Types and Property Tests for Software Resiliency

Agile India 2020

Tony Morris

Simple Machines

Simple
Machines

Just a bit of history

me

In the early 2000s, I was working at IBM
on the Java Development Kit

Just a bit of history

me

clumsily bumping into all the typical software engineering inefficiencies, bugs and limitations . . .

Simple
Machines

Just a bit of history

I had one simple thought

surely there is a better way . . .
surely someone smarter than me has figured it out

Just a bit of history

Yes

Searching far and with despair,
I found out that there is a better way to do software engineering

Simple **Machines**

Just a bit of history

It is called Functional Programming

Simple
Machines

What is Functional Programming?

What does it *mean*?

What is Functional Programming?

Suppose the following program ...

```
int wibble(int a, int b) {
  counter = counter + 1;
  return (a + b) * 2;
}

/* arbitrary code */

blobble(wibble(x, y), wibble(x, y));
```

Simple Machines

What is Functional Programming?

and we refactor out these common expressions ...

```
int wibble(int a, int b) {
  counter = counter + 1;
  return (a + b) * 2;
}

/* arbitrary code */

blobble(wibble(x, y), wibble(x, y));
```

Simple Machines

What is Functional Programming?

assign the expression to a value

```
int wibble(int a, int b) {
  counter = counter + 1;
  return (a + b) * 2;
}

/* arbitrary code */

int r = wibble(x, y);

blobble(r, r);
```

Simple Machines

What is Functional Programming?

Did the program just change?

Simple
Machines

What is Functional Programming?

Yes, the program changed . . .

```
int wibble(int a, int b) {
  counter = counter + 1;
  return (a + b) * 2;
}

/* arbitrary code */

int r = wibble(x, y);

blobble(r, r);
```

Simple
Machines

What is Functional Programming?

Suppose this slightly different program . . .

```
int pibble(int a, int b) {
  return (a + b) * 2;
}

/* arbitrary code */

globble(pibble(x, y), pibble(x, y));
```

Simple
Machines

What is Functional Programming?

and we refactor out these common expressions . . .

```
int pibble(int a, int b) {
  return (a + b) * 2;
}

/* arbitrary code */

globble(pibble(x, y), pibble(x, y));
```

Simple
Machines

What is Functional Programming?

assign the expression to a value

```
int pibble(int a, int b) {
  return (a + b) * 2;
}

/* arbitrary code */

int r = pibble(x, y);

globble(r, r);
```

Simple Machines

What is Functional Programming?

This time, did the program just change?

What is Functional Programming?

It's the same program

For given inputs, the same outputs are given, with no observable changes to the program

Simple Machines

What is Functional Programming?

Functional Programming is the idea that

We can **always replace expressions with a value, without affecting the program behaviour**

What is Functional Programming?

Let's start at a concrete example

How do I sum the integer values in a list?

What is Functional Programming?

Using a for loop

```
sum(list) {
  var r = 0;
  for(int i = 0; i < list.length; i++) {
    r = r + list[i];
  }
  return r;
}
```

Simple
Machines

What is Functional Programming?

Using a for loop

```
sum(list) {
  var r = 0;
  for(int i = 0; i < list.length; i++) {
    r = r + list[i];
  }
  return r;
}
```

Simple
Machines

What is Functional Programming?

Here is another way of looking at the problem

Simple
Machines

What is Functional Programming?

The sum of a list is . . .

- if the list is empty, return 0
- otherwise add the first element to the sum of the remainder of the list

What is Functional Programming?

The sum of a list is . . .

```
sum([6, 5, 9, 71, 3]) =
6 + sum ([5, 9, 71, 3]) =
6 + 5 + sum([9, 71, 3]) =
6 + 5 + 9 + sum([71, 3]) =
6 + 5 + 9 + 71 + sum([3]) =
6 + 5 + 9 + 71 + 3 + sum([]) =
6 + 5 + 9 + 71 + 3 + 0 =
94
```

What is Functional Programming?

Here is the Haskell source code

```
sum [] = 0
sum (first:rest) = first + sum rest
```

Simple Machines

Functional Programming

There are broader consequences once we commit to functional programming . . .

Software resiliency

One of them is an ability to efficiency and effectively ensure our software is correct

Simple
Machines

Software sandcastles

Who has one of these software systems?

- the tests take two hours to run
- so you commit, push and go to the coffee shop
- on your way back from the coffee shop you receive an email
- the tests failed because the build couldn't find the conf file
- you finally fix the build and tests so they pass
- you are onto your third coffee of the day
- you put the software into production
- the software falls over in a crumbling heap anyway
- "but it works on my machine!"
- go back to step 1

Software sandcastles

Who has one of these software systems?

- the tests take two hours to run
- so you commit, push and go to the coffee shop
- on your way back from the coffee shop you receive an email
- the tests failed because the build couldn't find the conf file
- you finally fix the build and tests so they pass
- you are onto your third coffee of the day
- you put the software into production
- the software falls over in a crumbling heap anyway
- "but it works on my machine!"
- go back to step 1

...nes

Software sandcastles

Who has one of these software systems?

- the tests take two hours to run
- so you commit, push and go to the coffee shop
- on your way back from the coffee shop you receive an email
- the tests failed because the build couldn't find the conf file
- you finally fix the build and tests so they pass
- you are onto your third coffee of the day
- you put the software into production
- the software falls over in a crumbling heap anyway
- "but it works on my machine!"
- go back to step 1

Software sandcastles

Who has one of these software systems?

- the tests take two hours to run
- so you commit, push and go to the coffee shop
- on your way back from the coffee shop you receive an email
- the tests failed because the build couldn't find the conf file
- you finally fix the build and tests so they pass
- you are onto your third coffee of the day
- you put the software into production
- the software falls over in a crumbling heap anyway
- "but it works on my machine!"
- go back to step 1

Software sandcastles

Who has one of these software systems?

- the tests take two hours to run
- so you commit, push and go to the coffee shop
- on your way back from the coffee shop you receive an email
- the tests failed because the build couldn't find the conf file
- you finally fix the build and tests so they pass
- you are onto your third coffee of the day
- you put the software into production
- the software falls over in a crumbling heap anyway
- "but it works on my machine!"
- go back to step 1

Software sandcastles

Who has one of these software systems?

- the tests take two hours to run
- so you commit, push and go to the coffee shop
- on your way back from the coffee shop you receive an email
- the tests failed because the build couldn't find the conf file
- you finally fix the build and tests so they pass
- you are onto your third coffee of the day
- you put the software into production
- the software falls over in a crumbling heap anyway
- "but it works on my machine!"
- go back to step 1

Software sandcastles

Who has one of these software systems?

- the tests take two hours to run
- so you commit, push and go to the coffee shop
- on your way back from the coffee shop you receive an email
- the tests failed because the build couldn't find the conf file
- you finally fix the build and tests so they pass
- you are onto your third coffee of the day
- you put the software into production
- the software falls over in a crumbling heap anyway
- "but it works on my machine!"
- go back to step 1

...ines

Software sandcastles

Who has one of these software systems?

- the tests take two hours to run
- so you commit, push and go to the coffee shop
- on your way back from the coffee shop you receive an email
- the tests failed because the build couldn't find the conf file
- you finally fix the build and tests so they pass
- you are onto your third coffee of the day
- you put the software into production
- the software falls over in a crumbling heap anyway
- "but it works on my machine!"
- go back to step 1

...nes

Software sandcastles

Who has one of these software systems?

- the tests take two hours to run
- so you commit, push and go to the coffee shop
- on your way back from the coffee shop you receive an email
- the tests failed because the build couldn't find the conf file
- you finally fix the build and tests so they pass
- you are onto your third coffee of the day
- you put the software into production
- the software falls over in a crumbling heap anyway
- "but it works on my machine!"
- go back to step 1

...nes

Software resiliency

it is because we are functional programming ...

- we can use **types** to determine the behaviour of our software
- we can use **automated testing** to make up for where types left gaps

Simple Machines

Software resiliency

it is because we are functional programming . . .

- we can use **types** to determine the behaviour of our software
- we can use **automated testing** to make up for where types left gaps

Fast and loose reasoning

Some programming languages have escape hatches . . .

- `null`
- `exceptions`
- `type-casting`
- `type-casing` e.g. `instanceof`
- `non-termination`

Fast and loose reasoning

We can (reasonably) disregard these

> *Functional programmers often reason about programs as if they were written in a total language, expecting the results to carry over to non-total (partial) languages. We justify such reasoning.*

Danielsson, Hughes, Jansson & Gibbons [DHJG06]

Consider the following Java function ...

```java
boolean boolean2boolean(boolean b) {
  // hidden from view
}
```

How many possible programs can be written that satisfy the type?
i.e. from the type, how much knowledge have we gained?

Simple
Machines

What about this Java function . . .

```java
String string2string(String s) {
  // hidden from view
}
```

from the type, how much knowledge have we gained?

OK now this Java function . . .

```
<A> A any2any(A a) {
  // hidden from view
}
```

How many possible programs can be written that satisfy the type?

Simple **Machines**

Polymorphic values

By utilising *polymorphic* values in a type . . .

we have gained a **lot** of knowledge of our function's behaviour
In this case, we have obtained *total* knowledge

Simple **Machines**

Polymorphic values

Parametricity

This idea of using parametric polymorphism to determine a function's behaviour is called *parametricity*

Simple **Machines**

What is Parametricity and Free Theorems

Philip Wadler [Wad89] tells us:

> *Write down the definition of a polymorphic function on a piece of paper. Tell me its type, but be careful not to let me see the function's definition. I will tell you a theorem that the function satisfies.*
>
> *The purpose of this paper is to explain the trick.*

Simple
Machines

Try this Java function . . .

```
List<String> strings2strings(List<String> s) {
  // hidden from view
}
```

from the type, how much knowledge have we gained?

Polymorphic values

This type has no polymorphic values

```
List<String> strings2strings(List<String> x) {
  // hidden from view
}
```

Simple
Machines

Polymorphic values

Can we determine function behaviour?

```
<T> List<T> anythings2anythings(List<T> x) {
  // hidden from view
}
```

Theorem

every element in the resulting list, appears in the input list

Polymorphic values

Some amount of function behaviour

```
<T> List<T> anythings2anythings(List<T> x) {
  // hidden from view
}
```

Theorem

We have **some** amount of information, but not **total** information

Let's write an *automated* test

Simple **Machines**

Polymorphic values and tests

Can we determine function behaviour?

```
<T> List<T> anythings2anythings(List<T> x) {
  // hidden from view
}
```

Tests

```
prop_anythings2anythings1 :: Property
prop_anythings2anythings1 =
  property $ do
    x <- forAll alpha
    anythings2anythings [x] == [x]
```

Simple Machines

Polymorphic values and tests

Can we determine function behaviour?

```
<T> List<T> anythings2anythings(List<T> x) {
  // hidden from view
}
```

Tests

```
prop_anythings2anythings2 :: Property
prop_anythings2anythings2 =
  property $ do
    x <- forAll (list (linear 0 100) alpha)
    y <- forAll (list (linear 0 100) alpha)
    anythings2anythings (x ++ y) ==
      anythings2anythings y ++ anythings2anythings x
```

Types and tests

By this method, it becomes very explicit that . . .

- Types alone provide a *proof* of a proposition
- Polymorphic types provide *additional theorems*
  i.e. free theorems
- Tests provide a *failed negative proof* of a proposition
- This outcome is the *only* difference between types and tests

Once-inhabitance

```
<T> T anything2anything(T x) {
  // hidden from view
}
```

This type is an example of *once-inhabitance*

There is only one function with this type

It is not possible to write tests for it —tests are redundant

Simple
Machines

Types and tests

But these are trivial examples

What about more realistic examples?

Simple
Machines

Types and tests

```haskell
-- the type implies this function does no I/O
validateWebForm ::
  f WebForm
  -> f (Either WebFormErrors ValidatedWebForm)

-- this function may do I/O
submitWebForm ::
  AppState
  -> WebForm
  -> IO (Response, AppState)
```

Simple
Machines

Types and tests

```
-- idempotence
prop_submitWebForm :: Property
prop_submitWebForm =
  property $ do
    w       <- forAll genWebForm
    s       <- forAll genAppState
    (_, s1) <- submitWebForm s (submitWebForm s w)
    (_, s2) <- submitWebForm s w
    s1 == s2
```

Simple
Machines

Types and tests

Types and tests . . .

- We use types **first**
- Where types fall short, we use **automated tests**
- Tests are written using the hedgehog[a] library
- Tests are deterministic
  "works on my machine today"
  $\longrightarrow$
  "works on all machines at all times"

[a] hedgehog ▸ Link

Algebra

```
boolean boolean2boolean(boolean b) {
  // hidden from view
}
```

How many possible programs can be written that satisfy the type?

We can calculate this *algebraically*

Simple
Machines

Counting inhabitants

```
boolean boolean2boolean(boolean b) {
  // hidden from view
}
```

The inhabitants of a function's type . . .
is the return type raised to the power of its argument type

Simple **Machines**

Counting inhabitants

```
boolean boolean2boolean(boolean b) {
  // hidden from view
}
```

$\text{boolean}^{\text{boolean}}$

$= 4$

Simple Machines

Counting inhabitants

```
boolean boolean2boolean(boolean b) {
  // hidden from view
}
```

Here are all the possible functions

```
return true;  // 1
return false; // 2
return b;     // 3
return !b;    // 4
```

Counting inhabitants

What about this one?

```
<A> A anything2anything(A a) {
  // hidden from view
}
```

Simple
Machines

Counting inhabitants

What about this one?

```
<A> A anything2anything(A a) {
  // hidden from view
}
```

- Assume = 1
- Prove = 1 using *the yoneda lemma*
- . . . using Java . . .
- you know, for giggles

Simple Machines

Counting inhabitants

What about this one?

```
<A> A anything2anything(A a) {
  // hidden from view
}
```

- Assume = 1
- Prove = 1 using *the yoneda lemma*
- . . . using Java . . .
- you know, for giggles

Proof

Yoneda Lemma

https://github.com/simple-machines/types-and-tests/blob/master/source/yoneda.java

Here are some more examples

One inhabitant

```
-- Haskell
(b -> c) -> (a -> b) -> a -> c

// Java
<A, B, C>
Function<A, C> c(Function<B, C> f, Function<A, B> g)
```

Here are some more examples

Two inhabitants

```
-- Haskell
a -> a -> a

// Java
<A>
A c(A a1, A a2)
```

Prove using the Yoneda lemma = `boolean` = 2

What tests can we write?

Simple Machines

Here are some more examples

One inhabitant

```
-- Haskell
Functor f => a -> f b -> f a
```

Simple Machines

Here are some more examples

Infinite inhabitants

```
-- Haskell
Applicative f => a -> f b -> f a
```

References

Nils Anders Danielsson, John Hughes, Patrik Jansson, and Jeremy Gibbons, *Fast and loose reasoning is morally correct*, ACM SIGPLAN Notices, vol. 41, ACM, 2006, pp. 206–217.

Philip Wadler, *Theorems for free!*, Proceedings of the fourth international conference on Functional programming languages and computer architecture, ACM, 1989, pp. 347–359.