Badger: Complexity Analysis with Fuzzing and Symbolic Execution

Yannic Noller[1], Rody Kersten[2], Corina S. Păsăreanu[3]

[1] Humboldt-Universität zu Berlin, Germany
[2] Synopsis, Inc., San Francisco, USA
[3] Carnegie Mellon University Silicon Valley, NASA Ames Research Center, Moffett Field, USA

Problem

Algorithmic complexity analysis enables developers to **reason** about their programs, understand **performance bottlenecks**, and reveal worst-case complexity **vulnerabilities.**

Hybrid testing approaches that involve **fuzzing** and **symbolic execution** have shown promising results in achieving **high code coverage**, uncovering **vulnerabilities**.

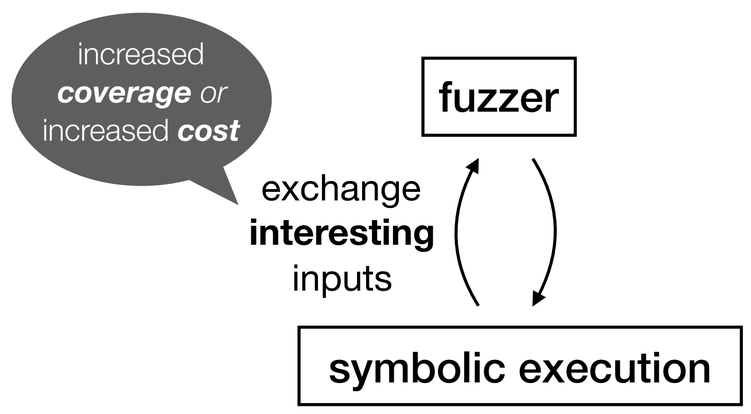
Contribution

- **combine** fuzzing and symbolic execution to find algorithmic complexity vulnerabilities
- **Badger**, a framework for analysis of Java applications
- handling of **user-defined cost**

```
int sumArg (int[] a) {
  int sum = 0;
  for (int i=0; i<a.length; i++){
    sum += a[i];
  }
  Kelinci.addCost(sum);
  return sum;
}
```

Overview

fuzzer and symbolic execution run in **parallel**



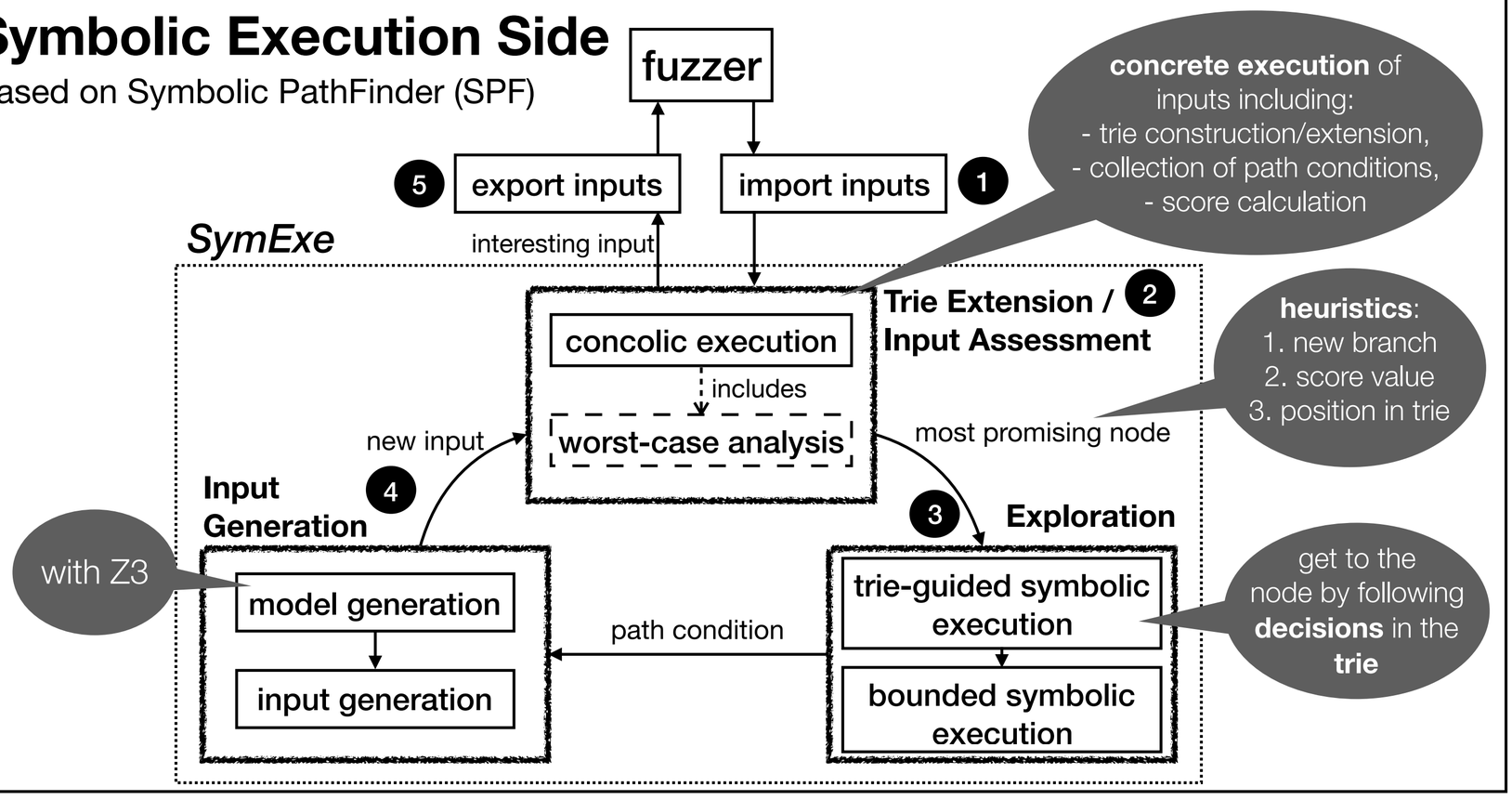
Fuzzing Side

KelinciWCA (based on AFL) is a cost-guided fuzzer; it prioritizes new inputs that increase **coverage** or **cost**. It supports the following cost metrics:

- **timing**, by counting jumps in Java byte-code
- **memory**, by frequently polling the current memory usage
- **user-defined**, by instrumenting program with special method call `Kelinci.addCost(int)`

Symbolic Execution Side

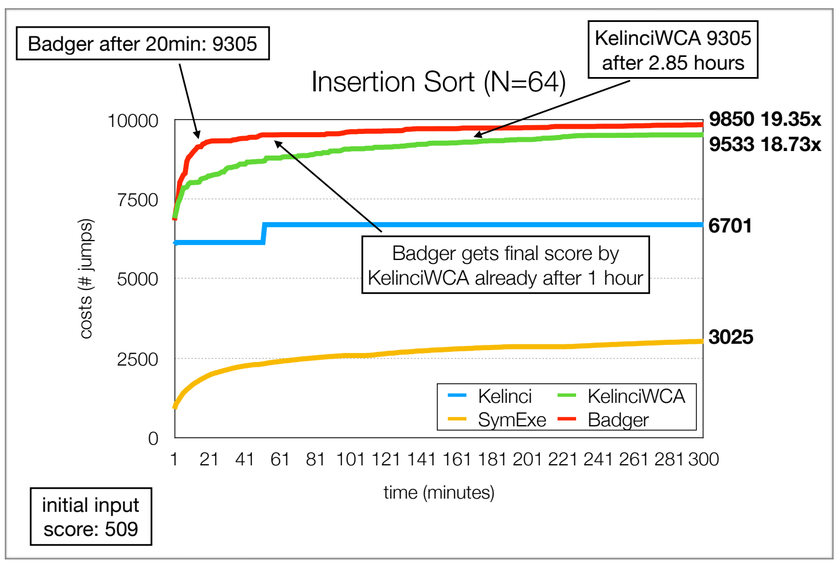
based on Symbolic PathFinder (SPF)

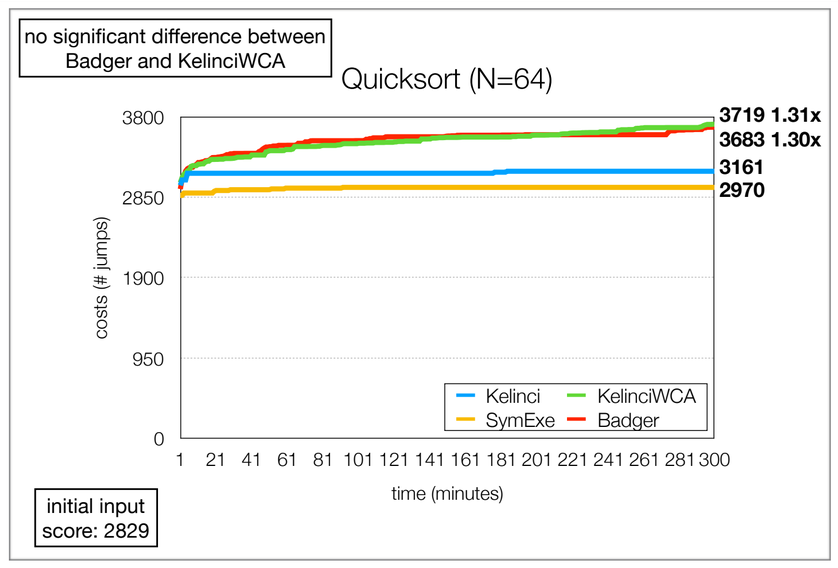


Evaluation

ID	Subject
1	Insertion Sort
2	Quicksort
3a	Regular Expression (fixed input)
3b	Regular Expression (fixed regex)
4	Hash Table
5	Compression
6	Image Processor
7	Smart Contract

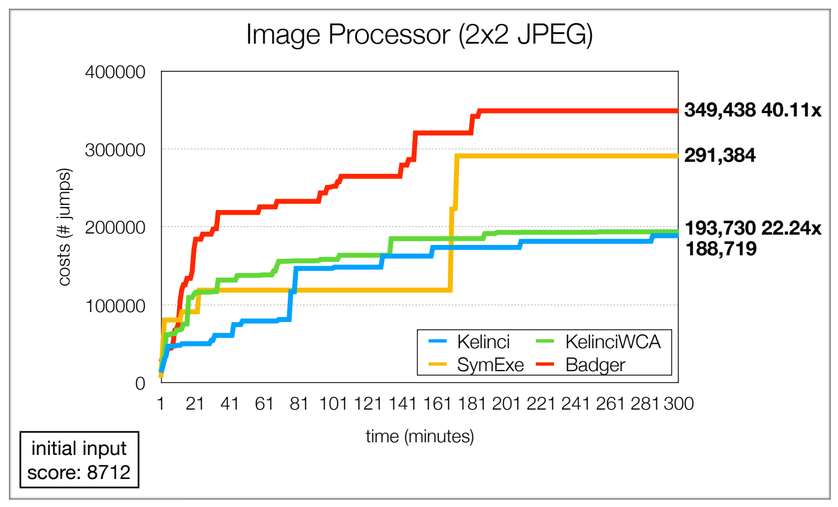
RQ1: *Since Badger combines fuzzing and symbolic execution, is it better than each part on their own (quality + speed)?* Badger always gets a better worst-case then SymExe, and almost always better than KelinciWCA. Badger ist faster than each component itself.





RQ2: *Is KelinciWCA better than Kelinci (quality + speed)?* Except for one case, KelinciWCA is always better than Kelinci.

RQ3: *Can Badger reveal worst-case vulnerabilities?* It performed well in slowdowns and it revealed the actual worst-case JPEG Image in Image Processor.



`git clone https://github.com/isstac/badger.git`

yannic.noller@hu-berlin.de