

Property Testing with derived idempotents

Brian Zeligson

April 2020

Table of Contents

Introduction to Property Testing

Isomorphisms: Ideal Property Testing Candidates

Idempotents: Practical Property Testing Candidates

Application: Deriving Idempotents for Property Testing

Live Coding: Property Testing FizzBuzz

Table of Contents

Introduction to Property Testing

Isomorphisms: Ideal Property Testing Candidates

Idempotents: Practical Property Testing Candidates

Application: Deriving Idempotents for Property Testing

Live Coding: Property Testing FizzBuzz

What is Property Testing?

$$\exists \Rightarrow \forall$$

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

Introduction to Property Testing

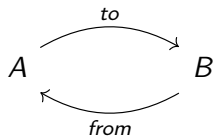
Isomorphisms: Ideal Property Testing Candidates

Idempotents: Practical Property Testing Candidates

Application: Deriving Idempotents for Property Testing

Live Coding: Property Testing FizzBuzz

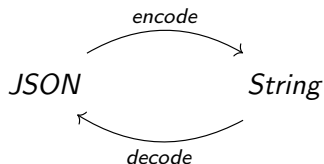
Isomorphism Defined



$$from(to(A)) = from \circ to = 1_A$$

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

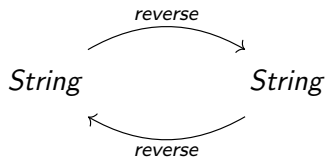
Isomorphism Example: encode \Longleftrightarrow decode



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

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

Isomorphism Example: reverse \iff reverse



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

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

Table of Contents

Introduction to Property Testing

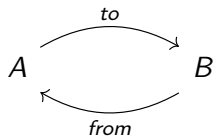
Isomorphisms: Ideal Property Testing Candidates

Idempotents: Practical Property Testing Candidates

Application: Deriving Idempotents for Property Testing

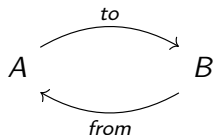
Live Coding: Property Testing FizzBuzz

Idempotent Defined



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

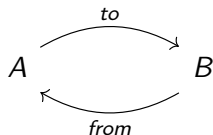
Idempotent Defined



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

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

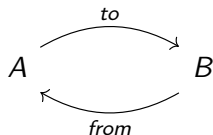
Idempotent Defined



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

$$from(to(from(to(A)))) = \\ from \circ to \circ from \circ to =$$

Idempotent Defined



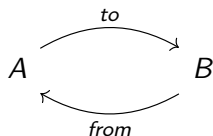
$$to(from(B)) = to \circ from = 1_B$$

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

$$from \circ to \circ from \circ to =$$

$$from \circ (to \circ from) \circ to =$$

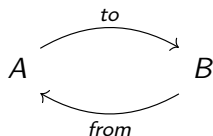
Idempotent Defined



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

$$\begin{aligned} from(to(from(to(A)))) &= \\ from \circ to \circ from \circ to &= \\ from \circ (to \circ from) \circ to &= \\ from \circ 1_B \circ to &= \end{aligned}$$

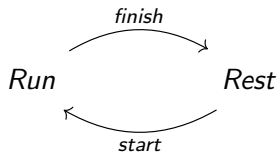
Idempotent Defined



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

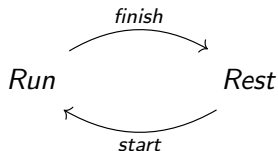
$$\begin{aligned} \text{from}(\text{to}(\text{from}(\text{to}(A)))) &= \\ \text{from} \circ \text{to} \circ \text{from} \circ \text{to} &= \\ \text{from} \circ (\text{to} \circ \text{from}) \circ \text{to} &= \\ \text{from} \circ 1_B \circ \text{to} &= \\ \text{from} \circ \text{to} \end{aligned}$$

Idempotent Example: $\text{Run} \hookrightarrow \text{Rest}$



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

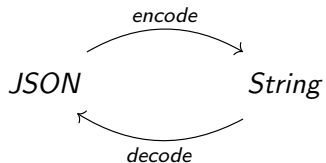
Idempotent Example: $\text{Run} \hookrightarrow \text{Rest}$



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

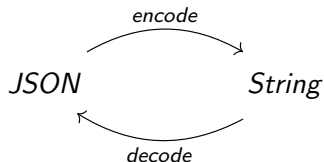
$$\begin{aligned} \text{finish}(\text{start}(\text{finish}(\text{start}(\text{Rest})))) &= \\ \text{finish} \circ \text{start} \circ \text{finish} \circ \text{start} &= \\ \text{finish} \circ (\text{start} \circ \text{finish}) \circ \text{start} &= \\ \text{finish} \circ (1_{\text{Run}}) \circ \text{start} &= \\ \text{finish} \circ \text{start} \end{aligned}$$

Idempotent Example: $\text{JSON} \hookrightarrow \text{String}$



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

Idempotent Example: $\text{JSON} \hookrightarrow \text{String}$



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

$$\begin{aligned} \text{encode}(\text{decode}(\text{encode}(\text{decode}(\text{String})))) &= \\ \text{encode} \circ \text{decode} \circ \text{encode} \circ \text{decode} &= \\ \text{encode} \circ (\text{decode} \circ \text{encode}) \circ \text{decode} &= \\ \text{encode} \circ (1_{\text{JSON}}) \circ \text{decode} &= \\ \text{encode} \circ \text{decode} \end{aligned}$$

Table of Contents

Introduction to Property Testing

Isomorphisms: Ideal Property Testing Candidates

Idempotents: Practical Property Testing Candidates

Application: Deriving Idempotents for Property Testing

Live Coding: Property Testing FizzBuzz

Making Properties Easy

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

```
prop_RevRev xs = reverse (reverse xs) == xs
  where types = xs :: [Int]
```

Making Properties Easy

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

```
prop_RevRev xs = reverse (reverse xs) == xs
  where types = xs :: [Int]
```

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.

```
prop_RevRev xs = reverse (reverse xs) == xs
  where types = xs :: [Int]
```

What about when we don't have an isomorphism?

Can we *find* an isomorphism?

Finding an isomorphism

FizzBuzz does not belong to an isomorphism.

```
object FizzBuzz {  
  def apply(nums: List[Int]): List[String] =  
    nums.map(n => (n % 3, n % 5) match {  
      case (0, 0) => "FizzBuzz"  
      case (0, _) => "Fizz"  
      case (_, 0) => "Buzz"  
      case _ => n.toString  
    })  
}
```

Finding an isomorphism

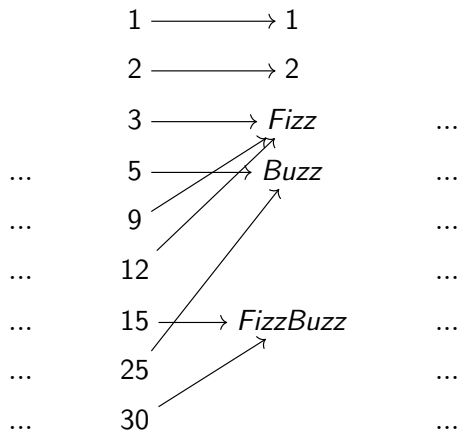
FizzBuzz does not belong to an isomorphism.

```
object FizzBuzz {  
  def apply(nums: List[Int]): List[String] =  
    nums.map(n => (n % 3, n % 5) match {  
      case (0, 0) => "FizzBuzz"  
      case (0, _) => "Fizz"  
      case (_, 0) => "Buzz"  
      case _ => n.toString  
    })  
}
```

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

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

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

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

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

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

$$\{7\} \longrightarrow 7 \quad \dots$$

$$\dots \quad \{15, 30, \dots\} \longrightarrow \textit{FizzBuzz}$$

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

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

FizzBuzz as a Set Function, Partitioned Domain

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

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

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

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

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

$$\{7\} \longrightarrow 7 \quad \dots$$

$$\dots \quad \{15, 30, \dots\} \longrightarrow \textit{FizzBuzz}$$

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

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 ...

... 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

Introduction to Property Testing

Isomorphisms: Ideal Property Testing Candidates

Idempotents: Practical Property Testing Candidates

Application: Deriving Idempotents for Property Testing

Live Coding: Property Testing FizzBuzz