

STOCKHOLM

HYBRID CONFERENCE

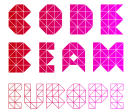
Improve your tests
with Makina

Luis Eduardo Bueso de Barrio

May 20 | 2022

#CodeBEAM

The problem



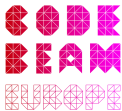
Before:

files	blank	comment	code
4	760	383	4513

After:

files	blank	comment	code
18	500	408	1692

Testing *stateful* programs



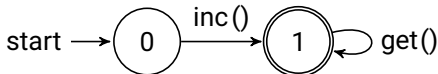
Imagine a simple counter:

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()
```

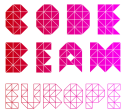
This test can be represented:



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

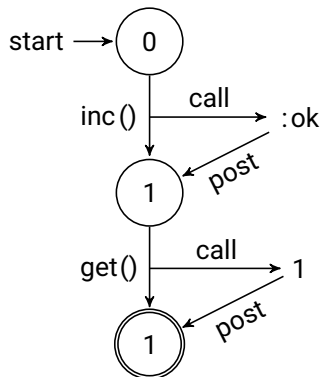


Basic property of *stateful* programs:

```
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.



PBT state machines



A PBT state machine is a program that specifies the behaviour of the implementation.

Behaviour:

- precondition.
- command generation.
- next state.
- postcondition

Executed in two phases:

- symbolic test generation.
- dynamic test execution.

Proven effectiveness.

Very slow adoption.

Why?

Problems:

- Cryptic errors.
- Usually buggy.
- Code reuse is very hard.
- Hard to maintain.

Makina is a DSL to write PBT state machines.

- Fully compatible with Erlang QuickCheck and PropEr.
- Better error messages.
- Better error detection in models.
- Improve code reuse.

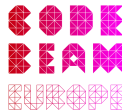
A Makina model contains:

- state
- command
- invariants

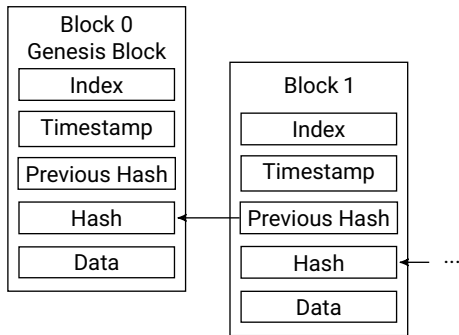
Provides code reuse mechanisms:

- imports
- extends
- composition

Ethereum Blockchain



A blockchain is a distributed ledger that enables peer-to-peer transactions.



Why Ethereum?

- Big community.
- Multiple implementations.

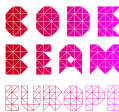
To interact we will use etherex:

<https://gitlab.com/babel-upm/blockchain/etherex>

The properties to test:

1. Mining blocks.
2. Account access.
3. Transactions between accounts.

Mining blocks



The API:

Command	Returns
mine/0	:ok
block_number/0	integer()

1. create module.
2. import Makina.
3. define state.
4. define invariants.
5. define commands.

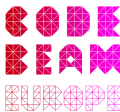
```
defmodule Blocks do
  use Makina

  state height: 0

  invariants non_neg_height: height > 0

  command block_number() do
    pre true
    args []
    call Etherex.block_number
    next []
    post height == result
  end
end
```

Mining blocks



The API:

Command	Returns
mine/0	:ok
block_number/0	integer()

1. create module.
2. import Makina.
3. define state.
4. define invariants.
5. define commands.

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

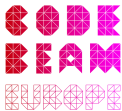
  state height: 0

  invariants non_neg_height: height > 0

  command block_number() do
    post height == result
  end

  command mine() do
    next height: height + 1
  end
end
```

Running the test



```
$ mix test
```

Failed! After 1 tests.

Postcondition crashed:

** invariant "non_neg_height" check failed

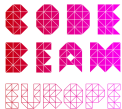
Shrinking x.(1 times)

```
[  
  Blocks.block_number/0  
]
```

Last state: %{height: 0}

```
defmodule Blocks do  
  use Makina, implemented_by: Etherex  
  
  state height: 0  
  
  invariants non_neg_height: height > 0  
  
  command block_number() do  
    post height == result  
  end  
  
  command mine() do  
    next height: height + 1  
  end  
end
```

Fixing the model



```
$ mix test
```

Failed! After 1 tests.

Postcondition crashed:

```
** invariant "non_neg_height" check failed
```

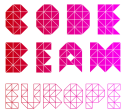
Shrinking x.(1 times)

```
[  
  Blocks.block_number/0  
]
```

Last state: %{height: 0}

```
defmodule Blocks do  
  use Makina, implemented_by: Etherex  
  
  state height: 0  
  
  invariants non_neg_height: height >= 0  
  
  command block_number() do  
    post height == result  
  end  
  
  command mine() do  
    next height: height + 1  
  end  
end
```

Running the test



```
$ mix test
```

```
.....  
.....
```

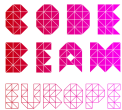
```
OK, passed 100 tests
```

```
51.5 Blocks.mine/0
```

```
48.5 Blocks.block_number/0
```

```
defmodule Blocks do  
  use Makina, implemented_by: Etherex  
  
  state height: 0  
  
  invariants non_neg_height: height >= 0  
  
  command block_number() do  
    post height == result  
  end  
  
  command mine() do  
    next height: height + 1  
  end  
end
```

Adding type information



```
$ mix gradient
```

```
$
```

Something changes in Etherex...

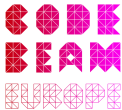
```
$ mix gradient
```

The function call `Etherex.block_number()` on line 8 is expected to have type `integer()` but it has type `{:ok, quantity()} | {:error, error()}`

```
$
```

```
1 defmodule Blocks do
2   use Makina, implemented_by: Etherex
3
4   state height: 0 :: integer()
5
6   invariants non_neg_height: height >= 0
7
8   command block_number() :: integer() do
9     post height == result
10  end
11
12  command mine() :: :ok do
13    next height: height + 1
14  end
15 end
```

Adding documentation



```
iex> h Blocks
```

```
Contains a Makina model called Blocks.
```

```
Checks blocks are mined correctly.
```

```
## Commands
```

- mine stored at Blocks.Command.Mine
- block_number stored at Blocks.Command.BlockNumber

```
## State attributes
```

- height

```
## Invariants
```

- non_neg_height

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

  @moduledoc """
  Checks blocks are mined correctly.
  """

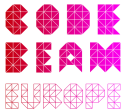
  state height: 0 :: integer()

  invariants non_neg_height: height >= 0

  command block_number() :: integer() do
    @moduledoc "Gets the block number."
    post {:ok, height} == result
  end

  command mine() :: :ok do
    @moduledoc "Mines a new block."
    next height: height + 1
  end
end
```

Adding documentation



```
iex> h Blocks.Command.Mine
```

This module contains the functions necessary to generate and execute the command mine.

Mines a new block.

Definitions

- next
- call
- weight
- post
- args
- pre

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

  @moduledoc """
  Checks blocks are mined correctly.
  """

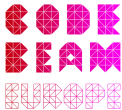
  state height: 0 :: integer()

  invariants non_neg_height: height >= 0

  command block_number() :: integer() do
    @moduledoc "Gets the block number."
    post {:ok, height} == result
  end

  command mine() :: :ok do
    @moduledoc "Mines a new block."
    next height: height + 1
  end
end
```


Adding documentation



```
iex> h Blocks.Command.Mine.post
```

This definition contains a predicate that should be true after the execution of mine

Available variables

State

- state
- height

Arguments

- arguments

Result

- result

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

  @moduledoc """
  Checks blocks are mined correctly.
  """

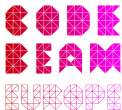
  state height: 0 :: integer()

  invariants non_neg_height: height >= 0

  command block_number() :: integer() do
    @moduledoc "Gets the block number."
    post {:ok, height} == result
  end

  command mine() :: :ok do
    @moduledoc "Mines a new block."
    next height: height + 1
  end
end
```

Account access



The API:

Command	Returns
balance/1	integer()

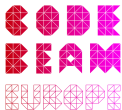
1. create module.
2. import Makina.
3. define state.
4. define commands.

```
defmodule Accounts do
  use Makina, implemented_by: Etherex

  state accounts: Etherex.accounts() :: [address()],
        balances: Etherex.balances() :: %{address() => integer()}

  command balance(account :: address()) :: integer() do
    pre accounts != []
    post balances[account] == result
  end
end
```

Running the test

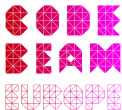


```
$ mix test
```

```
** (Makina.Error) argument  
  'account' missing in command  
  get_balance
```

```
defmodule Accounts do  
  use Makina, implemented_by: Etherex  
  
  state accounts: Etherex.accounts() :: [address()],  
        balances: Etherex.balances() :: %{address() => integer()}  
  
  command balance(account :: address()) :: integer() do  
    pre accounts != []  
    post balances[account] == result  
  end  
end
```

Fixing the model

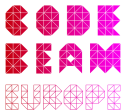


```
$ mix test
```

```
** (Makina.Error) argument  
  'account' missing in command  
  get_balance
```

```
defmodule Accounts do  
  use Makina, implemented_by: Etherex  
  
  state accounts: Etherex.accounts() :: [address()],  
        balances: Etherex.balances() :: %{address() => integer()}  
  
  command balance(account :: address()) :: integer() do  
    args account: oneof(accounts)  
    pre accounts != []  
    post balances[account] == result  
  end  
end
```

Running the test



```
$ mix test
```

```
.....  
.....  
.....  
.....
```

```
OK, passed 100 tests
```

```
'100.0 Accounts.get_balance/1
```

```
defmodule Accounts do  
  use Makina, implemented_by: Etherex  
  
  state accounts: Etherex.accounts() :: [address()],  
        balances: Etherex.balances() :: %{address() => integer()}  
  
  command balance(account :: address()) :: integer() do  
    args account: oneof(accounts)  
    pre accounts != []  
    post balances[account] == result  
  end  
end
```

Generating transactions



The API to generate and check transactions:

Command	Returns
mine/0	:ok
block_number/0	integer()
get_balance/1	integer()
transfer/3	hash()

We can compose Blocks and Accounts!

```
defmodule Transactions do
  use Makina,
    extends: [Blocks, Accounts],
    implemented_by: Etherex
end
```

Generates a model Transactions.Composed.

```
iex(1)> h Transactions.Composed
```

```
# Transactions.Composed
```

```
## Commands
```

- mine stored
- get_balance
- block_number

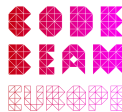
```
## State attributes
```

- height
- balances
- accounts

```
## Invariants
```

- non_neg_height

Generating transactions



Command	Returns
transfer/3	hash()

```
defmodule Transactions do
  use Makina,
    implemented_by: Etherex,
    extends: [Accounts, Blocks]

  command transfer(from, to, value) :: hash() do
    pre accounts != []
    args from: oneof(accounts),
          to: oneof(accounts),
          value: pos_integer()
    next balances: update(balances, from, to, value)
  end
end
```

Running the test



```
$ mix test
```

```
transfer("0xffcf8fdee72ac11",  
         "0x90f8bf6a479f320",  
         1)  
block_number()
```

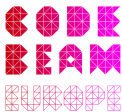
```
Postcondition failed.
```

```
block_number() -> {:ok, 1}
```

```
Last state: %{height: 0, ...}
```

```
defmodule Transactions do  
  use Makina,  
    implemented_by: Etherex,  
    extends: [Accounts, Blocks]  
  
  command transfer(from, to, value) :: hash() do  
    pre accounts != []  
    args from: oneof(accounts),  
          to: oneof(accounts),  
          value: pos_integer()  
    next balances: update(balances, from, to, value)  
  end  
end
```


Fixing the model



```
$ mix test
```

```
Postcondition failed.
```

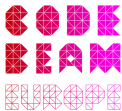
```
transfer("0xffcf8fdee72ac11",  
         "0x90f8bf6a479f320",  
         1)
```

```
block_number() -> {:ok, 1}
```

```
Last state: %{height: 0, ...}
```

```
defmodule Transactions do  
  use Makina,  
    implemented_by: Etherex,  
    extends: [Accounts, Blocks]  
  
  command transfer(from, to, value) :: hash() do  
    pre accounts != []  
    args from: oneof(accounts),  
          to: oneof(accounts),  
          value: pos_integer()  
    next height: height + 1,  
          balances: update(balances, from, to, value)  
  end  
end
```

Running the test



```
$ mix test
```

```
transfer("0x90f8bf6a479f320",
         "0x90f8bf6a479f320",
         1),
get_balance("0x90f8bf6a479f320")
```

```
Postcondition failed.
```

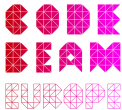
```
get_balance("0x90f8bf6a479f320")
-> {:ok, 979000}
```

```
Last state: %{
  balances: %{
    "0x90f8bf6a479f320" => 1000000
    .. }
  .. }
```

```
defmodule Transactions do
  use Makina,
    implemented_by: Etherex,
    extends: [Accounts, Blocks]

  command transfer(from, to, value) :: hash() do
    pre accounts != []
    args from: oneof(accounts),
          to: oneof(accounts),
          value: pos_integer()
    next height: height + 1,
          balances: update(balances, from, to, value)
  end
end
```

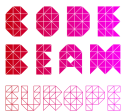
Fixing the model



```
defmodule Transactions do
  use Makina,
    implemented_by: Etherex,
    extends: [Accounts, Blocks]

  command transfer(from, to, value) :: hash() do
    pre accounts != []
    args from: oneof(accounts),
          to: oneof(accounts),
          value: pos_integer()
    next height: height + 1,
          balances: update(balances, from, to, value)
  end
end
```

Fixing the model



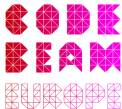
```
defmodule Transactions do
  use Makina,
    implemented_by: Etherex,
    extends: [Accounts, Blocks]

  state transactions: [] :: [symbolic(hash())]

  command transfer(from, to, value) :: hash() do
    pre accounts != []
    args from: oneof(accounts),
          to: oneof(accounts),
          value: pos_integer()
    next height: height + 1,
          transactions: [result | transactions],
          balances: update(balances, from, to, value)
                  |> symbolic(),
    end

  command get_balance() do
    pre transactions == []
  end
end
```

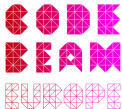
Fixing the model



```
defmodule Transactions.GasCost do
  use Makina, extends: Transactions

  command gas_cost(hash :: hash())
    :: {address(), quantity()} do
    pre transactions != []
    args hash: oneof(transactions)
    next do
      from = symbolic(elem(result, 0))
      gas = symbolic(elem(result, 1))
      [transactions: List.delete(transactions, hash),
       balances: update(balances, from, gas)
       |> symbolic()]
    end
  end
end
```

Running the test



```
$ mix test
```

```
.....
```

```
.....
```

```
OK, passed 100 tests
```

```
'25.5 Transactions.mine/0
```

```
'24.9 Transactions.block_number/0
```

```
'23.6 Transactions.transfer/3
```

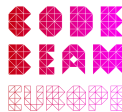
```
'14.3 Transactions.gas_cost/1
```

```
'11.8 Transactions.get_balance/1
```

```
defmodule Transactions.GasCost do
  use Makina, extends: Transactions

  command gas_cost(hash :: hash())
    :: {address(), quantity()} do
    pre transactions != []
    args hash: oneof(transactions)
    next do
      from = symbolic(elem(result, 0))
      gas = symbolic(elem(result, 1))
      [transactions: List.delete(transactions, hash),
       balances: update(balances, from, gas)
       |> symbolic()]
    end
  end
end
```

Results and conclusions



Before Makina:

files	blank	comment	code
4	760	383	4513

After Makina:

files	blank	comment	code
18	500	408	1692

Libraries:

- <https://gitlab.com/babel-upm/makina/makina>
- <https://gitlab.com/babel-upm/blockchain/etherex>

Slides and code:

- https://gitlab.com/babel-upm/makina/code_beam_2022