Property based Testing (in C#)

Bart Wullems

https://bartwullems.blogspot.com

bart.wullems@ordina.be

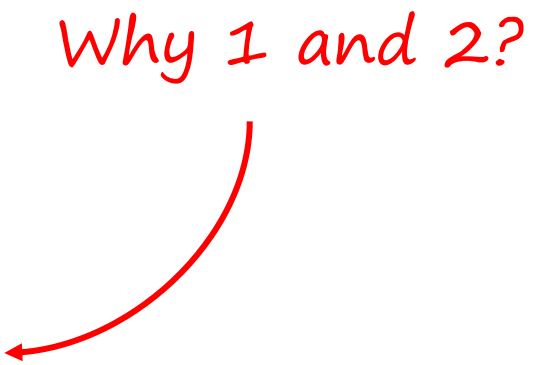
Some questions…

ORDINA
a Sopra Steria company

Example Based Testing

Example based testing

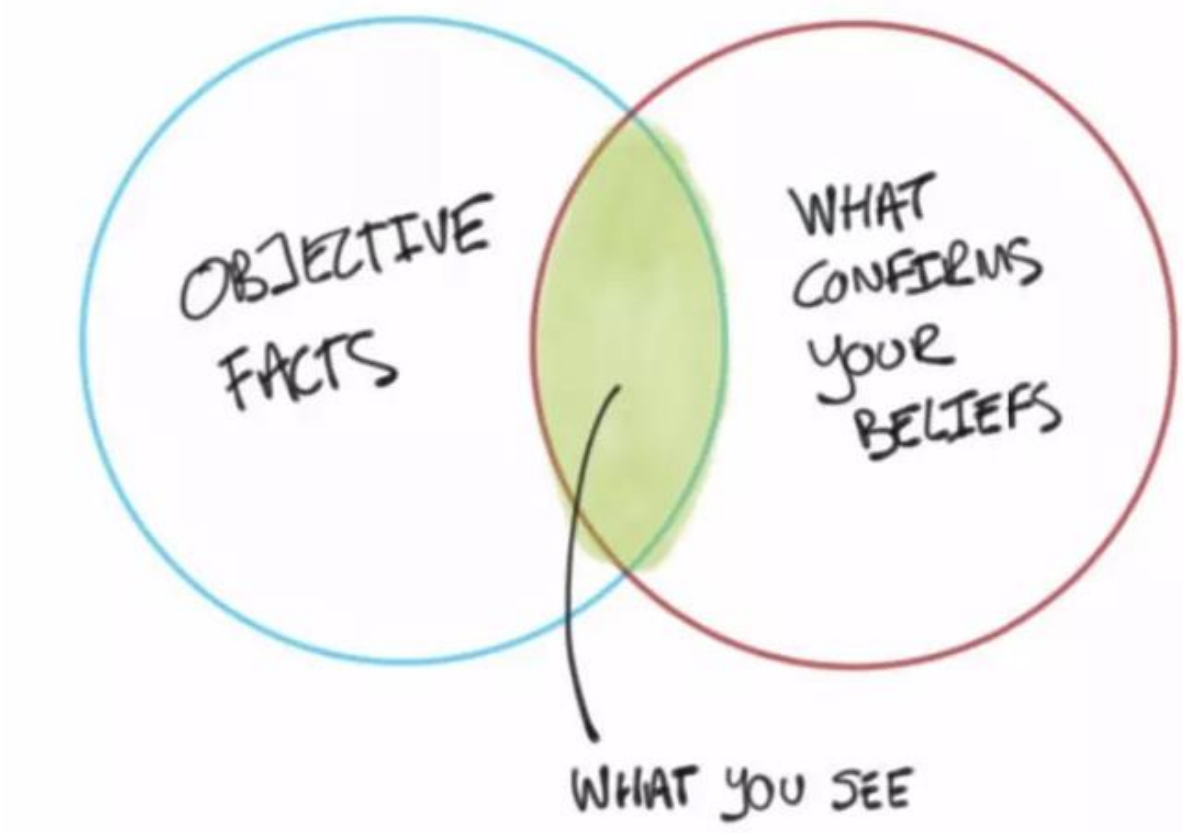
```
[Fact]
| 0 references | 0 changes | 0 authors, 0 changes
public void WhenIAdd1and2IExpect3()
{
    var result = Calculator.Add(1, 2);
    Assert.Equal(3, result);
}
```



Let’s write some tests…

ORDINA
a Sopra steria company

Example based testing



ORDINA
a Sopra steria company

Property Based Testing

Property Based Testing

Property based testing allows you to:

- Specify what values to generate
- Assert on **properties** that are true regardless of the exact value

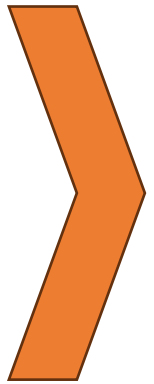
What is a ‘property’?

Property ≈ Requirement

What is a ‘property’?

You don’t know the exact input

So you don’t know what the exact output should be



But you can describe a relationships between input and output

What are the properties of the ‘add’ function?

ORDINA
a Sopra steria company

Properties?

```
[Fact]
| 0 references | 0 changes | 0 authors, 0 changes
public void WhenIAddTwoRandomNumbersTheResultShouldNotDependOnParameterOrder()
{
    for (int i = 0; i < 100; i++)
    {
        //Arrange
        var random = new Random();
        int input1 = random.Next();
        int input2 = random.Next();

        //Act
        var result1 = Calculator.Add(input1, input2);
        var result2 = Calculator.Add(input2, input1);

        //Assert
        Assert.Equal(result1, result2);
    }
}
```

ORDINA
a Sopra steria company

Properties

```
[Fact]
0 references | 0 changes | 0 authors, 0 changes
public void WhenIAdd1TwiceTheResultIsTheSameAsWhenAdding2()
{
    for (int i = 0; i < 100; i++)
    {
        //Arrange
        var random = new Random();
        int input = random.Next();

        //Act
        var result1 = Calculator.Add(Calculator.Add(input,1),1);
        var result2 = Calculator.Add(input, 2);

        //Assert
        Assert.Equal(result1, result2);
    }
}
```

Properties

```
[Fact]
| 0 references | 0 changes | 0 authors, 0 changes
public void WhenIAddZeroTheInputIsNotChanged()
{
    for (int i = 0; i < 100; i++)
    {
        //Arrange
        var random = new Random();
        int input = random.Next();

        //Act
        var result1 = Calculator.Add(input,0);
        var result2 = input;

        //Assert
        Assert.Equal(result1, result2);
    }
}
```

Properties

When I add 2 random numbers, the result should not depend on parameter order

Properties [edit]

Commutativity [edit]

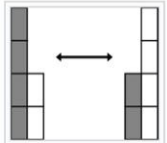
Addition is commutative, meaning that one can change the order of the terms in a sum, but still get the same result. Symbolically, if a and b are any two numbers, then

$$a + b = b + a.$$

The fact that addition is commutative is known as the "commutative law of addition" or "commutative property of addition". Some other binary operations are commutative, such as multiplication, but many others are not, such as subtraction and division.

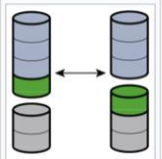
jump" that has a distance of 2 followed by another that is long as 4, is the same as a translation by 6.

A number-line visualization of the unary addition 2 + 4 = 6. A translation by 4 is equivalent to four translations by 1.



4 + 2 = 2 + 4 with blocks

Associativity [edit]



2 + (1 + 3) = (2 + 1) + 3 with segmented rods

Addition is associative, which means that when three or more numbers are added together, the order of operations does not change the result.

As an example, should the expression $a + b + c$ be defined to mean $(a + b) + c$ or $a + (b + c)$? Given that addition is associative, the choice of definition is irrelevant. For any three numbers a, b, and c, it is true that $(a + b) + c = a + (b + c)$. For example, $(1 + 2) + 3 = 3 + 3 = 6 = 1 + 5 = 1 + (2 + 3)$.

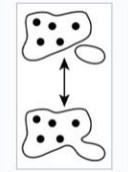
When addition is used together with other operations, the order of operations becomes important. In the standard order of operations, addition is a lower priority than exponentiation, nth roots, multiplication and division, but is given equal priority to subtraction.[21]

Identity element [edit]

Adding zero to any number, does not change the number; this means that zero is the identity element for addition, and is also known as the additive identity. In symbols, for every a, one has

$$a + 0 = 0 + a = a.$$

This law was first identified in Brahmagupta's *Brahmasphutasiddhanta* in 628 AD, although he wrote it as three separate laws, depending on whether a is negative, positive, or zero itself, and he used words rather than algebraic symbols. Later Indian mathematicians refined the concept; around the year 830, Mahavira wrote, "zero becomes the same as what is added to it", corresponding to the unary statement $0 + a = a$. In the 12th century, Bhaskara wrote, "In the addition of cipher, or subtraction of it, the quantity, positive or negative, remains the same", corresponding to the unary statement $a + 0 = a$.[22]



5 + 0 = 5 with bags of dots

ORDINA
a Sopra Steria company

Properties

When I add 1 twice, the result is the same as when adding 2

Properties [edit]

"jump" that has a distance of 2 followed by another that is long as 4, is the same as a translation by 6.

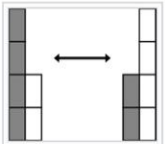
Commutativity [edit]

Addition is commutative, meaning that one can change the order of the terms in a sum, but still get the same result. Symbolically, if a and b are any two numbers, then

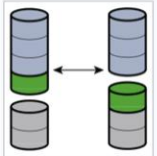
$$a + b = b + a.$$

The fact that addition is commutative is known as the "commutative law of addition" or "commutative property of addition". Some other binary operations are commutative, such as multiplication, but many others are not, such as subtraction and division.

A number-line visualization of the unary addition 2 + 4 = 6. A translation by 4 is equivalent to four translations by 1.


4 + 2 = 2 + 4 with blocks

Associativity [edit]


2 + (1 + 3) = (2 + 1) + 3 with segmented rods

Addition is associative, which means that when three or more numbers are added together, the order of operations does not change the result.

As an example, should the expression $a + b + c$ be defined to mean $(a + b) + c$ or $a + (b + c)$? Given that addition is associative, the choice of definition is irrelevant. For any three numbers a, b, and c, it is true that $(a + b) + c = a + (b + c)$. For example, $(1 + 2) + 3 = 3 + 3 = 6 = 1 + 5 = 1 + (2 + 3)$.

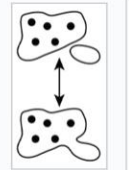
When addition is used together with other operations, the order of operations becomes important. In the standard order of operations, addition is a lower priority than exponentiation, nth roots, multiplication and division, but is given equal priority to subtraction.[21]

Identity element [edit]

Adding zero to any number, does not change the number; this means that zero is the identity element for addition, and is also known as the additive identity. In symbols, for every a, one has

$$a + 0 = 0 + a = a.$$

This law was first identified in Brahmagupta's *Brahmasphutasiddhanta* in 628 AD, although he wrote it as three separate laws, depending on whether a is negative, positive, or zero itself, and he used words rather than algebraic symbols. Later Indian mathematicians refined the concept; around the year 830, Mahavira wrote, "zero becomes the same as what is added to it", corresponding to the unary statement $0 + a = a$. In the 12th century, Bhaskara wrote, "In the addition of cipher, or subtraction of it, the quantity, positive or negative, remains the same", corresponding to the unary statement $a + 0 = a$.[22]


5 + 0 = 5 with bags of dots

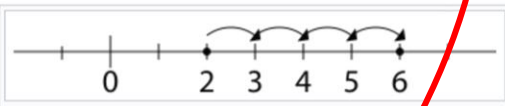
ORDINA
a Sopra Steria company

Properties

When I add 0, the input is not changed

Properties [edit]

"jump" that has a distance of 2 followed by another that is long as 4, is the same as a translation by 6.



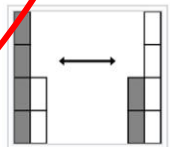
A number-line visualization of the unary addition 2 + 4 = 6. A translation by 4 is equivalent to four translations by 1.

Commutativity [edit]

Addition is commutative, meaning that one can change the order of the terms in a sum, but still get the same result. Symbolically, if *a* and *b* are any two numbers, then

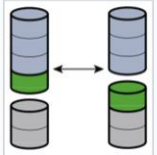
$$a + b = b + a.$$

The fact that addition is commutative is known as the "commutative law of addition" or "commutative property of addition". Some other binary operations are commutative, such as multiplication, but many others are not, such as subtraction and division.



4 + 2 = 2 + 4 with blocks

Associativity [edit]



2 + (1 + 3) = (2 + 1) + 3 with segmented rods

Addition is associative, which means that when three or more numbers are added together, the order of operations does not change the result.

As an example, should the expression $a + b + c$ be defined to mean $(a + b) + c$ or $a + (b + c)$? Given that addition is associative, the choice of definition is irrelevant. For any three numbers a, b, and c, it is true that $(a + b) + c = a + (b + c)$. For example, $(1 + 2) + 3 = 3 + 3 = 6 = 1 + 5 = 1 + (2 + 3)$.

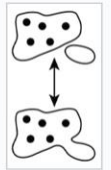
When addition is used together with other operations, the order of operations becomes important. In the standard order of operations, addition is a lower priority than exponentiation, nth roots, multiplication and division, but is given equal priority to subtraction.[21]

Identity element [edit]

Adding zero to any number, does not change the number; this means that zero is the identity element for addition, and is also known as the additive identity. In symbols, for every a, one has

$$a + 0 = 0 + a = a.$$

This law was first identified in Brahmagupta's *Brahmasphutasiddhanta* in 628 AD, although he wrote it as three separate laws, depending on whether *a* is negative, positive, or zero itself, and he used words rather than algebraic symbols. Later Indian mathematicians refined the concept; around the year 830, Mahavira wrote, "zero becomes the same as what is added to it", corresponding to the unary statement $0 + a = a$. In the 12th century, Bhaskara wrote, "In the addition of cipher, or subtraction of it, the quantity, positive or negative, remains the same", corresponding to the unary statement $a + 0 = a$.[22]



5 + 0 = 5 with bags of dots

ORDINA
a Sopra Steria company

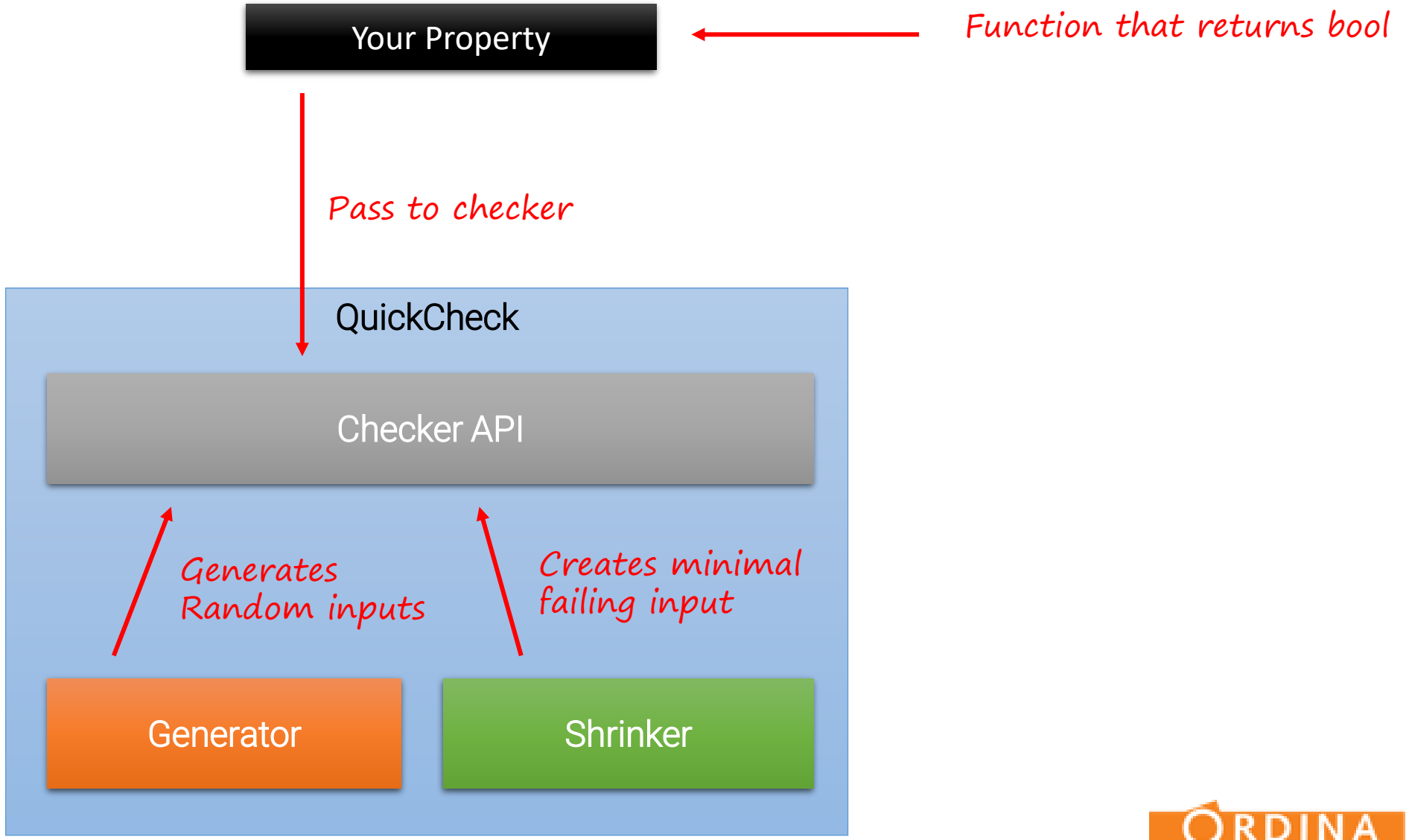
Introducing FSCheck

QuickCheck

FSCheck

Don't forget to mention CSCheck!

ORDINA
a Sopra steria company

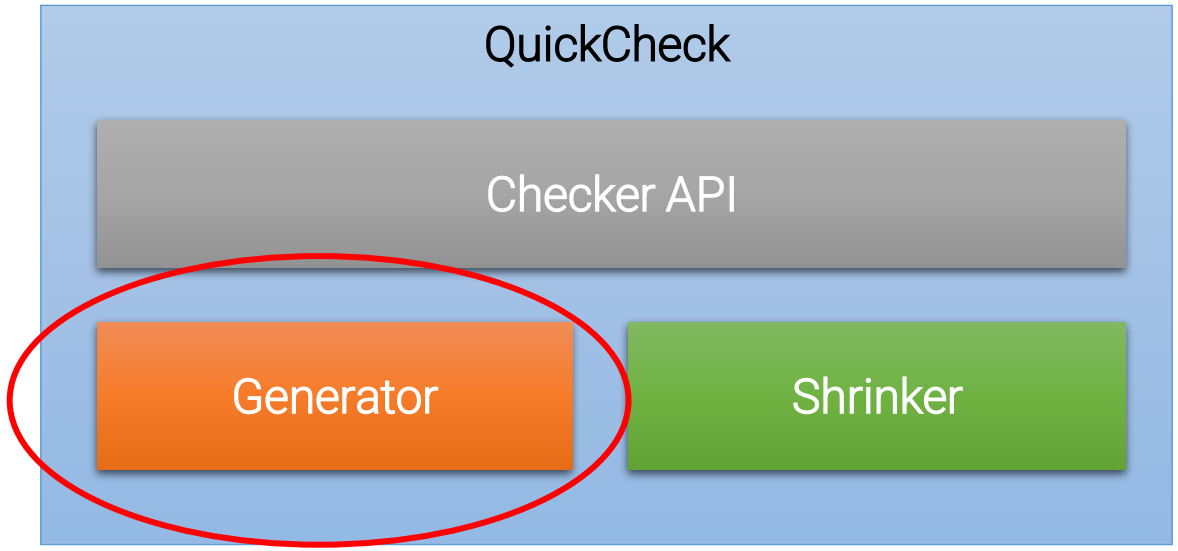


Time to integrate FSCheck in our tests…

ORDINA
a Sopra steria company

Generators

Generators: Making random input



Generate types

“int” generator	0, 1, 3, -2, …
“string” generator	“”, “eiX$a[”, “U%0Ika&r”, …
“bool” generator	true, false, false, true, …
“Tuple<int,int>” generator	(0,0), (1,0), (2,0), (-1,1), (-1,2), …
“Color” generator	Red, Green, Blue, Green, …

How it works in practice

```
[Property(Verbose =true)]
| 0 references | Wullems Bart, 1 day ago | 1 author, 1 change
public Property Adding_Two_Numbers_Doesnt_Depend_On_Parameter_Order_2(Tuple<int,int> values)
{
    return (Add(values.Item1, values.Item2) == Add(values.Item1, values.Item2)).ToProperty();
}
```

(1) Checker detects that the input is a tuple of (int,int)

(2) Appropriate generator is created

(3) Valid values are generated

(0,0) (1,0) (2,0) (-1, 1) (100, -99) ...

(4) Values are passed to the property

ORDINA
a Sopra steria company

Let’s look at some generators…

ORDINA
a Sopra steria company

Debugging a disproven property

Debugging a disproven property

```
PropertyBasedTesting.Tests.BuiltInGeneratorsTests.Generator2
  Source: BuiltInGeneratorsTests.cs line 41
  Duration: 202 ms

  Message:
    FsCheck.Xunit.PropertyFailedException :
    Falsifiable, after 1 test (1 shrink) (StdGen (2052518649,297245404)):
    Original:
    NonEmptyArray [|null; ""|]
    Shrunk:
    NonEmptyArray [|""|]
```

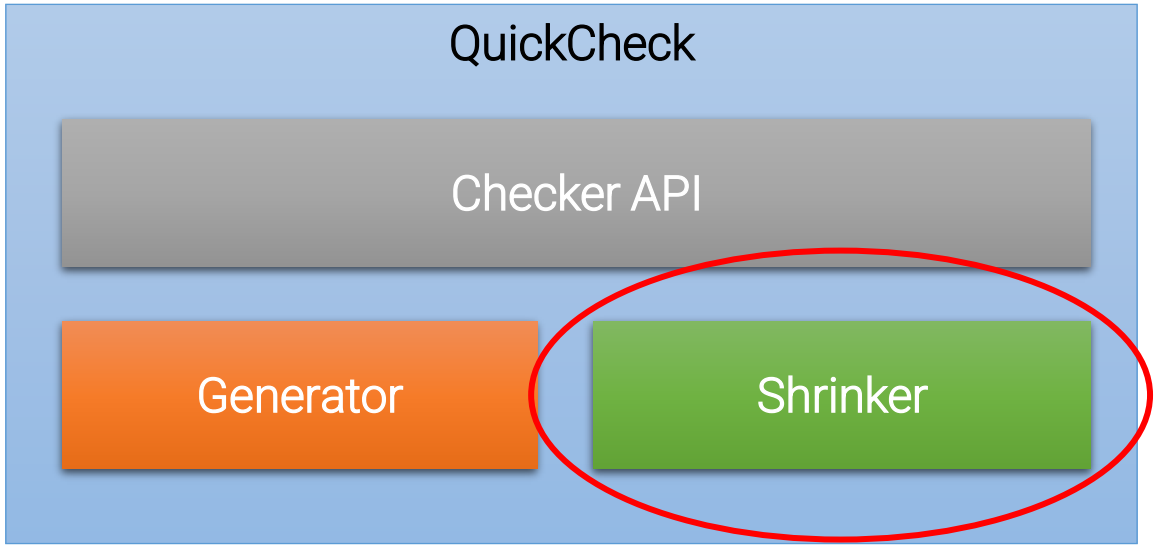
Locking input

Shrinking

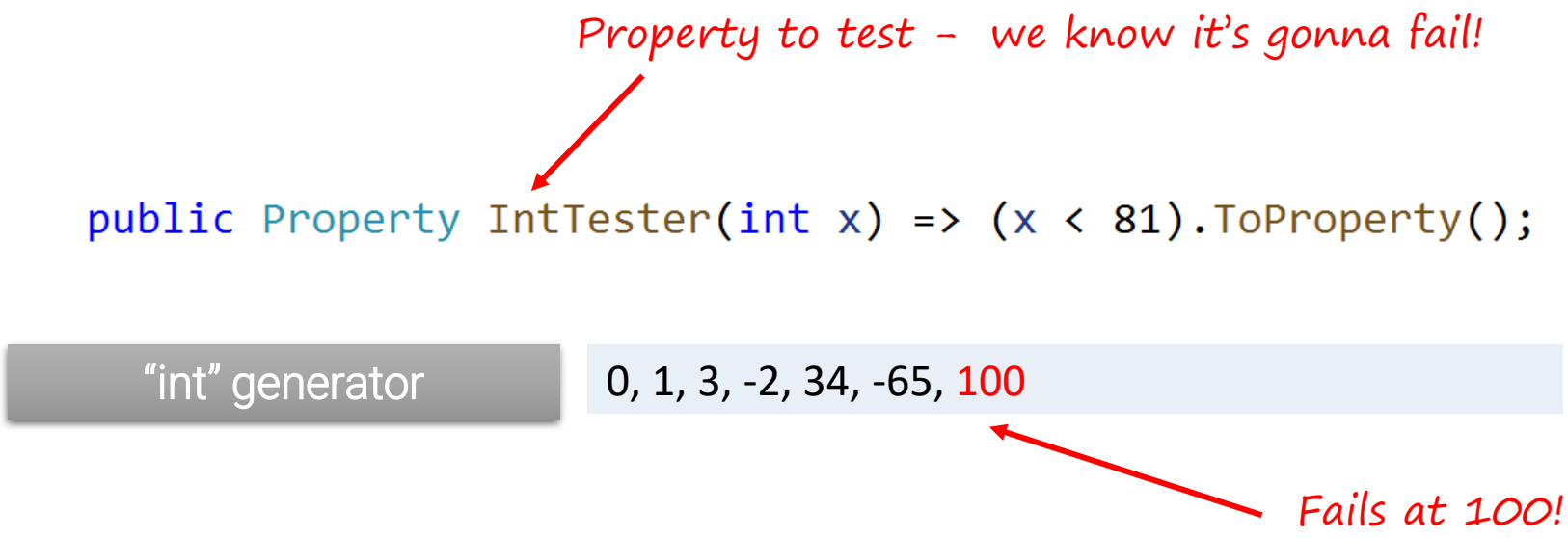
Locking our input…

ORDINA
a Sopra steria company

Shrinking



How shrinking works



How shrinking works

```
public Property IntTester(int x) => (x < 81).ToProperty();
```

Shrink list for 100

0, 50, 75, 88, 94, 97, 99

Fails at 88!

Generate a new sequence up to 100

How shrinking works

```
public Property IntTester(int x) => (x < 81).ToProperty();
```

Shrink list for 88

0, 44, 66, 77, 83, 86, 87

Fails at 83!

Generate a new sequence up to 88

How shrinking works

```
public Property IntTester(int x) => (x < 81).ToProperty();
```

Shrink list for 83

0, 42, 63, 73, 78, 81, 82

Fails at 81!

Generate a new sequence up to 83

How shrinking works

```
public Property IntTester(int x) => (x < 81).ToProperty();
```

Shrink list for 81

0, 41, 61, 71, 76, 79, 80

All pass!

Generate a new sequence up to 81

Shrink has determined that 81 is the smallest failing input

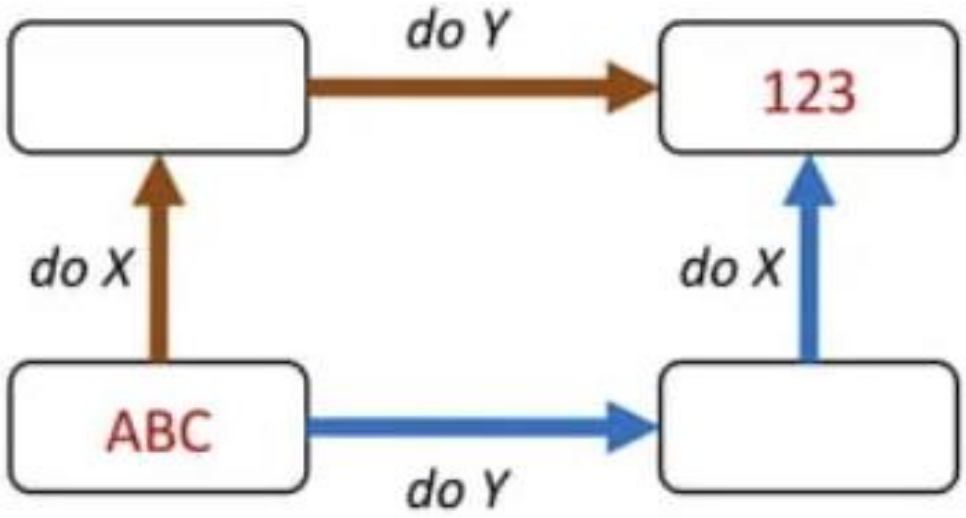
ORDINA
a Sopra steria company

Shrinking time…

ORDINA
a Sopra steria company

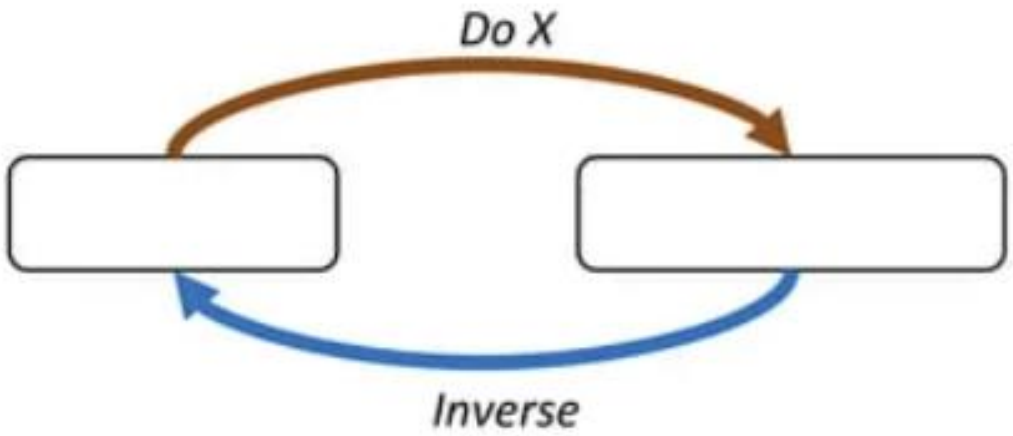
Identifying properties

Different paths, same destination



- Examples
  - Commutivity
  - Associativity
  - Map

There and back again

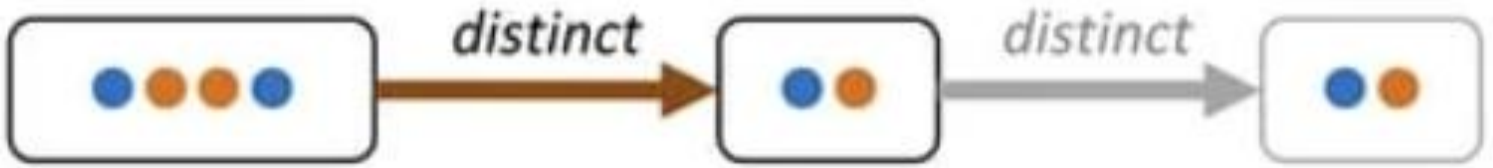


- Examples
  - Serialization/Deserialization
  - Addition/Subtraction
  - Write/Read
  - Set/Get

Some things never change

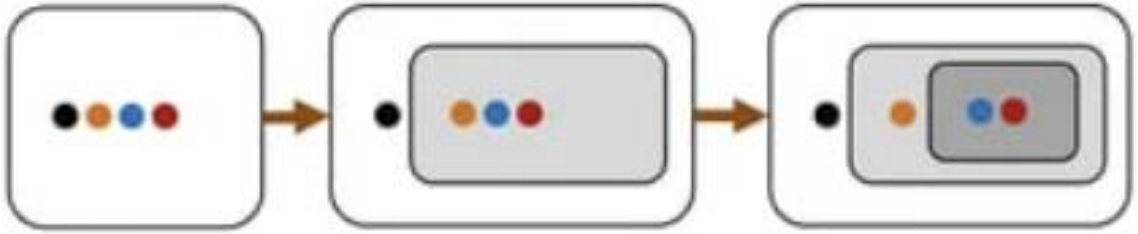
- Examples
  - Size of a collection
  - Contents of a collection
  - Balanced trees

The more things change, the more they stay the same



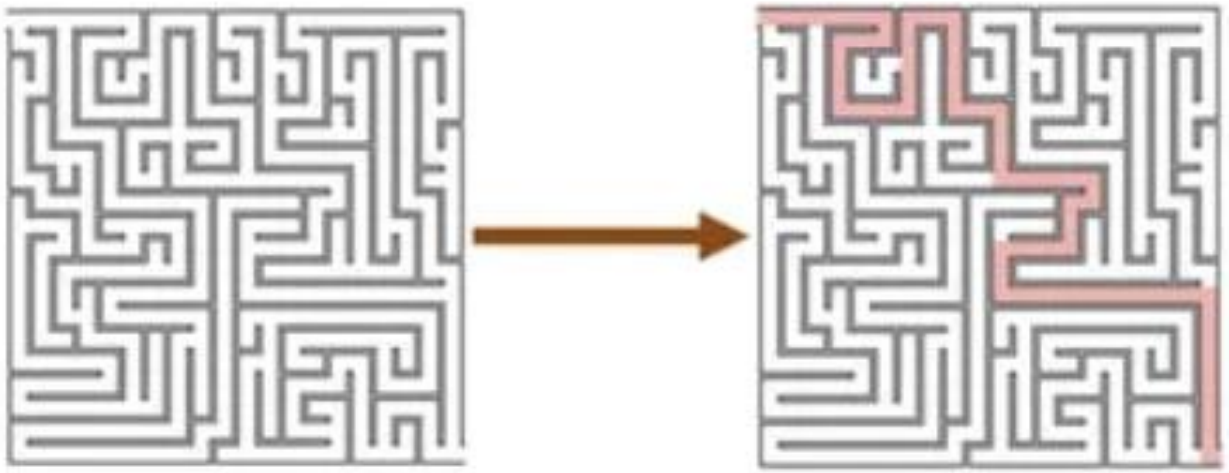
- Idempotence
  - Sort
  - Filter
  - Event processing
  - Required for distributed designs

Solve a smaller problem first



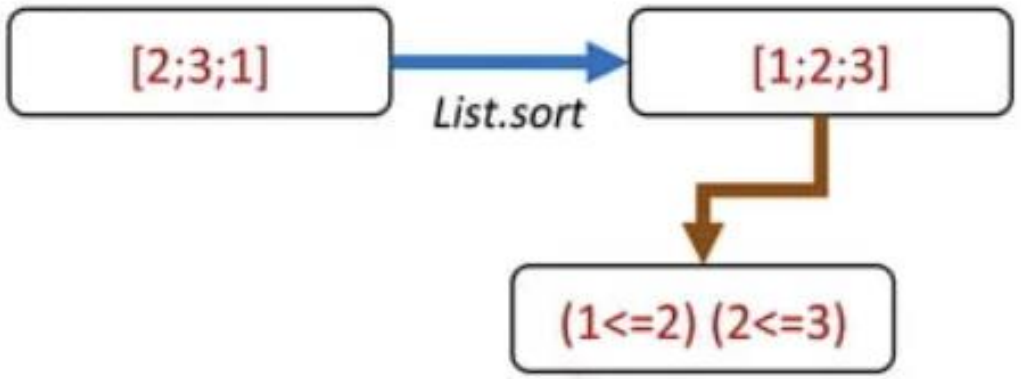
- Divide and conquer algorithms(e.g. quicksort)
- Structural induction (recursive data structures)

Hard to prove, easy to verify



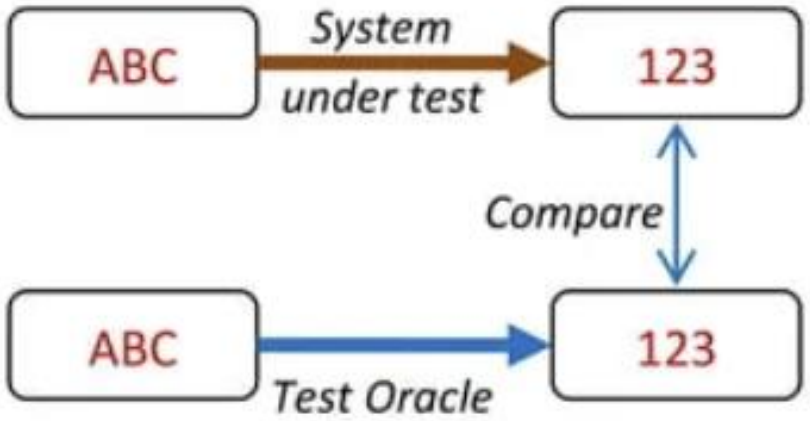
- Algorithms

Hard to prove, easy to verify



To verify the sort,
Check that each pair is ordered

The test oracle



- Compare optimized with slow brute-force version
- Compare parallel with single thread version

Conclusion

Conclusion

- PBT’s are more general
  - One property-based tests can replace many example-based tests
- PBT’s can reveal overlooked edge cases
  - Nulls, negative numbers, weird strings, etc…
- PBT’s require deep understanding of requirements
  - Property-based tests force you to think ☺
- Example-based tests are still helpful though!
  - Easier to understand for newcomers

Go write some tests!

ORDINA
a Sopra Steria company