crypto++** • CVE-2016-9939
- Programming language interpreters
  - **PHP/Python/Ruby**
  - PHP WDDX
  - PHP
  - Perl: #1, #2, #3
- Double-free in **LibXMP**
- Heap buffer overflow in SAPCAR • CVE-2017-8852
- Crashes in **libbass**
- **FreeTyp**
  - CVI
  - CVI
  - CVI
  - CVI
  - CVI
  - CVI
  - CVI
- Stack cc
- Infinite l
- A couple
- **Samba**
- Crash in
- Multiple
- Buffer o

- Heap buffer-overflow (or UAF) in **MPV**
- Heap buffer-overflow in **picoc**
- Crashes in **OpenCOBOL**: #1, #2
- DoS in **ProFTPD**: #1, #2
- Memory corruption in **htmldoc**
- Memory corruption in **OpenDetex**
- Memory corruption in **Yabasic**
- Memory corruption in **Xfig**
- **Rust**:
  - panic() in regex #1, #2, #3
  - panic() in h2 #1, #2, #3
  - panic() in sleep-parser #1
  - panic() in lewton #1
  - panic()/DoS in Ethereum-Parity #1
  - crash() in Parts - a GPT partition manager #1
  - crashes in rust-bitcoin/rust-lightning #1



The bug-o-rama trophy case

IJG jpeg
libtiff 1 2 3
Mozilla Firefox
Adobe Flash / PCRE
LibreOffice 1
GnuTLS
PuTTY 1
bash (post-Shells
pdfium 1
libarchive 1 2 3
BIND 1 2 3
Oracle Berkeley
FLAC audio lib
strings (+ related tool
rcs 1
Info-Zip unzi

NetBSD bpf 1
clamav 1 2 3 4
clang / llvm 1 2 3 4
mutt 1
pdksh 1 2
redis / lua-cmsgp
perl 1 2 3 4 5 6
SleuthKit 1
exifprobe 1
Xerces-C 1 2
exiv 1 2
curl 1 2 3
dnsmasq 1
libwmf 1
imlib2 1 2 3
libsass 1
VLC 1 2
screen 1 2 3
UPX 1

man & mandoc 1 2 3 4 5 …
MMIX 1
dhcpcd 1
mbed TLS 1
Linux xfs 1
Adobe Reader 1
OpenBSD kernel 1
lz4 1
libpcre 1 2 3
openexr 1
lrzip 1 2 3
ytnef 1 2 3 4 …
Apache httpd 1
pev 1 2 3 4
Mongoose OS 1

Linux mem mgmt
iOS kernel 1

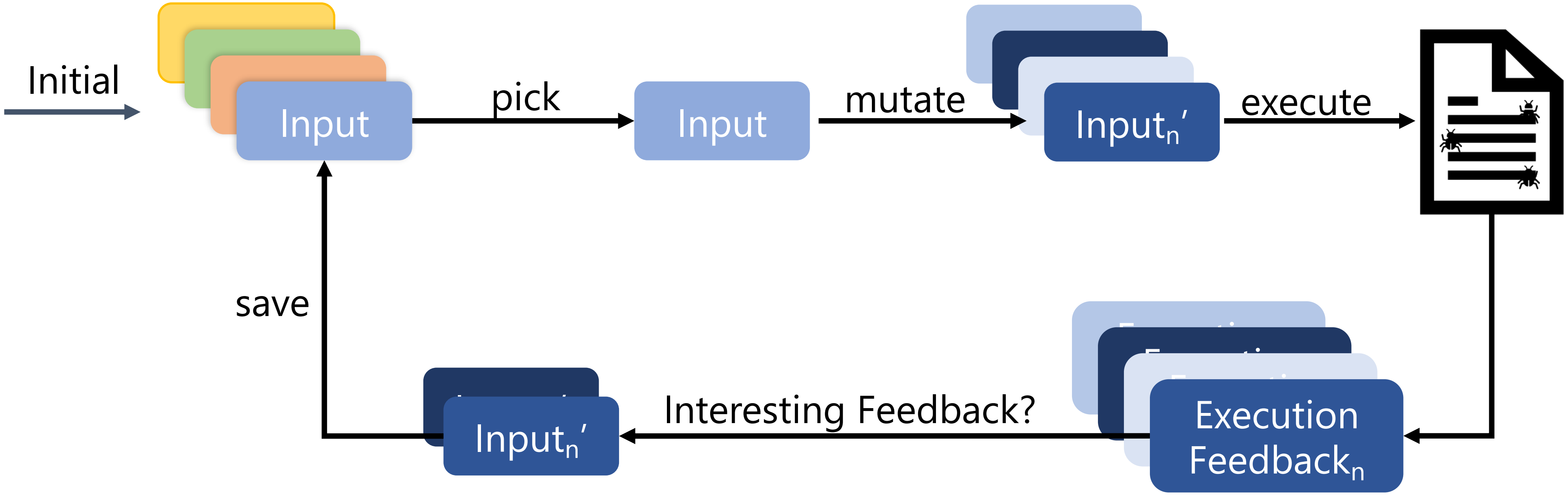
Trophies

Honggfuzz has been used to find a few interesting security problems in major software packages; An incomplete list:

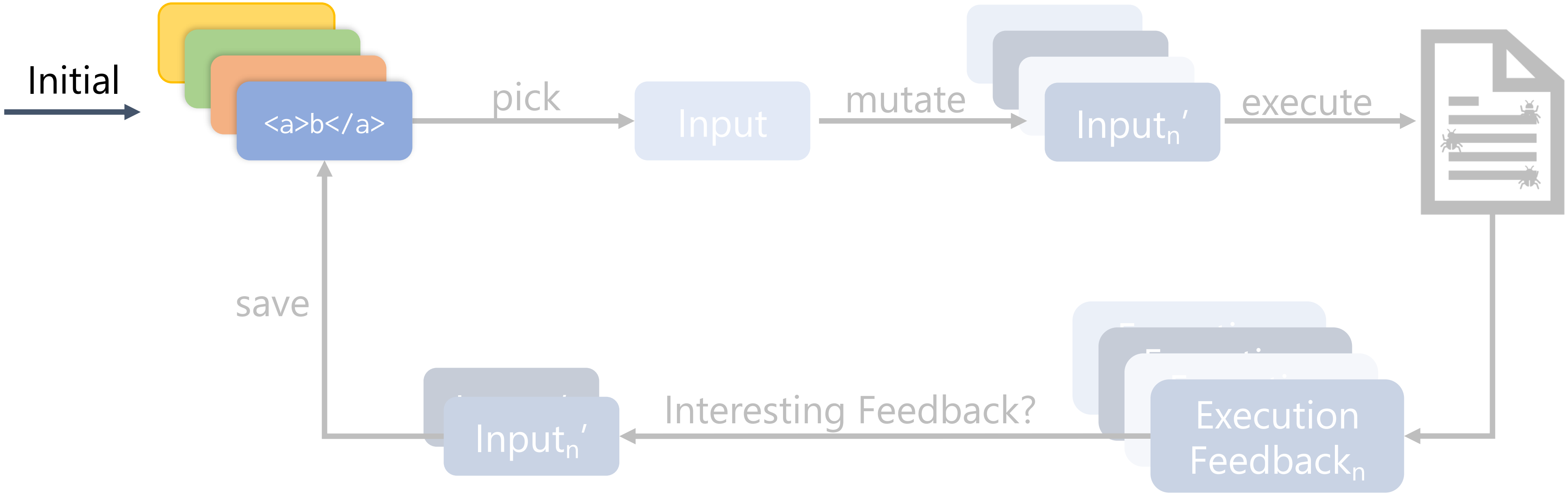
- Multiple exploitable bugs in **IDA-Pro**
- Remote DoS in **Crypto++** • CVE-2016-9939
- Programming language interpreters
  - **PHP/Python/Ruby**
  - PHP WDDX
  - PHP
  - Perl: #1, #2, #3
- Double-free in **LibXMP**
- Heap buffer overflow in SAPCAR - CVE-2017-8852
- Stack cc
- Infinite l
- A couple
- **Samba**
- Crash in
- Multiple
- Buffer o
- Memory corruption in **Yabasic**
- Memory corruption in **Xfig**
- **Rust**:
  - panic() in regex #1, #2, #3
  - panic() in h2 #1, #2, #3
  - panic() in sleep-parser #1
  - panic() in lewton #1
  - panic()/DoS in Ethereum-Parity #1
  - crash() in Parts - a GPT partition manager #1
  - crashes in rust-bitcoin/rust-lightning #1

How does it work?

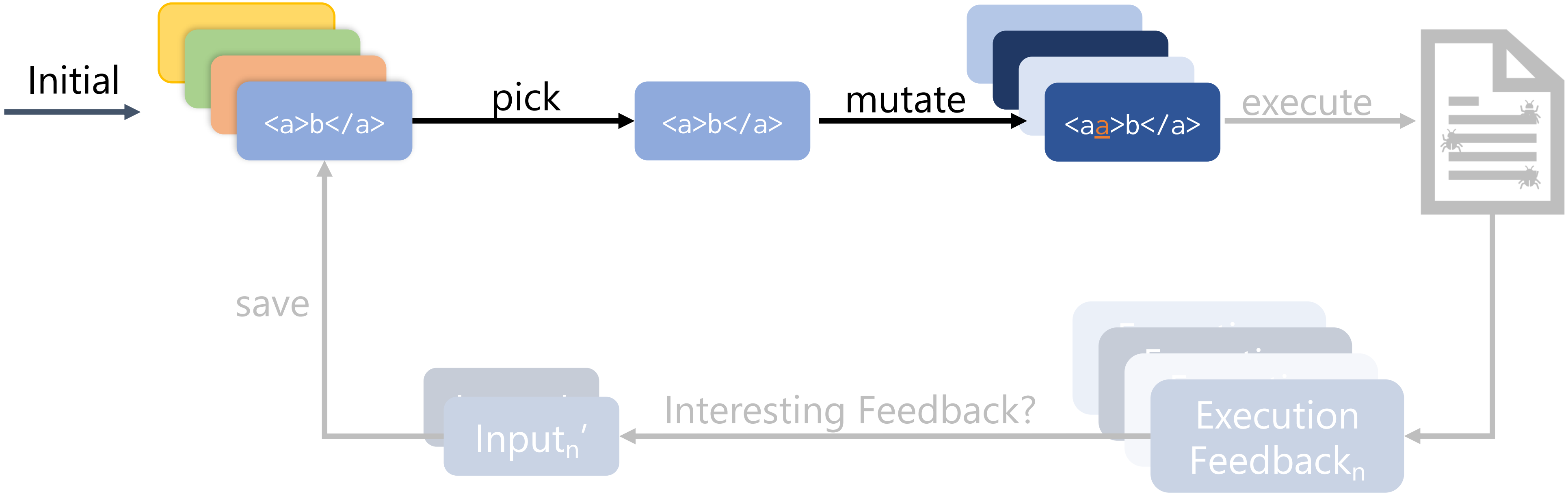
Coverage-Guided Fuzzing



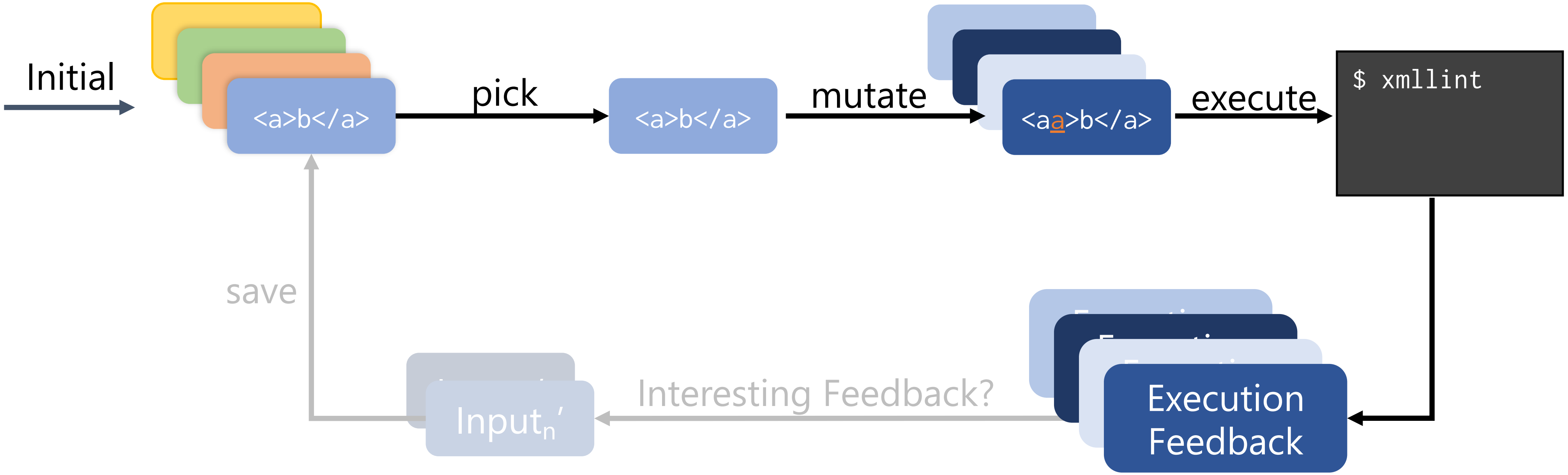
Coverage-Guided Fuzzing



Coverage-Guided Fuzzing



Coverage-Guided Fuzzing



Initial

<a>b</a>

pick

<a>b</a>

mutate

<aa>b</a>

execute

```
$ xmllint
```

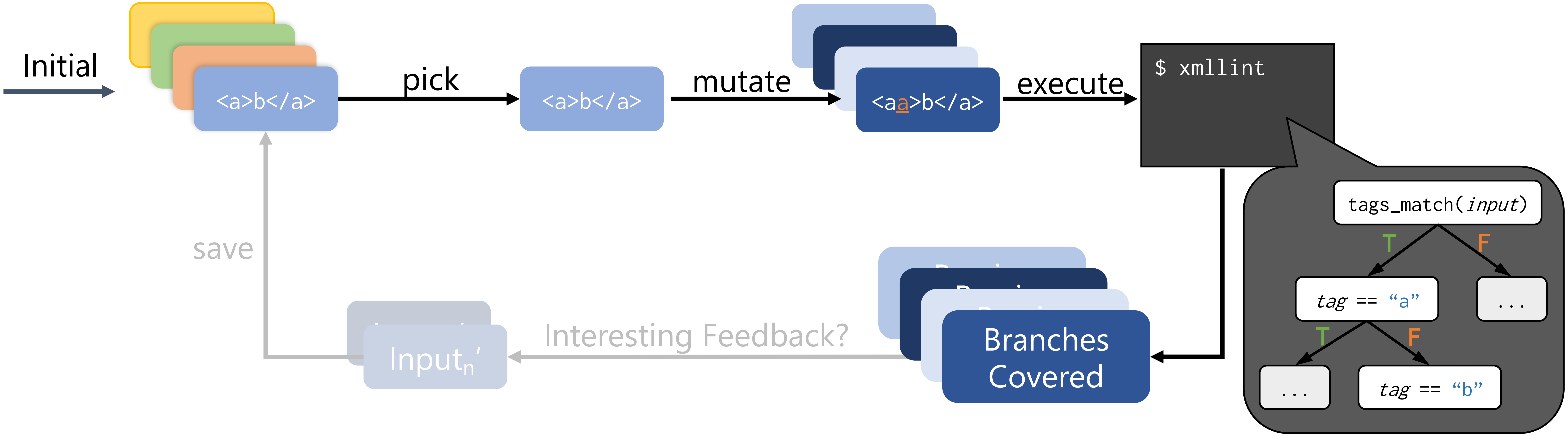
Execution Feedback

Interesting Feedback?

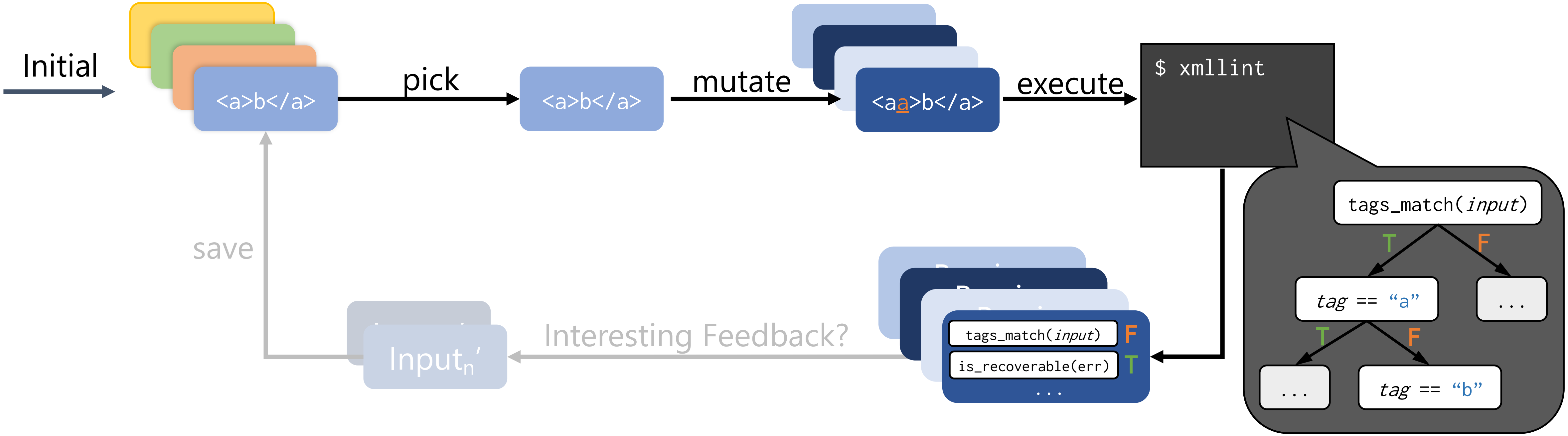
Input_n'

save

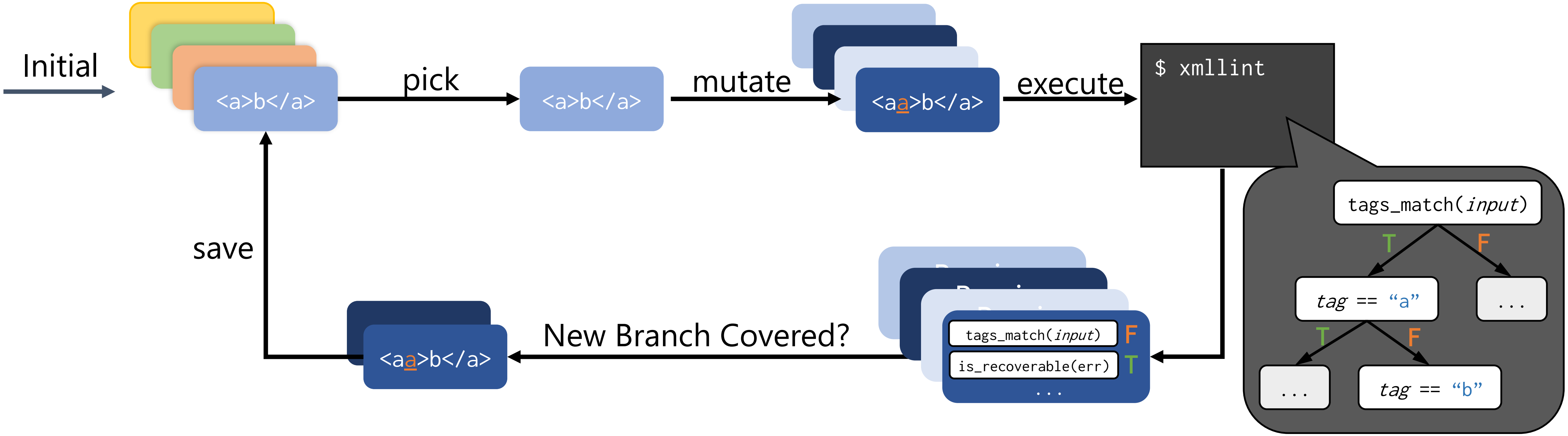
Coverage-Guided Fuzzing



Coverage-Guided Fuzzing



Coverage-Guided Fuzzing



Initial

<a>b</a>

pick

<a>b</a>

mutate

<aa>b</a>

execute

```
$ xmllint
```

tags_match(*input*)

T

F

tag == “a”

. . .

T

F

. . .

tag == “b”

tags_match(*input*) F

is_recoverable(err) T

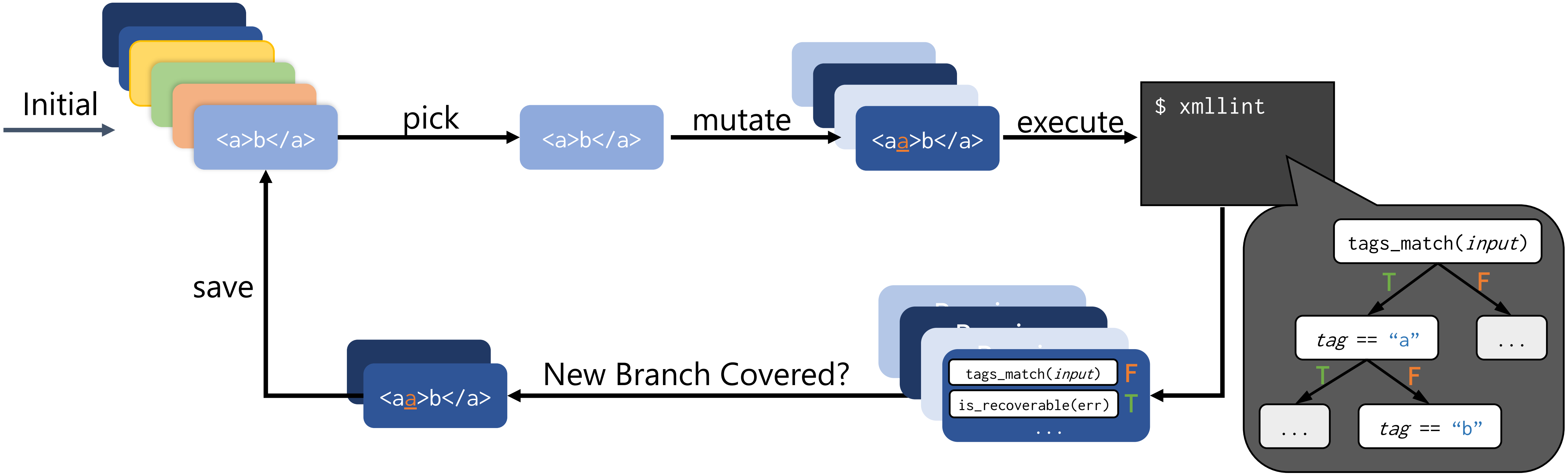
. . .

New Branch Covered?

<aa>b</a>

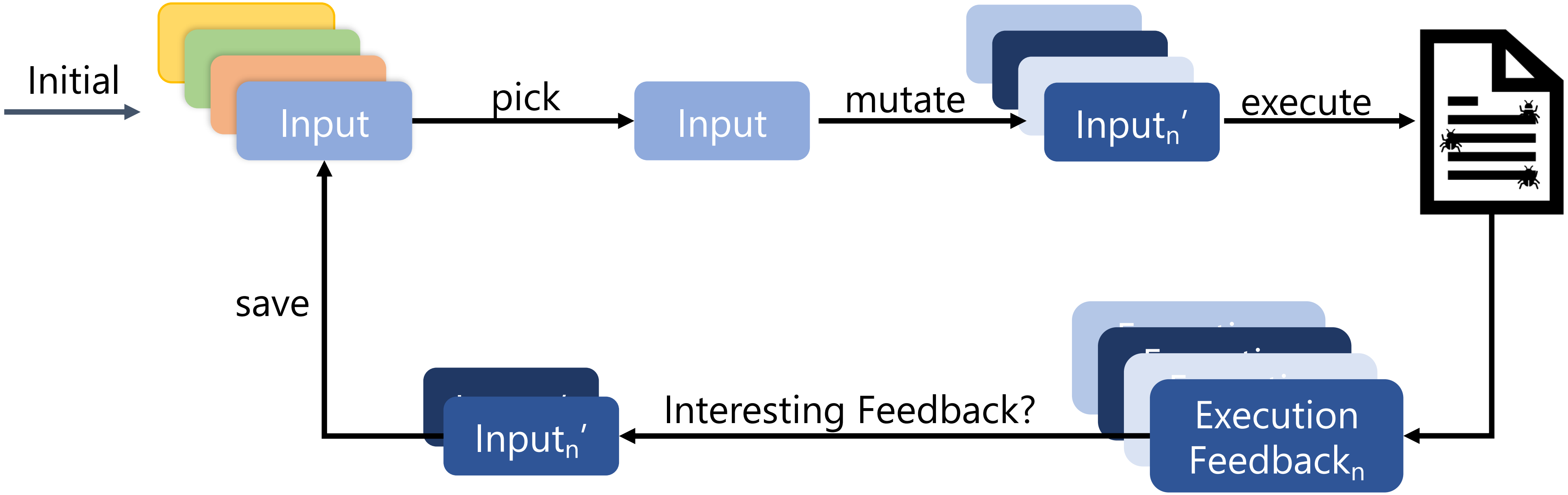
save

Coverage-Guided Fuzzing

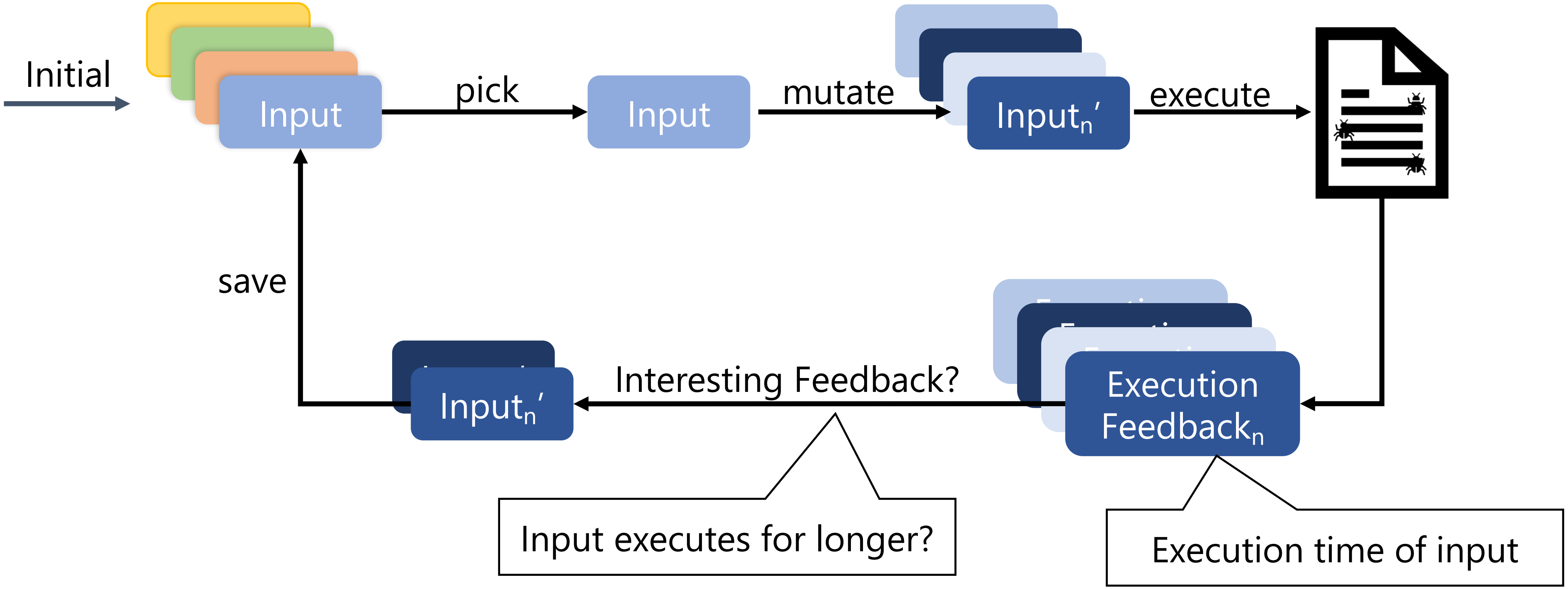


```
$ xmllint
```

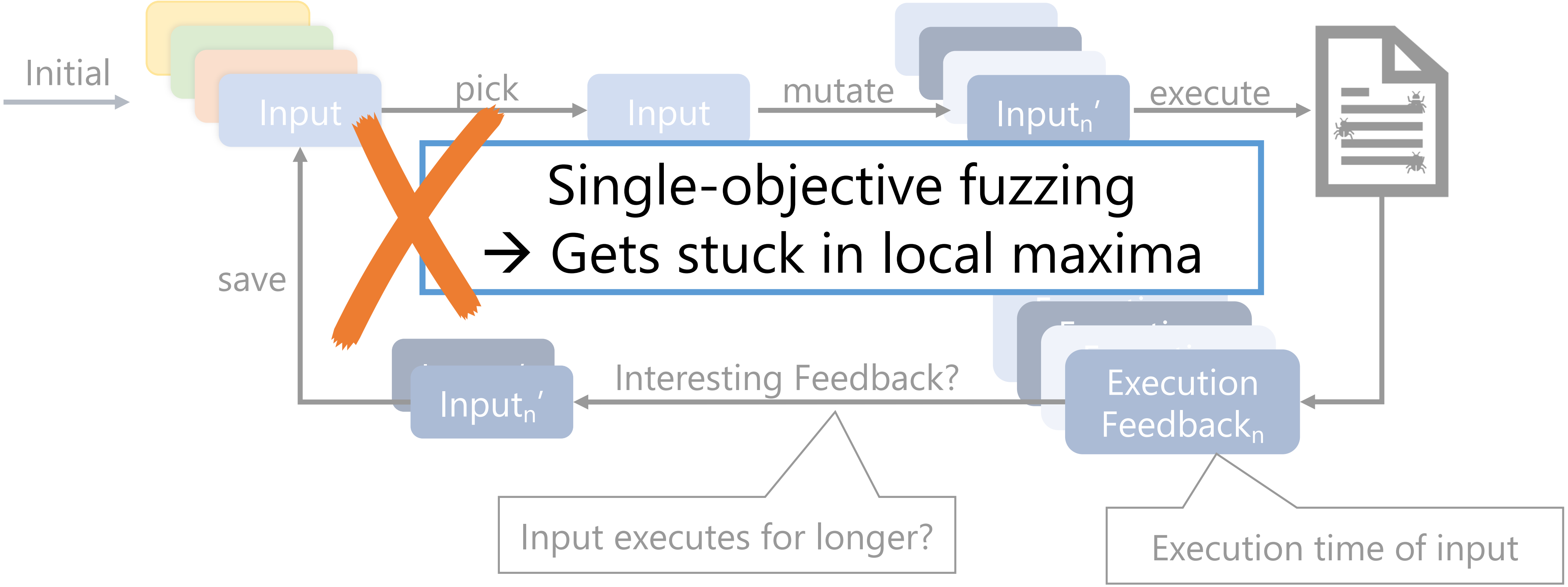
Coverage-Guided Fuzzing



Fuzzing to Find Performance Bugs?

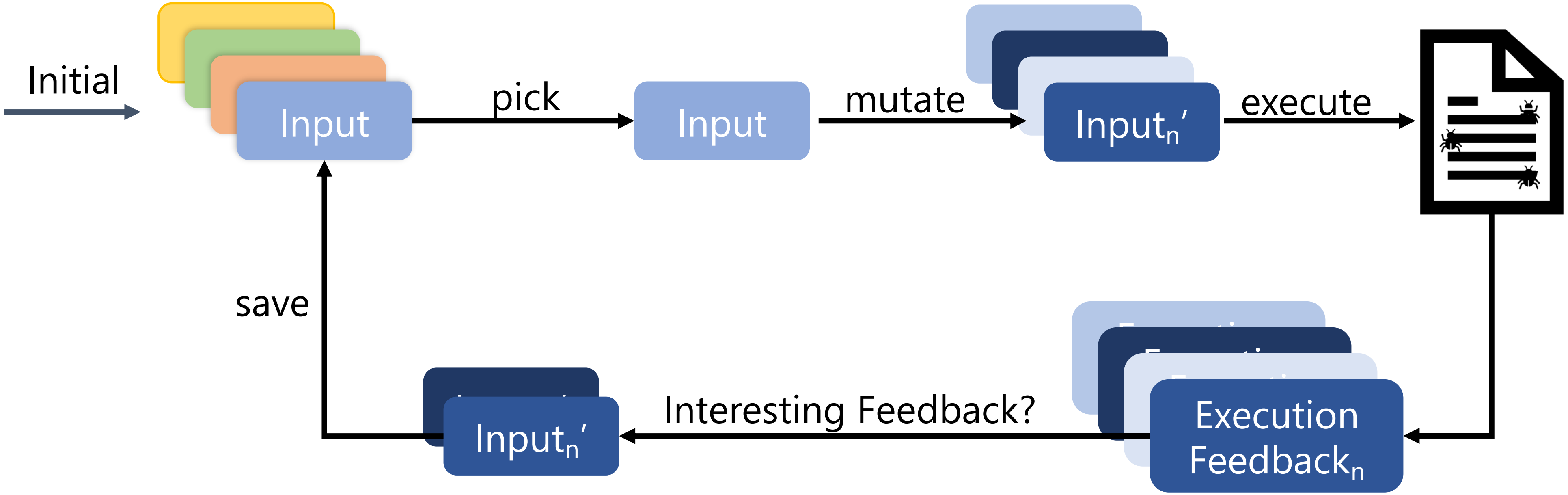


Fuzzing to Find Performance Bugs?

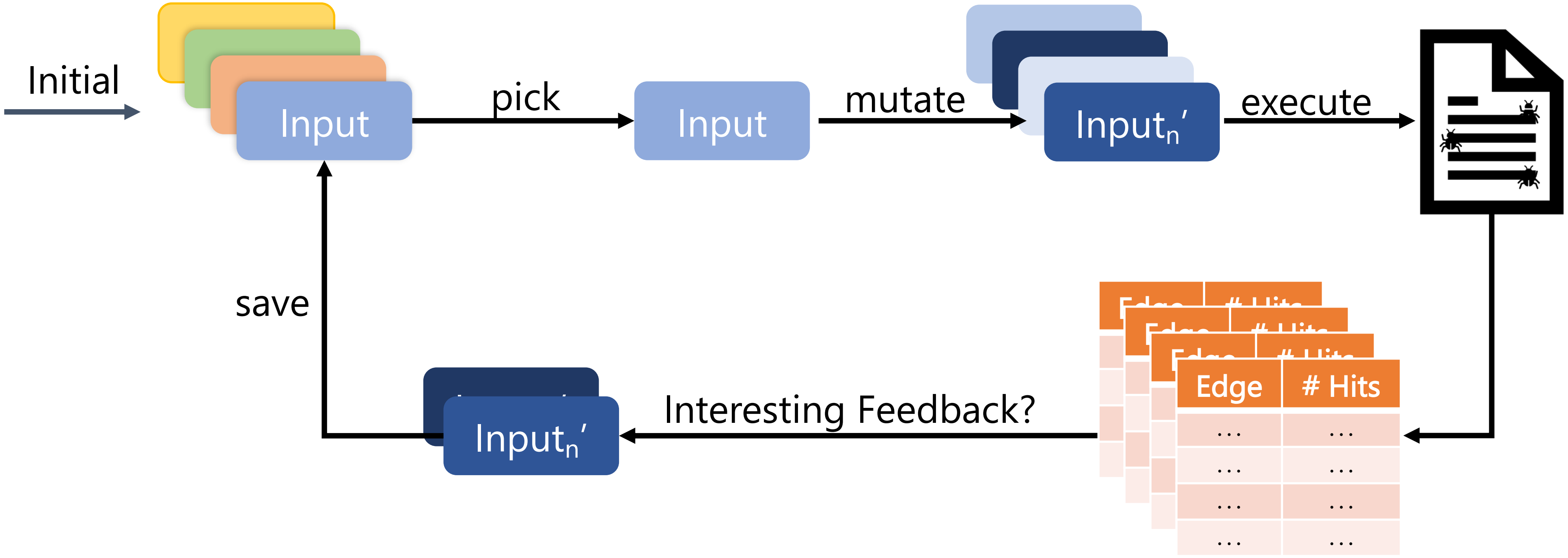


Single-objective fuzzing
→ Gets stuck in local maxima

PerfFuzz

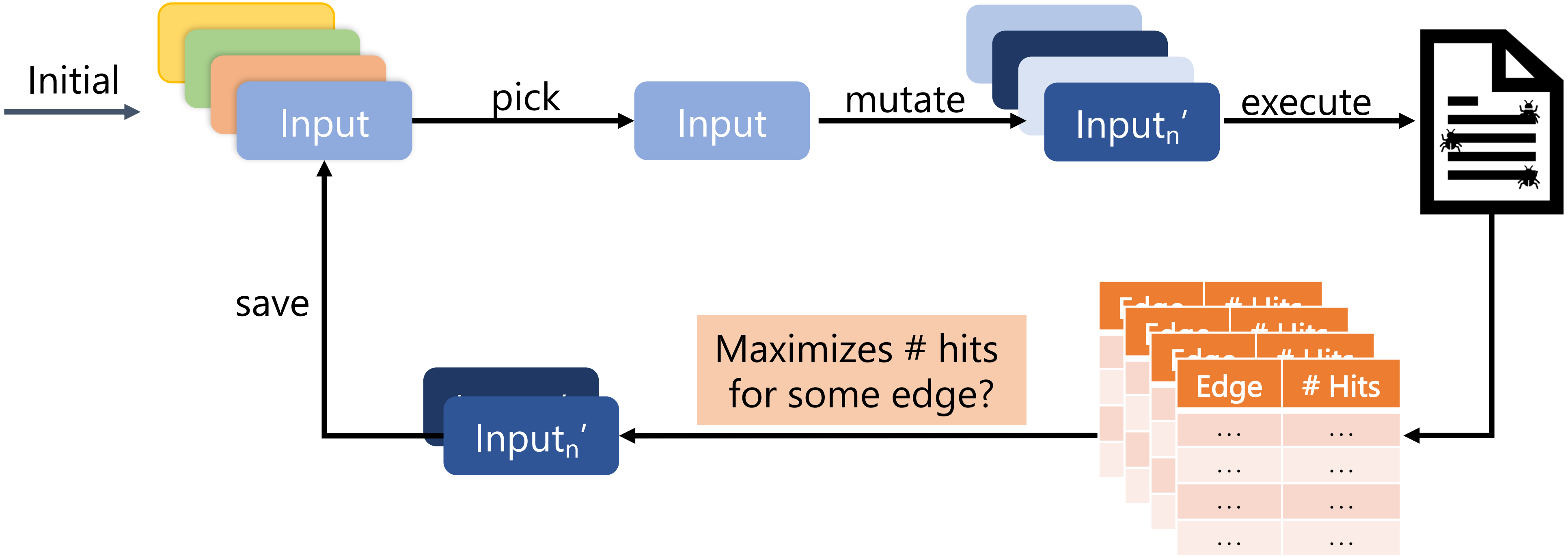

Initial
Input
pick
Input
mutate
Input$_n$'
execute
Execution Feedback$_n$
Interesting Feedback?
Input$_n$'
save

PerfFuzz

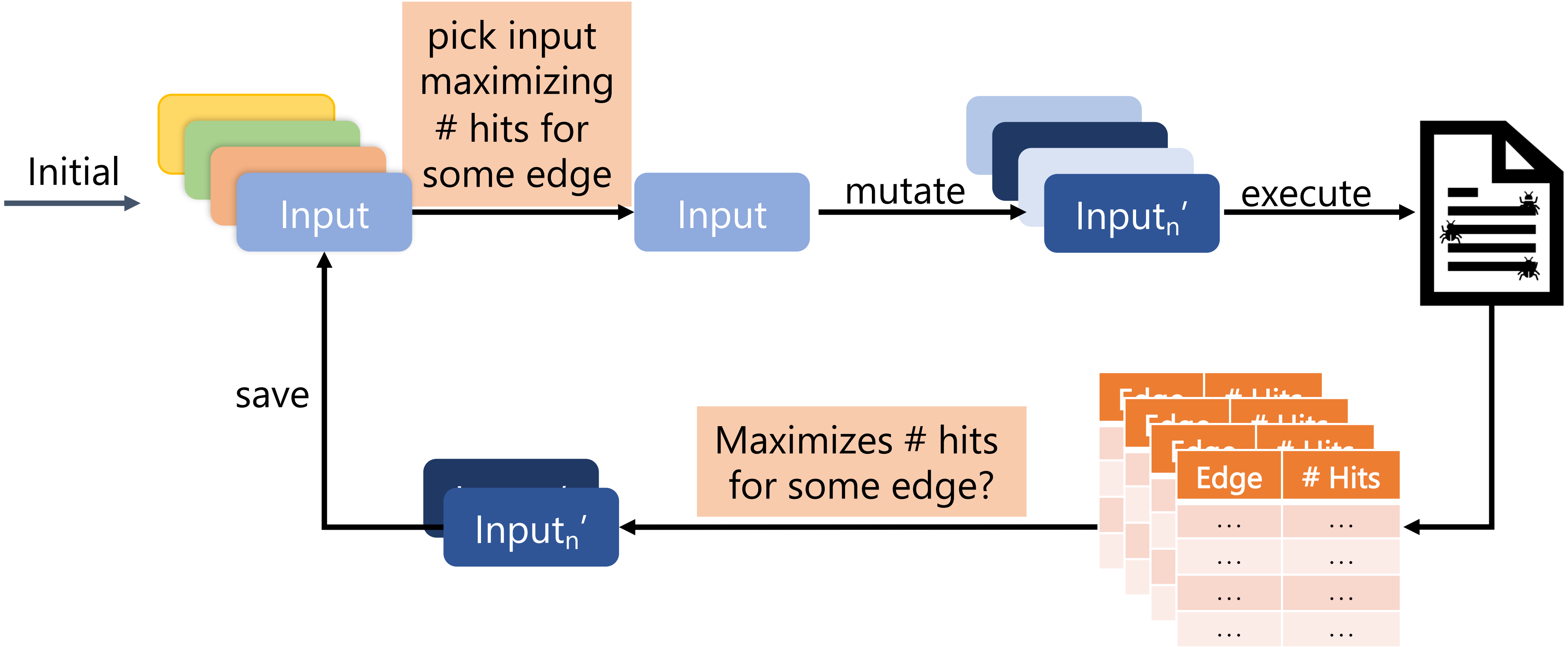


Initial
Input
pick
Input
mutate
Input$_n$'
execute
Edge
# Hits
…
Interesting Feedback?
Input$_n$'
save

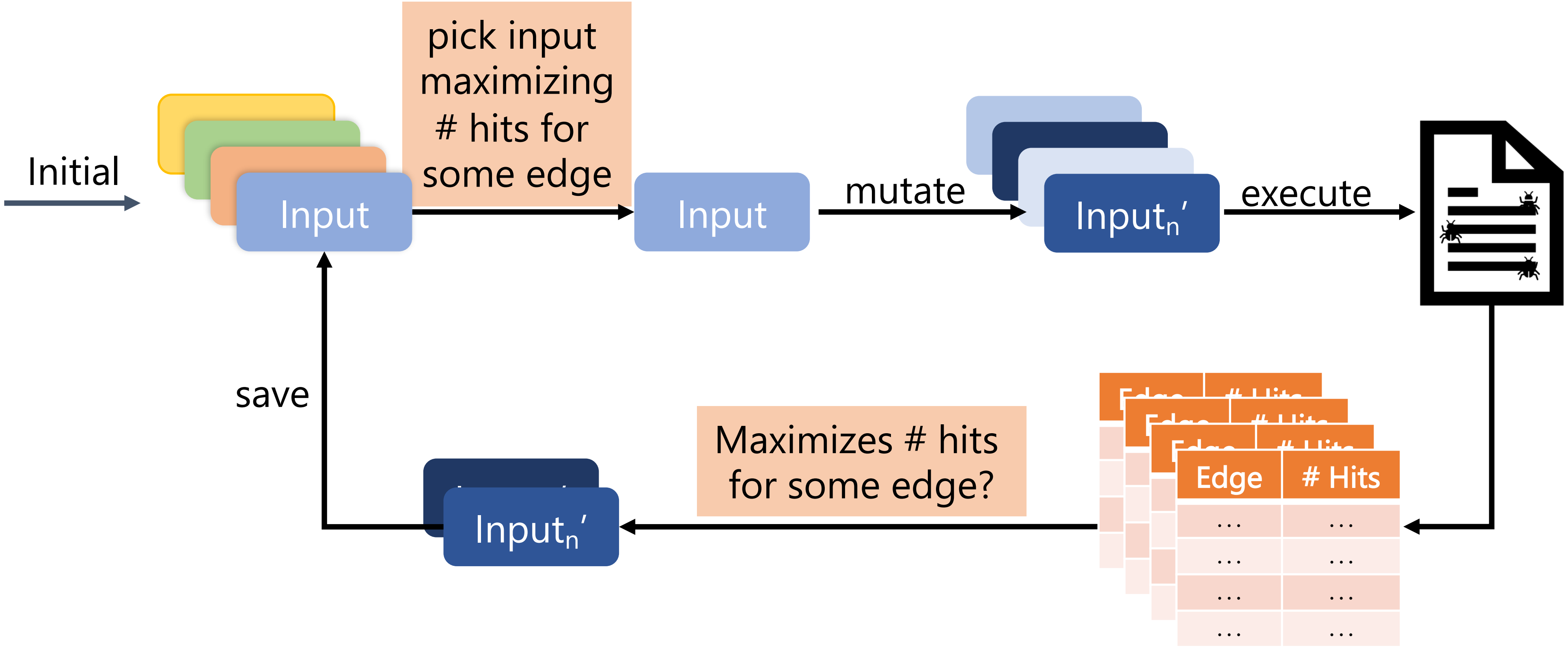
PerfFuzz


Initial
Input
pick
Input
mutate
Input$_n$′
execute
Edge
# Hits
…
save
Input$_n$′
Maximizes # hits for some edge?

PerfFuzz


Initial
Input
pick input maximizing # hits for some edge
Input
mutate
Input$_n$'
execute
Edge
# Hits
…
Maximizes # hits for some edge?
Input$_n$'
save

PerfFuzz



Initial

Input

pick input maximizing # hits for some edge

Input

mutate

Input_n'

execute

Edge	# Hits
…	…
…	…
…	…
…	…

Maximizes # hits for some edge?

Input_n'

save

Example program: `wf`

- Count frequency of words in string

input:

```
the quick brown the dog
```

output:

```
brown: 1
dog: 1
quick: 1
the: 2
```

Example program: `wf`

- Count frequency of words in string

input:

```
the quick brown the dog
```

output:

```
brown: 1
dog: 1
quick: 1
the: 2
```

Fedora Linux implementation: linked list hash table.
Quadratic worst-case behavior ☹
(when all words hash collide)

`wf` results: PerfFuzz Finds True Worst Cases

`wf` results: PerfFuzz Finds True Worst Cases

SlowFuzz (single objective maximization) worst case:

```
t r t t s f o Öe r t s f o r t x x t s f o r t x x
```

`wf` results: PerfFuzz Finds True Worst Cases

SlowFuzz (single objective maximization) worst case:

```
t r t t s f o Öe r t s f o r t x x t s f o r t x x
```

PerfFuzz worst case:

```
t <81>v ^?@t <80>!^?@t <80>!t t^Rn t t t t t t t t t
```

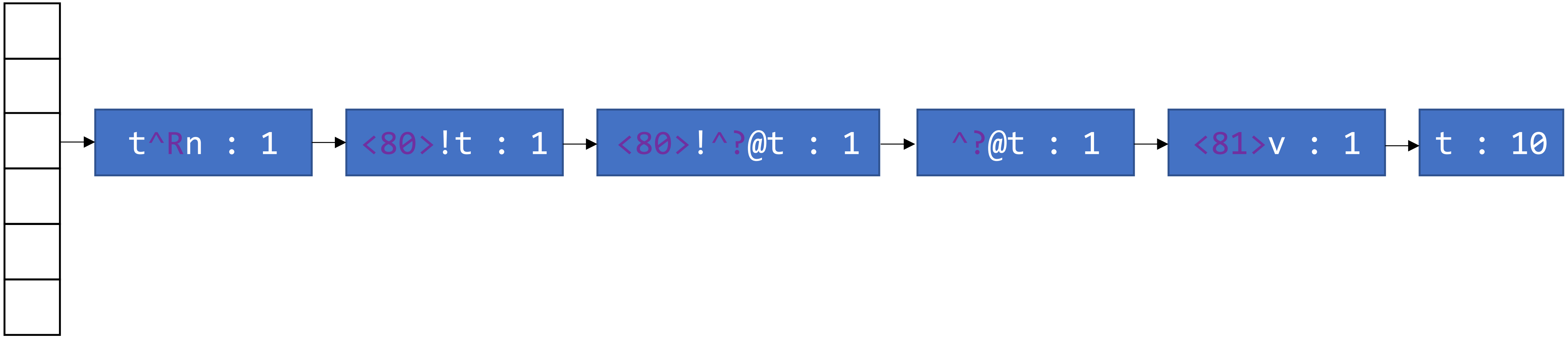
`wf` results: PerfFuzz Finds True Worst Cases

SlowFuzz (single objective maximization) worst case:

```
t r t t s f o Öe r t s f o r t x x t s f o r t x x
```

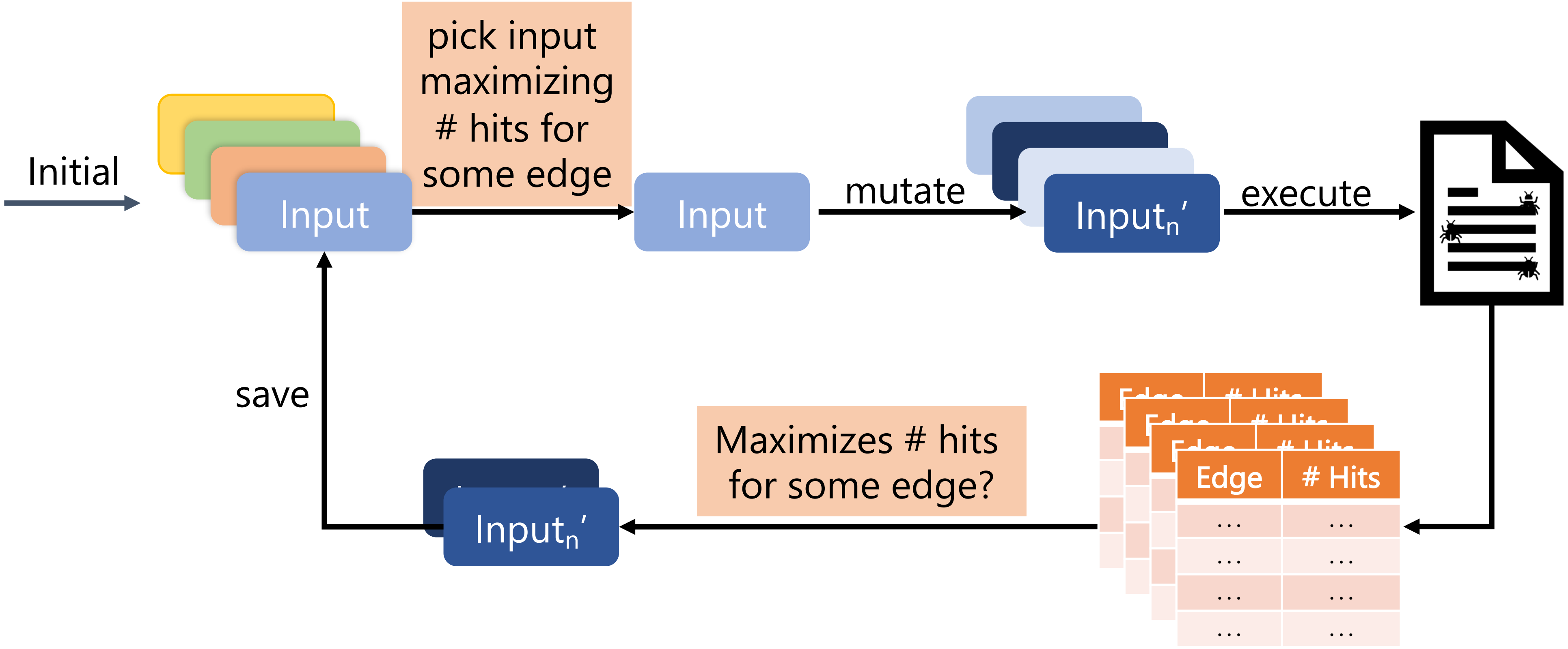
PerfFuzz worst case:

```
t <81>v ^?@t <80>!^?@t <80>!t t^Rn t t t t t t t t t
```

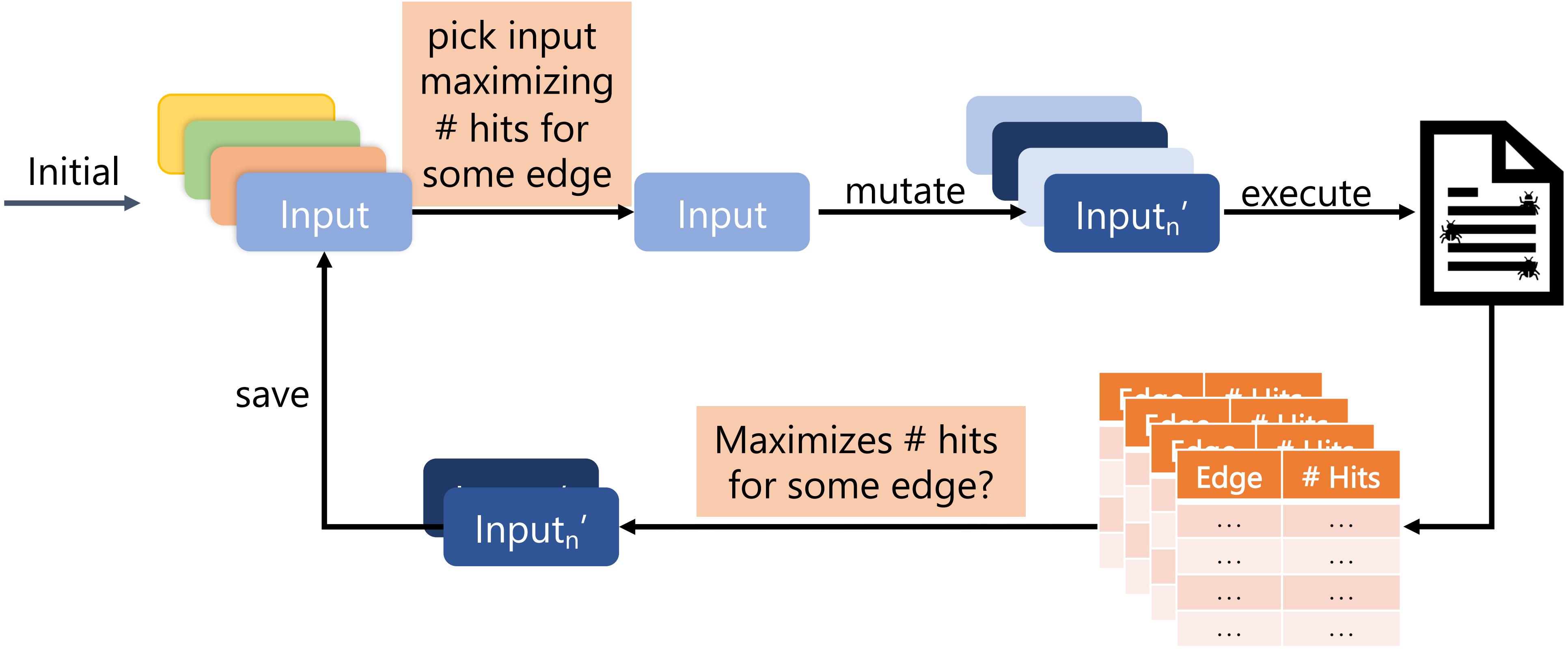


t^Rn : 1 → <80>!t : 1 → <80>!^?@t : 1 → ^?@t : 1 → <81>v : 1 → t : 10

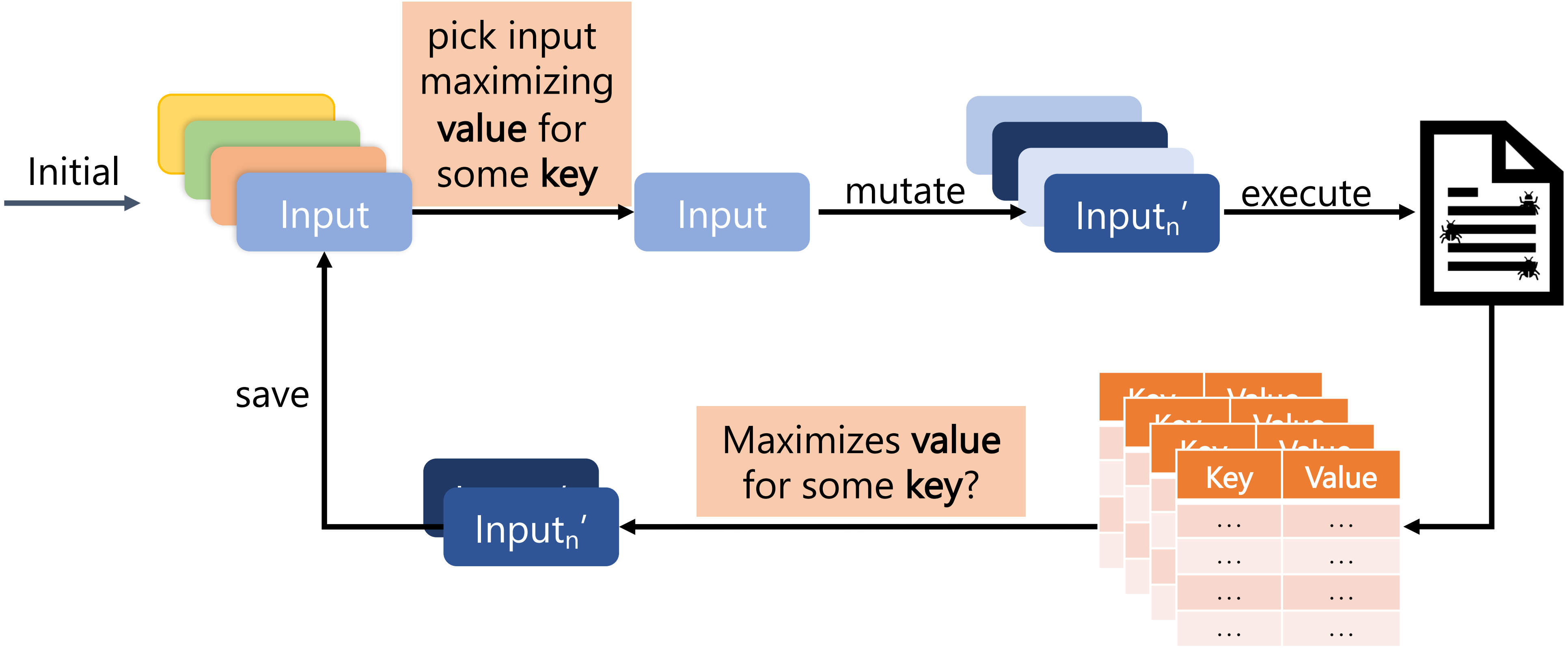
PerfFuzz


Initial
Input
pick input maximizing # hits for some edge
Input
mutate
Input$_n$'
execute
Edge
# Hits
…
Maximizes # hits for some edge?
Input$_n$'
save

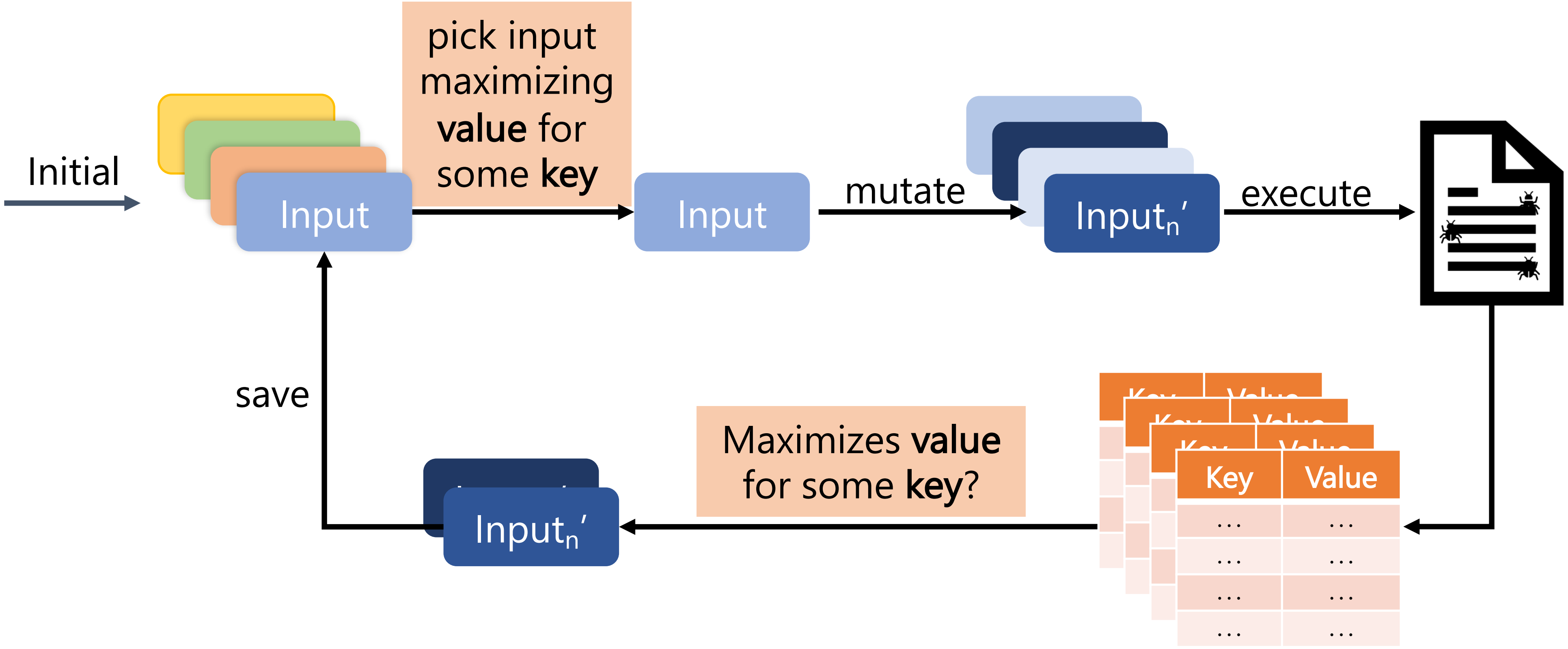
Observation: Algorithm is More General


Initial
Input
pick input maximizing # hits for some edge
Input
mutate
Input$_n$'
execute
Edge
# Hits
…
save
Maximizes # hits for some edge?
Input$_n$'

Observation: Algorithm is More General

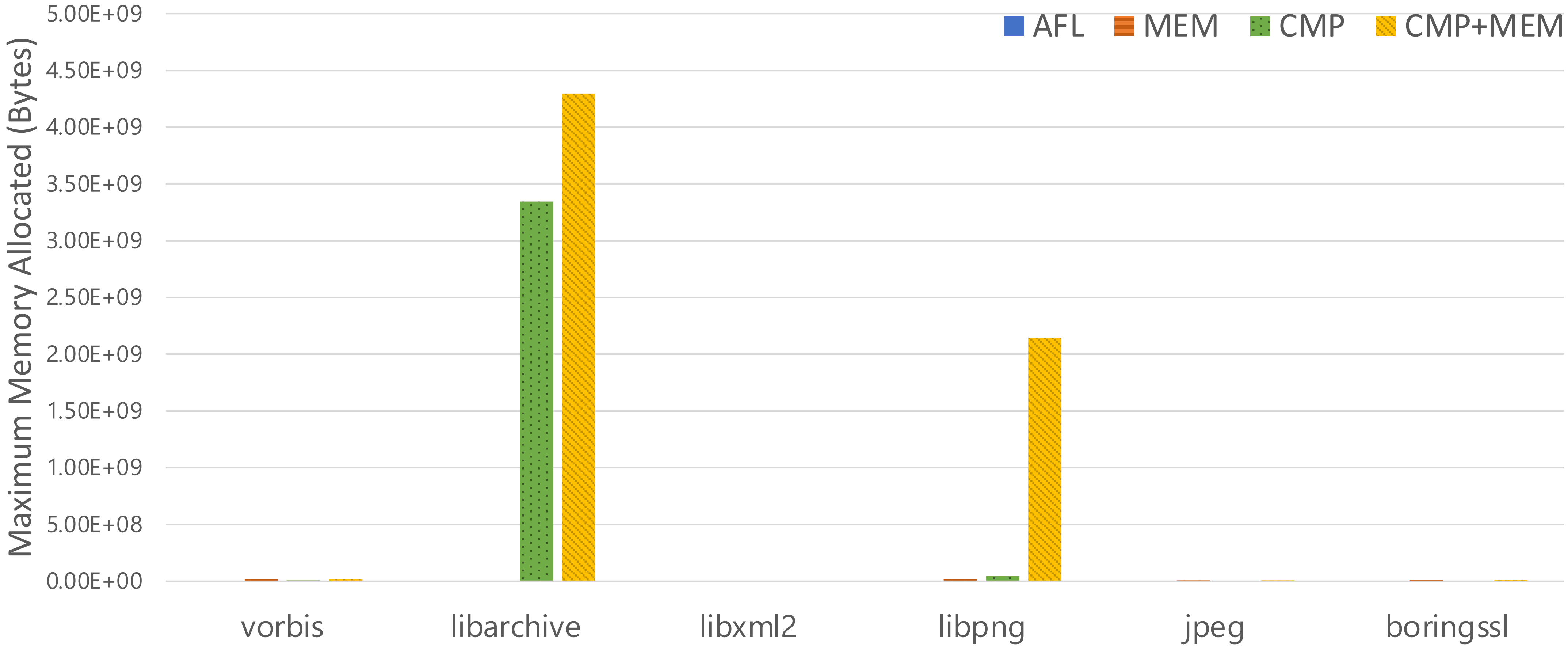

Initial
Input
pick input maximizing **value** for some **key**
Input
mutate
Input$_n$'
execute
Key
Value
…
Maximizes **value** for some **key**?
Input$_n$'
save

FuzzFactory

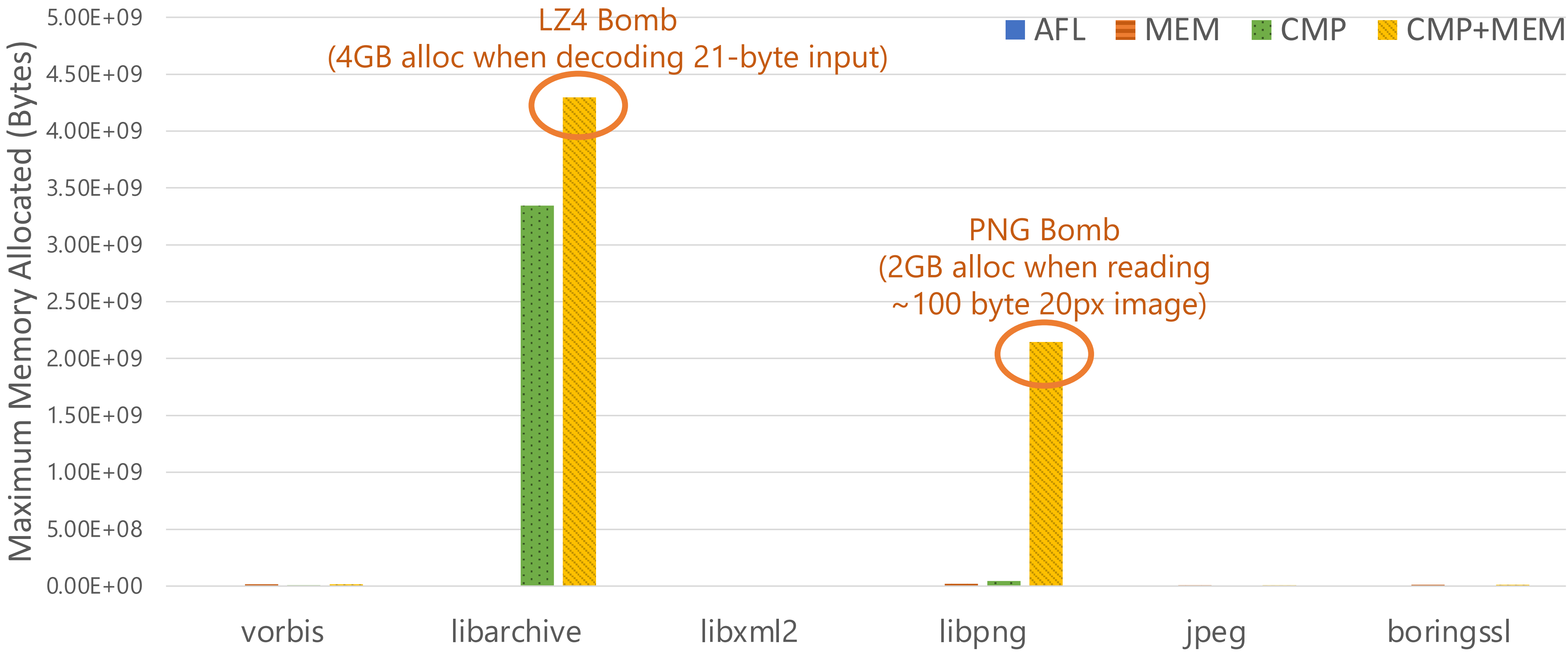


Initial
Input
pick input maximizing **value** for some **key**
Input
mutate
Input$_n$'
execute
Key
Value
…
Maximizes **value** for some **key**?
Input$_n$'
save

Super Fuzzer: Hard Comparisons + Memory Allocations

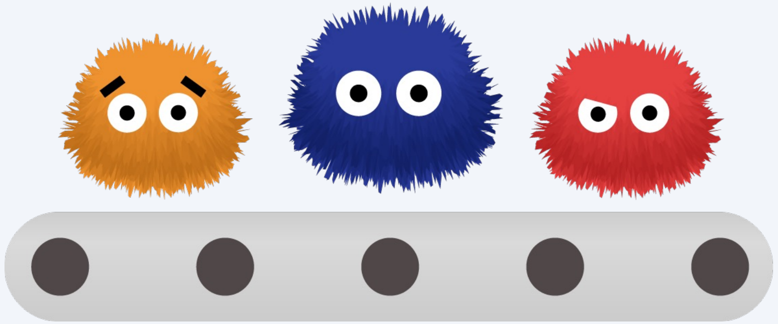


Super Fuzzer: Hard Comparisons + Memory Allocations



LZ4 Bomb
(4GB alloc when decoding 21-byte input)

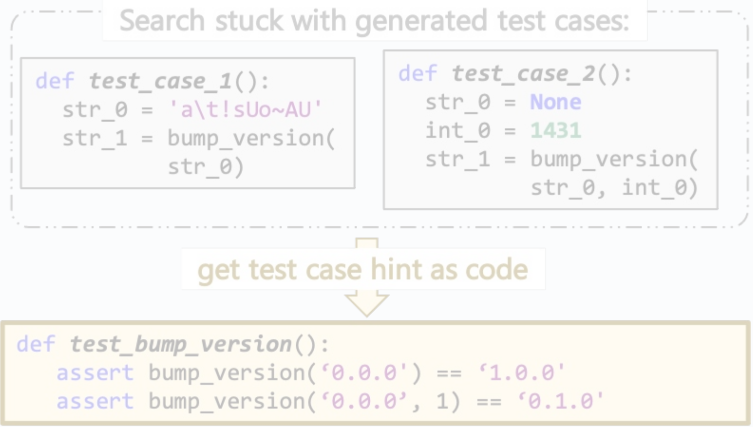
PNG Bomb
(2GB alloc when reading ~100 byte 20px image)



Using ***generalized feedback maps*** to expand ***bugs findable by fuzz testing***

PerfFuzz (ISSTA’18)

FuzzFactory (OOPSLA’19)



Search stuck with generated test cases:

```
def test_case_1():
  str_0 = 'a\t!sUo~AU'
  str_1 = bump_version(
            str_0)
```

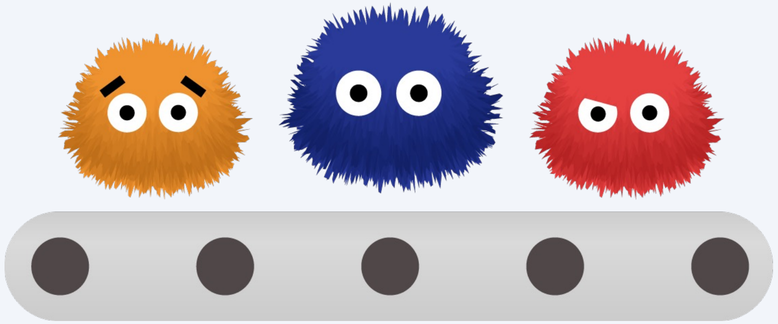
```
def test_case_2():
  str_0 = None
  int_0 = 1431
  str_1 = bump_version(
            str_0, int_0)
```

get test case hint as code

```
def test_bump_version():
  assert bump_version(‘0.0.0') == ‘1.0.0'
  assert bump_version(‘0.0.0’, 1) == ‘0.1.0'
```

Using ***large language models*** to improve ***automated test suite generation***

CodaMOSA (ICSE’23)

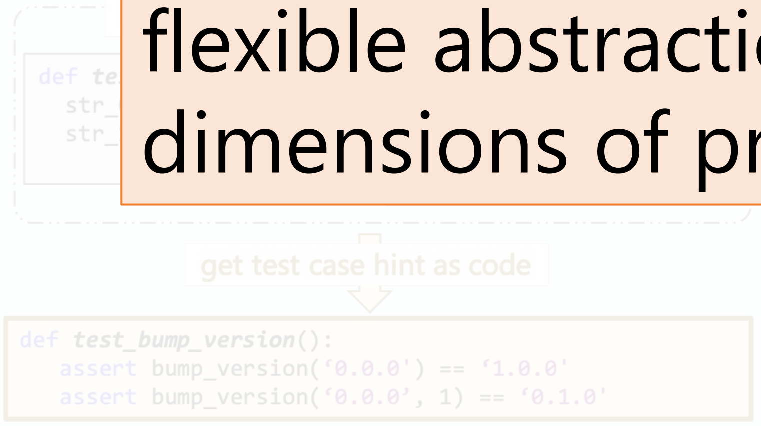


Using ***generalized feedback maps*** to expand ***bugs findable by fuzz testing***

PerfFuzz
(ISSTA'18)

FuzzFactory
(OOPSLA'19)

→ Feedback-directed fuzzing is a flexible abstraction to explore different dimensions of program behavior



```
def te
  str_
  str_
```

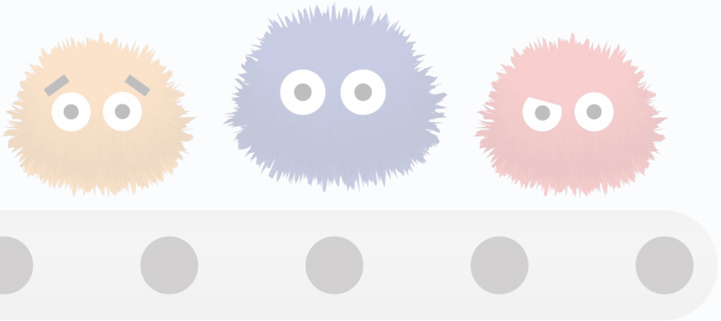
get test case hint as code

```
def test_bump_version():
   assert bump_version('0.0.0') == '1.0.0'
   assert bump_version('0.0.0', 1) == '0.1.0'
```

e

automated test suite generation

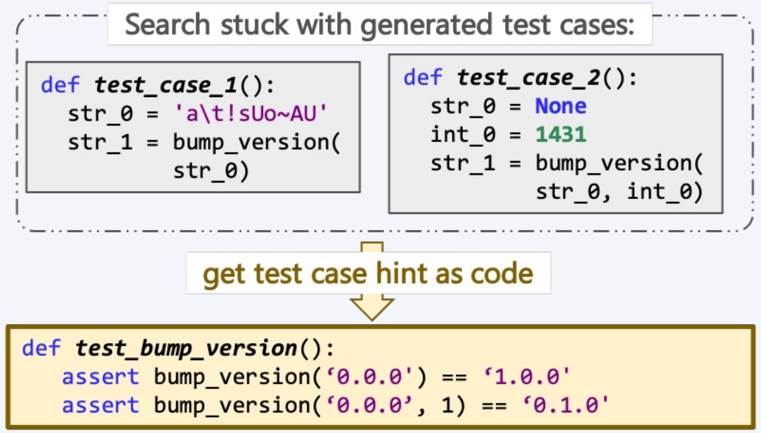
CodaMOSA
(ICSE'23)



Using *generalized feedback maps* to expand *bugs findable by fuzz testing*

PerfFuzz (ISSTA’18)

FuzzFactory (OOPSLA’19)



Search stuck with generated test cases:

```
def test_case_1():
  str_0 = 'a\t!sUo~AU'
  str_1 = bump_version(
          str_0)
```

```
def test_case_2():
  str_0 = None
  int_0 = 1431
  str_1 = bump_version(
          str_0, int_0)
```

get test case hint as code

```
def test_bump_version():
   assert bump_version(‘0.0.0') == ‘1.0.0'
   assert bump_version(‘0.0.0’, 1) == ‘0.1.0'
```

Using *large language models* to improve *automated test suite generation*

CodaMOSA (ICSE’23)

Test Input Generation (PerfFuzz, etc.)

Generate **inputs** to a **parameterized** test function

```
the quick brown the dog
```

```
t <81>v <80>!^?@t t t
```

```
t e q I k b o n t e d g
```

Parameterized Test Function

```
def test_wf(document):
      frequencies = wf(document)
      print(frequencies)
```

Test *Suite* Generation

Generate **test cases** for a file **(e.g., python module, java class)** under test

```
def test_BST_insert():
    tree = None
    tree = BST_insert(tree, 5)
    tree = BST_insert(tree, 3)
    tree = BST_insert(tree, 7)
```

```
def test_BST_search():
    tree = Node(5)
    tree.left = Node(3)
    tree.left.left = Node(1)
    res = BST_search(tree, 3)
```

```
def test_BST_delete():
    tree = Node(5)
    tree.left = Node(3)
    tree.left.left = Node(1)
    tree.left.right = Node(4)
    BST_delete(tree, 4)
```

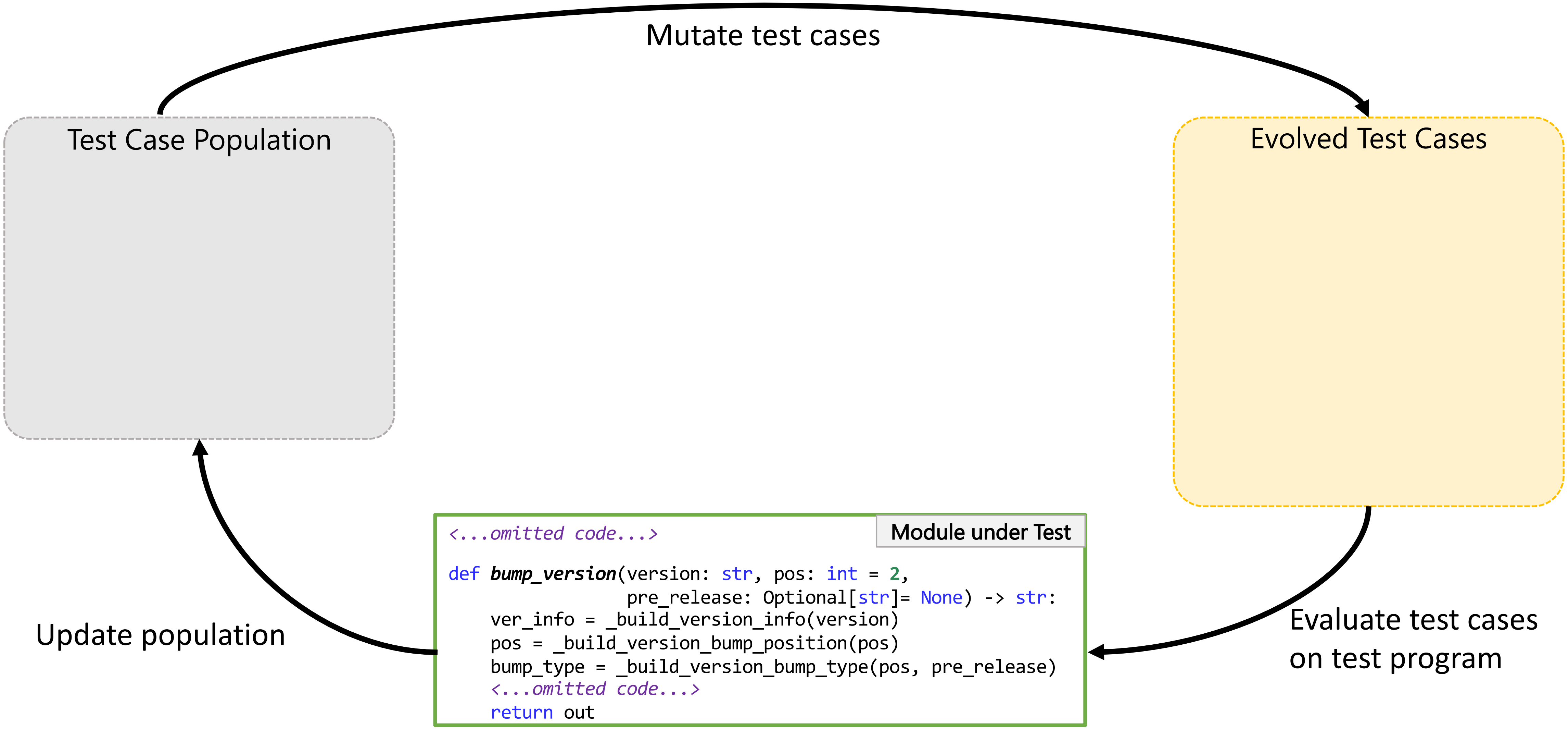
Module Under test

```
def BST_insert(tree, to_add):
    # Insert to_add into tree
    <...>

def BST_search(tree, to_search):
    # Search tree for to_search
    <...>

def BST_delete(tree, to_delete):
    # Delete to_delete from tree
    <...>
```

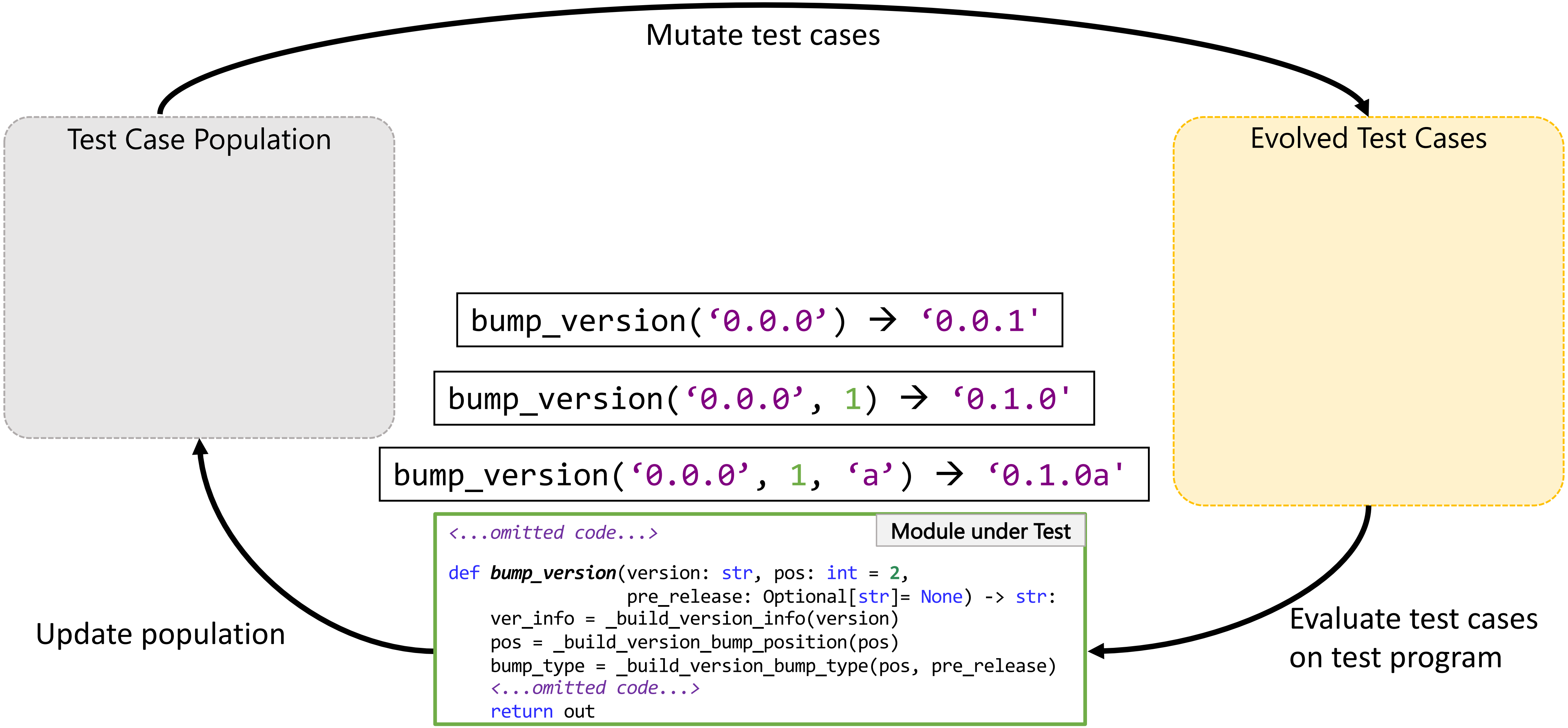
Search-Based Test Suite Generation



```
<...omitted code...>

def bump_version(version: str, pos: int = 2,
                 pre_release: Optional[str]= None) -> str:
    ver_info = _build_version_info(version)
    pos = _build_version_bump_position(pos)
    bump_type = _build_version_bump_type(pos, pre_release)
    <...omitted code...>
    return out
```

Example: Expected Behavior of Function



Mutate test cases

Test Case Population

Evolved Test Cases

bump_version(‘0.0.0’) → ‘0.0.1'

bump_version(‘0.0.0’, 1) → ‘0.1.0'

bump_version(‘0.0.0’, 1, ‘a’) → ‘0.1.0a'

Module under Test

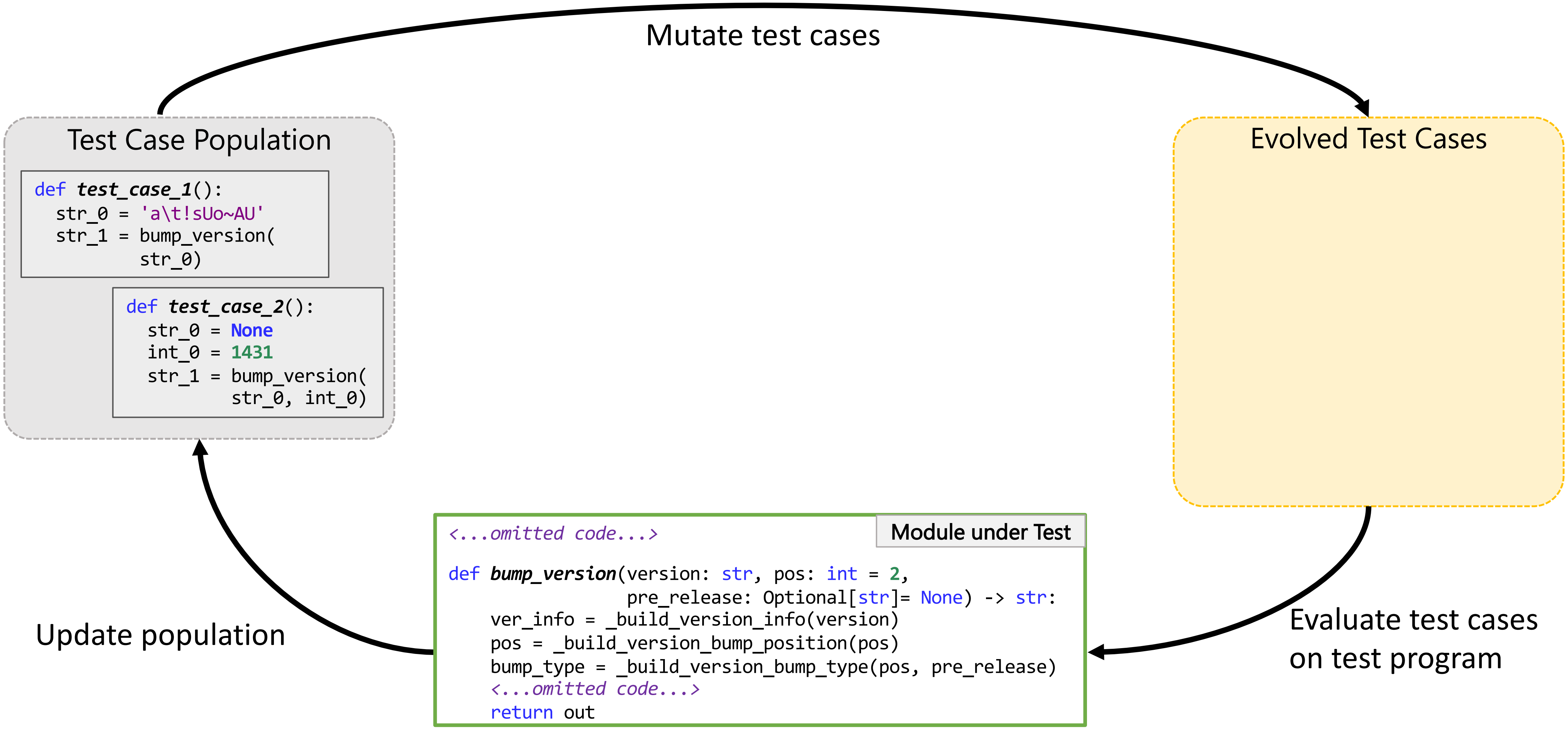
```
<...omitted code...>

def bump_version(version: str, pos: int = 2,
                 pre_release: Optional[str]= None) -> str:
    ver_info = _build_version_info(version)
    pos = _build_version_bump_position(pos)
    bump_type = _build_version_bump_type(pos, pre_release)
    <...omitted code...>
    return out
```

Update population

Evaluate test cases
on test program

Initialize Test Population



Mutate test cases

Test Case Population

```
def test_case_1():
  str_0 = 'a\t!sUo~AU'
  str_1 = bump_version(
            str_0)
```

```
def test_case_2():
  str_0 = None
  int_0 = 1431
  str_1 = bump_version(
            str_0, int_0)
```

Evolved Test Cases

Evaluate test cases on test program

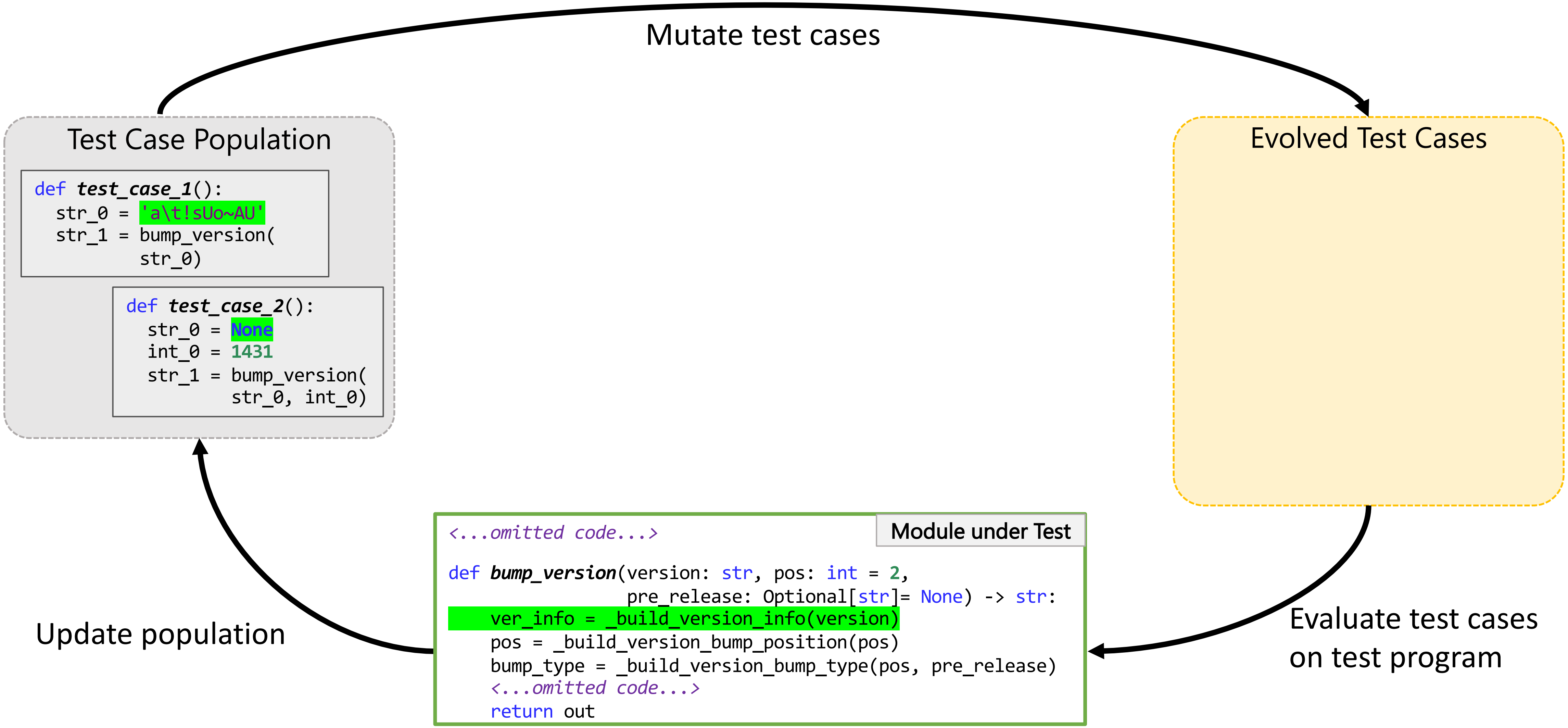
Module under Test

```
<...omitted code...>

def bump_version(version: str, pos: int = 2,
                 pre_release: Optional[str]= None) -> str:
    ver_info = _build_version_info(version)
    pos = _build_version_bump_position(pos)
    bump_type = _build_version_bump_type(pos, pre_release)
    <...omitted code...>
    return out
```

Update population

Current Tests have Low Coverage



Mutate test cases

Test Case Population

```
def test_case_1():
  str_0 = 'a\t!sUo~AU'
  str_1 = bump_version(
          str_0)
```

```
def test_case_2():
  str_0 = None
  int_0 = 1431
  str_1 = bump_version(
          str_0, int_0)
```

Evolved Test Cases

Evaluate test cases on test program

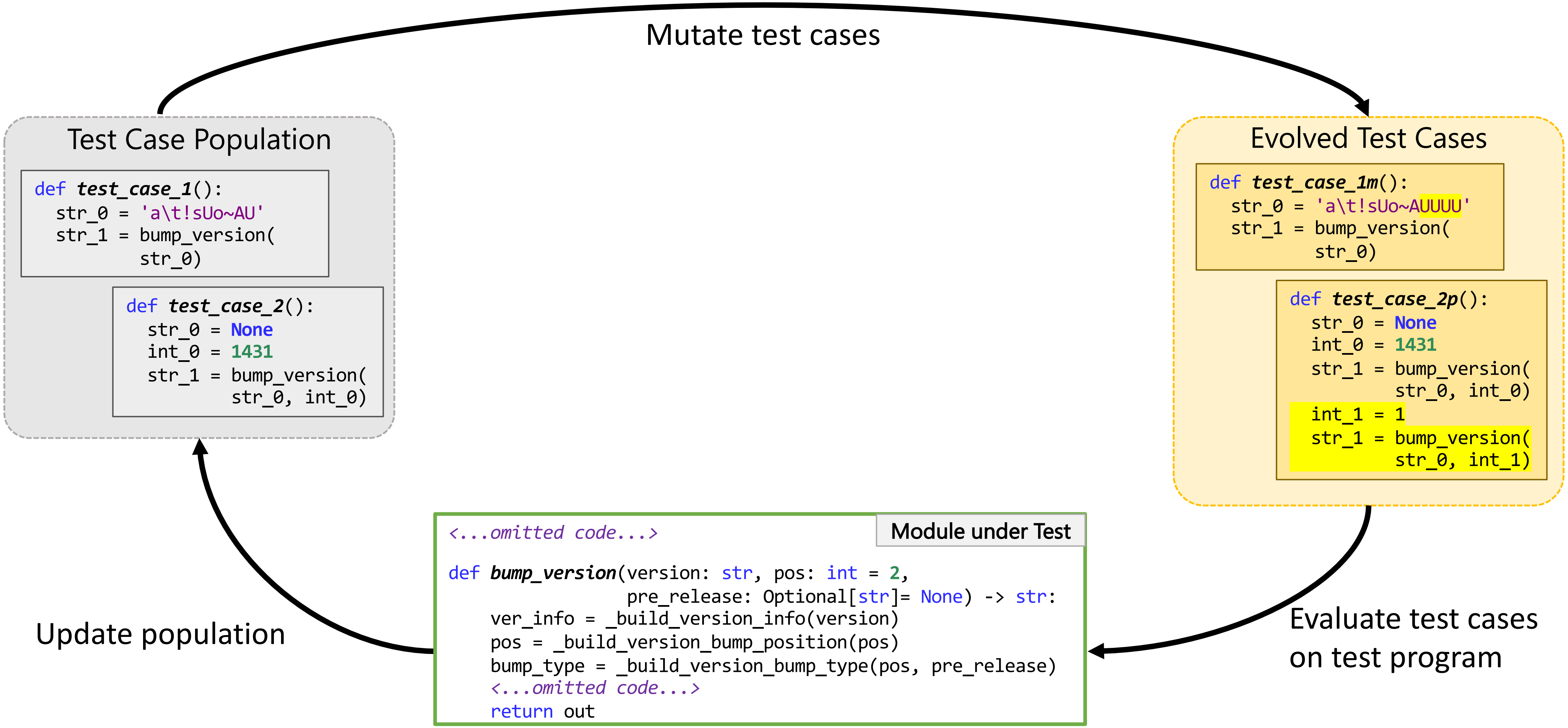
Update population

Module under Test

```
<...omitted code...>

def bump_version(version: str, pos: int = 2,
                 pre_release: Optional[str]= None) -> str:
    ver_info = _build_version_info(version)
    pos = _build_version_bump_position(pos)
    bump_type = _build_version_bump_type(pos, pre_release)
    <...omitted code...>
    return out
```

Create New Test Cases via Mutation



Mutate test cases

Test Case Population

```
def test_case_1():
  str_0 = 'a\t!sUo~AU'
  str_1 = bump_version(
          str_0)
```

```
def test_case_2():
  str_0 = None
  int_0 = 1431
  str_1 = bump_version(
          str_0, int_0)
```

Evolved Test Cases

```
def test_case_1m():
  str_0 = 'a\t!sUo~AUUUU'
  str_1 = bump_version(
          str_0)
```

```
def test_case_2p():
  str_0 = None
  int_0 = 1431
  str_1 = bump_version(
          str_0, int_0)
  int_1 = 1
  str_1 = bump_version(
          str_0, int_1)
```

Evaluate test cases on test program

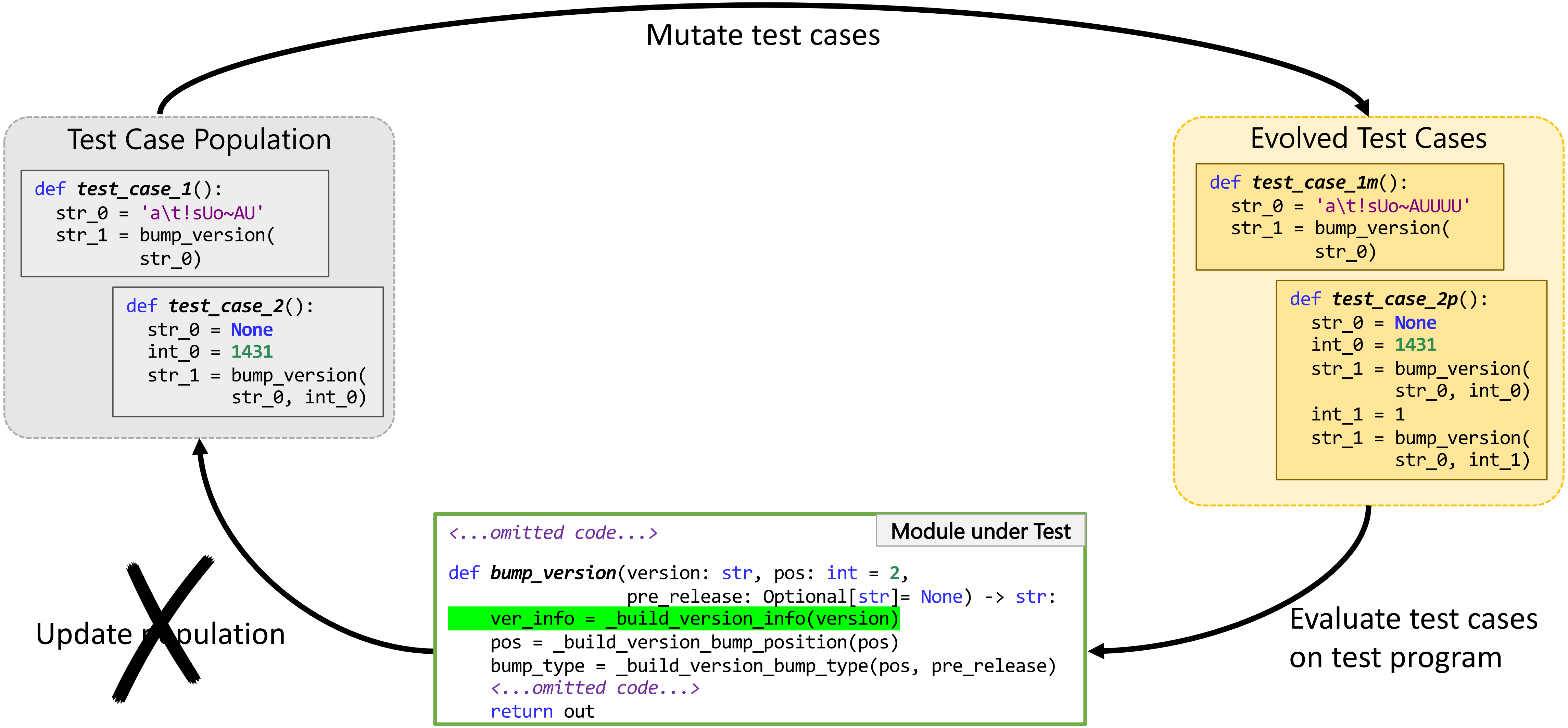
Module under Test

```
<...omitted code...>

def bump_version(version: str, pos: int = 2,
                 pre_release: Optional[str]= None) -> str:
    ver_info = _build_version_info(version)
    pos = _build_version_bump_position(pos)
    bump_type = _build_version_bump_type(pos, pre_release)
    <...omitted code...>
    return out
```

Update population

Mutation Unable to Increase Coverage



Mutate test cases

Test Case Population

```
def test_case_1():
  str_0 = 'a\t!sUo~AU'
  str_1 = bump_version(
          str_0)
```

```
def test_case_2():
  str_0 = None
  int_0 = 1431
  str_1 = bump_version(
          str_0, int_0)
```

Evolved Test Cases

```
def test_case_1m():
  str_0 = 'a\t!sUo~AUUUU'
  str_1 = bump_version(
          str_0)
```

```
def test_case_2p():
  str_0 = None
  int_0 = 1431
  str_1 = bump_version(
          str_0, int_0)
  int_1 = 1
  str_1 = bump_version(
          str_0, int_1)
```

Evaluate test cases on test program

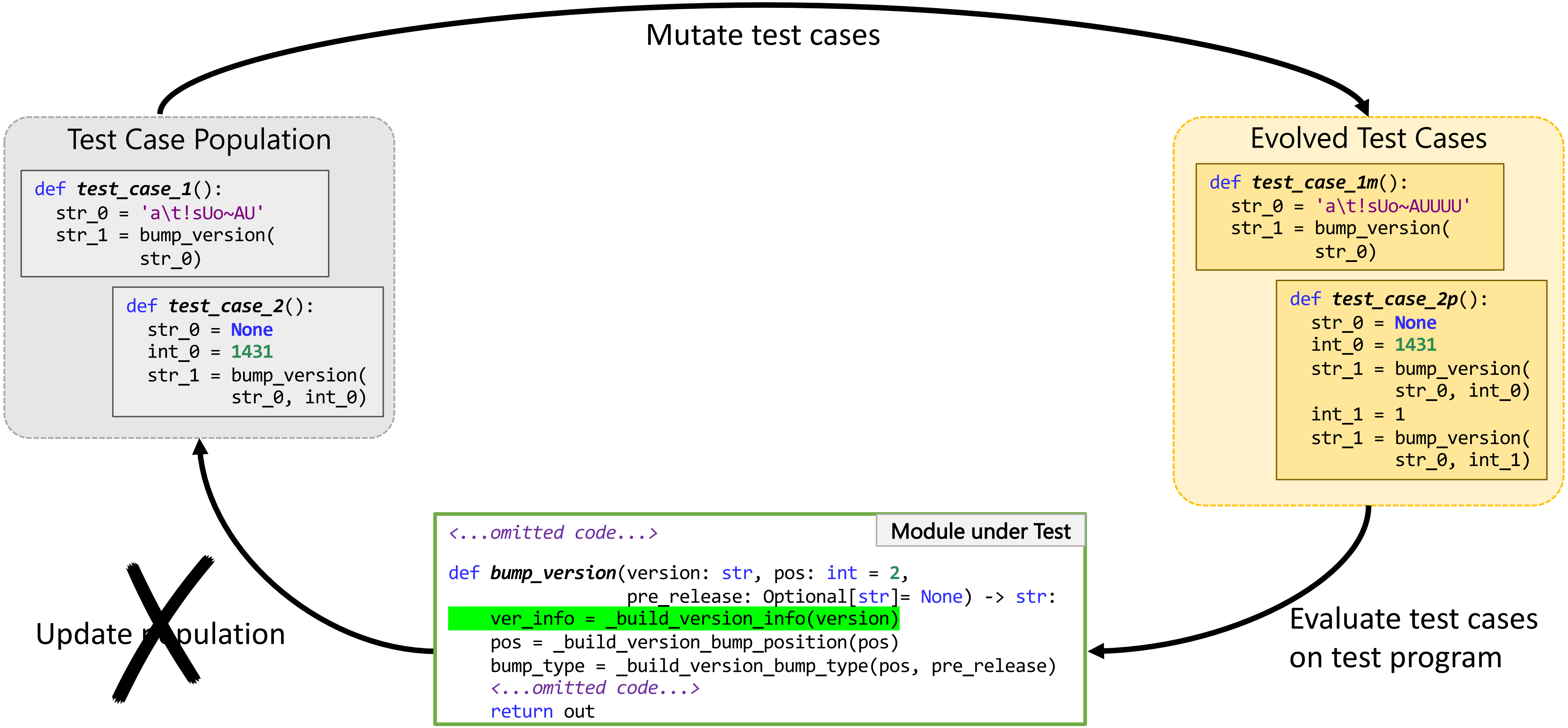
Module under Test

```
<...omitted code...>

def bump_version(version: str, pos: int = 2,
                 pre_release: Optional[str]= None) -> str:
    ver_info = _build_version_info(version)
    pos = _build_version_bump_position(pos)
    bump_type = _build_version_bump_type(pos, pre_release)
    <...omitted code...>
    return out
```

~~Update population~~

Search stalled. What to do now?



Mutate test cases

Test Case Population

```python
def test_case_1():
  str_0 = 'a\t!sUo~AU'
  str_1 = bump_version(
          str_0)
```

```python
def test_case_2():
  str_0 = None
  int_0 = 1431
  str_1 = bump_version(
          str_0, int_0)
```

Evolved Test Cases

```python
def test_case_1m():
  str_0 = 'a\t!sUo~AUUUU'
  str_1 = bump_version(
          str_0)
```

```python
def test_case_2p():
  str_0 = None
  int_0 = 1431
  str_1 = bump_version(
          str_0, int_0)
  int_1 = 1
  str_1 = bump_version(
          str_0, int_1)
```

Evaluate test cases on test program

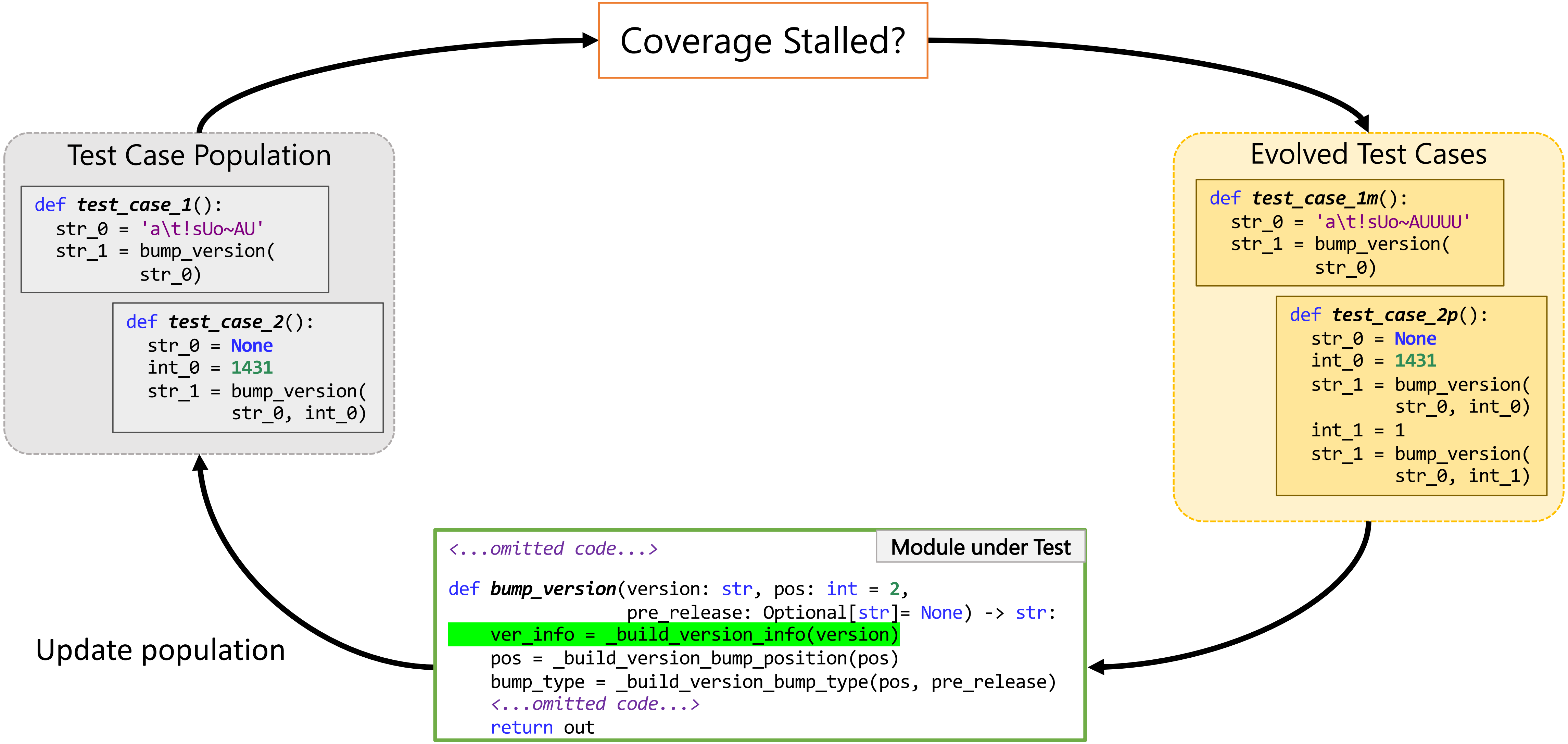
Module under Test

```python
<...omitted code...>

def bump_version(version: str, pos: int = 2,
                 pre_release: Optional[str]= None) -> str:
    ver_info = _build_version_info(version)
    pos = _build_version_bump_position(pos)
    bump_type = _build_version_bump_type(pos, pre_release)
    <...omitted code...>
    return out
```

~~Update population~~

CodaMOSA: Asks for hints when stuck



Coverage Stalled?

Test Case Population

```
def test_case_1():
  str_0 = 'a\t!sUo~AU'
  str_1 = bump_version(
          str_0)
```

```
def test_case_2():
  str_0 = None
  int_0 = 1431
  str_1 = bump_version(
          str_0, int_0)
```

Evolved Test Cases

```
def test_case_1m():
  str_0 = 'a\t!sUo~AUUUU'
  str_1 = bump_version(
          str_0)
```

```
def test_case_2p():
  str_0 = None
  int_0 = 1431
  str_1 = bump_version(
          str_0, int_0)
  int_1 = 1
  str_1 = bump_version(
          str_0, int_1)
```

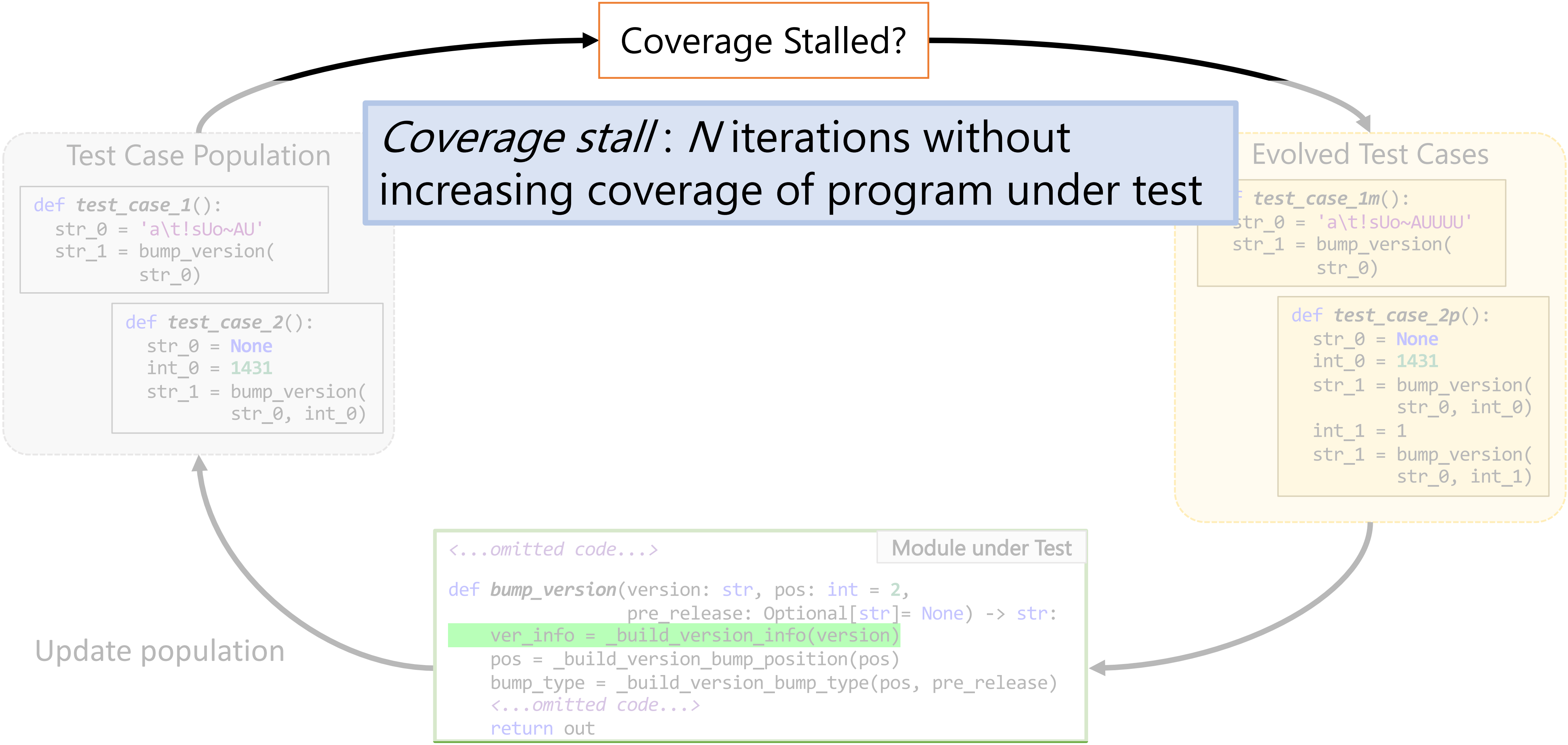
Module under Test

```
<...omitted code...>

def bump_version(version: str, pos: int = 2,
                 pre_release: Optional[str]= None) -> str:
    ver_info = _build_version_info(version)
    pos = _build_version_bump_position(pos)
    bump_type = _build_version_bump_type(pos, pre_release)
    <...omitted code...>
    return out
```

Update population

CodaMOSA: Asks for hints when stuck



Coverage stall: *N* iterations without increasing coverage of program under test

```
def test_case_1():
  str_0 = 'a\t!sUo~AU'
  str_1 = bump_version(
          str_0)
```

```
def test_case_2():
  str_0 = None
  int_0 = 1431
  str_1 = bump_version(
          str_0, int_0)
```

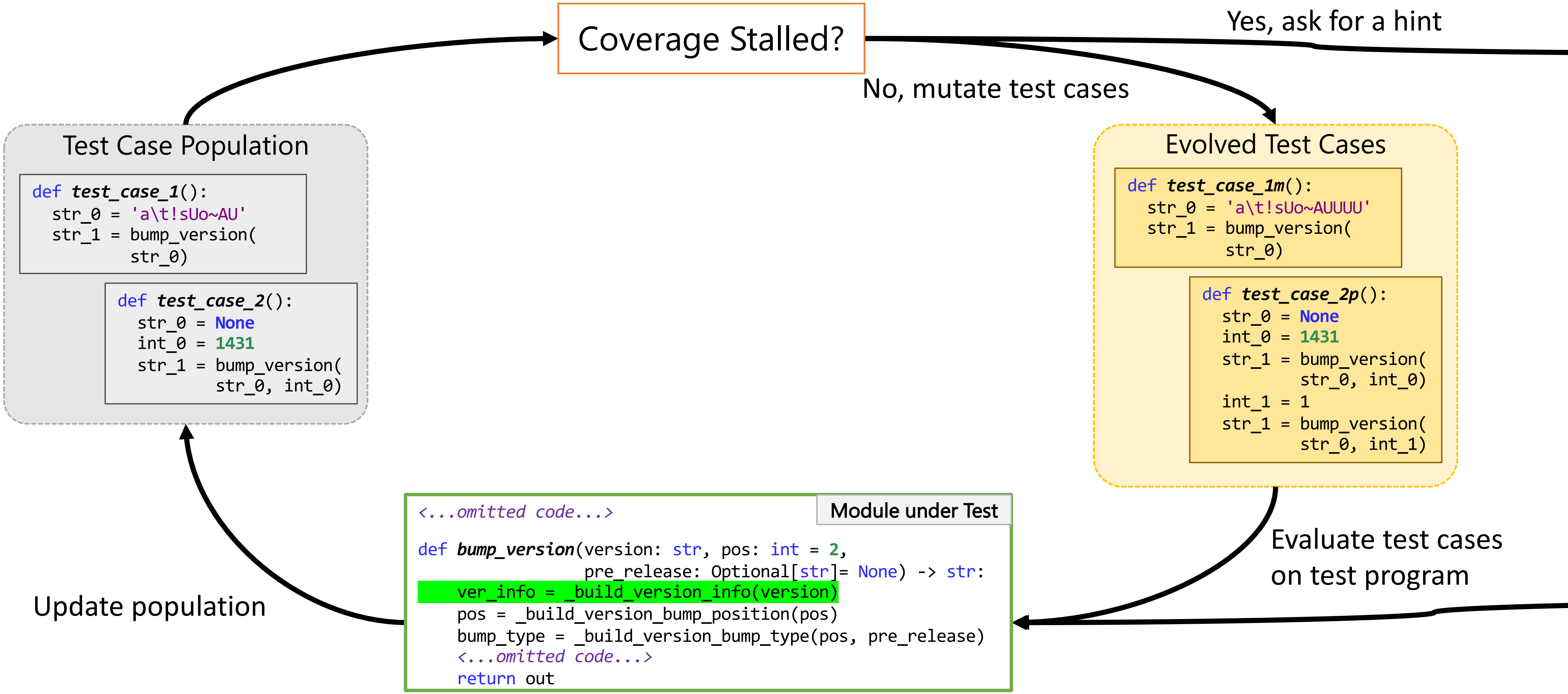
```
f test_case_1m():
str_0 = 'a\t!sUo~AUUUU'
str_1 = bump_version(
        str_0)
```

```
def test_case_2p():
  str_0 = None
  int_0 = 1431
  str_1 = bump_version(
          str_0, int_0)
  int_1 = 1
  str_1 = bump_version(
          str_0, int_1)
```

```
<...omitted code...>

def bump_version(version: str, pos: int = 2,
                 pre_release: Optional[str]= None) -> str:
    ver_info = _build_version_info(version)
    pos = _build_version_bump_position(pos)
    bump_type = _build_version_bump_type(pos, pre_release)
    <...omitted code...>
    return out
```

CodaMOSA: Asks for hints when stuck



Coverage Stalled?

Yes, ask for a hint

No, mutate test cases

Test Case Population

```
def test_case_1():
  str_0 = 'a\t!sUo~AU'
  str_1 = bump_version(
          str_0)
```

```
def test_case_2():
  str_0 = None
  int_0 = 1431
  str_1 = bump_version(
          str_0, int_0)
```

Evolved Test Cases

```
def test_case_1m():
  str_0 = 'a\t!sUo~AUUUU'
  str_1 = bump_version(
          str_0)
```

```
def test_case_2p():
  str_0 = None
  int_0 = 1431
  str_1 = bump_version(
          str_0, int_0)
  int_1 = 1
  str_1 = bump_version(
          str_0, int_1)
```

Evaluate test cases on test program

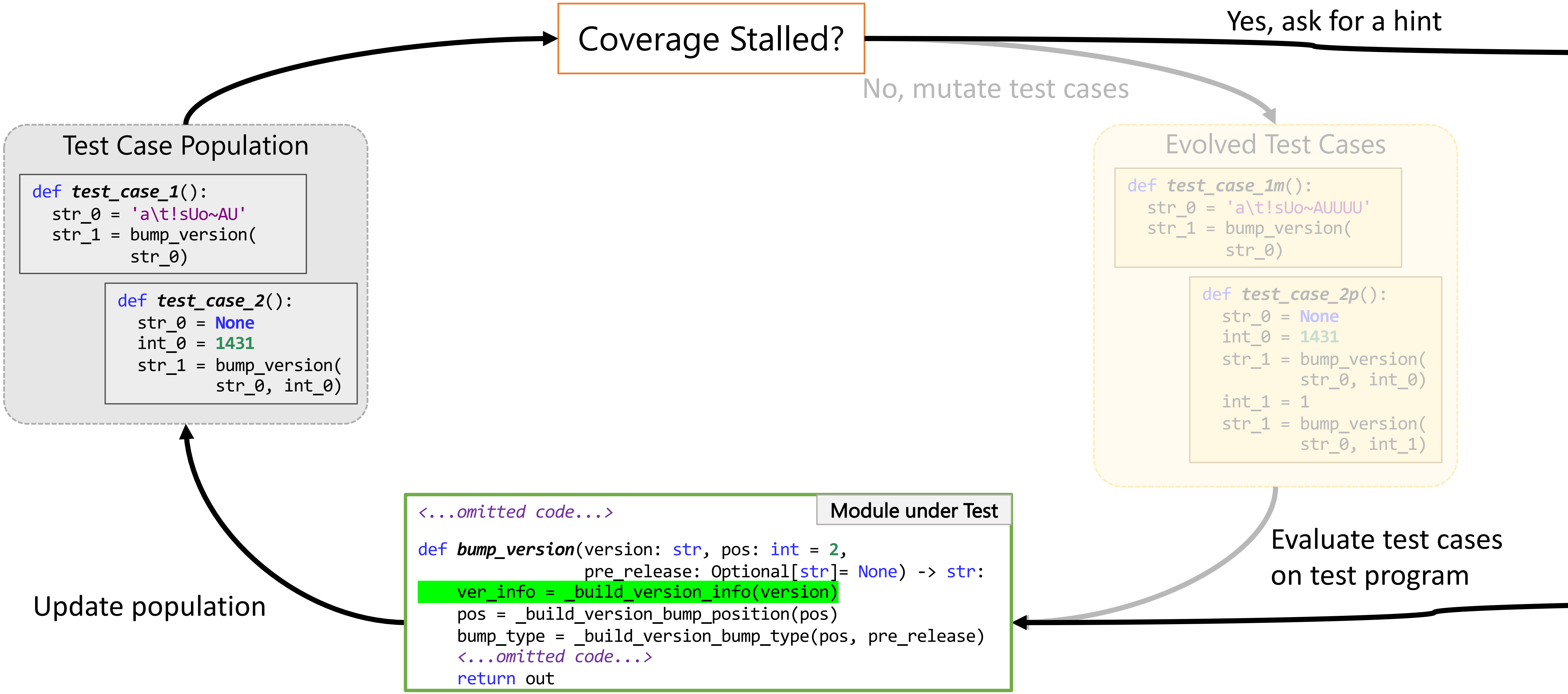
Module under Test

```
<...omitted code...>

def bump_version(version: str, pos: int = 2,
                 pre_release: Optional[str]= None) -> str:
    ver_info = _build_version_info(version)
    pos = _build_version_bump_position(pos)
    bump_type = _build_version_bump_type(pos, pre_release)
    <...omitted code...>
    return out
```

Update population

Search Stalled



Coverage Stalled?

Yes, ask for a hint

No, mutate test cases

Test Case Population

```
def test_case_1():
  str_0 = 'a\t!sUo~AU'
  str_1 = bump_version(
          str_0)
```

```
def test_case_2():
  str_0 = None
  int_0 = 1431
  str_1 = bump_version(
          str_0, int_0)
```

Evolved Test Cases

```
def test_case_1m():
  str_0 = 'a\t!sUo~AUUUU'
  str_1 = bump_version(
          str_0)
```

```
def test_case_2p():
  str_0 = None
  int_0 = 1431
  str_1 = bump_version(
          str_0, int_0)
  int_1 = 1
  str_1 = bump_version(
          str_0, int_1)
```

Evaluate test cases on test program

Module under Test

```
<...omitted code...>

def bump_version(version: str, pos: int = 2,
                 pre_release: Optional[str]= None) -> str:
    ver_info = _build_version_info(version)
    pos = _build_version_bump_position(pos)
    bump_type = _build_version_bump_type(pos, pre_release)
    <...omitted code...>
    return out
```

Update population

Time to Ask for a Hint

Coverage Stalled?

Yes, ask for a hint

Prompt to create low-coverage functions tests

Ask Codex

aw Model Output ("natural" code)

No mutate test cases

Test Case Population

test_case_1

test_case_2

Codex, LLM trained on code
Released Aug 2021

Module unde

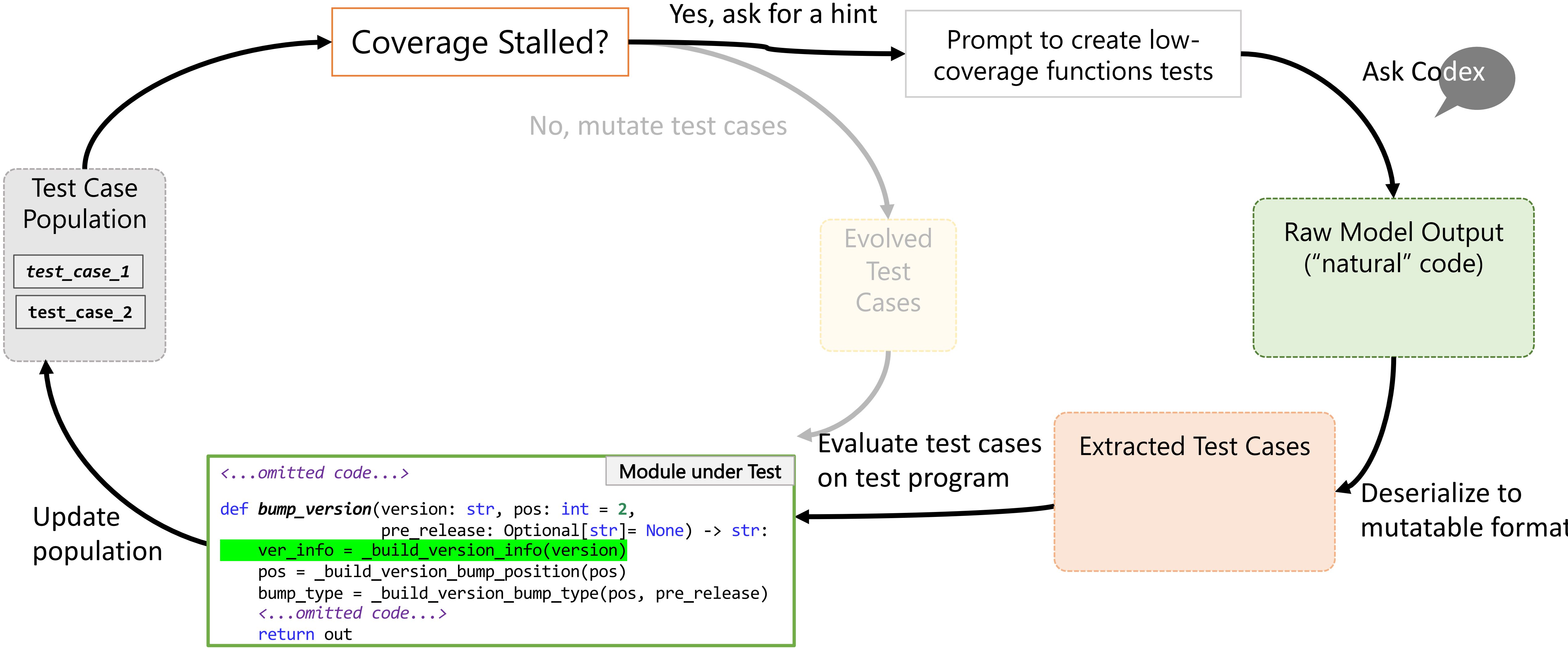
```
<...omitted code...>

def bump_version(version: str, pos: int = 2,
                 pre_release: Optional[str]= None) -> s
    ver_info = _build_version_info(version)
    pos = _build_version_bump_position(pos)
    bump_type = _build_version_bump_type(pos, pre_release)
    <...omitted code...>
    return out
```

Update population

Deserialize to mutatable format

Time to Ask for a Hint



Coverage Stalled?

Yes, ask for a hint

Prompt to create low-coverage functions tests

Ask Codex

Raw Model Output (“natural” code)

Deserialize to mutatable format

Extracted Test Cases

Evaluate test cases on test program

No, mutate test cases

Evolved Test Cases

Module under Test

```
<...omitted code...>

def bump_version(version: str, pos: int = 2,
                 pre_release: Optional[str]= None) -> str:
    ver_info = _build_version_info(version)
    pos = _build_version_bump_position(pos)
    bump_type = _build_version_bump_type(pos, pre_release)
    <...omitted code...>
    return out
```

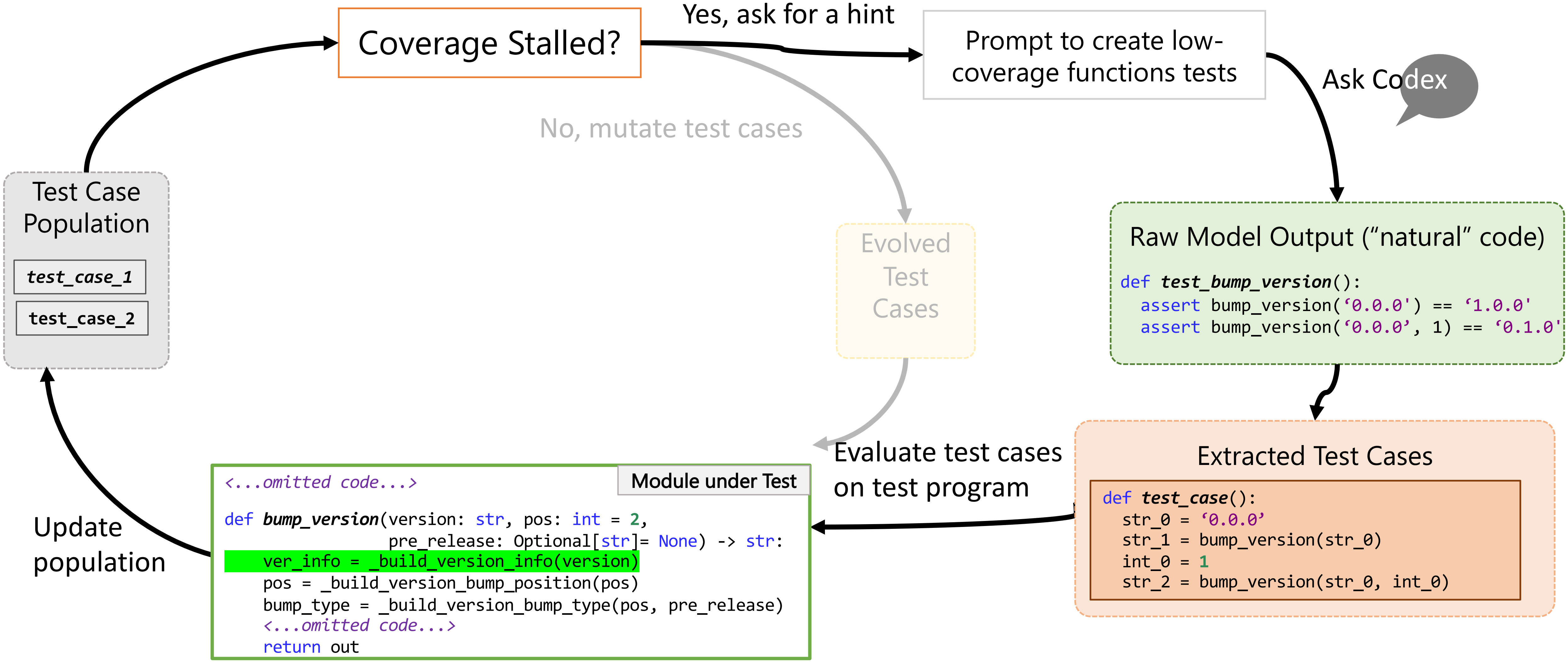
Update population

Test Case Population

test_case_1

test_case_2

Time to Ask for a Hint



Coverage Stalled?

Yes, ask for a hint

Prompt to create low-coverage functions tests

Ask Codex

No, mutate test cases

Test Case Population

test_case_1

test_case_2

Evolved Test Cases

Raw Model Output (“natural” code)

```
def test_bump_version():
  assert bump_version(‘0.0.0') == ‘1.0.0'
  assert bump_version(‘0.0.0’, 1) == ‘0.1.0'
```

Extracted Test Cases

```
def test_case():
  str_0 = ‘0.0.0’
  str_1 = bump_version(str_0)
  int_0 = 1
  str_2 = bump_version(str_0, int_0)
```

Evaluate test cases on test program

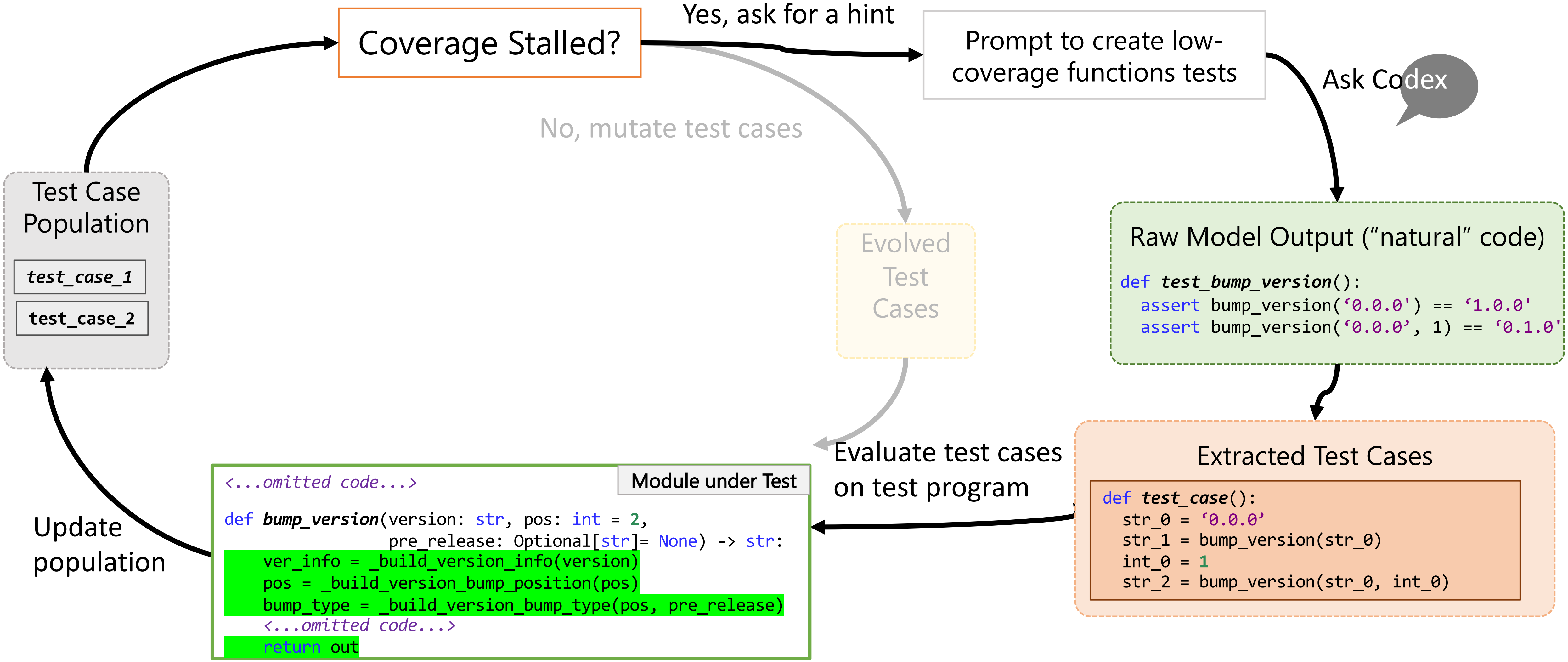
Update population

Module under Test

```
<...omitted code...>

def bump_version(version: str, pos: int = 2,
                 pre_release: Optional[str]= None) -> str:
    ver_info = _build_version_info(version)
    pos = _build_version_bump_position(pos)
    bump_type = _build_version_bump_type(pos, pre_release)
    <...omitted code...>
    return out
```

Suggested Test Case Increases Coverage



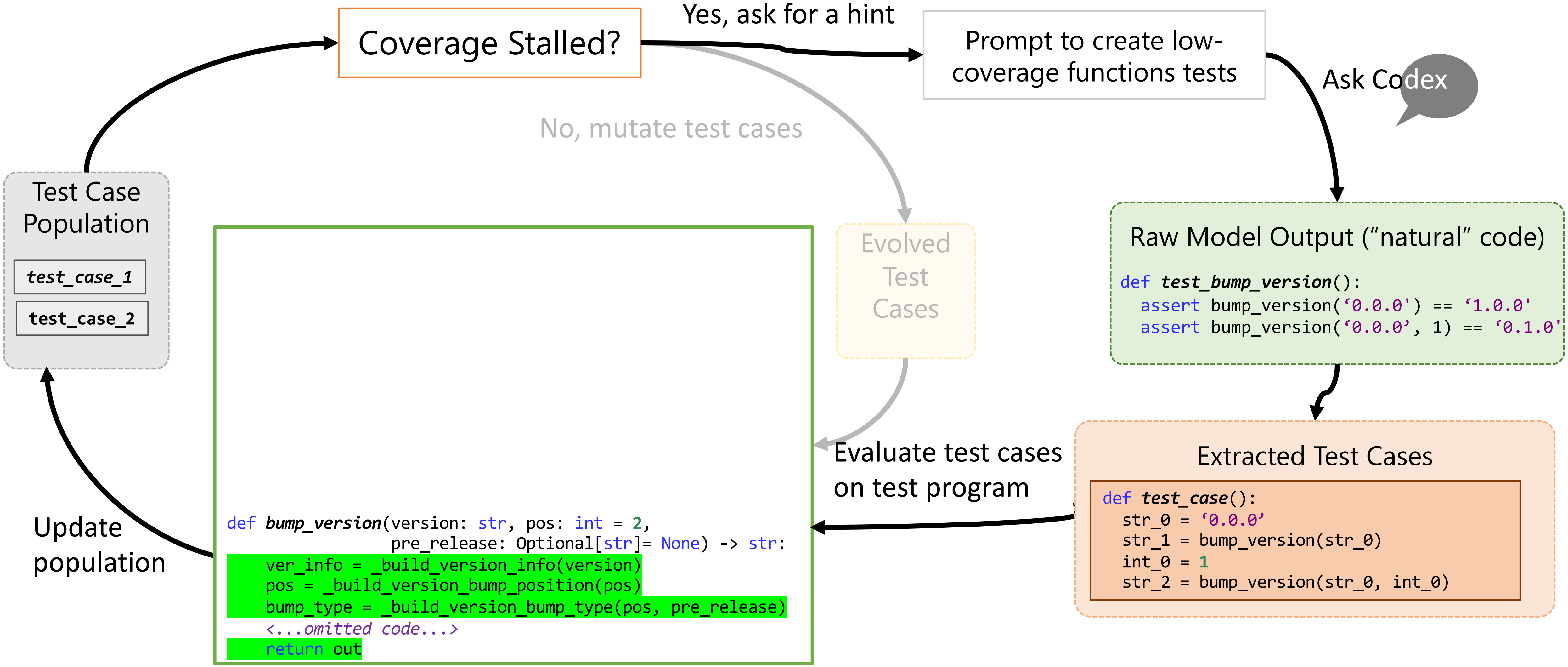
```
def test_bump_version():
  assert bump_version(‘0.0.0') == ‘1.0.0'
  assert bump_version(‘0.0.0’, 1) == ‘0.1.0'
```

```
def test_case():
  str_0 = ‘0.0.0’
  str_1 = bump_version(str_0)
  int_0 = 1
  str_2 = bump_version(str_0, int_0)
```

```
<...omitted code...>

def bump_version(version: str, pos: int = 2,
                 pre_release: Optional[str]= None) -> str:
    ver_info = _build_version_info(version)
    pos = _build_version_bump_position(pos)
    bump_type = _build_version_bump_type(pos, pre_release)
    <...omitted code...>
    return out
```

Suggested Test Case Increases Coverage



Coverage Stalled?

Yes, ask for a hint

Prompt to create low-coverage functions tests

Ask Codex

Raw Model Output (“natural” code)

```
def test_bump_version():
  assert bump_version(‘0.0.0') == ‘1.0.0'
  assert bump_version(‘0.0.0’, 1) == ‘0.1.0'
```

Extracted Test Cases

```
def test_case():
  str_0 = ‘0.0.0’
  str_1 = bump_version(str_0)
  int_0 = 1
  str_2 = bump_version(str_0, int_0)
```

Evaluate test cases on test program

```
def bump_version(version: str, pos: int = 2,
                 pre_release: Optional[str]= None) -> str:
    ver_info = _build_version_info(version)
    pos = _build_version_bump_position(pos)
    bump_type = _build_version_bump_type(pos, pre_release)
    <...omitted code...>
    return out
```

Update population

Test Case Population

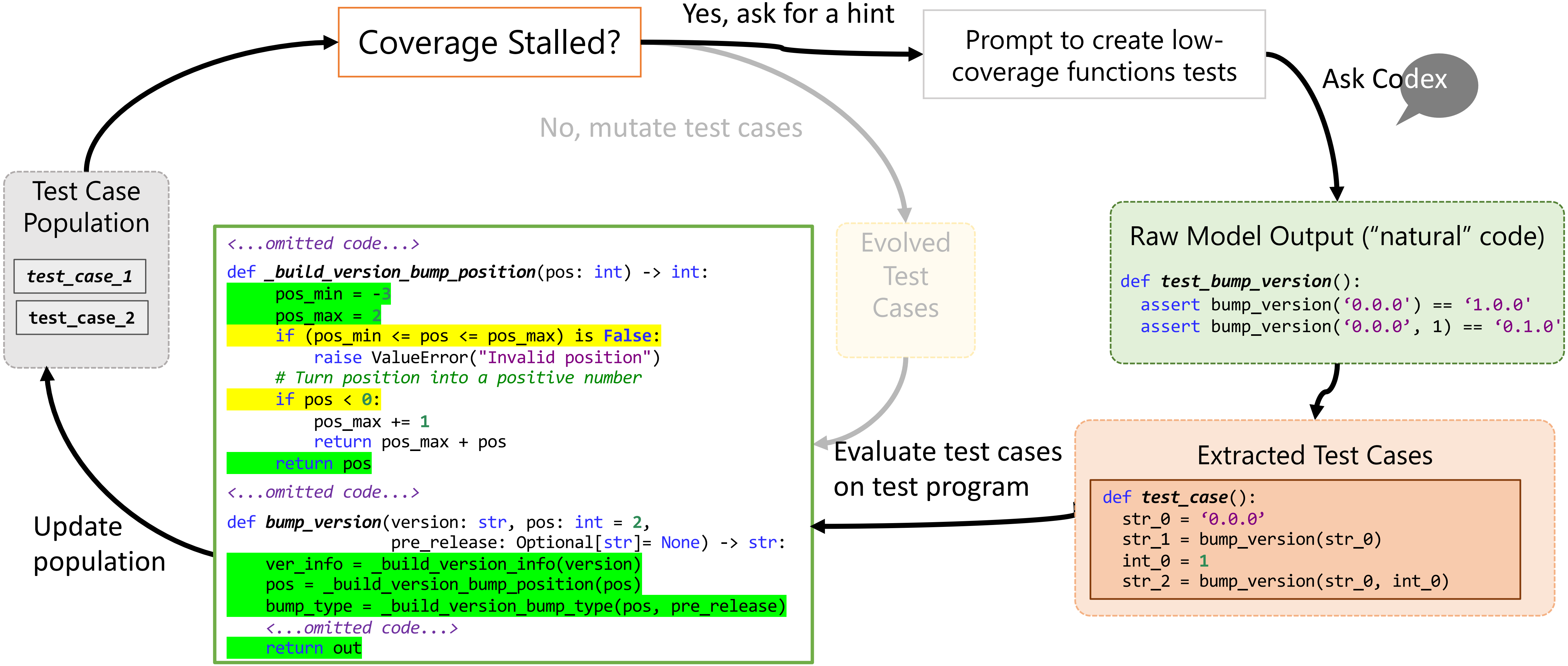
test_case_1

test_case_2

No, mutate test cases

Evolved Test Cases

Suggested Test Case Increases Coverage



Test Case Population

test_case_1

test_case_2

Coverage Stalled?

Yes, ask for a hint

Prompt to create low-coverage functions tests

Ask Codex

No, mutate test cases

Evolved Test Cases

Raw Model Output (“natural” code)

```
def test_bump_version():
  assert bump_version(‘0.0.0') == ‘1.0.0'
  assert bump_version(‘0.0.0’, 1) == ‘0.1.0'
```

Extracted Test Cases

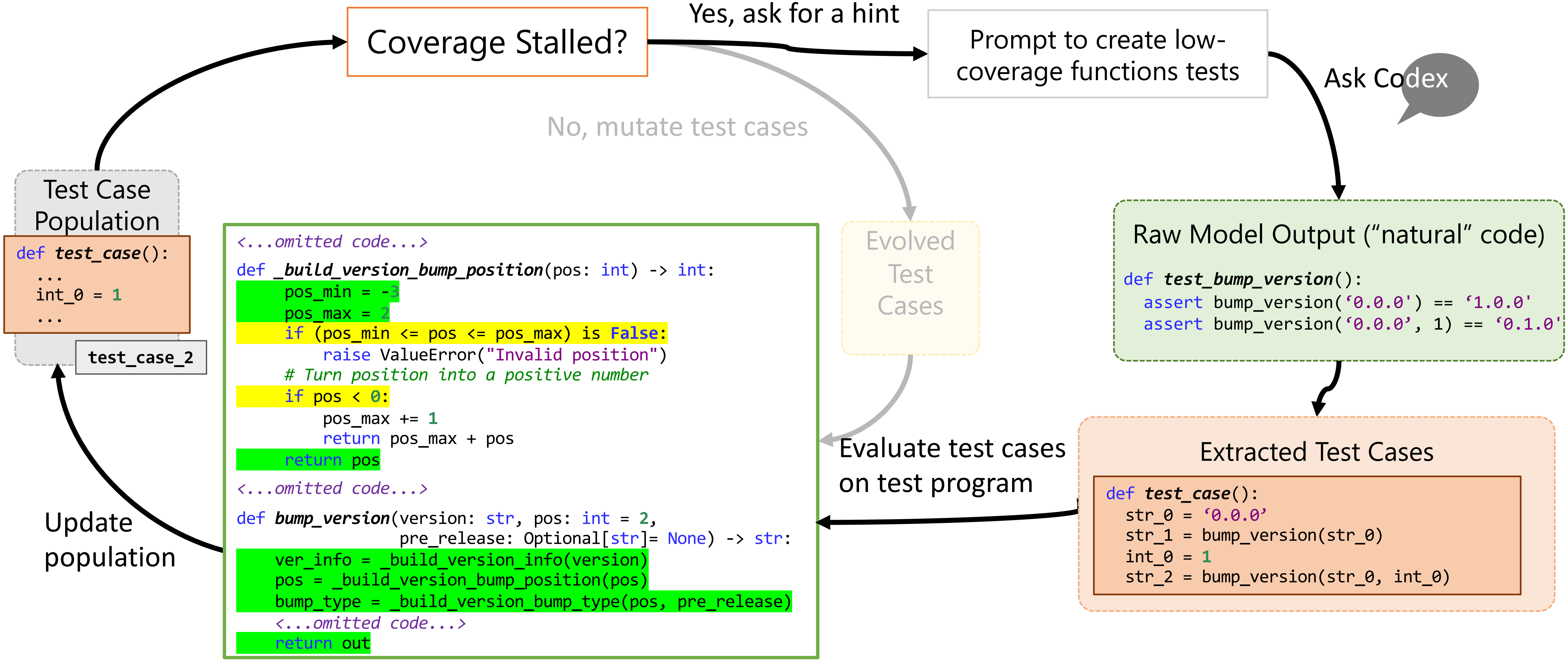
```
def test_case():
  str_0 = ‘0.0.0’
  str_1 = bump_version(str_0)
  int_0 = 1
  str_2 = bump_version(str_0, int_0)
```

Evaluate test cases on test program

```
<...omitted code...>
def _build_version_bump_position(pos: int) -> int:
    pos_min = -3
    pos_max = 2
    if (pos_min <= pos <= pos_max) is False:
        raise ValueError("Invalid position")
    # Turn position into a positive number
    if pos < 0:
        pos_max += 1
        return pos_max + pos
    return pos
<...omitted code...>
def bump_version(version: str, pos: int = 2,
                 pre_release: Optional[str]= None) -> str:
   ver_info = _build_version_info(version)
   pos = _build_version_bump_position(pos)
   bump_type = _build_version_bump_type(pos, pre_release)
   <...omitted code...>
   return out
```

Update population

Update Population



Coverage Stalled?

Yes, ask for a hint

Prompt to create low-coverage functions tests

Ask Codex

No, mutate test cases

Test Case Population

```
def test_case():
  ...
  int_0 = 1
  ...
```

test_case_2

Raw Model Output (“natural” code)

```
def test_bump_version():
  assert bump_version(‘0.0.0') == ‘1.0.0'
  assert bump_version(‘0.0.0’, 1) == ‘0.1.0'
```

Evolved Test Cases

Extracted Test Cases

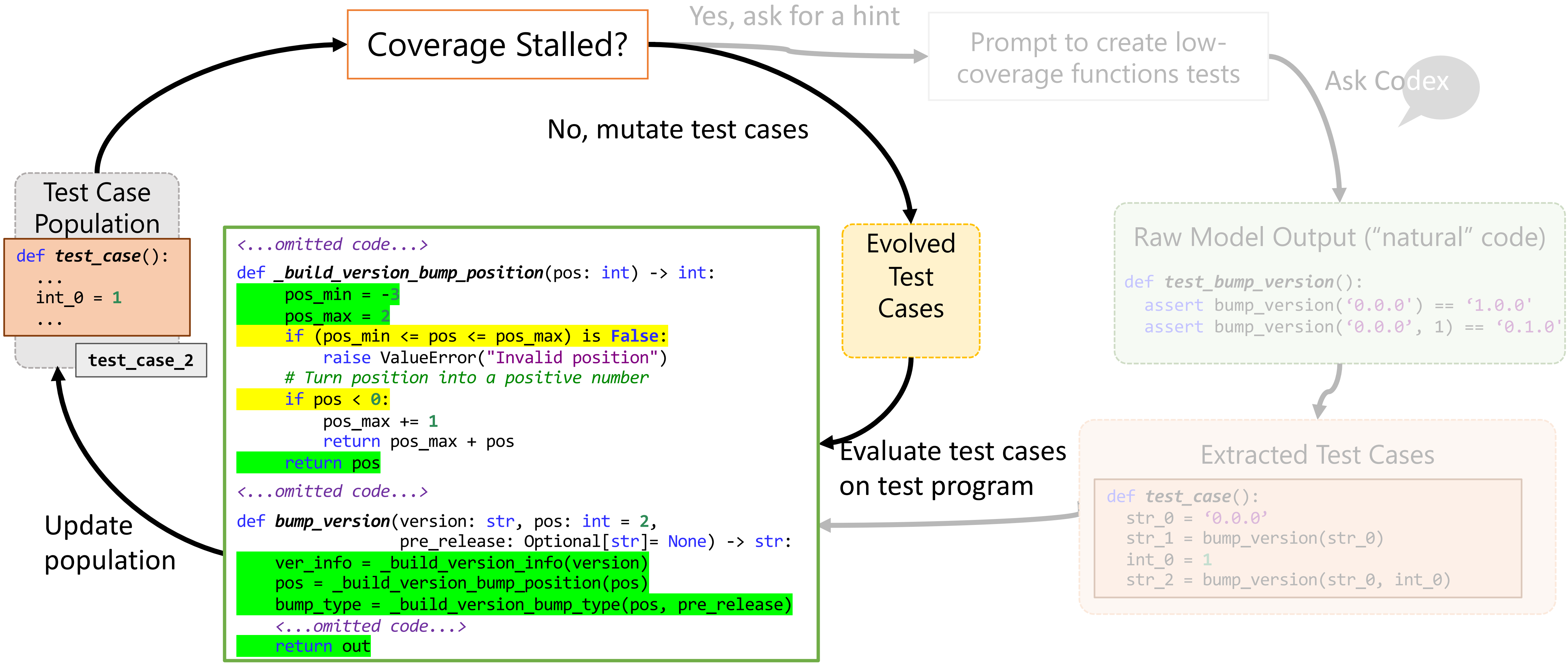
```
def test_case():
  str_0 = ‘0.0.0’
  str_1 = bump_version(str_0)
  int_0 = 1
  str_2 = bump_version(str_0, int_0)
```

Evaluate test cases on test program

```
<...omitted code...>
def _build_version_bump_position(pos: int) -> int:
    pos_min = -3
    pos_max = 2
    if (pos_min <= pos <= pos_max) is False:
        raise ValueError("Invalid position")
    # Turn position into a positive number
    if pos < 0:
        pos_max += 1
        return pos_max + pos
    return pos
<...omitted code...>
def bump_version(version: str, pos: int = 2,
                 pre_release: Optional[str]= None) -> str:
   ver_info = _build_version_info(version)
   pos = _build_version_bump_position(pos)
   bump_type = _build_version_bump_type(pos, pre_release)
   <...omitted code...>
   return out
```

Update population

Search No Longer Stalled



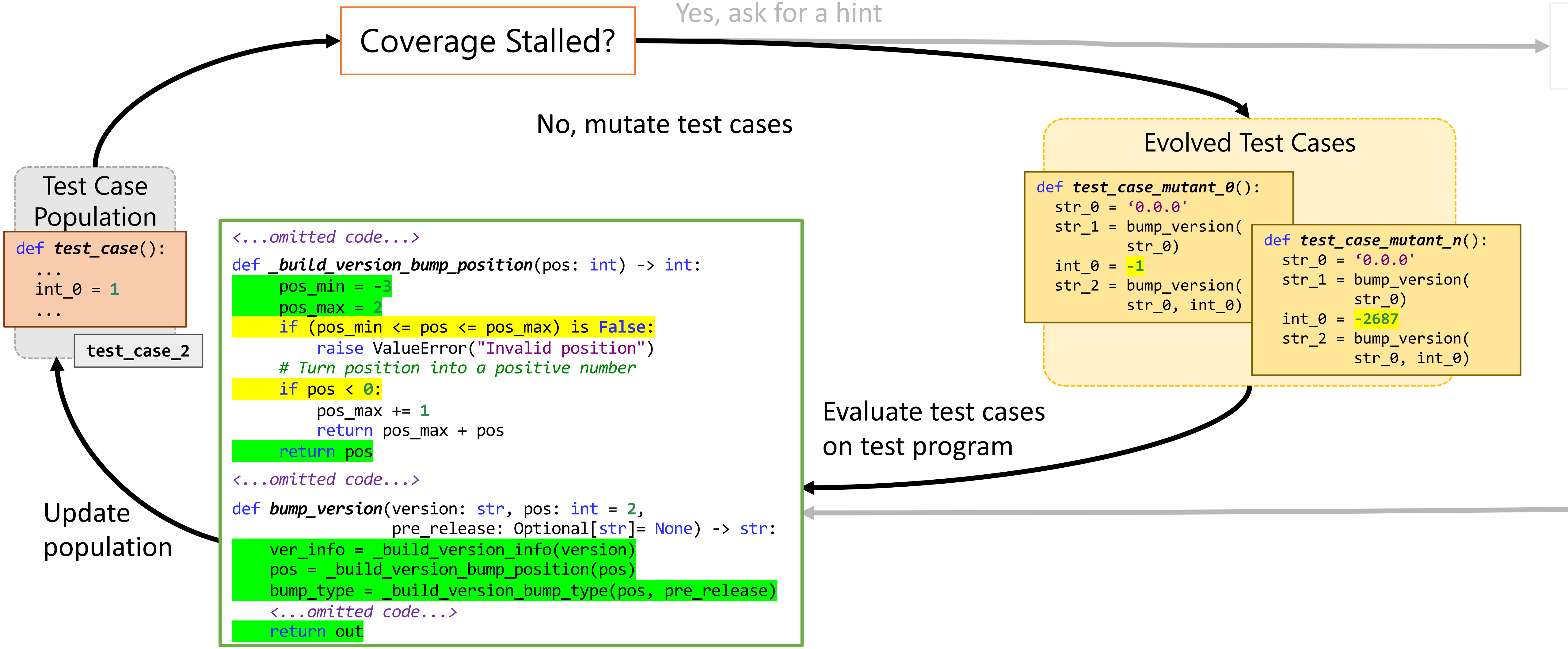
```
def test_case():
  ...
  int_0 = 1
  ...
```

```
<...omitted code...>
def _build_version_bump_position(pos: int) -> int:
    pos_min = -3
    pos_max = 2
    if (pos_min <= pos <= pos_max) is False:
        raise ValueError("Invalid position")
    # Turn position into a positive number
    if pos < 0:
        pos_max += 1
        return pos_max + pos
    return pos
<...omitted code...>
def bump_version(version: str, pos: int = 2,
                 pre_release: Optional[str]= None) -> str:
    ver_info = _build_version_info(version)
    pos = _build_version_bump_position(pos)
    bump_type = _build_version_bump_type(pos, pre_release)
    <...omitted code...>
    return out
```

```
def test_bump_version():
  assert bump_version(‘0.0.0') == ‘1.0.0'
  assert bump_version(‘0.0.0’, 1) == ‘0.1.0'
```

```
def test_case():
  str_0 = ‘0.0.0’
  str_1 = bump_version(str_0)
  int_0 = 1
  str_2 = bump_version(str_0, int_0)
```

Search No Longer Stalled



Coverage Stalled?

Yes, ask for a hint

No, mutate test cases

Test Case Population

```
def test_case():
  ...
  int_0 = 1
  ...
```

test_case_2

```
<...omitted code...>
def _build_version_bump_position(pos: int) -> int:
    pos_min = -3
    pos_max = 2
    if (pos_min <= pos <= pos_max) is False:
        raise ValueError("Invalid position")
    # Turn position into a positive number
    if pos < 0:
        pos_max += 1
        return pos_max + pos
    return pos
<...omitted code...>
def bump_version(version: str, pos: int = 2,
                 pre_release: Optional[str]= None) -> str:
   ver_info = _build_version_info(version)
   pos = _build_version_bump_position(pos)
   bump_type = _build_version_bump_type(pos, pre_release)
   <...omitted code...>
   return out
```

Evolved Test Cases

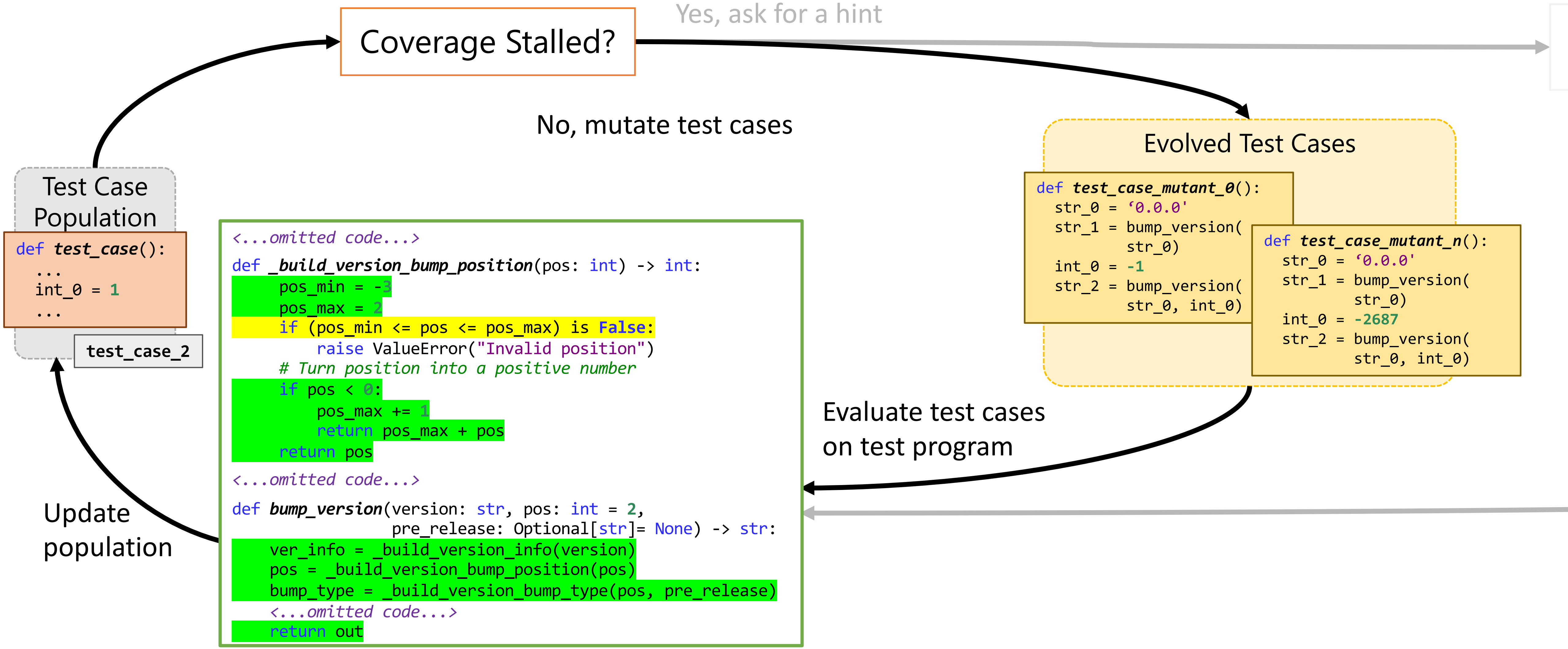
```
def test_case_mutant_0():
  str_0 = ‘0.0.0'
  str_1 = bump_version(
          str_0)
  int_0 = -1
  str_2 = bump_version(
          str_0, int_0)
```

```
def test_case_mutant_n():
  str_0 = ‘0.0.0'
  str_1 = bump_version(
          str_0)
  int_0 = -2687
  str_2 = bump_version(
          str_0, int_0)
```

Evaluate test cases on test program

Update population

Search No Longer Stalled



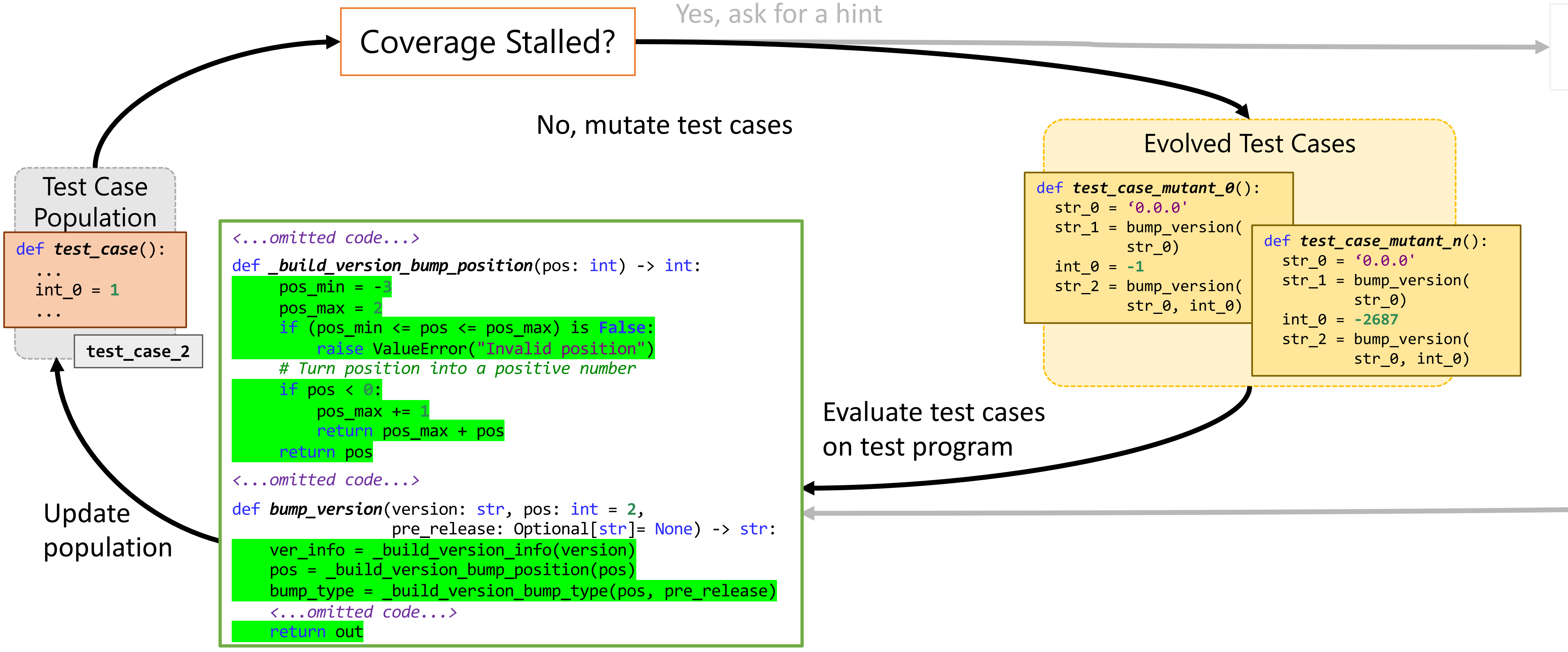
```python
def test_case():
  ...
  int_0 = 1
  ...
```

```python
<...omitted code...>
def _build_version_bump_position(pos: int) -> int:
    pos_min = -3
    pos_max = 2
    if (pos_min <= pos <= pos_max) is False:
        raise ValueError("Invalid position")
    # Turn position into a positive number
    if pos < 0:
        pos_max += 1
        return pos_max + pos
    return pos
<...omitted code...>
def bump_version(version: str, pos: int = 2,
                 pre_release: Optional[str]= None) -> str:
    ver_info = _build_version_info(version)
    pos = _build_version_bump_position(pos)
    bump_type = _build_version_bump_type(pos, pre_release)
    <...omitted code...>
    return out
```

```python
def test_case_mutant_0():
  str_0 = '0.0.0'
  str_1 = bump_version(
          str_0)
  int_0 = -1
  str_2 = bump_version(
          str_0, int_0)
```

```python
def test_case_mutant_n():
  str_0 = '0.0.0'
  str_1 = bump_version(
          str_0)
  int_0 = -2687
  str_2 = bump_version(
          str_0, int_0)
```

Search No Longer Stalled



Coverage Stalled?

Yes, ask for a hint

No, mutate test cases

Test Case Population

```
def test_case():
  ...
  int_0 = 1
  ...
```

test_case_2

```
<...omitted code...>
def _build_version_bump_position(pos: int) -> int:
    pos_min = -3
    pos_max = 2
    if (pos_min <= pos <= pos_max) is False:
        raise ValueError("Invalid position")
    # Turn position into a positive number
    if pos < 0:
        pos_max += 1
        return pos_max + pos
    return pos
<...omitted code...>
def bump_version(version: str, pos: int = 2,
                 pre_release: Optional[str]= None) -> str:
    ver_info = _build_version_info(version)
    pos = _build_version_bump_position(pos)
    bump_type = _build_version_bump_type(pos, pre_release)
    <...omitted code...>
    return out
```

Evolved Test Cases

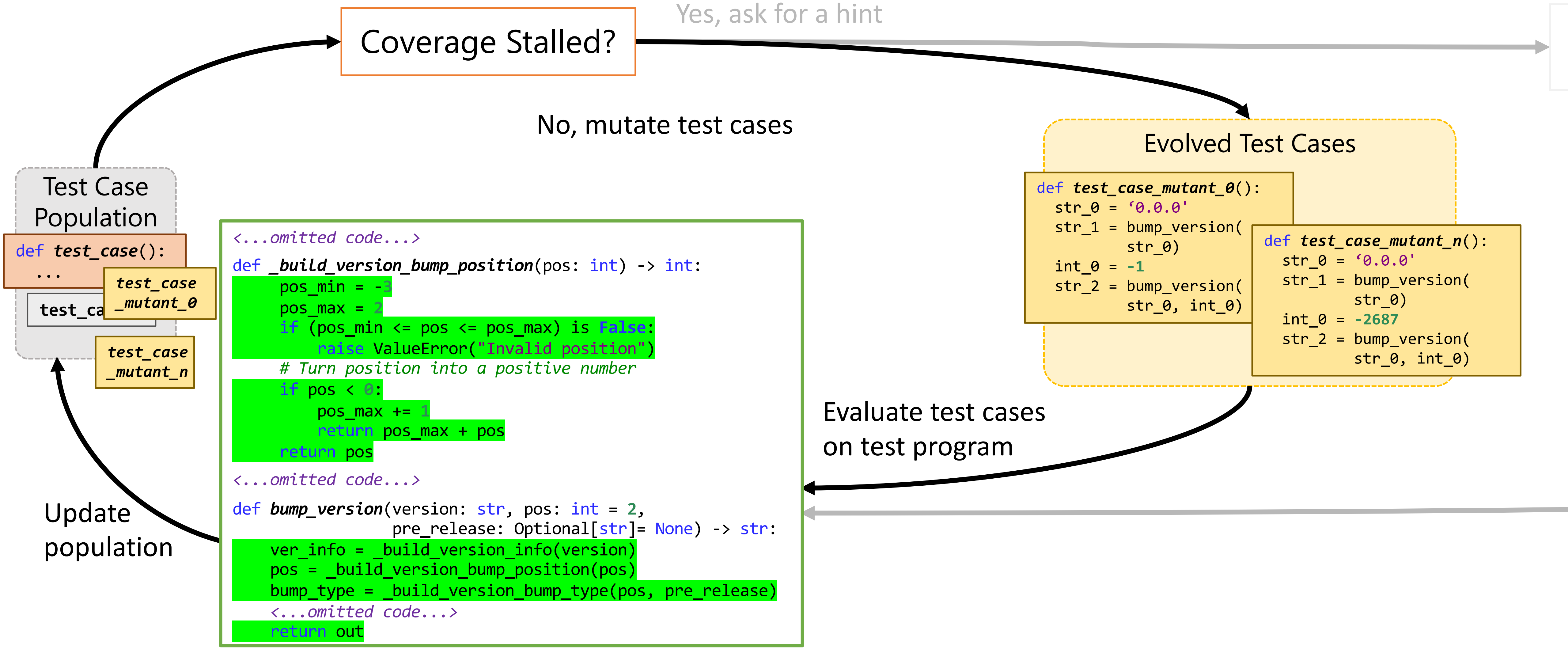
```
def test_case_mutant_0():
  str_0 = ‘0.0.0'
  str_1 = bump_version(
          str_0)
  int_0 = -1
  str_2 = bump_version(
          str_0, int_0)
```

```
def test_case_mutant_n():
  str_0 = ‘0.0.0'
  str_1 = bump_version(
          str_0)
  int_0 = -2687
  str_2 = bump_version(
          str_0, int_0)
```

Evaluate test cases on test program

Update population

Search No Longer Stalled



```
def test_case():
  ...
```

```
test_case
_mutant_0
```

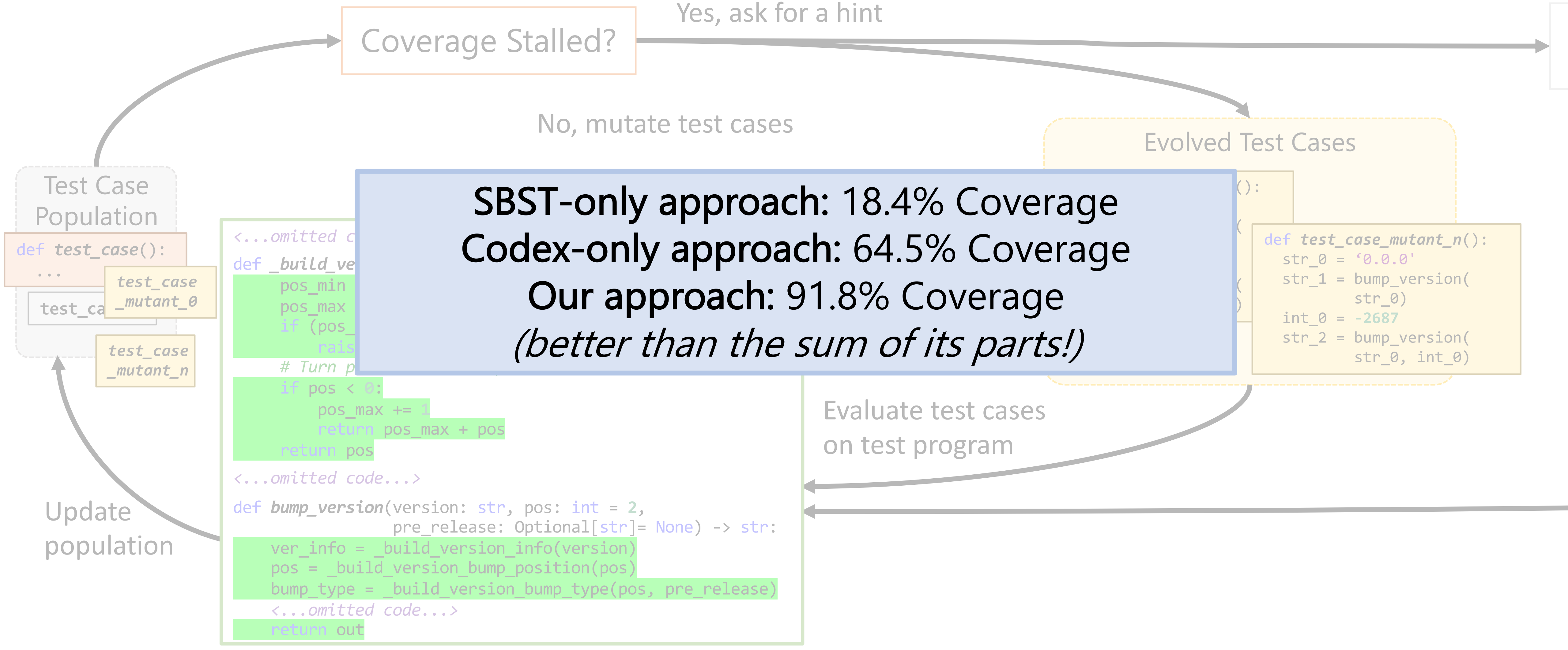
```
test_case
_mutant_n
```

```
def test_case_mutant_0():
  str_0 = '0.0.0'
  str_1 = bump_version(
          str_0)
  int_0 = -1
  str_2 = bump_version(
          str_0, int_0)
```

```
def test_case_mutant_n():
  str_0 = '0.0.0'
  str_1 = bump_version(
          str_0)
  int_0 = -2687
  str_2 = bump_version(
          str_0, int_0)
```

```
<...omitted code...>
def _build_version_bump_position(pos: int) -> int:
    pos_min = -3
    pos_max = 2
    if (pos_min <= pos <= pos_max) is False:
        raise ValueError("Invalid position")
    # Turn position into a positive number
    if pos < 0:
        pos_max += 1
        return pos_max + pos
    return pos
<...omitted code...>
def bump_version(version: str, pos: int = 2,
                 pre_release: Optional[str]= None) -> str:
   ver_info = _build_version_info(version)
   pos = _build_version_bump_position(pos)
   bump_type = _build_version_bump_type(pos, pre_release)
   <...omitted code...>
   return out
```

Spoiler: Results on this Benchmark



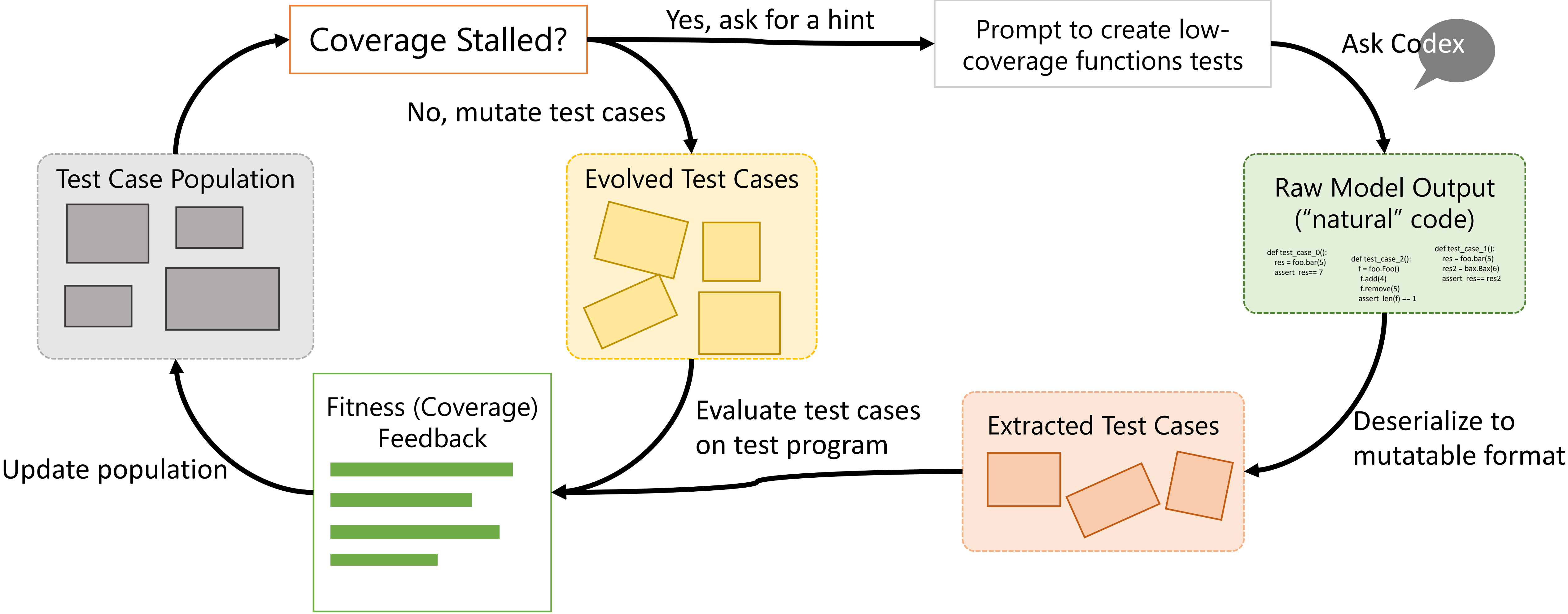
```
def test_case():
    ...
```

```
<...omitted c
def _build_ve
        pos_min
        pos_max
        if (pos_
            rais
        # Turn p
        if pos < 0:
            pos_max += 1
            return pos_max + pos
        return pos
<...omitted code...>
def bump_version(version: str, pos: int = 2,
                 pre_release: Optional[str]= None) -> str:
    ver_info = _build_version_info(version)
    pos = _build_version_bump_position(pos)
    bump_type = _build_version_bump_type(pos, pre_release)
    <...omitted code...>
    return out
```

```
def test_case_mutant_n():
  str_0 = '0.0.0'
  str_1 = bump_version(
          str_0)
  int_0 = -2687
  str_2 = bump_version(
          str_0, int_0)
```

SBST-only approach: 18.4% Coverage
Codex-only approach: 64.5% Coverage
Our approach: 91.8% Coverage
(better than the sum of its parts!)

CodaMOSA Approach



Coverage Stalled?

Yes, ask for a hint

Prompt to create low-coverage functions tests

Ask Codex

Raw Model Output (“natural” code)

```
def test_case_0():
   res = foo.bar(5)
   assert  res== 7

def test_case_2():
   f = foo.Foo()
    f.add(4)
    f.remove(5)
   assert  len(f) == 1

def test_case_1():
   res = foo.bar(5)
   res2 = bax.Bax(6)
   assert  res== res2
```

Deserialize to mutatable format

Extracted Test Cases

Evaluate test cases on test program

Fitness (Coverage) Feedback

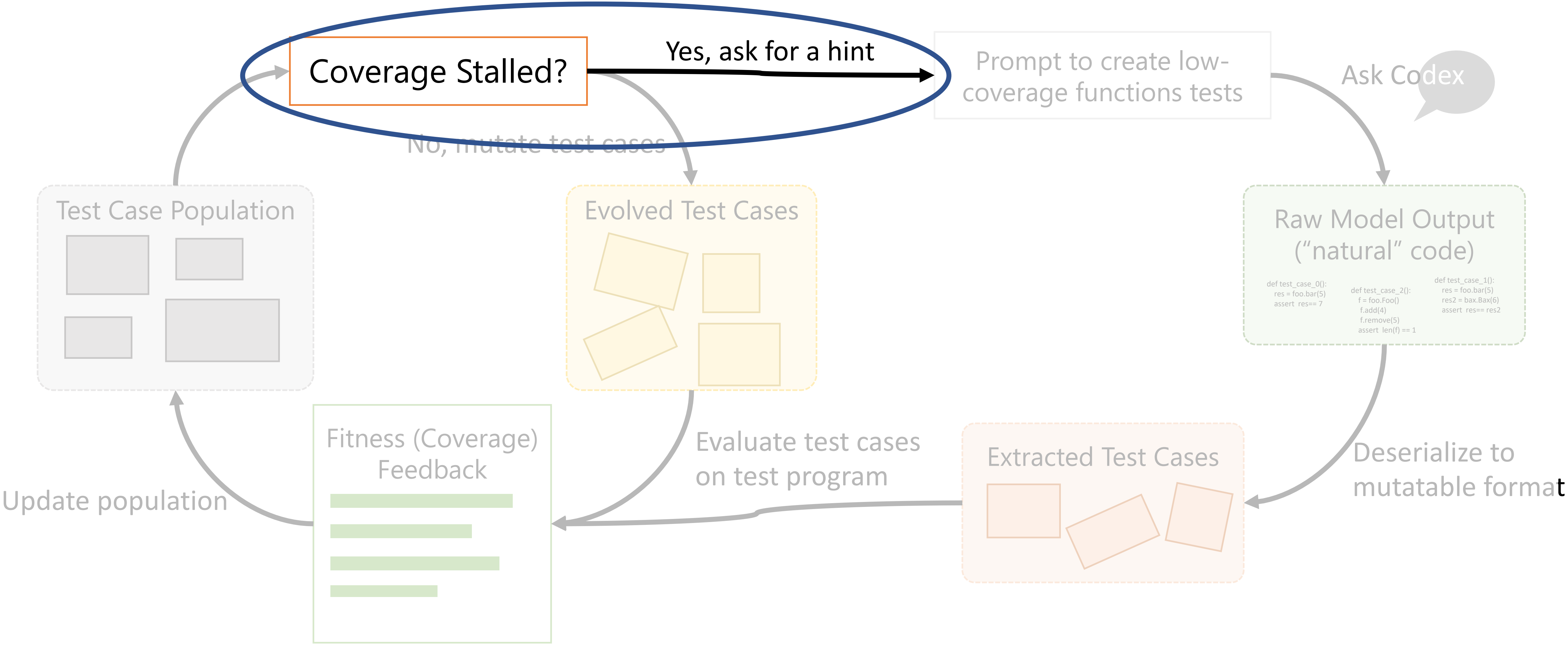
Update population

Test Case Population

No, mutate test cases

Evolved Test Cases

Challenge: When to ask for a hint?

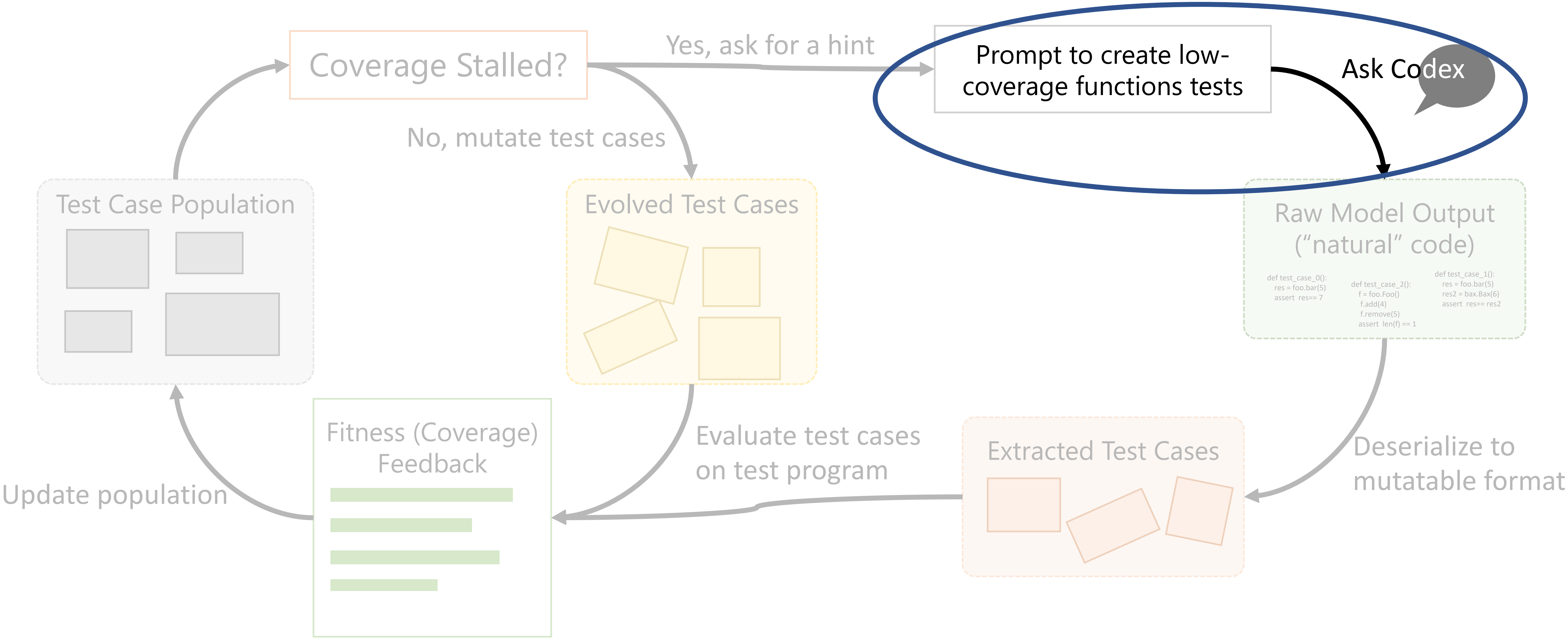

Coverage Stalled?
Yes, ask for a hint
No, mutate test cases
Prompt to create low-coverage functions tests
Ask Codex
Test Case Population
Evolved Test Cases
Raw Model Output ("natural" code)
Deserialize to mutatable format
Extracted Test Cases
Evaluate test cases on test program
Fitness (Coverage) Feedback
Update population

```
def test_case_0():
  res = foo.bar(5)
  assert  res== 7

def test_case_2():
  f = foo.Foo()
  f.add(4)
  f.remove(5)
  assert  len(f) == 1

def test_case_1():
  res = foo.bar(5)
  res2 = bax.Bax(6)
  assert  res== res2
```

Challenge: How to ask for a hint?

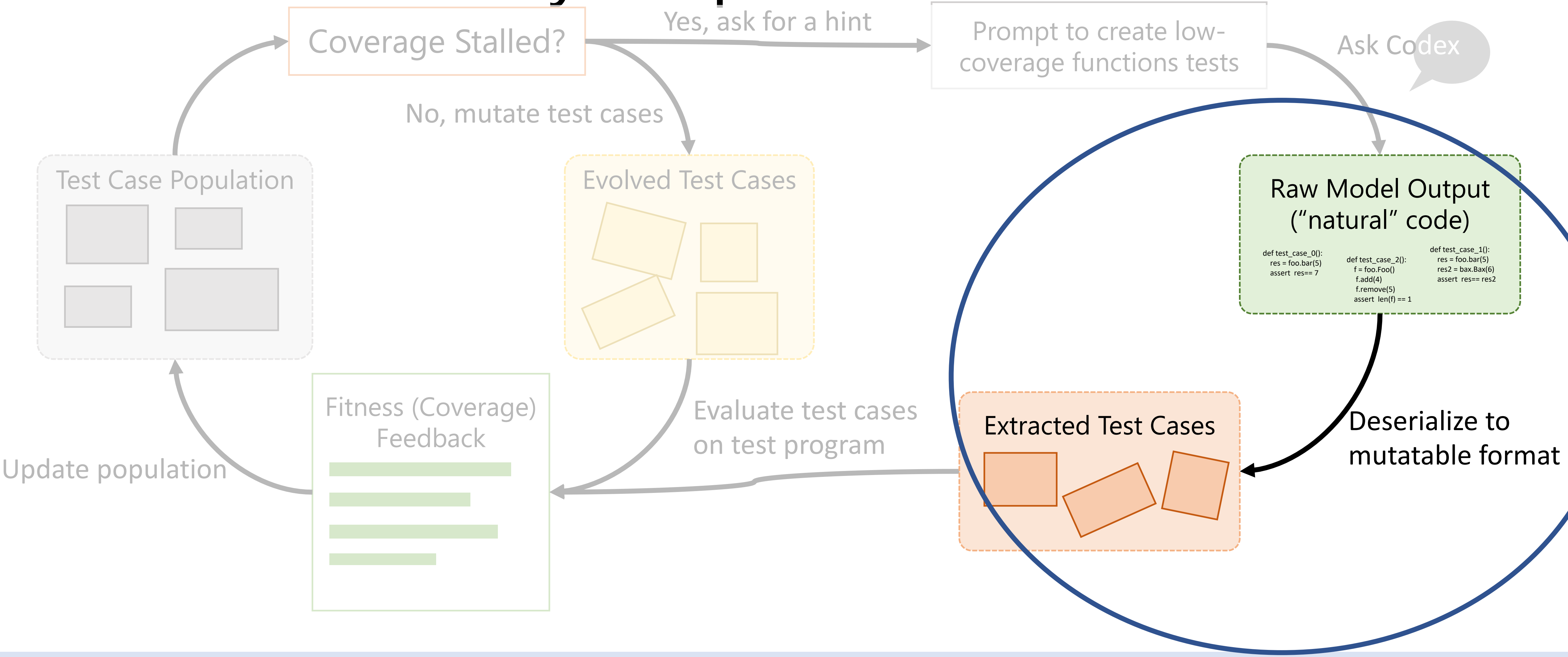

Coverage Stalled?
Yes, ask for a hint
Prompt to create low-coverage functions tests
Ask Codex
No, mutate test cases
Test Case Population
Evolved Test Cases
Raw Model Output ("natural" code)
Deserialize to mutatable format
Extracted Test Cases
Evaluate test cases on test program
Fitness (Coverage) Feedback
Update population

```
def test_case_0():
   res = foo.bar(5)
   assert  res== 7

def test_case_2():
   f = foo.Foo()
    f.add(4)
    f.remove(5)
   assert  len(f) == 1

def test_case_1():
   res = foo.bar(5)
   res2 = bax.Bax(6)
   assert  res== res2
```

Challenge: How to handle (potentially) arbitrary output from Codex?



Coverage Stalled?

Yes, ask for a hint

Prompt to create low-coverage functions tests

Ask Codex

No, mutate test cases

Test Case Population

Evolved Test Cases

Raw Model Output (“natural” code)

```
def test_case_0():
   res = foo.bar(5)
   assert  res== 7
```

```
def test_case_2():
   f = foo.Foo()
    f.add(4)
    f.remove(5)
   assert  len(f) == 1
```

```
def test_case_1():
   res = foo.bar(5)
   res2 = bax.Bax(6)
   assert  res== res2
```

Fitness (Coverage) Feedback

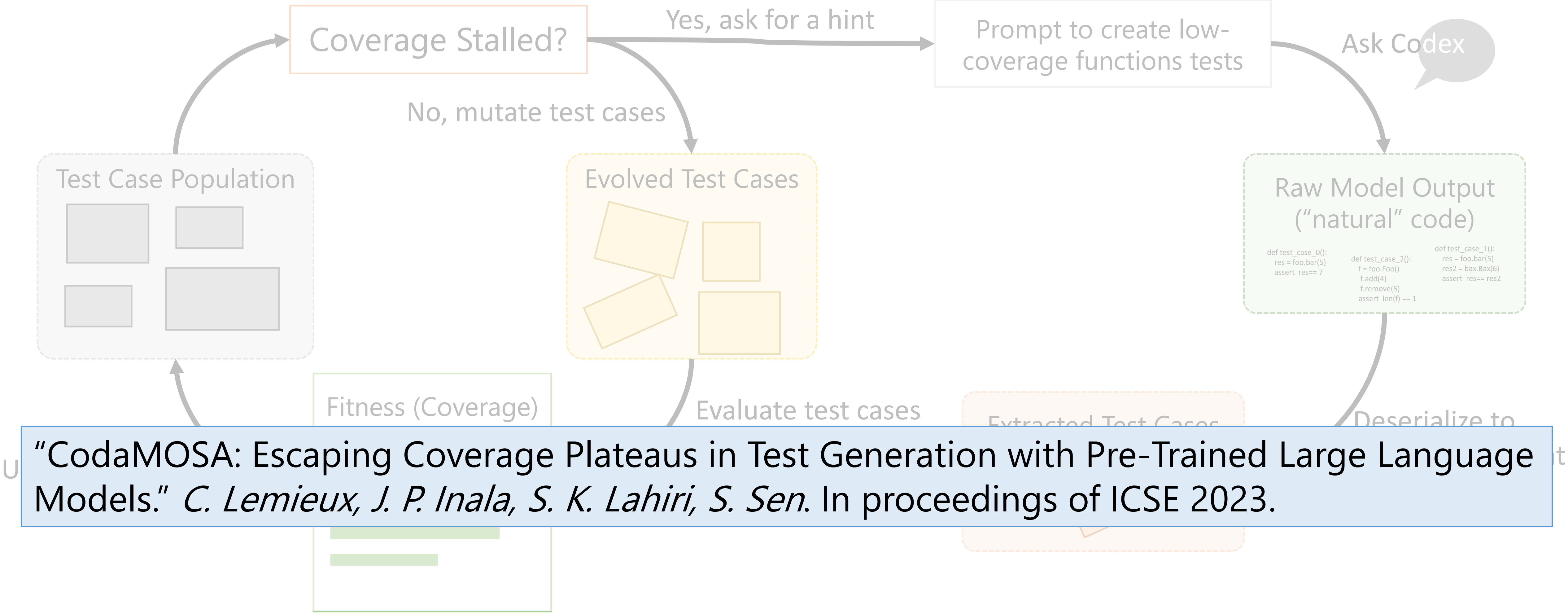
Evaluate test cases on test program

Extracted Test Cases

Deserialize to mutatable format

Update population

Solutions Discussed Further in Paper

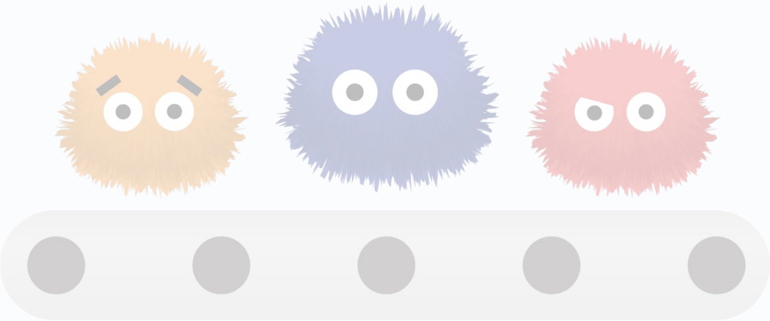


```
def test_case_0():
  res = foo.bar(5)
  assert res== 7

def test_case_2():
  f = foo.Foo()
  f.add(4)
  f.remove(5)
  assert len(f) == 1

def test_case_1():
  res = foo.bar(5)
  res2 = bax.Bax(6)
  assert res== res2
```

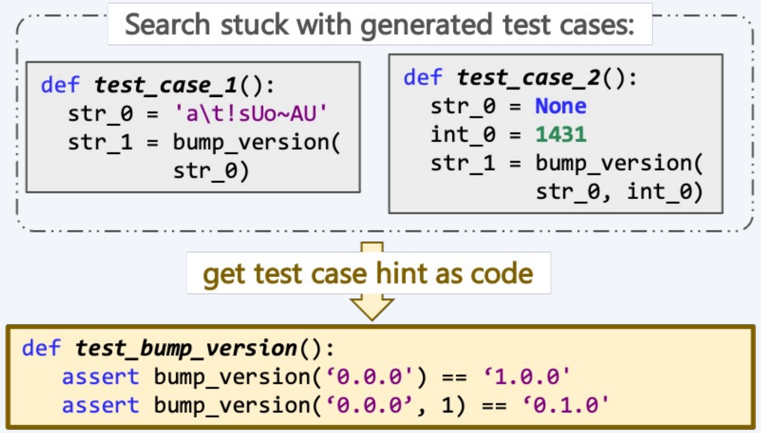
“CodaMOSA: Escaping Coverage Plateaus in Test Generation with Pre-Trained Large Language Models.” *C. Lemieux, J. P. Inala, S. K. Lahiri, S. Sen*. In proceedings of ICSE 2023.



Using *generalized feedback maps* to expand *bugs findable by fuzz testing*

PerfFuzz (ISSTA'18)

FuzzFactory (OOPSLA'19)



Search stuck with generated test cases:

```
def test_case_1():
  str_0 = 'a\t!sUo~AU'
  str_1 = bump_version(
          str_0)
```

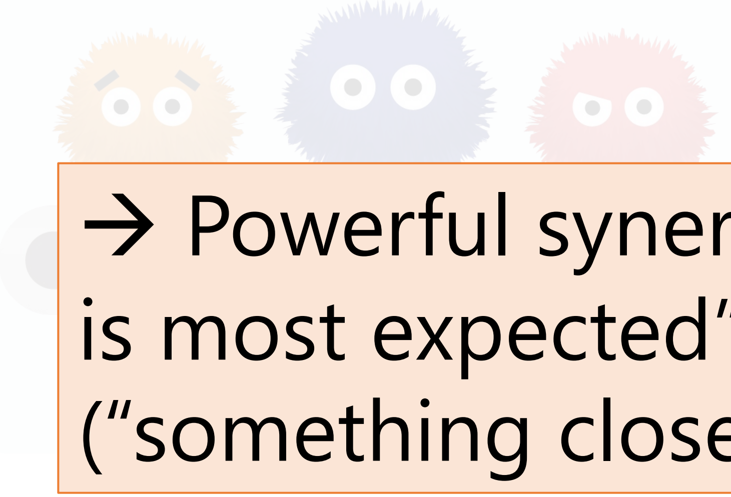
```
def test_case_2():
  str_0 = None
  int_0 = 1431
  str_1 = bump_version(
          str_0, int_0)
```

get test case hint as code

```
def test_bump_version():
   assert bump_version('0.0.0') == '1.0.0'
   assert bump_version('0.0.0', 1) == '0.1.0'
```

Using *large language models* to improve *automated test suite generation*

CodaMOSA (ICSE'23)

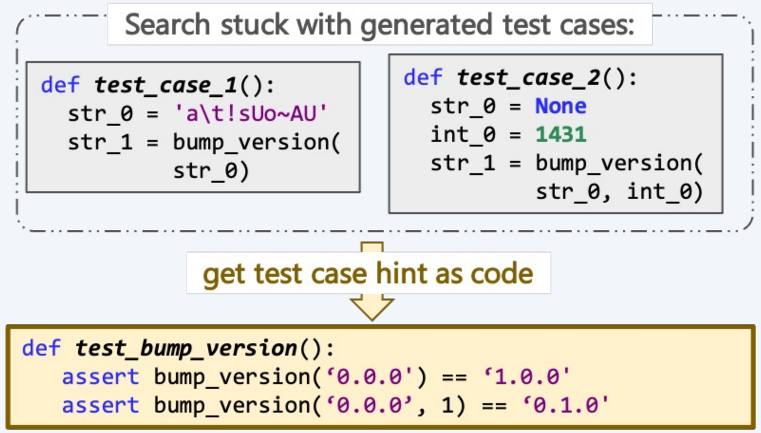


Using *generalized feedback maps* to

PerfFuzz (ISSTA’18)

FuzzFactory (OOPSLA’19)

→ Powerful synergy between LLMs (“what is most expected”) and mutative search (“something close, but unexpected”)



Search stuck with generated test cases:

```
def test_case_1():
  str_0 = 'a\t!sUo~AU'
  str_1 = bump_version(
          str_0)
```

```
def test_case_2():
  str_0 = None
  int_0 = 1431
  str_1 = bump_version(
          str_0, int_0)
```

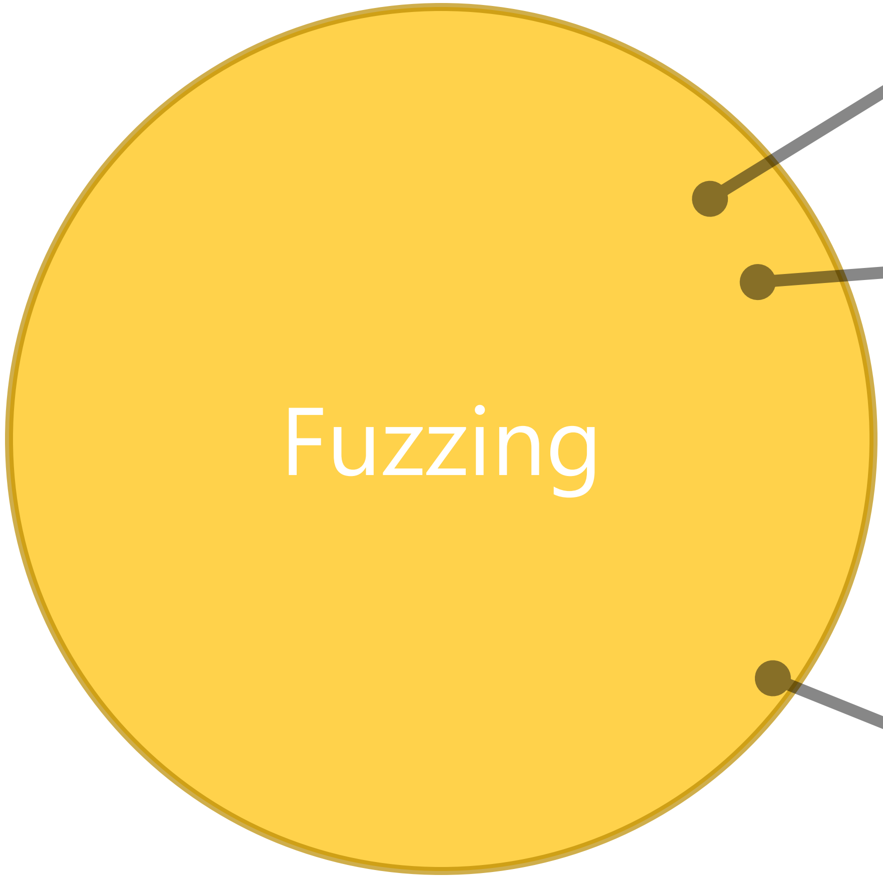
get test case hint as code

```
def test_bump_version():
   assert bump_version(‘0.0.0') == ‘1.0.0'
   assert bump_version(‘0.0.0’, 1) == ‘0.1.0'
```

Using ***large language models*** to improve ***automated test suite generation***

CodaMOSA (ICSE’23)

So Far: Innovations in Fuzzing Algorithms

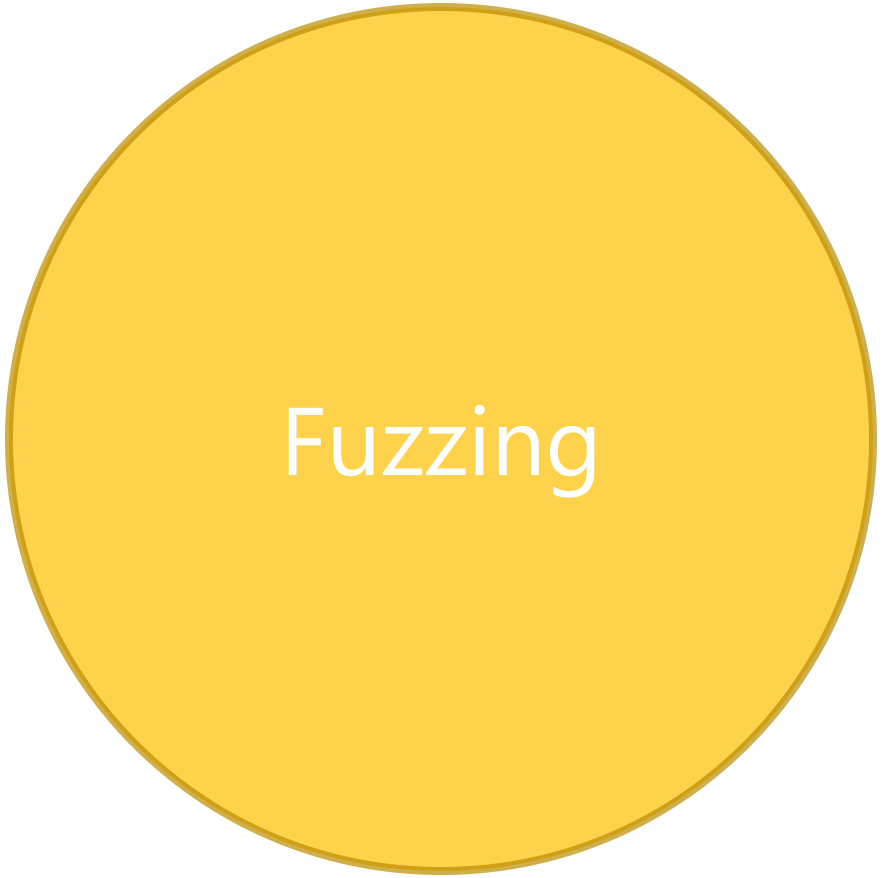


PerfFuzz
(ISSTA’18)

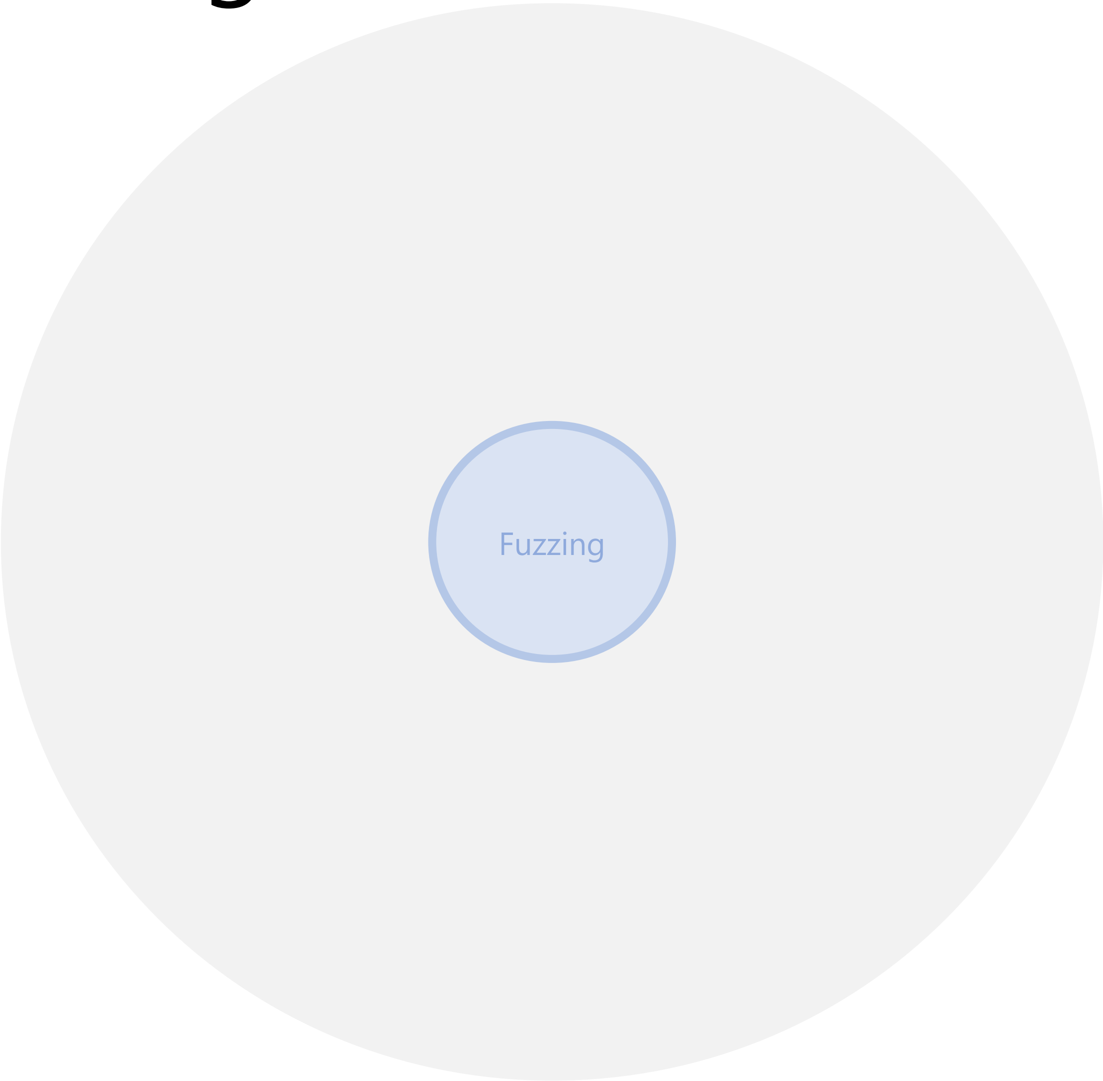
FuzzFactory
(OOPSLA’19)

CodaMOSA
(ICSE’23)

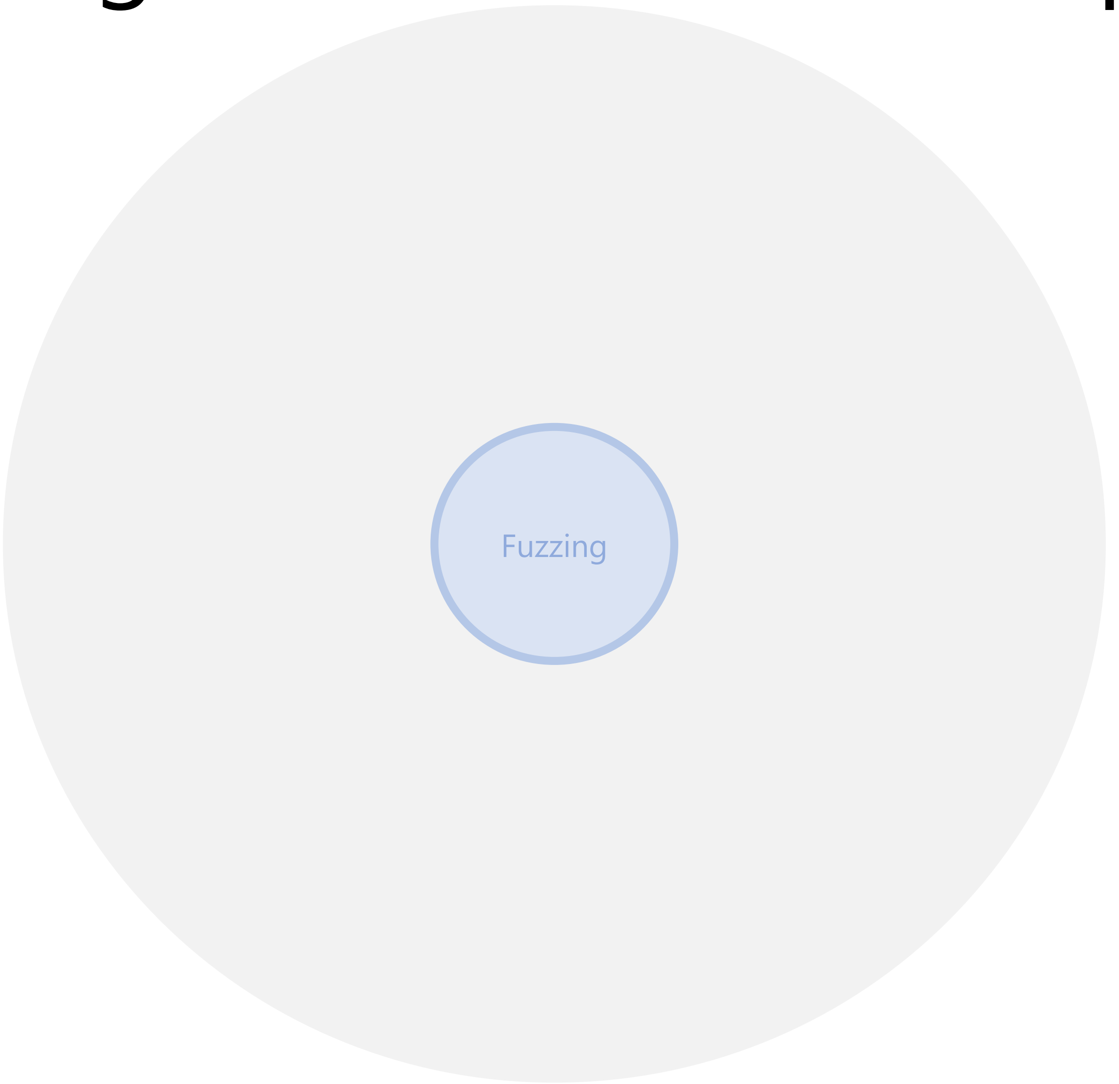
Next: Solving Problems **Around** Fuzzing



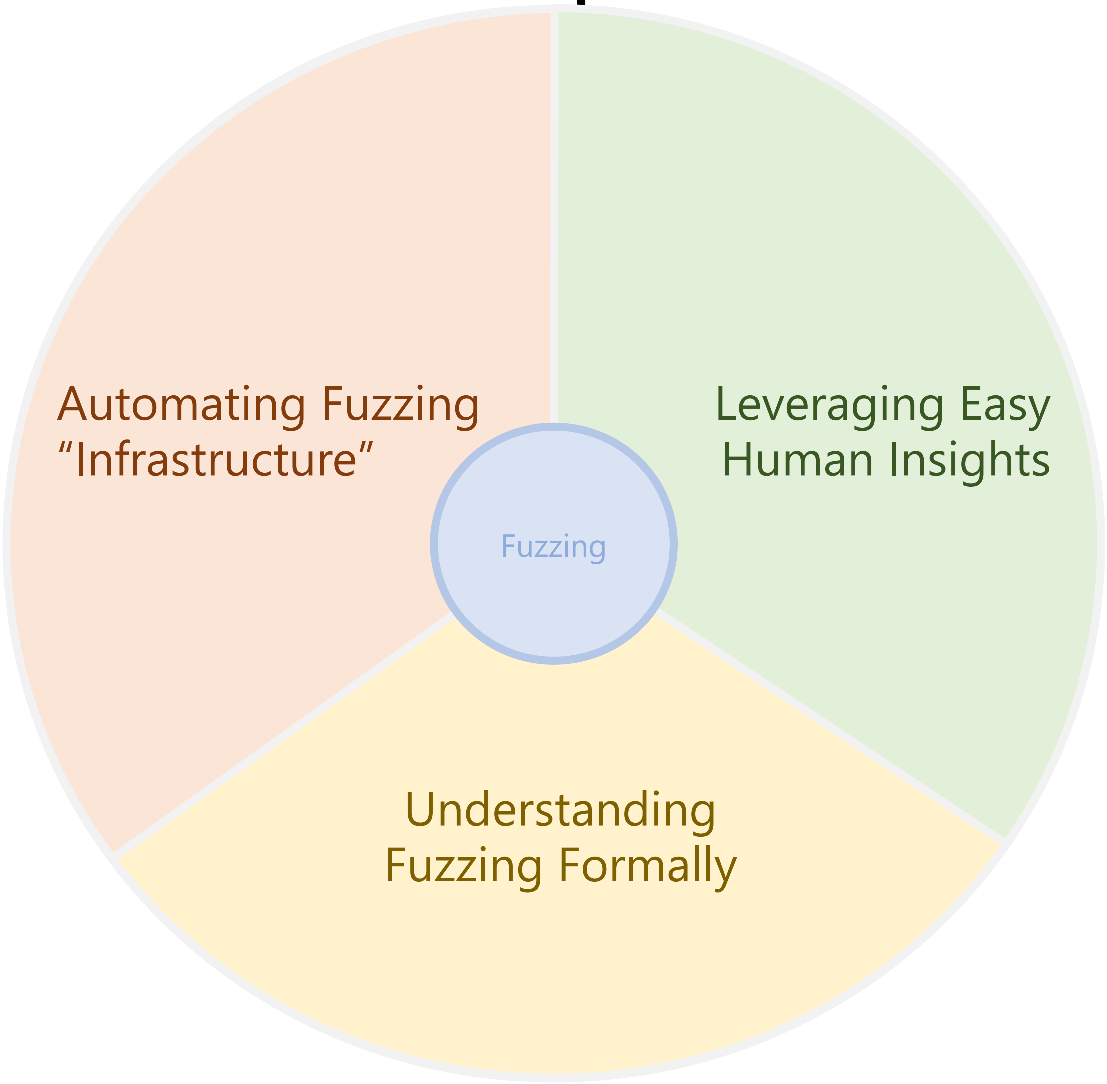
Next: Solving Problems **Around** Fuzzing



Enabling Fuzz-Driven Development



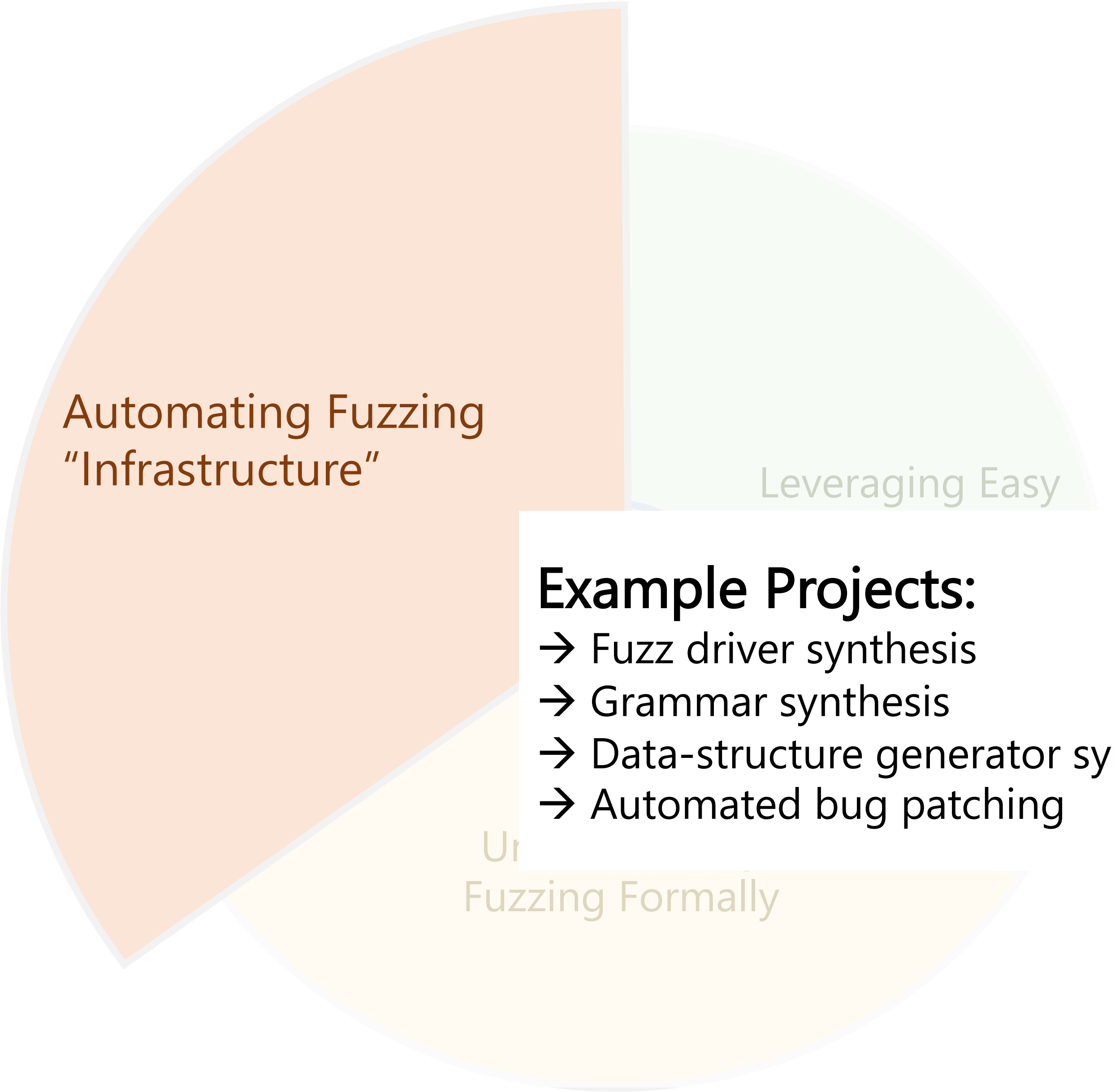
Fuzz-Driven Development: Three Pillars



Automating Fuzzing “Infrastructure”

Leveraging Easy Human Insights

Understanding Fuzzing Formally



Example Projects:

→ Fuzz driver synthesis
→ Grammar synthesis
→ Data-structure generator synthesis
→ Automated bug patching

Automating Fuzzing
“Infrastructure”

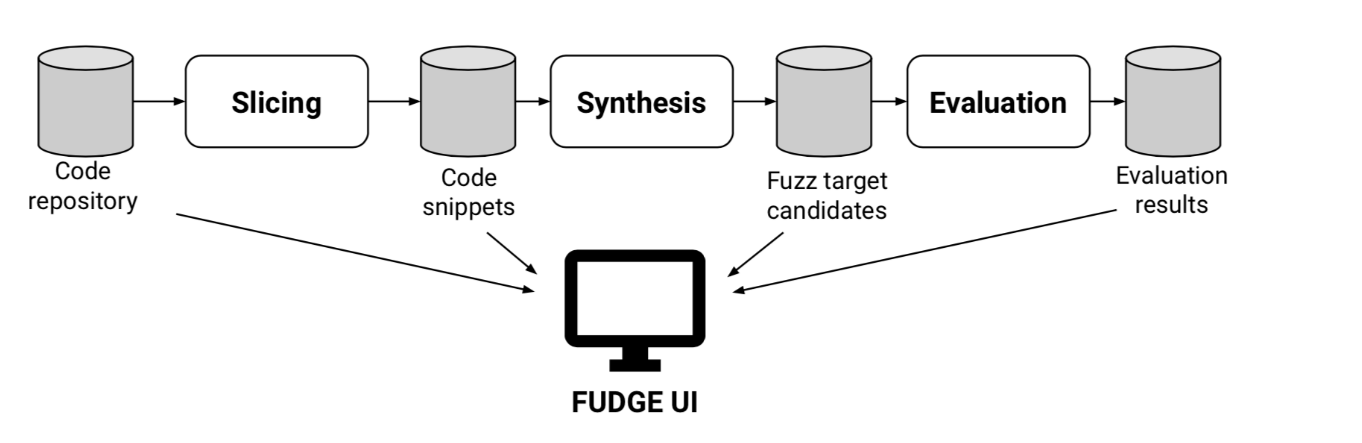
Leveraging Easy

FUDGE

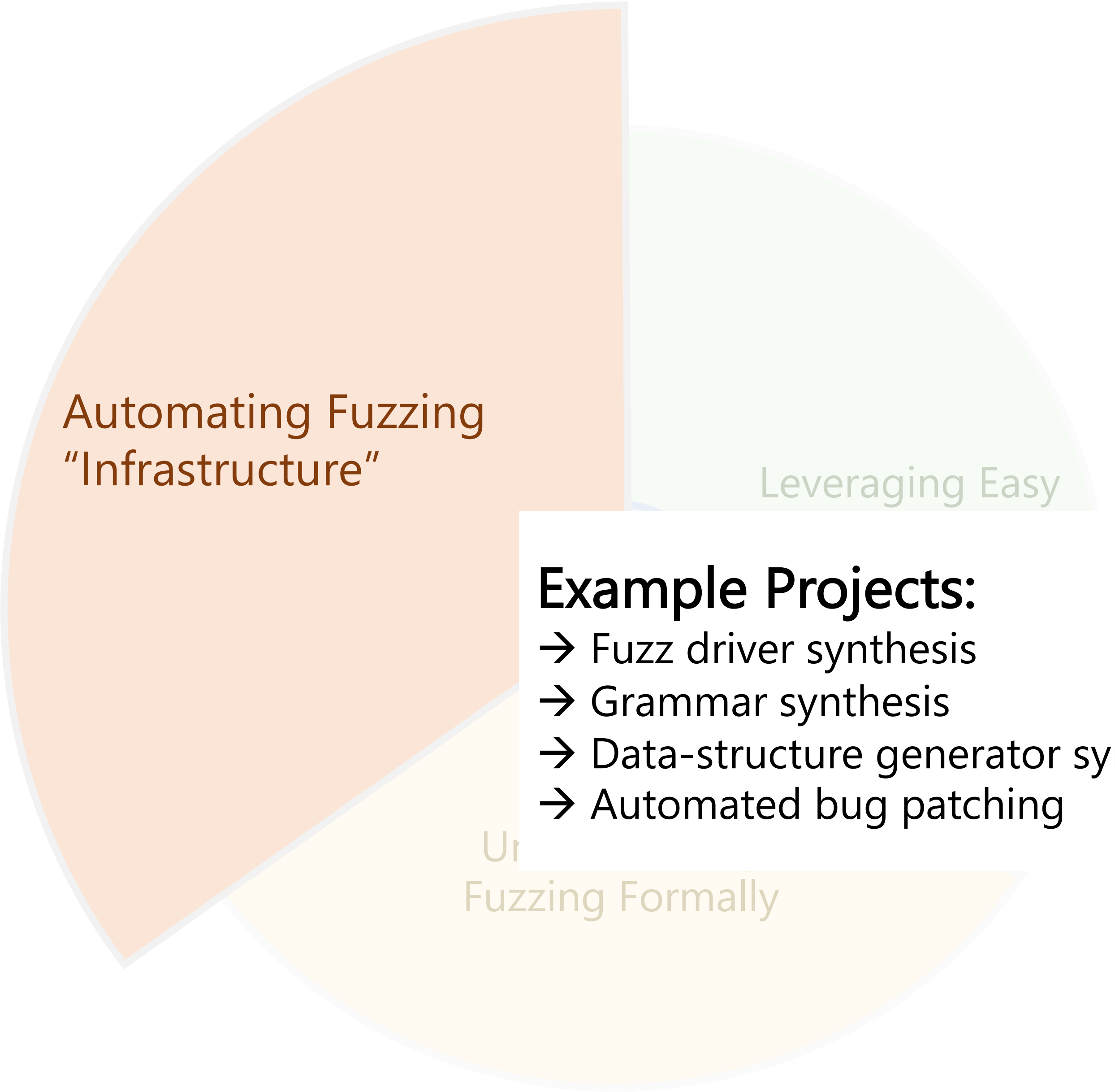
Babic, Bucur, Chen, Ivancic, King,
Kusano, <u>Lemieux</u>, Szekeres, Wang
ESEC/FSE’19 **(Industry Track)**

Best Paper Award
(Industry Track)

U
Fu

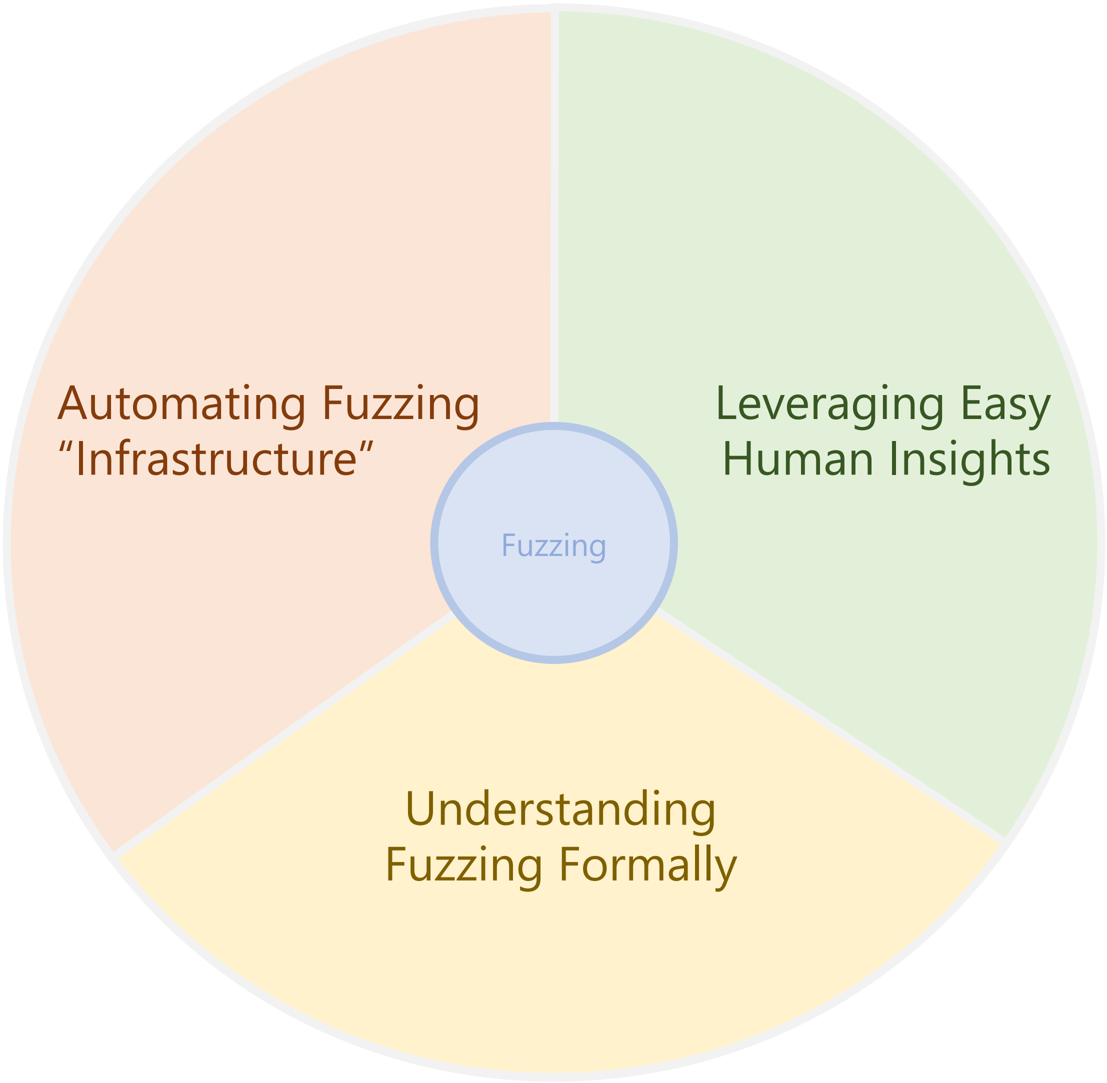


200 drivers integrated into open-source projects
→<u>150 security-improving fixes</u>



Example Projects:

→ Fuzz driver synthesis
→ Grammar synthesis
→ Data-structure generator synthesis
→ Automated bug patching

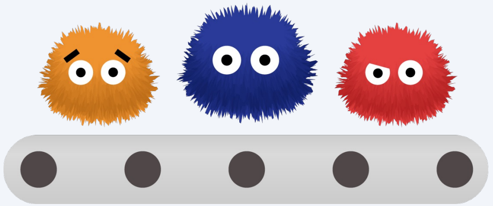

Automating Fuzzing
“Infrastructure”

Leveraging Easy
Human Insights

Fuzzing

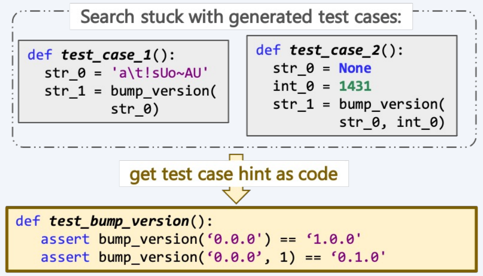
Understanding
Fuzzing Formally

PerfFuzz (ISSTA'18) + FuzzFactory (OOPSLA'19)



Using ***generalized feedback maps*** to expand ***bugs findable by fuzz testing***

CodaMOSA (ICSE'23)



Search stuck with generated test cases:

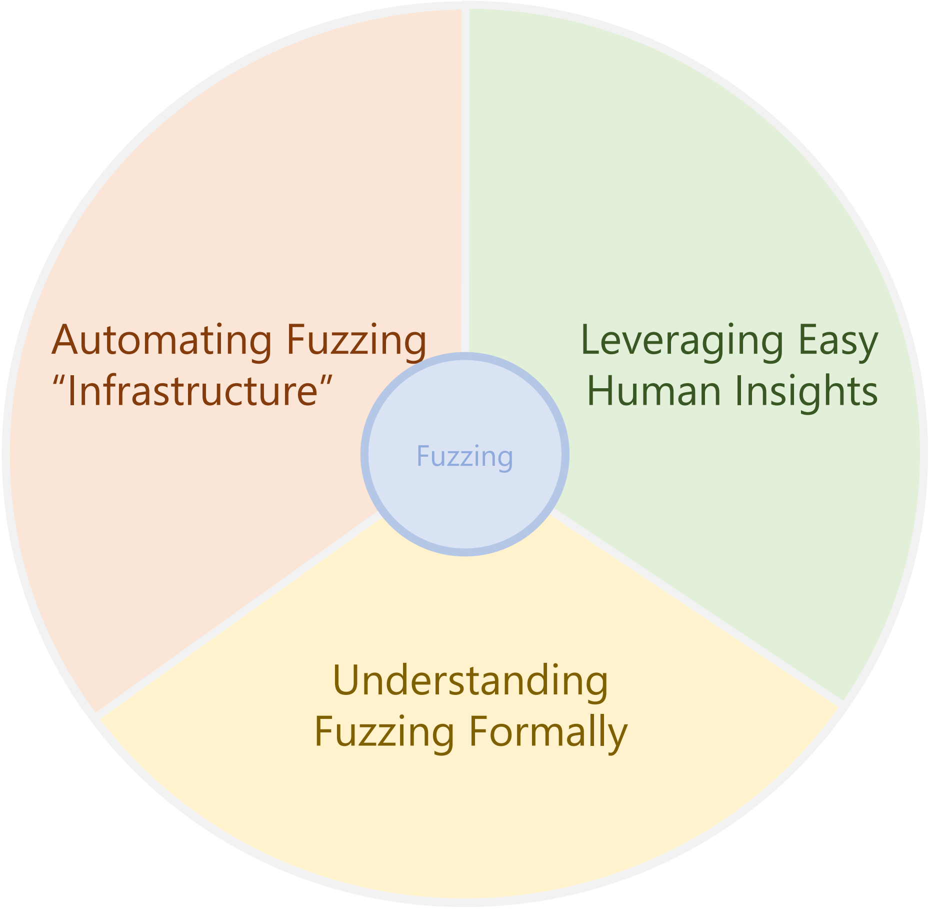
```
def test_case_1():
  str_0 = 'a\t!sUo~AU'
  str_1 = bump_version(
          str_0)
```

```
def test_case_2():
  str_0 = None
  int_0 = 1431
  str_1 = bump_version(
          str_0, int_0)
```

get test case hint as code

```
def test_bump_version():
   assert bump_version('0.0.0') == '1.0.0'
   assert bump_version('0.0.0', 1) == '0.1.0'
```

Using ***large language models*** to improve ***automated test suite generation***



Automating Fuzzing "Infrastructure"
Leveraging Easy Human Insights
Fuzzing
Understanding Fuzzing Formally

clemieux@cs.ubc.ca

carolemieux.com

@cestlemieux