Contributions:

1. A technique to generate concurrent tests that exercise a CUT from multiple threads.
2. A generic test oracle that finds concurrent executions that expose thread safety bugs in the CUT
3. Empirical evidence that combining concurrent test generation with the thread safety oracle is effective in automatically finding concurrency bugs

Overview:

1. Generation of tests that lead to an error
2. Oracle which can recognize thread safety problems

Generation of concurrent tests:

1. Class C is an input (A is a set of auxiliary classes that C depend on)
2. Generator creates call sequences
3. The call sequence is well-defined if each input variable of a call c\_j is the output variable of a call c\_i with i<j, that is, if each call uses only variables defined in prior calls.
4. A test consists of a prefix and a set of suffixes
5. Tasks
   1. Each task takes a call sequence sin = (c1,…,ci) and returns a new sequence sout = (c1,…,ci,cj,…cn)
   2. Three kind of tasks: instantiateCUTTask, callCUTTask and parameterTask
   3. A task succeeds if it extends sin with additional calls in such a way that sout is well-defined and that sout executes in a single thread without throwing an uncaught exception

Test Generation Algorithm:

1. Three global variables: the set P of prefixes, the map M assigning a prefix to its set of suffixes and the set T of already generated but not yet returned tests
2. Three main steps: creates a new prefix or choses a previously created prefix, creates a new suffix for the prefix, creates tests by combining the new suffix which each existing suffix.
3. Creating prefixes
   1. Create a prefix -> invokes instantiateCUTTask -> randomly chooses a method m in C and A -> invokes parameterTasks (method parameters)
   2. instantiateCUTTask returns a call sequence that creates all required parameters, stores them into output variables and returns m.
   3. callCUTTask randomly choses a method among all methods on C and invokes the parameterTask for each required parameter.
4. Creating suffixes
   1. Creates a suffix for a prefix p (repeatedly invokes callCUTTask)
   2. Append a call to CUT instance, the new suffix is added for the set of suffixes for prefix p
5. Creating tests
   1. Combines the new suffix with each existing suffix for the prefix into a test.
   2. Stores created tests in T and on further invocations returns a randomly selected test from T until T is empty
6. Thread safety oracle:
   1. Compares concurrent execution to test linearization
   2. Linearization of a test (p, s1, s2) appends to p calls from s1 and s2 preserving an order
   3. A class is a thread-safe if each concurrent test execution has an equivalent linearization
   4. The test oracle tries to find a test that exposes behavior not possible with any linearization of the test
   5. Two executions are equivalent if neither e1 or e2 results in an exception or a deadlock, or both e1 and e2 fail for the same reason (the same type of exception is thrown or both executions end with a deadlock)
   6. For a failed concurrent test, the test oracle executes all possible linearizations of the concurrent test. If the error does not repeat during the linearization replay, a bug is reported.
   7. Two assumptions: exceptions and deadlocks are problems if they occur only in a concurrent test, sequential executions behave deterministically
7. Implementation
   1. Reflection based approach
   2. Takes a random seed as an input
   3. Deadlocks are detected using management interface of JVM
   4. Acceptable performance
8. Evaluation
   1. Code bases: Java stl, Apache Commons DBCP, XStream, LingPipe, JFreeChart, Joda-Time
   2. Bugs found: Java stl, Apache Commons DBCP
9. Finding concurrency bugs
   1. Data races: happens before, unsynchronized access to shared data
   2. Atomicity violations: specifications of atomic blocks
   3. Deadlocks
   4. Active testing: uses a scheduler to provoke an exception
   5. Linearizability: correctness criterion for concurrent objects
   6. Other correctness criteria: type state errors
10. Conclusions:
    1. Automatic technique
    2. Technique for generating tests and an oracle