KING'S College LONDON

Week 7: Property-Based Testing

Dr Gunel Jahangirova

6CCS3SMT/7CCSMASE Software Measurement and Testing

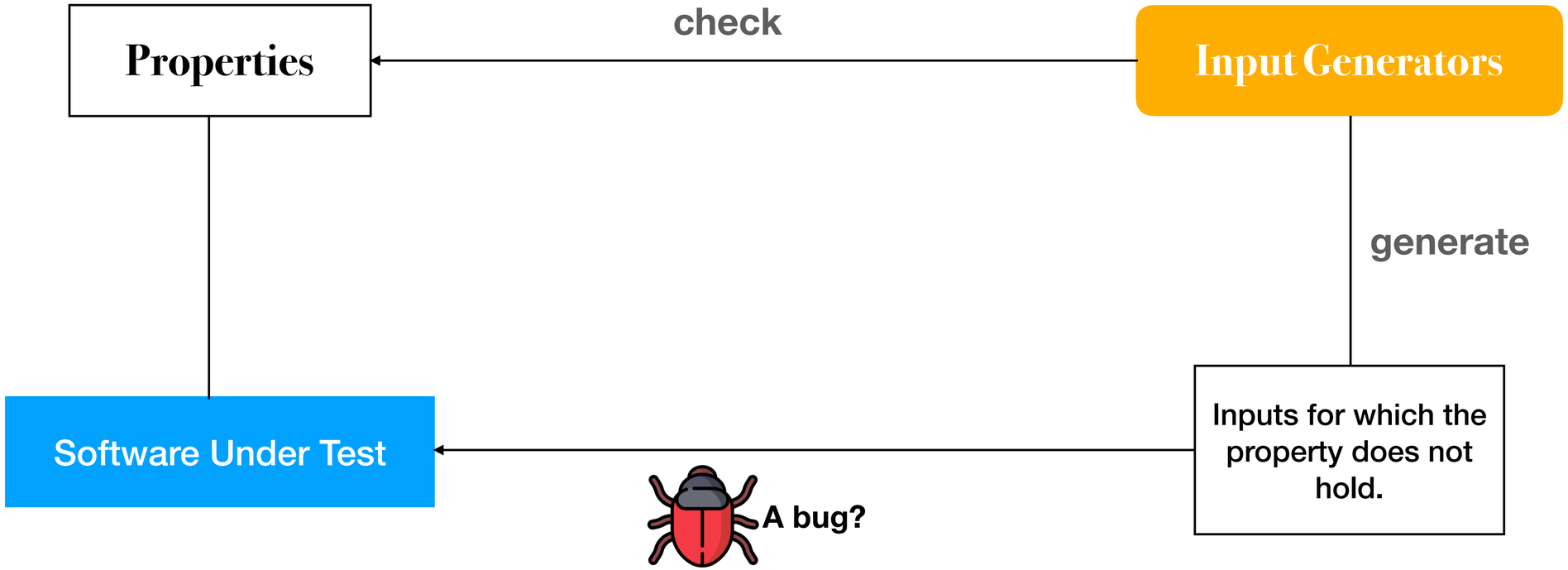
Faculty of Natural, Mathematical & Engineering Sciences
Department of Informatics

Property-Based Testing

KING'S College LONDON

Property-Based Testing (PBT) is a software testing approach where instead of specifying individual test cases with specific inputs and expected outputs, you define general properties or invariants that should always hold true for a wide range of inputs. The testing framework then generates numerous random or structured inputs to automatically verify that these properties are consistently satisfied by the system under test (SUT).

Property-Based Testing



Property-Based Testing

KING'S College LONDON

Example scenario: public int multiply(int a, int b)

Example-based Testing

```
@Test
public void testPositiveNumbers() {
    int a = 3;
    int b = 5;
    int expected = 15;
    assertEquals(expected, multiply(a, b));
}
```

```
@Test
public void testZeroMultiplication() {
    int a = 0;
    int b = 7;
    int expected = 0;
    assertEquals(expected, multiply(a, b));
}
```

Property-based Testing

Property 1: "Multiplication is commutative (a * b == b * a)"

```
@Property
void multiplicationIsCommutative(@ForAll int a, @ForAll int b) {
    assertThat(multiply(a, b)).isEqualTo(multiply(b, a));
}
```

Property 2: "Multiplying any number by 0 should result in 0"

```
@Property
void multiplyingByZeroGivesZero(@ForAll int a) {
    assertThat(multiply(a, 0)).isEqualTo(0);
    assertThat(multiply(0, a)).isEqualTo(0);
}
```

Property-Based Testing

KING'S College LONDON

Example scenario: public int[] sortArray(int[] array)

Example-based Testing

```
@Test
public void testSortArrayWithPositiveNumbers() {
    int[] input = {3, 1, 2};
    int[] expectedOutput = {1, 2, 3};
    assertArrayEquals(expectedOutput, sortArray(input));
}
```

```
@Test
public void testSortArrayWithEmptyArray() {
    int[] input = {};
    int[] expectedOutput = {};
    assertArrayEquals(expectedOutput, sortArray(input));
}
```

Property-based Testing

Property 1: "Sorted output should always be ordered"

```
@Property
void sortedArrayIsOrdered(@ForAll int[] array) {
    int[] sortedArray = sortArray(array);
    for (int i = 0; i < sortedArray.length - 1; i++) {
        assertThat(sortedArray[i]).isLessThanOrEqualTo(sortedArray[i + 1]);
    }
}
```

Property 2: "Sorting an already sorted array should give the same result"

```
@Property
void sortingAnAlreadySortedArray(@ForAll int[] array) {
    int[] sortedArray = sortArray(array.clone());
    int[] sortedAgain = sortArray(sortedArray);
    assertThat(sortedAgain).containsExactly(sortedArray);
}
```

Property-Based Testing Libraries

KING'S College LONDON

Programming Language	PBT Libraries
Haskell	QuickCheck
Java	jqwik
Python	Hypothesis
JavaScript	Fast-check, jsverify
C	Theft
C++	RapidCheck