AXOLOTL

OR HOW TO AUTOMATICALLY FIX PROGRAMS' FAULT ON THE FLY

YOUNGJAE KIM, FÉLIX YVONNET, JOOYONG YI

OVERVIEW

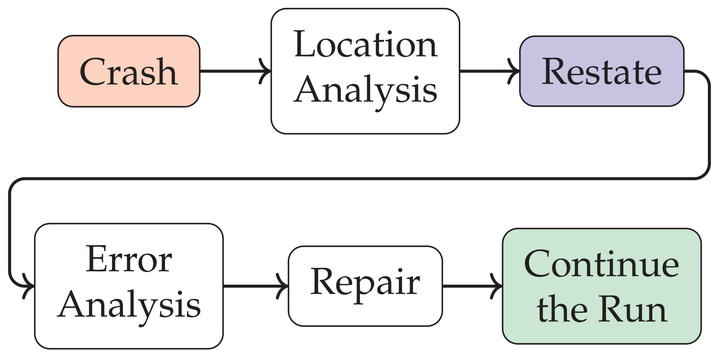


Figure 1: Pipeline of the framework

Cost-free program repairing python interpreter

- Cost-free if the program is crash-free
- Abstraction of the plugin used for fault location, error analysis and program repair
- Rewrites the source code to avoid any bug repetition

INTEGERS

- Generate a huge number of examples with a fuzzer
- A dense neural network takes as input the observed outputs and guess the corresponding input state
- Do not work well with other types (*e.g.* strings)

STRINGS

- Use the same generated set of example than the neural network
- Synthesize[1] the inverse program to guess the initial input
- Does not work well with information loss

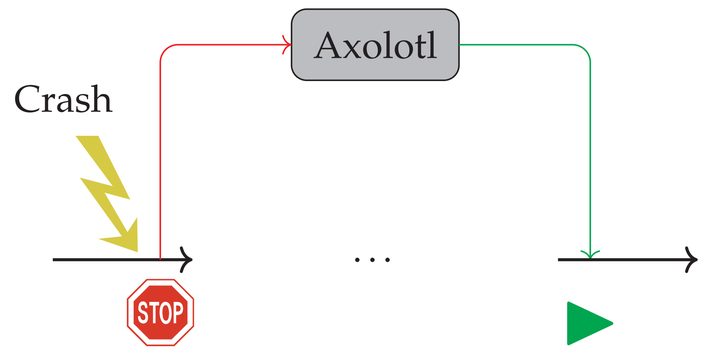
RESTATE

State-of-the-art: state backup. Doubles the required memory.

Idea: using PBE with type-specific tools, restore the initial state only when a crash occurs.

Method: generate a set of pairs $(input, output)$, then retrieve the initial input using:

- for integers a neural network.
- for strings a synthesizer.



MERGING THE STATES

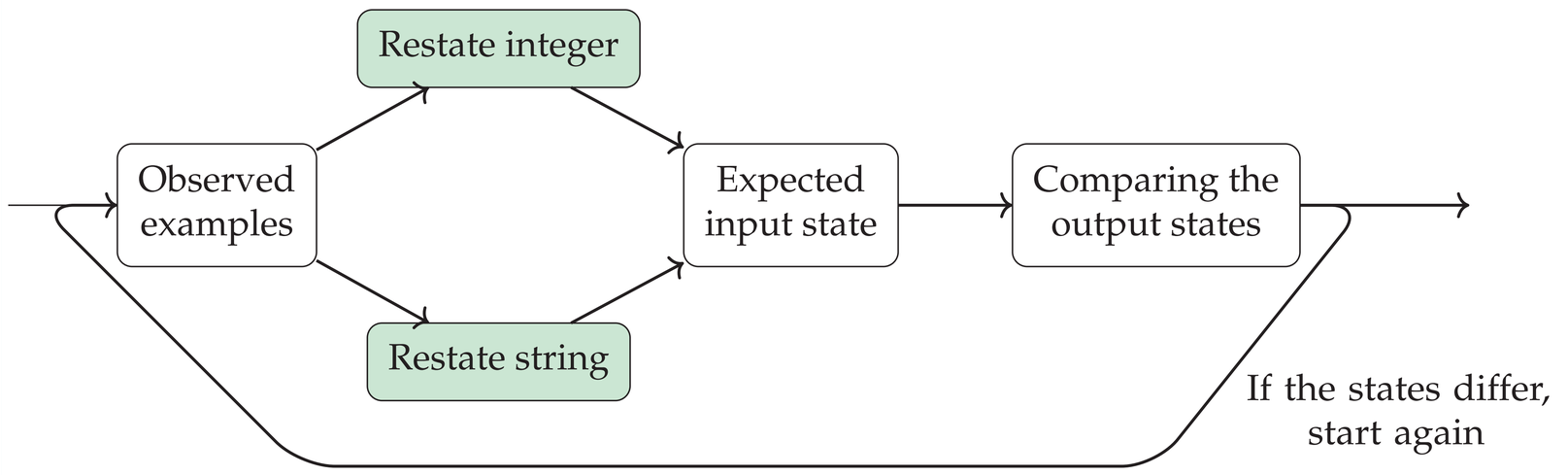


Figure 2: Restate pipeline

RESULTS

```python
# simple_example.py
my_str = "a string"
my_int = 2

def f() -> str:
    global my_str, my_int
    my_str = my_str * 2
    my_int += 5
    print(my_str + my_int)

f()
```

```
$ python3 -m axolotl\
        simple_example.py
```

```
print(my_str + my_int)
TypeError: can only concatenate str (
    not "int") to str

The program has been synthesized:
(define-fun f ( (_arg_0 String) (
    _arg_1 Int)) String
(str.substr
    (str.++ " " _arg_0)
    1
    (/ (str.len _arg_0) 2)))

Training the model...
Function entry found:
arg: None
kwarg: None
global:
        my_str: a string
        my_int: 2
```

Application to real life examples with BugsInPy's benchmark

We selected only bugs that led to a crash and checked if the program runs correctly

- Black: 6/6
- Pandas: 28/79

FUTURE WORKS

- Extend to other types (float, list...)
- Better handling of interleaved variables
- Extend to other programming languages

REFERENCES

[1] Woosuk Lee. Combining the top-down propagation and bottom-up enumeration for inductive program synthesis. *Proc. ACM Program. Lang.*, 5(POPL), jan 2021.

[2] YoungJae Kim, Seungheon Han, Askar Yeltayuly Khamit, and Jooyong Yi. Automated program repair from fuzzing perspective. In *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis*, ISSTA 2023, page 854–866. Association for Computing Machinery, 2023.