Program Testing and Analysis: Random and Fuzz Testing

Prof. Dr. Michael Pradel
Software Lab, TU Darmstadt
Outline

■ Feedback-directed random test generation
   Based on Feedback-Directed Random Test Generation, Pacheco et al., ICSE 2007

■ Adaptive random testing
   Based on ARTOO: Adaptive Random Testing for Object-oriented Software, Ciupa et al., ICSE 2008

■ Fuzz testing
   Based on Fuzzing with Code Fragments, Holler et al., USENIX Security 2012
Feedback-directed Test Generation

Idea: **Guide** randomized **creation** of new test inputs by **feedback** about execution of previous inputs

- Avoid redundant inputs
- Avoid illegal inputs

- Test input here means **sequence of method calls**
- Software under test: Classes in Java-like language
Motivating Examples

Two randomly generated tests:

Set s = new HashSet();
s.add("hi");
assertTrue(s.equals(s));

Set s = new HashSet();
s.add("hi");
s.isEmpty();
s.isEmpty();
assertTrue(s.equals(s));
Motivating Examples

Two randomly generated tests:

Set s = new HashSet();
s.add("hi");
assertTrue(s.equals(s));

Set s = new HashSet();
s.add("hi");
\boxed{s.isEmpty();}
assertTrue(s.equals(s));
Motivating Examples

Two randomly generated tests:

Set s = new HashSet();
s.add("hi");
assertTrue(s.equals(s));

Set s = new HashSet();
s.add("hi");
s.isEmpty();
assertTrue(s.equals(s));

Redundant test
Motivating Examples (2)

Three randomly generated tests:

```java
Date d = new Date(2006, 2, 14);
assertTrue(d.equals(d));

Date d = new Date(2006, 2, 14);
d.setMonth(-1);
assertTrue(d.equals(d));

Date d = new Date(2006, 2, 14);
d.setMonth(-1);
d.setDay(5);
assertTrue(d.equals(d));
```
Motivating Examples (2)

Three randomly generated tests:

```java
Date d = new Date(2006, 2, 14);
assertTrue(d.equals(d));

Date d = new Date(2006, 2, 14);
  d.setMonth(-1);
assertTrue(d.equals(d));

Date d = new Date(2006, 2, 14);
  d.setMonth(-1);
  d.setDay(5);
assertTrue(d.equals(d));
```

Violates pre-condition
Motivating Examples (2)

Three randomly generated tests:

```java
Date d = new Date(2006, 2, 14);
assertTrue(d.equals(d));
```

```java
Date d = new Date(2006, 2, 14);
d.setMonth(-1);
assertTrue(d.equals(d));
```

```java
Date d = new Date(2006, 2, 14);
d.setMonth(-1);
d.setDay(5);
assertTrue(d.equals(d));
```

Illegal tests
Approach

- Build test inputs **incrementally**
  - New test inputs extend previous ones
- As soon as test input is created, **execute it**
- Use execution results to guide generation
  - away from redundant or illegal method sequences
  - toward sequences that create **new object states**
Randoop: Implementation of feedback-directed random test generation

- **Input:**
  - Classes under test
  - Time limit
  - Set of contracts
    - Method contracts, e.g., `o.hashCode()` throws no exception
    - Object invariants, e.g., `o.equals(o) == true`

- **Output:** Test cases with assertions
Example

```java
HashMap h = new HashMap();
Collection c = h.values();
Object[] a = c.toArray();
LinkedList l = new LinkedList();
l.addFirst(a);
TreeSet t = new TreeSet(l);
Set u = Collections.unmodifiableSet(t);
assertTrue(u.equals(u));
```
Example

```java
HashMap h = new HashMap();
Collection c = h.values();
Object[] a = c.toArray();
LinkedList l = new LinkedList();
l.addFirst(a);
TreeSet t = new TreeSet(l);
Set u = Collections.unmodifiableSet(t);
assertTrue(u.equals(u));
```

Fails when executed
Example

```java
HashMap h = new HashMap();
Collection c = h.values();
Object[] a = c.toArray();
LinkedList l = new LinkedList();
l.addFirst(a);
TreeSet t = new TreeSet(l);
Set u = Collections.unmodifiableSet(t);
assertTrue(u.equals(u));
```

No contracts violated up to last method call

Fails when executed
**Algorithm**

1. Initialize **seed components**: \( i=0; \ b=false; \ ... \)

2. Do until time limit expires:
   - Create a new sequence
     - Randomly pick a method \( T_0.m(T_1, ..., T_k)/T_{ret} \)
     - For each \( T_i \), randomly pick a sequence \( S_i \) from the components that constructs a value \( v_i \) of type \( T_i \)
     - Create new sequence
       \[ S_{new} = S_1; \ldots; S_k; T_{ret} \ v_{new} = m(v_1, ..., v_k); \]
     - If \( S_{new} \) was previously created (lexically), go to
   - **Classify the sequence** \( S_{new} \)
     - May discard, output as test case, or add to components
Classifying a Sequence

1. Start
2. Execute and check contracts
3. Contract violated?
   - Yes: Minimize sequence
   - No: Check components
4. Sequence redundant?
   - No: Contract-violating test case
   - Yes: Discard sequence

Image source: Slides by Pacheco et al.
Redundant Sequences

- During generation, maintain a set of all objects created
- Sequence is redundant if all objects created during its execution are in the above set (using equals() to compare)
- Could also use more sophisticated state equivalence methods
  - E.g., heap canonicalization
Classes under test: java.util.*

1) Pick a method
   → No values are needed
   → New sequence

2) Classify sequence
   → No contract violated
   → Not redundant

⇒ Add to components
3) Pick method
   \[ \rightarrow \text{Sequence} \]

4) Clarify sequence
   \[ \rightarrow \text{No contract violated} \]
   \[ \rightarrow \text{Redundant} \]
   \[ \Rightarrow \text{Discard sequence} \]
5) Pick method
   $\text{HashMap.values}$
   $\rightarrow$ Need sequence that constructs
   value of type $\text{HashMap}$
   $\rightarrow$ Use sequence of step 2)

4) Create sequence
   $\text{HashMap } h = \text{new HashMap}()$
   $\text{Collection } c = h.\text{values}()$

6) Clarify sequence
   $\rightarrow$ No contract violated
   $\rightarrow$ Not redundant

$\Rightarrow$ Add to components
Test Oracles

- Testing only useful if there is an oracle

- Randoop outputs two kinds of oracles
  - Oracle for contract-violating test cases:
    ```java
    assertTrue(u.equals(u));
    ```
  - Oracle for normal-behavior test cases:
    ```java
    assertEquals(2, l.size());
    assertEquals(false, l.isEmpty());
    ```
Quiz

Which of these tests may be created by Randoop?

Test 1:  
```java
LinkedList l = new LinkedList(); 
l.add(23);
```

Test 2:  
```java
LinkedList l = new LinkedList(); 
l.get(-5);
```

Test 3:  
```java
LinkedList l = new LinkedList(); 
l.add(7); 
assertEquals(l.getFirst(), 7);
```
Quiz

Which of these tests may be created by Randoop?

Test 1:
```java
LinkedList l = new LinkedList();
l.add(23);  // (oracle missing)
```

Test 2:
```java
LinkedList l = new LinkedList();
l.get(-5);  // (crashes)
```

Test 3:
```java
LinkedList l = new LinkedList();
l.add(7);
assertEquals(l.getFirst(), 7);
```
Results

- Applied to data structure implementations and popular library classes
- Achieves 80-100% basic block coverage
- Finds various bugs in JDK collections, classes from the .NET framework, and Apache libraries

Read Pacheco et al.’s paper for details
Outline

- Feedback-directed random test generation
  Based on *Feedback-Directed Random Test Generation*, Pacheco et al., ICSE 2007

- Adaptive random testing
  Based on *ARTOO: Adaptive Random Testing for Object-oriented Software*, Ciupa et al., ICSE 2008

- Fuzz testing
  Based on *Fuzzing with Code Fragments*, Holler et al., USENIX Security 2012