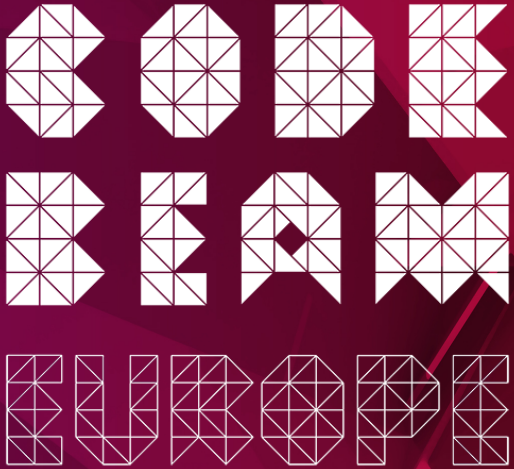


STOCKHOLM

HYBRID CONFERENCE

Improve your tests with Makina

Luis Eduardo Bueso de Barrio

May 20 | 2022

#CodeBEAM

The problem

files	blank	comment	code
4	760	383	4513

files	blank	comment	code
18	500	408	1692

Introduction to PBT

Writing unit-tests is hard and time-consuming:

```
reverse([]) == []
reverse([1]) == [1]
reverse([1 ,2]) == [2 ,1]
...
```

Property-Based Testing (PBT) philosophy:

Don't write tests, generate them.

A test execution in PBT consists of:

1. Data generation.
2. Property checking.
3. An shrinking strategy (if the property doesn't hold).

A property:

```
forall list <- list() do
  list == reverse(reverse(list))
end
```

In each test:

1. `list()` generates a random list:

   `[8 ,10 ,6] ...`

2. Checks the property:

   `[8, 10, 6] == reverse(reverse([8, 10, 6]))`

3. If the property doesn't hold returns a counter-example.

Testing *stateful* programs

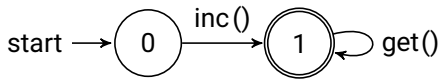
Imagine a simple counter:

Command	Returns
`start/1`	`:ok | :error`
`stop/0`	`:ok | :error`
`inc/0`	`:ok`
`get/0`	`integer()`

Unit test:

```
:ok = start(0)
:ok = inc()
1   = get()
:ok = stop()
```

This test can be represented:



To successfully test this program we need to:

- Generate sequences of commands.
- An internal state to track the changes in the program.
- A way to interact with the program under test.

PBT of *stateful* programs

Basic property of *stateful* programs:

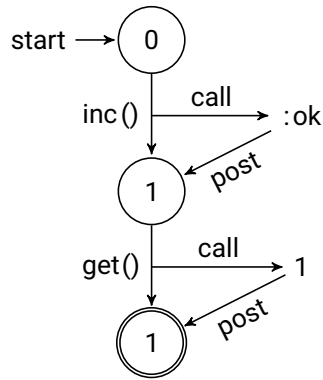
```
forall cmds <- commands(Counter) do
  :ok == run_commands(Counter, cmds)
end
```

where:

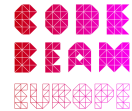
- `commands/1` generates sequences commands:

  ```
  [start(0), inc(), get(), stop()]
  ...
  ```

- `run_commands` executes the generated sequence.



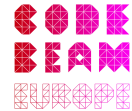
Introduction to Makina



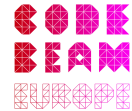
`Makina` is a DSL to write PBT state machines.

- Fully compatible with `Erlang QuickCheck` and `PropEr`.
-

Ethereum Blockchain

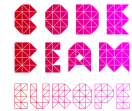


Mining blocks



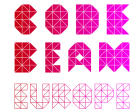
```elixir
defmodule Blocks do
  use Makina, implemented_by: Etherex

  state height: 0

  invariants non_neg_height: height >= 0

  command block_number() do
    post {:ok, height} == result
  end

  command mine() do
    call Etherex.Time.mine()
    next height: height + 1
  end
end
```

Consulting accounts



Generating transactions

An abstract model for contracts



A basic model to test a contract