Property Testing with derived idempotents

Brian Zeligson

April 2020

Table of Contents

Table of Contents

What is Property Testing?

$$\exists \implies \forall$$

What is Property Testing?

$\exists \implies \forall$

Unit tests on cartoon steroids.

What is Property Testing?

$\exists \implies \forall$

Unit tests on cartoon steroids.

Take from the source:

QuickCheck
Hypothesis
JSVerify

Property Testing - A closer look at the examples

You don't know what your inputs are.

Property Testing - A closer look at the examples

You don't know what your inputs are.

Your properties are meant to hold over a broad set of inputs, they must be general.

Property Testing - A closer look at the examples

You don't know what your inputs are.

Your properties are meant to hold over a broad set of inputs, they must be general.

How do you make meaningful assertions without re-implementing the code under test?

Property Testing - A closer look at the examples

You don't know what your inputs are.

Your properties are meant to hold over a broad set of inputs, they must be general.

How do you make meaningful assertions without re-implementing the code under test?

Revisit:

QuickCheck
Hypothesis
JSVerify

Table of Contents

Isomorphism Defined

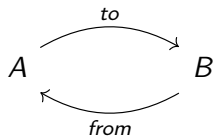$$\text{from(to(A))} = \mathit{from} \circ \mathit{to} = 1_A$$
$$\text{to(from(B))} = \mathit{to} \circ \mathit{from} = 1_B$$

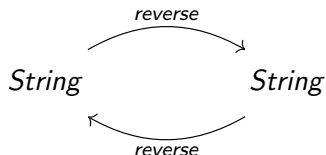# Isomorphism Example: encode $\iff$ decode
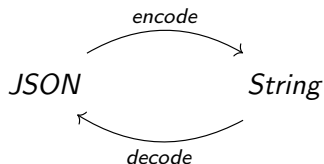
$$\text{decode(encode(JSON))} = decode \circ encode = 1_{JSON}$$
$$\text{encode(decode(String))} = encode \circ decode = 1_{String}$$

Isomorphism Example: reverse $\Longleftrightarrow$ reverse

$\text{reverse}(\text{reverse}(\text{String})) = reverse \circ reverse = 1_{String}$

$\text{reverse}(\text{reverse}(\text{String})) = reverse \circ reverse = 1_{String}$

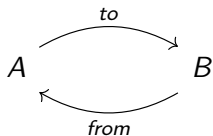# Table of Contents

Idempotent Defined
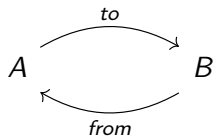
$$\text{to}(\text{from}(B)) = to \circ from = 1_B$$

Idempotent Defined

$$to(from(B)) = to \circ from = 1_B$$

$$from(to(from(to(A)))) =$$

Idempotent Defined
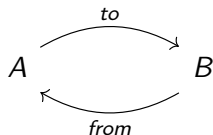
$$\text{to}(\text{from}(B)) = to \circ from = 1_B$$

$$\text{from}(\text{to}(\text{from}(\text{to}(A)))) =$$
$$from \circ to \circ from \circ to =$$

Idempotent Defined

$$\text{to}(\text{from}(B)) = to \circ from = 1_B$$

$$\text{from}(\text{to}(\text{from}(\text{to}(A)))) =$$
$$from \circ to \circ from \circ to =$$
$$from \circ (to \circ from) \circ to =$$

Idempotent Defined

$$\text{to}(\text{from}(B)) = \textit{to} \circ \textit{from} = 1_B$$

$$\text{from}(\text{to}(\text{from}(\text{to}(A)))) =$$
$$\textit{from} \circ \textit{to} \circ \textit{from} \circ \textit{to} =$$
$$\textit{from} \circ (\textit{to} \circ \textit{from}) \circ \textit{to} =$$
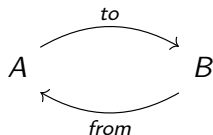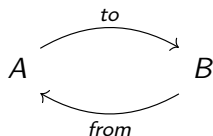$$\textit{from} \circ 1_B \circ \textit{to} =$$

Idempotent Defined

$$\text{to(from(B))} = to \circ from = 1_B$$

$$\text{from(to(from(to(A))))} =$$
$$from \circ to \circ from \circ to =$$
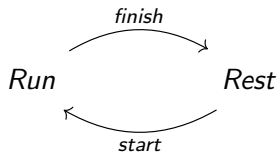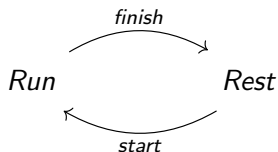$$from \circ (to \circ from) \circ to =$$
$$from \circ 1_B \circ to =$$
$$from \circ to$$

Idempotent Example: Run $\hookrightarrow$ Rest

$$\text{start}(\text{finish}(\text{Run})) = \textit{start} \circ \textit{finish} = 1_{Run}$$

Idempotent Example: Run $\hookrightarrow$ Rest

$$\text{start}(\text{finish}(\text{Run})) = \textit{start} \circ \textit{finish} = 1_{Run}$$

$$\text{finish}(\text{start}(\text{finish}(\text{start}(\text{Rest})))) =$$
$$\textit{finish} \circ \textit{start} \circ \textit{finish} \circ \textit{start} =$$
$$\textit{finish} \circ (\textit{start} \circ \textit{finish}) \circ \textit{start} =$$
$$\textit{finish} \circ (1_{Run}) \circ \textit{start} =$$
$$\textit{finish} \circ \textit{start}$$

Idempotent Example: JSON ↪ String
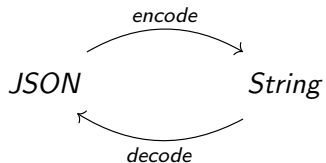
$$\text{decode}(\text{encode}(\text{JSON})) = decode \circ encode = 1_{JSON}$$

Idempotent Example: JSON $\hookrightarrow$ String
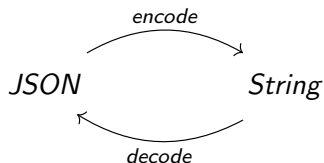
$$\text{decode}(\text{encode}(\text{JSON})) = decode \circ encode = 1_{JSON}$$

$$\text{encode}(\text{decode}(\text{encode}(\text{decode}(\text{String})))) =$$
$$encode \circ decode \circ encode \circ decode =$$
$$encode \circ (decode \circ encode) \circ decode =$$
$$encode \circ (1_{JSON}) \circ decode =$$
$$encode \circ decode$$

Table of Contents

Making Properties Easy

We know that properties are easy and effective when we have an isomorphism.

```python
from hypothesis import given
from hypothesis.strategies import text


@given(text())
def test_decode_inverts_encode(s):
    assert decode(encode(s)) == s
```

Making Properties Easy

We know that properties are easy and effective when we have an isomorphism.

```
from hypothesis import given
from hypothesis.strategies import text


@given(text())
def test_decode_inverts_encode(s):
    assert decode(encode(s)) == s
```

What about when we don't have an isomorphism?

Making Properties Easy

We know that properties are easy and effective when we have an isomorphism.

```
from hypothesis import given
from hypothesis.strategies import text


@given(text())
def test_decode_inverts_encode(s):
    assert decode(encode(s)) == s
```

What about when we don't have an isomorphism?

Can we *find* an isomorphism?

Finding an isomorphism

FizzBuzz does not belong to an isomorphism.

```python
from typing import List
def fizzbuzz(nums: List[int]) -> List[str]:
  res: List[str] = []
  for num in nums:
    s = ""
    if num % 3:
      s += "Fizz"
    if num % 5:
      s += "Buzz"
    if s == "":
      s = str(num)
    res.append(s)
  return res
```

Finding an isomorphism

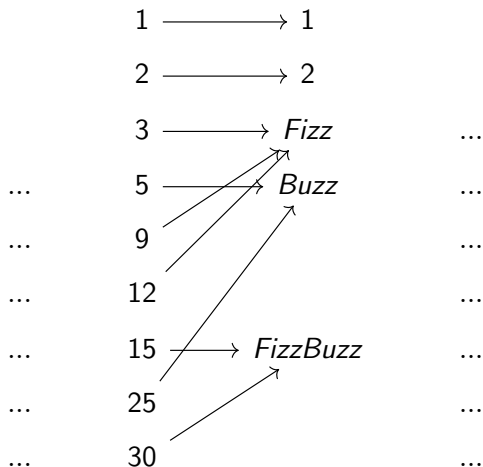FizzBuzz does not belong to an isomorphism.

```python
from typing import List
def fizzbuzz(nums: List[int]) -> List[str]:
  res: List[str] = []
  for num in nums:
    s = ""
    if num % 3:
      s += "Fizz"
    if num % 5:
      s += "Buzz"
    if s == "":
      s = str(num)
    res.append(s)
  return res
```

What is the closest isomorphism we can find?

It helps to take a different perspective.

FizzBuzz as a Set Function

We cannot have an isomorphism because inputs *collapse* onto outputs.

This prevents construction of an inverse.

FizzBuzz as a Set Function, Partitioned Domain

$$\{1\} \longrightarrow 1$$

$$\{2\} \longrightarrow 2$$

$$\{3, 6, 9, 12, ...\} \longrightarrow \textit{Fizz}$$

$$\{4\} \longrightarrow 4$$

$$\{5, 10, 20, 25, ...\} \longrightarrow \textit{Buzz}$$

$$\{7\} \longrightarrow 7 \qquad ...$$

$$... \qquad \{15, 30, ...\} \longrightarrow \textit{FizzBuzz}$$

$$\{16\} \longrightarrow 16$$

We have an isomorphism, can we fix the input type?

$\{1\} \longrightarrow 1$

$\{2\} \longrightarrow 2$

$\{3, 6, 9, 12, ...\} \longrightarrow$ *Fizz*

$\{4\} \longrightarrow 4$

$\{5, 10, 20, 25, ...\} \longrightarrow$ *Buzz*

$\{7\} \longrightarrow 7$ ...

... $\{15, 30, ...\} \longrightarrow$ *FizzBuzz*

$\{16\} \longrightarrow 16$

We have an isomorphism, can we fix the input type?

With an idempotent.

FizzBuzz^{-1} as a Set Function, Idempotent

We just pick one value from each input set.

$$1 \longleftarrow \{1\} \longleftarrow 1$$

$$2 \longleftarrow \{2\} \longleftarrow 2$$

$$3 \longleftarrow \{3, 6, 9, 12, ...\} \longleftarrow Fizz$$

$$4 \longleftarrow \{4\} \longleftarrow 4$$

$$5 \longleftarrow \{5, 10, 20, 25, ...\} \longleftarrow Buzz$$

$$7 \longleftarrow \{7\} \longleftarrow 7 \qquad ...$$

$$... \quad 15 \longleftarrow \{15, 30, ...\} \longleftarrow FizzBuzz$$

$$16 \longleftarrow \{16\} \longleftarrow 16$$

This can be pre-composed with FizzBuzz to create an identity on the output set.

This means we have an idempotent on the input set.

Table of Contents