Property based testing in Elixir

Let’s start with a disclaimer

Property based testing ~~in Elixir~~

Tonći Galić / @Tuxified
@tuxified@mastodon.social

A little intro

- Tonći Galić / @Tuxified
- Live in 🇳🇱 (near Amsterdam)
- Family 🐶 🐱🐱👨‍🍼👨‍🍼👨‍🍼
- Like (computer) languages 🤓
- Like doing sports (🏃, 🚣 ..)

Sometimes feel 👨‍🦳

TL;DL:

Generate

All kinds of cases, often edge cases

Shrinking

Once an issue is found, search for minimal input

Complexity

Combination of features (hence tests) possible

Rewires 🧠

Makes you think harder, not more

Table of contents

01

Unit testing

A.k.a example testing

Why talk about Unit testing?

Good

Testing is good as it gives us confidence, prevents disasters and helps drive design

Boring

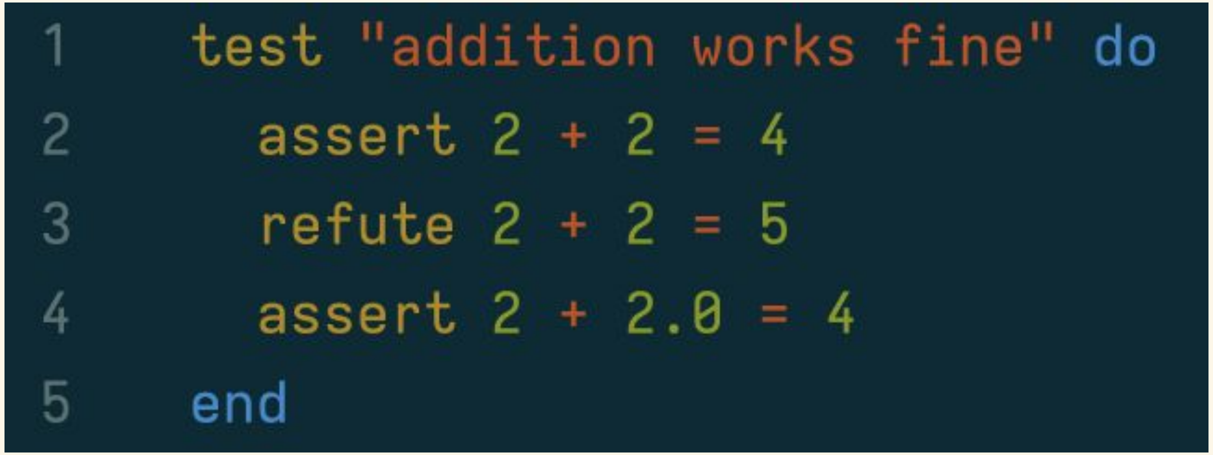
Coming up with good examples for our tests is boring and tedious

Hard

How many tests should we write? How will our test suite grow as we add features

Write tests for the plus operator (Kernel.+/2)

Write tests for the plus operator (Kernel.+/2)



```elixir
test "addition works fine" do
  assert 2 + 2 = 4
  refute 2 + 2 = 5
  assert 2 + 2.0 = 4
end
```

Testing can be boring

How many tests?

N features	O(n) tests	No problem

How many tests?

N features	O(n) tests	No problem
Pairs of N features	O(n^2)	This is a step up, but doable

How many tests?

N features	O(n) tests	No problem
Pairs of N features	$O(n^2)$	This is a step up, but doable
Triples of N features	$O(n^3)$	Starting to get out of hand

How many tests?

N features	O(n) tests	No problem
Pairs of N features	$O(n^2)$	This is a step up, but doable
Triples of N features	$O(n^3)$	Starting to get out of hand
M x N features	💥	🤯

Testing can be hard

How to fix?

02

Property based testing

"Instead of writing examples, we define properties and let the computer come up with cases"

– Folks at Quviq

“Instead of writing examples, we define properties and let the computer come up with cases”
… and some more.

– Folks at Quviq

Write tests for string reversal

```elixir
test "String.reverse reverses a string" do
  assert String.reverse("FOSDEM") == "MEDSOF"
  refute String.reverse("FOSDEM") == "FOSDEM"
  assert String.reverse("") == ""
end
```

Does this spark confidence?

»»» Write properties for string reversal

Write properties for string reversal

```elixir
property "reversing a string twice returns original" do
  check all shirt <- string(:ascii) do
    assert String.reverse(String.reverse(shirt)) == shirt
  end
end
```

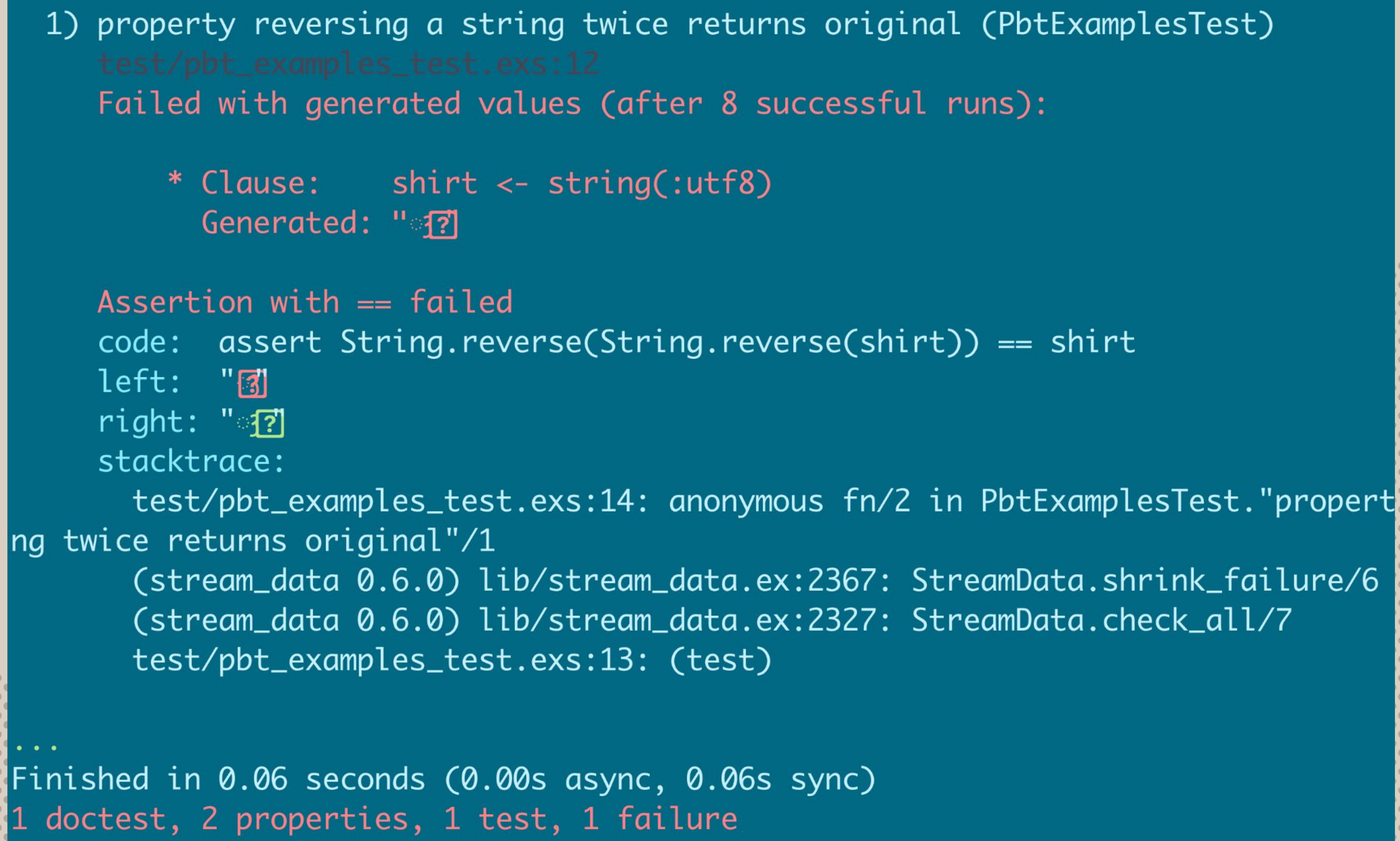
»»» What are other properties of Any.reverse/1 ?

- first item becomes last item (for non-empty list)
- last item becomes first item (for non-empty list)
- palindromes stay the same after reversal
- amount of items stay consistent
- ....

Write properties for string reversal

```elixir
property "reversing a string twice returns original" do
  check all shirt <- string(:utf8) do
    assert String.reverse(String.reverse(shirt)) == shirt
  end
end
```

Run tests



```
  1) property reversing a string twice returns original (PbtExamplesTest)
     test/pbt_examples_test.exs:12
     Failed with generated values (after 8 successful runs):

         * Clause:    shirt <- string(:utf8)
           Generated: "◌�

     Assertion with == failed
     code:  assert String.reverse(String.reverse(shirt)) == shirt
     left:  "�"
     right: "◌�"
     stacktrace:
       test/pbt_examples_test.exs:14: anonymous fn/2 in PbtExamplesTest."propert
ng twice returns original"/1
       (stream_data 0.6.0) lib/stream_data.ex:2367: StreamData.shrink_failure/6
       (stream_data 0.6.0) lib/stream_data.ex:2327: StreamData.check_all/7
       test/pbt_examples_test.exs:13: (test)


...
Finished in 0.06 seconds (0.00s async, 0.06s sync)
1 doctest, 2 properties, 1 test, 1 failure
```

Edge case found !!

Case shrinked

03

Examples*

*Use cases

Volvo's AUTOSAR

3000 pages

specification

1_000_000

Lines of vendor code tested

200

Issues found

Klarna's heisenbug

6 weeks

No result

1GB

Files

< 3 day

To find issue with QuickCheck

What are other good occasions to use PBT

- symmetrical functions (serialize <-> deserialize)
- mathematical proof
- compare systems
- testing concurrency

04

Conclusion

Conclusion

Generate

All kinds of cases, often edge cases

Shrinking

Once an issue is found, search for minimal input

Complexity

Combination of features (hence tests) possible

Makes you think harder, not more

Thanks!

Does anyone have any questions?
Tonći Galić / @Tuxified
@tuxified@mastodon.social

Additional resources

BEAM: PropEr, QuickCheck, StreamData, Triq, etc etc

Haskell: QuickCheck (by Quviq)

Python: https://hypothesis.works/

Book: https://propertesting.com/

Talk: John Hughes - Keynote: How to specify it! https://www.youtube.com/watch?v=G0NUOst-53U