

Korat: Automated Testing Based on Java Predicates

Written by Chandrasekhar Boyapati,
Sarfraz Khurshid, and Darko Marinov
(MIT Laboratory for Computer Science)

Topics Covered

- Problem Identified
- Approach
- Test Cases
- Testing Methods
- Results
- Conclusions
- Discussion

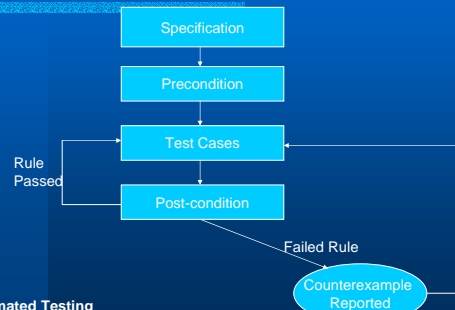
2

Problem

Software development and maintenance costs can be exorbitant when implementing labor-intensive, time-consuming manual software testing and test case generation.

3

How Does Korat Work?



Automated Testing

4

Korat Characteristics

- Predicate and a bound
 - Inputs where predicate is true
- Backtracking
 - Explore bounded input space of the predicate
- Candidate inputs
- Pruning

5

Binary Tree Example

```
import java.util.*;
class BinaryTree{
  private Node root;
  private int size;
  static class Node {
    private Node left;
    private Node right;
  }
  public boolean repOk(){
    if (root == null) return size == 0;
    Set visited = new Hash Set();
    visited.add(root);
    LinkedList workList = new LinkedList();
    workList.add(root);
    while (!workList.isEmpty()) {
      Node current = (Node) workList.removeFirst();
      if (current.left != null) {
        if (! visited.add(current.left))
          return false;
        workList.add(current.left);
      }
      .....
    }
  }
}
```

6

Example – Korat Test Case

```
public static Finitization finBinaryTree(int
NUM_Node) {
    Finitization f = new Finitization(BinaryTree.class);
    ObjSet nodes = f.createObject("Node",
NUM_NODE);

    nodes.add(null);
    f.set("root",nodes);
    f.set("size",NUM_NODE);
    f.set("Node.left",nodes);
    f.set("Node.right",nodes);
    return f;
}
```

7

Korat method correctness

```
//@public invariant repOk();

/*@public normal_behavior
@ requires has(n);
@ ensures !has(n);
@*/
public void remove(Node n){
    .....
}
```

8

Finitization

- Generate a “finite” space of a predicate’s inputs
 - set of bounds that limits the size of the inputs
- Class Domain for each type of class
- Set of classes whose objects the field can point to
- Korat uses Java type declarations
- Can be specialized or generalized

9

State Space

```
public static Finitization finBinaryTree(int NUM_Node) {
    NUM_NODE = 3
    Finitization f = new Finitization(BinaryTree.class);
    ObjSet nodes = f.createObject("Node", NUM_NODE);

    nodes.add(null);
    f.set("root",nodes);
    f.set("size",NUM_NODE);
    f.set("Node.left",nodes);
    f.set("Node.right",nodes);
    return f;
}
```

field

field

field for 3 nodes

field for 3 nodes

10

Searching Process

- All candidate vector sets start at zero
- Korat sets fields in the objects according to the values
- repOk is invoked to check the validity of the candidate
- Korat monitors the fields that repOk accesses via field-ordering list
- “Prunes” large portions of the search space

11

Testing Methods

- JML annotations
 - requires, ensures, invariant
 - Java syntax and semantics for expression
- Normal and Exceptional Behaviors

12

Correctness

- Each output is checked with a test oracle
- Partial correctness
 - invoke repOk in the post state
 - determines if method preserves its class invariant
 - false- >counterexample
- JML vs JML+JUNIT

13

Results

- javac modification
- Linux machine PIII 800MHz processor
- finitization parameters set to the same size
 - Korat generated all valid structures
 - large state spaces
 - Pruning to avoid generating unnecessary candidates

14

Alloy or Korat?

- Alloy: declarative language based on relations
 - models are translated into boolean formulas
 - symmetry breaking predicates
 - different satisfying assignments represent non isomorphic instances
 - can not handle arbitrary arithmetic

15

Related Work

- Specification based Testing
 - TestEra
- Static Analysis
 - Extended Static Checker
 - Three-Valued-Logic Analyzer (TVLA)
- Software Model Checking
 - JavaPathFinder
 - VeriSoft
 - Bandera
 - JCAT

16

Conclusions

- Implementations of abstract data types should have an accurate means of testing
- Korat operates faster than AA
- Easier to implement with JML because less learning curve
- Operates in both small and large state spaces

17

Discussion

- Where is the balance for testing for “normal errors” and “data structure errors”?
- How does this resolve the problem statement identified in this paper?
- Based on the testing and results, where does the burden of testing fall? Where should it fall?

18

References

- *International Symposium on Software Testing and Analysis (ISSTA 2002), Rome, Italy. Jul 2002*