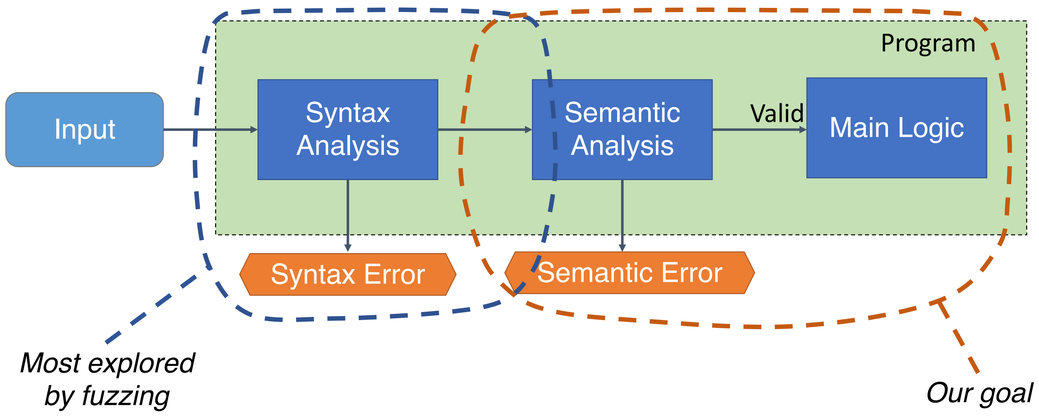
Validity Fuzzing and Parametric Generators for Effective Random Testing

Full paper!

Rohan Padhye*, Caroline Lemieux*, Koushik Sen*, Mike Papadakis†, Yves Le Traon†
*University of California, Berkeley †University of Luxembourg

https://github.com/rohanpadhye/jqf

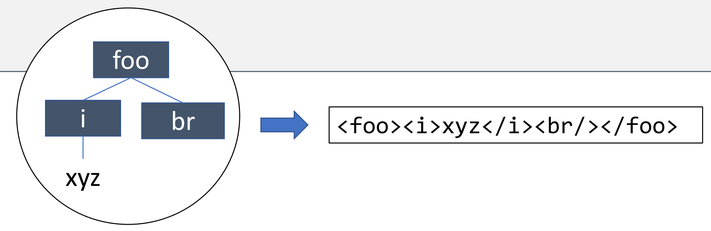
Fuzz testers mostly explore syntax analysis;



we want to test *deeper* stages of programs.

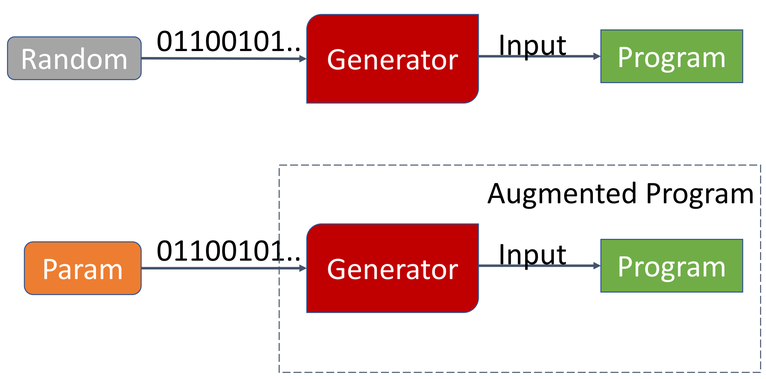
QuickCheck-like generators allow us to create random *highly-structured* inputs.

```
public XMLElement genXML(Random random, int depth=0) {
  String name = random.nextString(MAX_TAG_LENGTH); // Generate random tag name
  XMLElement node = new XMLElement(name);
  if (depth < maxDepth) {
     int n = random.nextInt(MAX_CHILDREN);          // Generate random integer n
     for (int i = 0; i < n; i++) {
       node.addChild(genXML(random), depth + 1);    // Generate child nodes recursively
     }
  }
  if (random.nextBoolean()) {                       // Maybe insert text inside element
    node.addText(random.nextString(MAX_TEXT_LENGTH));
  }
  return node;
}
```



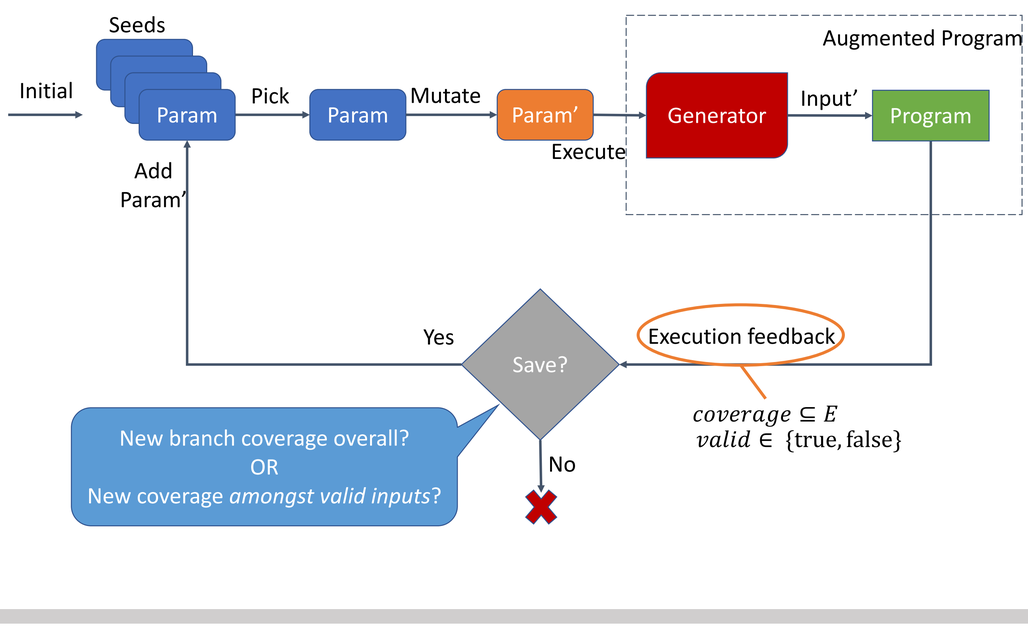
Syntactically valid by construction

Key idea 1: **Parametric Generators**



Bit flips on parameters → Structural mutations on inputs

Key idea 2: **Validity Fuzzing** (with Parametric Generators)

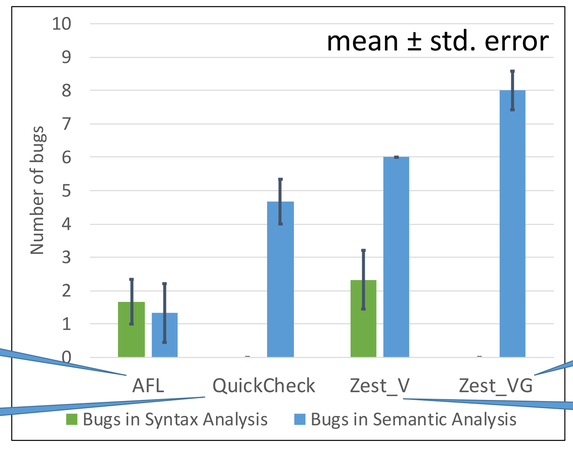


Evaluation

More results in full paper (ISSTA'19)
Semantic Fuzzing with Zest

Generator	Benchmarks
XML	Apache Maven Apache Ant
JavaScript	Google Closure Compiler Mozilla Rhino
Chess FEN	ScalaChess
JVM .class	Apache BCEL

Bugs Found (3 hour timeout)



Total **18** new bugs reported
➔ **7 unique** to Zest!
➔ Within *10 minutes* on average

Example:
Semantic Bug in Closure

```
while ((l_0)){
  while ((l_0)){
    if ((l_0))
    { break;var l_0;continue }
    { break;var l_0 }
 }
}
```

Unreachable statement, but not dead code!

JavaScript generated by Zest_VG