Symbolic Execution and Fuzzing on Guava

Zekai Zhao, Junxiong Lin

12-09-2019
18737

Outline

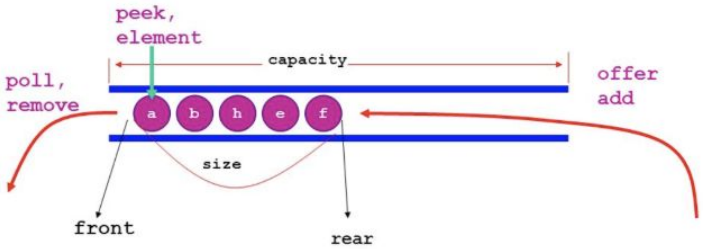
- Context & background
- Plan
- Symbolic execution approach
- Fuzzing approach
- Future work

Context

- **Guava**
  - Google Core Libraries for Java that includes new collection types
- **MinMaxPriorityQueue**
  - Serves as typical double-ended min-max Priority Queue in Java
- **Our repo**
  - https://github.com/zkzhao33/TestingPracticeOnGuava

What we have

- 10 Mutated bugs in source class
  - 4 found by previous unit tests
- 100 unit test cases focusing on behaviors of Priority Queue
  - Behavior
    - create
    - offer, add, addAll
    - poll, pollfirst
    - peekLast, peekFirst
    - removeAt, removeLast, removeFirst
    - toArray
    - clearqueue
    - capacity
  - Testing assertion
    - Ordering is correct
    - Peek/poll the correct element
    - Capacity: initialization/growth as expected



What we are going to do

- **Structural/White box testing with symbolic execution**

  To cover the program code

- **More Spec/Black box testing with fuzzing**

  To cover the specs/input space

Symbolic Execution

- Analyzing a program to determine what inputs cause each part of a program to execute
- Many applications: test-case generation, error detection
- Using Symbolic PathFinder

Configuration

- Using JAVA 8
- Download jpf-core and jpf-symbc
- .jpf/site.properties
  - jpf-core = ${user.home}/CMU/18737/jpf-core
  - Jpf-symbc = ${user.home}/CMU/18737/jpf-symbc
  - extensions=${jpf-core},${jpf-symbc}
- Run a small test

```java
public class Test {
    public static int myMethod(int x, int y){
        int z = x + y;
        if (x > 0) {
            if(y>0){
                z = x/z;
            } else{
                z = x/z;
            }
        } else {
            if(x>0){
                z = z - x;
            } else{
                z = z + x;
            }
        }
        z = 2 * z;
        return z;
    }
    public static void main(String[] args){
        myMethod(0,1);
    }
}
```

```
========================================================== search started: 19-12-8 下午10:57
New sym int x_1_SYMINT min=-2147483648, max=2147483647
New sym int y_2_SYMINT min=-2147483648, max=2147483647
numeric PC: constraint # = 1
(y_2_SYMINT + x_1_SYMINT) != CONST_0 -> false

### PCs: total:1 sat:0 unsat:1

numeric PC: constraint # = 1
(y_2_SYMINT + x_1_SYMINT) = CONST_0 -> true

### PCs: total:2 sat:1 unsat:1

string analysis: SPC # = 0
NPC constraint # = 1
(y_2_SYMINT + x_1_SYMINT) = CONST_0
```

Model a simple version of MinMaxPriorityQueue

```java
public static void main(String[] args){
    PriorityQueue pq=new PriorityQueue();

    for (int i = 0; i < 4; i++) {
        Verify.beginAtomic();
        try {
            switch (Verify.getInt(0,3)) {
            case 0:
                pq.peek();
                break;
            case 1:
                pq.push(114514);
                break;
            case 2:
                pq.contains(114514);
                break;
            case 3:
                pq.pop();
                break;
            }

        } catch (Throwable t) {
            // don't care
        }

        Verify.endAtomic();
    }
}
```

```
target=PriorityQueue
symbolic.method=PriorityQueue.contains(sym),PriorityQueue.push(sym),PriorityQueue.pop(),PriorityQueue.peek();
classpath=${jpf-symbc}/build/examples
symbolic.debug=true
symbolic.min_int=-100
symbolic.max_int=100
symbolic.min_long=-100
symbolic.max_long=100
symbolic.lazy=on
#symbolic.arrays=true
#search.depth_limit = 25
cg.randomize_choices= VAR_SEED
search.class = .search.heuristic.BFSHeuristic
#symbolic.dp=no_solver
vm.storage.class=nil
listener = gov.nasa.jpf.symbc.sequences.SymbolicSequenceListener
```

```
========================================================= search started: 19-12-8 下午9:47
New sym int value_1_SYMINT min=-100, max=100
New sym int value_2_SYMINT min=-100, max=100
New sym int value_3_SYMINT min=-100, max=100
New sym int value_4_SYMINT min=-100, max=100
New sym int value_5_SYMINT min=-100, max=100
New sym int value_6_SYMINT min=-100, max=100
New sym int value_7_SYMINT min=-100, max=100
New sym int value_8_SYMINT min=-100, max=100
New sym int value_9_SYMINT min=-100, max=100
New sym int value_10_SYMINT min=-100, max=100
numeric PC: constraint # = 1
value_1_SYMINT != value_7_SYMINT -> true

### PCs: total:1 sat:1 unsat:0

string analysis: SPC # = 0
NPC constraint # = 1
value_1_SYMINT != value_7_SYMINT
numeric PC: constraint # = 1
value_1_SYMINT = value_7_SYMINT -> true

### PCs: total:2 sat:2 unsat:0

string analysis: SPC # = 0
NPC constraint # = 1
value_1_SYMINT = value_7_SYMINT
New sym int value_11_SYMINT min=-100, max=100
New sym int value_12_SYMINT min=-100, max=100
New sym int value_13_SYMINT min=-100, max=100
New sym int value_14_SYMINT min=-100, max=100
New sym int value_15_SYMINT min=-100, max=100
New sym int value_16_SYMINT min=-100, max=100
New sym int value_17_SYMINT min=-100, max=100
New sym int value_18_SYMINT min=-100, max=100
New sym int value_19_SYMINT min=-100, max=100
New sym int value_20_SYMINT min=-100, max=100
New sym int value_21_SYMINT min=-100, max=100
New sym int value_22_SYMINT min=-100, max=100
New sym int value_23_SYMINT min=-100, max=100
New sym int value_24_SYMINT min=-100, max=100
New sym int value_25_SYMINT min=-100, max=100
```

Push only

```
[push(-100), push(-99), push(-98), push(-97), push(-9223372036854775808)]
[push(-100), push(-99), push(-98), push(-98), push(-9223372036854775808)]
[push(-100), push(-99), push(-99), push(-100), push(-100)]
[push(-100), push(-99), push(-99), push(-100), push(-99)]
[push(-100), push(-99), push(-99), push(-98), push(-9223372036854775808)]
[push(-100), push(-98), push(-99), push(-97)]
[push(-100), push(-98), push(-99), push(-98), push(-9223372036854775808)]
[push(-100), push(-99), push(-100), push(-99), push(-99)]
[push(-100), push(-99), push(-100), push(-99), push(-100)]
[push(-100), push(-99), push(-100), push(-98), push(-100)]
[push(-100), push(-99), push(-100), push(-98), push(-99)]
[push(-99), push(-98), push(-100), push(-100), push(-99)]
[push(-100), push(-100), push(-100), push(-99), push(-99)]
[push(-100), push(-100), push(-100), push(-99), push(-100)]
[push(-99), push(-100), push(-100), push(-99), push(-100)]
[push(-99), push(-100), push(-100), push(-99), push(-99)]
[push(-99), push(-99), push(-100), push(-100), push(-99)]
[push(-100), push(-100), push(-99), push(-99), push(-9223372036854775808)]
[push(-100), push(-100), push(-99), push(-98), push(-9223372036854775808)]
```

```
        @Test
        public void test143() {
                priorityqueue.push(-99);
                priorityqueue.push(-100);
                priorityqueue.push(-99);
                priorityqueue.push(-98);
                priorityqueue.push(-97);
        }

        @Test
        public void test144() {
                priorityqueue.push(-97);
                priorityqueue.push(-100);
                priorityqueue.push(-99);
                priorityqueue.push(-98);
                priorityqueue.push(-97);
        }
}

============================================================ results
no errors detected

============================================================ statistics
elapsed time:       00:00:01
states:             new=3411,visited=0,backtracked=3411,end=1942
search:             maxDepth=19,constraints=0
choice generators:  thread=1 (signal=0,lock=1,sharedRef=0,threadApi=0,reschedule=0), data=1469
heap:               new=1732,released=24048,maxLive=370,gcCycles=2939
instructions:       128000
max memory:         309MB
loaded code:        classes=65,methods=1434

============================================================ search finished: 19-12-9 下午1:34
```

Trying Random Execution Order with Push and Contains

```
========================================================= Method Sequences
[push(-100), contains(-100)]
[push(-100), contains(-99)]
[push(-99), contains(-100)]
[peek(), push(-100), contains(-99)]
[peek(), push(-100), contains(-100)]
[pop(), push(-100), contains(-99)]
[pop(), push(-100), contains(-100)]
[push(-100), contains(-100), peek()]
[push(-100), contains(-100), contains(-9223372036854775808)]
[push(-100), contains(-100), pop()]
[push(-100), contains(-100), push(-9223372036854775808)]
[push(-100), peek(), contains(-100)]
[push(-100), peek(), contains(-99)]
[push(-100), push(-9223372036854775808), contains(-99)]
[push(-100), push(-9223372036854775808), contains(-100)]
[push(-100), push(-100), push(-9223372036854775808)]
[push(-100), push(-99), push(-9223372036854775808)]
[contains(-9223372036854775808), push(-100), contains(-100)]
[contains(-9223372036854775808), push(-100), contains(-99)]
```

Results

```
        public void test419() {
                priorityqueue.push(-100);
                priorityqueue.push(-98);
                priorityqueue.contains(-97);
                priorityqueue.contains(-99);
        }

        @Test
        public void test420() {
                priorityqueue.push(-100);
                priorityqueue.push(-98);
                priorityqueue.contains(-99);
                priorityqueue.contains(-97);
        }
}

====================================================== results
no errors detected

====================================================== statistics
elapsed time:       00:00:00
states:             new=1265,visited=0,backtracked=1265,end=519
search:             maxDepth=17,constraints=0
choice generators:  thread=1 (signal=0,lock=1,sharedRef=0,threadApi=0,reschedule=0), data=746
heap:               new=1522,released=7915,maxLive=384,gcCycles=809
instructions:       30086
max memory:         245MB
loaded code:        classes=69,methods=1449
```

Pros and Cons

- Pros
  - Explore different kind of work flow
  - Analysis without real execution
  - Pre-processing to eliminate unsatisfiable test cases
- Cons
  - In heap, the symbol of exact number isn’t really related to the process
  - Lack of documentation
  - Need pruning
  - Fully implementation

Fuzzing

- What is fuzzing ?
  - Testing software with invalid and possibly malicious data.
- What is the goal of fuzzing?
  - Evaluate program response to invalid input, rather than “common case” inputs (what we used to do!) used for plain functional testing.

Radamsa

- An open source input generator that mutates given input by applying pre-defined mutation rules and patterns.

```
$ echo 192.168.106.103 | radamsa --count 10 --seed 0
-107.167.106.103
192.168.8407971865571866.-9?51547373063626639424131940 69
191.1A1.1A1.106.1
192.129.18.106.103
192.168.0.103
192.170141183460.106.1802311213346089.104
-3402823669209.106.168.106.16.103
192093846346337460765704.192.65704.-1.?-18446744073709518847
192.106.0
191.168.106.103
$ echo 192.168.106.103 | radamsa --count 1 --seed 0 | xargs ping
ping: invalid option -- 1
```

Mutations

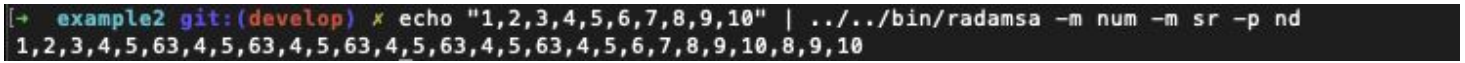
```
$ ./radamsa --list
Mutations (-m)
   ...
   bd: drop a byte
   bf: flip one bit
   bi: insert a random byte
   ...
   sr: repeat a sequence of bytes
   sd: delete a sequence of bytes
   ld: delete a line
   ...
   ls: swap two lines
   ...
   num: try to modify a textual number
   xp: try to parse XML and mutate it
   ...
Mutation patterns (-p)
   od: Mutate once
   nd: Mutate possibly many times
   bu: Make several mutations closeby once
```

Mutations

- Generate random long length inputs by given format

- Generate inputs with randomly modified integer numbers by given pattern.

- Combine previous arguments together.



Application

```java
public void testPollFirstWithIntegerMinHeapContainsNElements() {
    MinMaxPriorityQueue<Integer> q = MinMaxPriorityQueue.create();
    Collections.addAll(q, ...elements: 1,2,3,4,5,6,7,8,9,10);
    assertEquals(q.pollFirst().intValue(), actual: 1);
    assertEquals(q.peek().intValue(), actual: 2);
}
```

```java
public void testPollFirstWithIntegerMinHeapContainsNElements2() {
    MinMaxPriorityQueue<Integer> q = MinMaxPriorityQueue.create();
    Collections.addAll(q, ...elements: 1,2,3,4,5,6,5,6,5,6,5,6,5,6,5,
            6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,
            5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,
            6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,
            5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,
            6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,
            5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,
            6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,
            5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,
            6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,
            5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,
            6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,
            5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,
            6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,
            5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,5,6,32769,-4,0,7,8,9,10);
    assertEquals(q.pollFirst().intValue(), actual: -4);
    assertEquals(q.peek().intValue(), actual: 0);
}
```

Pros & Cons

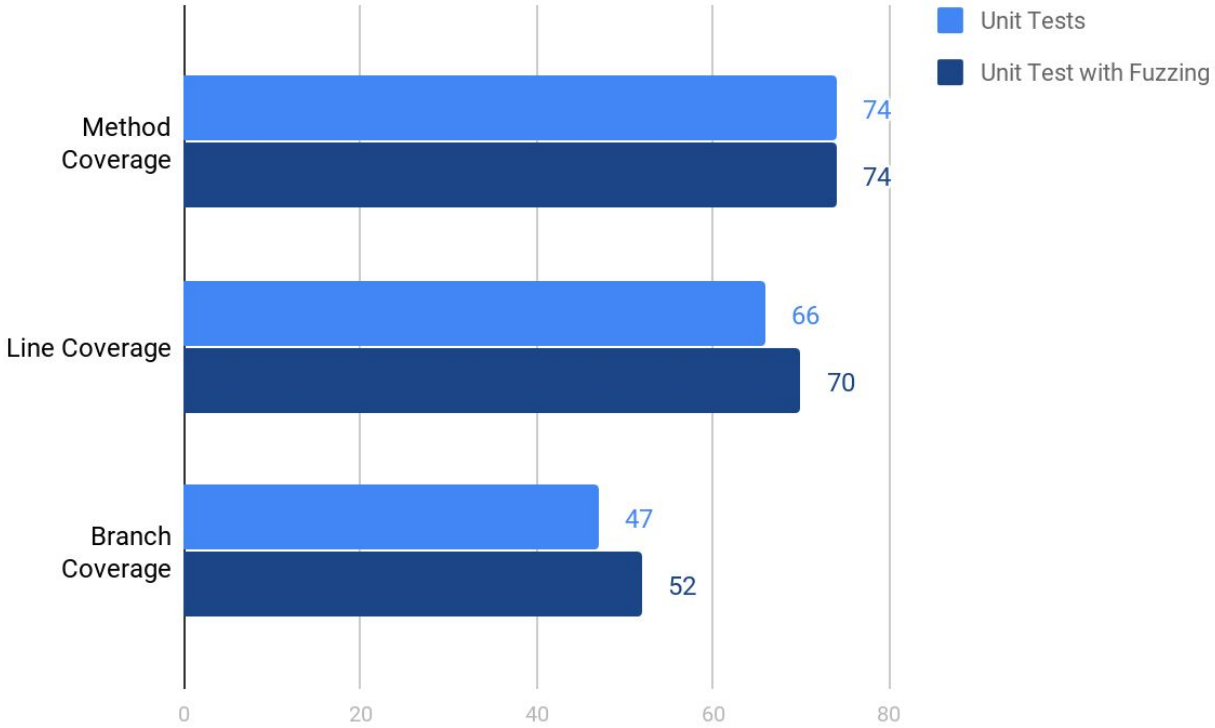
Pros:

- Easily generate inputs with given format.
- Able to generate special inputs (corner cases).

Cons:

- Need manually writing assertion (semi-automatic).
- Sometimes generate out-of-bound inputs.

Coverage


Unit Tests
Unit Test with Fuzzing
Method Coverage
74
74
Line Coverage
66
70
Branch Coverage
47
52
0
20
40
60
80

Results

- Coverage improved
  - corner inputs made the tests cover more branches
- One extra mutation bug found
  - A new mutation bug that is related to heap size growing was found

Future work

- Explore more function of SPF
- Try different fuzzer
  - Random behavior
  - Automatic fuzzing

Q&A