QUICK! Check Your Properties

(and Write Better Software)

Paul Blasucci

paul@statmuse.com

RANDOM TESTING

"Properties are described as … **functions**, and can be **automatically tested on random input**... [or] custom test data generators."

from ICFP'00 – Claessen, Hughes

FROM UNIT TESTING…

```
[<Fact>]
let PlusIgnoresTime () =
  let days  = time.FromDays   7
  let hours = time.FromHours (7 * 24)
  let civil = date.Now

  Assert.Equal (civil + days
                ,civil + hours)
```

TEST EXECUTION SUMMARY

Tests run: 1, Errors: 0, Failed: 0, Ignored: 0

✓ **PlusIgnoresTime**
OK, Elapsed time: 0.0527666s

TO PROPERTY TESTING!

```
[<Property>]
let ``plus ignores time`` (civil:date) =
  let days  = time.FromDays   7
  let hours = time.FromHours (7 * 24)

  civil + days = civil + hours
```

TEST EXECUTION SUMMARY

Tests run: 1, Errors: 0, Failed: 0, Ignored: 0

✓ **plus ignores time**
OK, passed 100 tests

PATTERNS: INVERSION & IDEMPOTENCE

```
// inversion ... an action and its inverse cancel each other out
[Property]
public static Boolean AddAndSubtract_AreInverses (date civil, PositiveInt total)
{
  var days = time.FromDays(total.Item);
  return ((civil + days) - days == civil);
}
```

```
// idempotence ... an action has the same effect no matter how many times it occurs
[Property]
public static Boolean Taking_TimeDuration_IsIdempotent (time value)
{
  var once  = value.Duration();
  var twice = value.Duration().Duration();
  return (once == twice);
}
```

PATTERNS: INTERCHANGE & INVARIANCE

```
' interchange ... the order of two or more actions does not affect the outcome
<[Property]>
Public Function Add_Change_CanBeReordered (civil As Dated, total As PositiveInt)
  Dim pacStd       = "Pacific Standard Time“
  Dim days         = Time.FromDays(total)
  Dim addThenShift = Zone.ConvertTimeBySystemTimeZoneId(civil + days, pacStd)
  Dim shiftThenAdd = Zone.ConvertTimeBySystemTimeZoneId(civil, pacStd) + days
  Return addThenShift = shiftThenAdd
End Function

' invariance ... something remains constant, despite action being taken
<[Property] >
Public Function Add_DoesNotChange_Offset (civil As Dated, months As PositiveInt)
  Dim offset  = civil.Offset
  Dim shifted = civil.AddMonths(months)
  Return shifted.Offset = offset
End Function
```

INPUT CONTROL

Conditional Properties

```
[<Property>]
let ``DST test oracle (naive)`` (civil :date) =
  let eastern = zone.FindSystemTimeZoneById EST
  let etDate  = zone.ConvertTime (civil,eastern)

  Zone.inUnitedStatesDaylightTime etDate
    = eastern.IsDaylightSavingTime etDate
```

TEST EXECUTION SUMMARY

Tests run: 1, Errors: 0, Failed: 1, Ignored: 0

✗ **DST test oracle (naive)**
Falsifiable, after 1 test (4 shrinks), (StdGen (2119435949,296213433))

Original: 1908-04-23 23:48:57 -04:02
Shrunk: 1908-04-23 00:00:00 +00:00

INPUT CONTROL

Conditional Properties

```
[<Property>]
let ``DST test oracle`` (civil :date) =
  let eastern = zone.FindSystemTimeZoneById EST
  let etDate  = zone.ConvertTime (civil,eastern)

  ( civil.Year >= 2007
    && eastern.IsDaylightSavingTime etDate )
      ==> lazy
            Zone.inUnitedStatesDaylightTime etDate
```

TEST EXECUTION SUMMARY

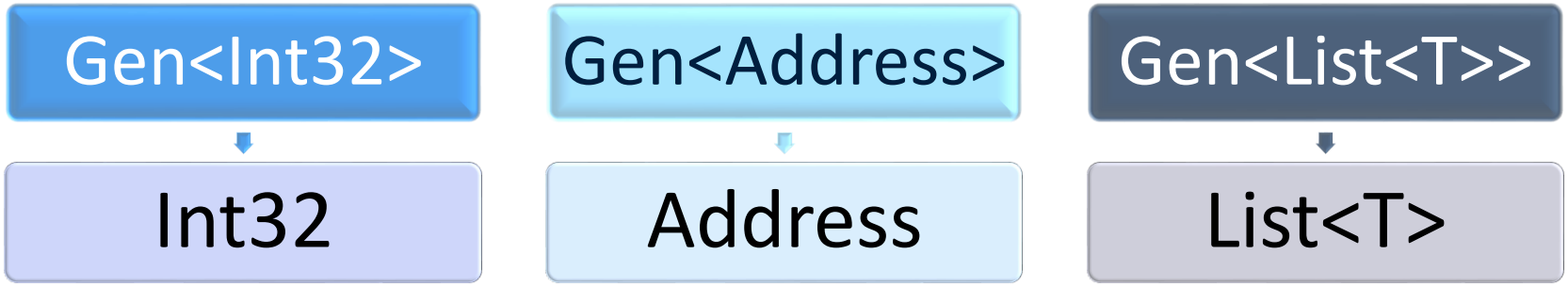
Tests run: 1, Errors: 0, Failed: 0, Ignored: 0

✓ **DST test oracle**
OK, passed 100 tests

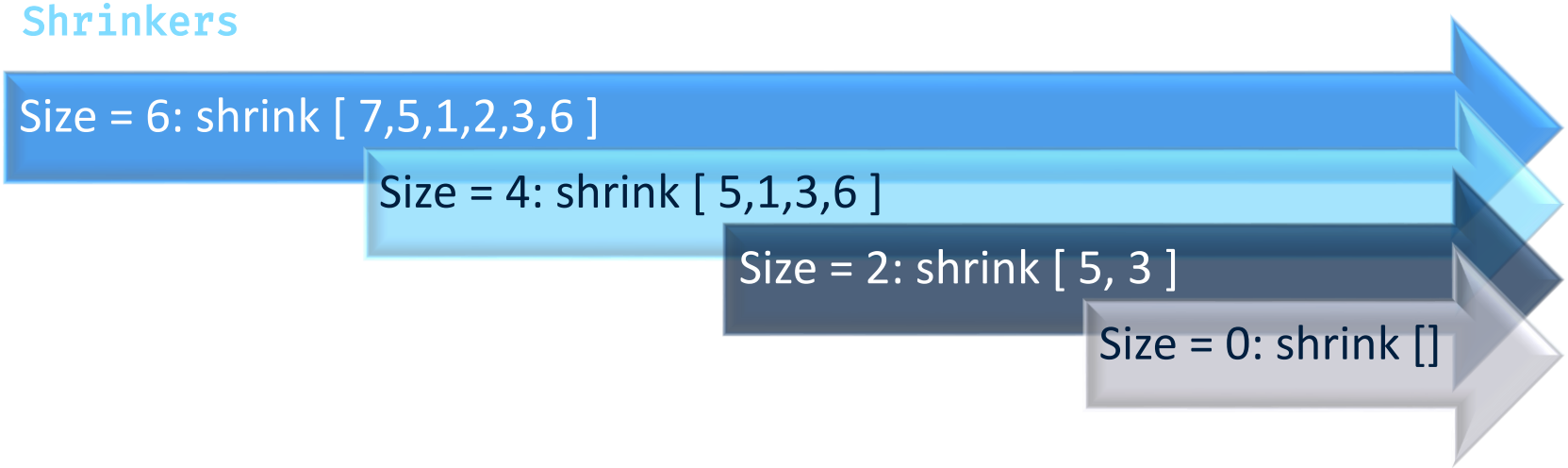
CUSTOM DATA GENERATION

CUSTOM DATA GENERATION

Generators



Shrinkers



INPUT CONTROL

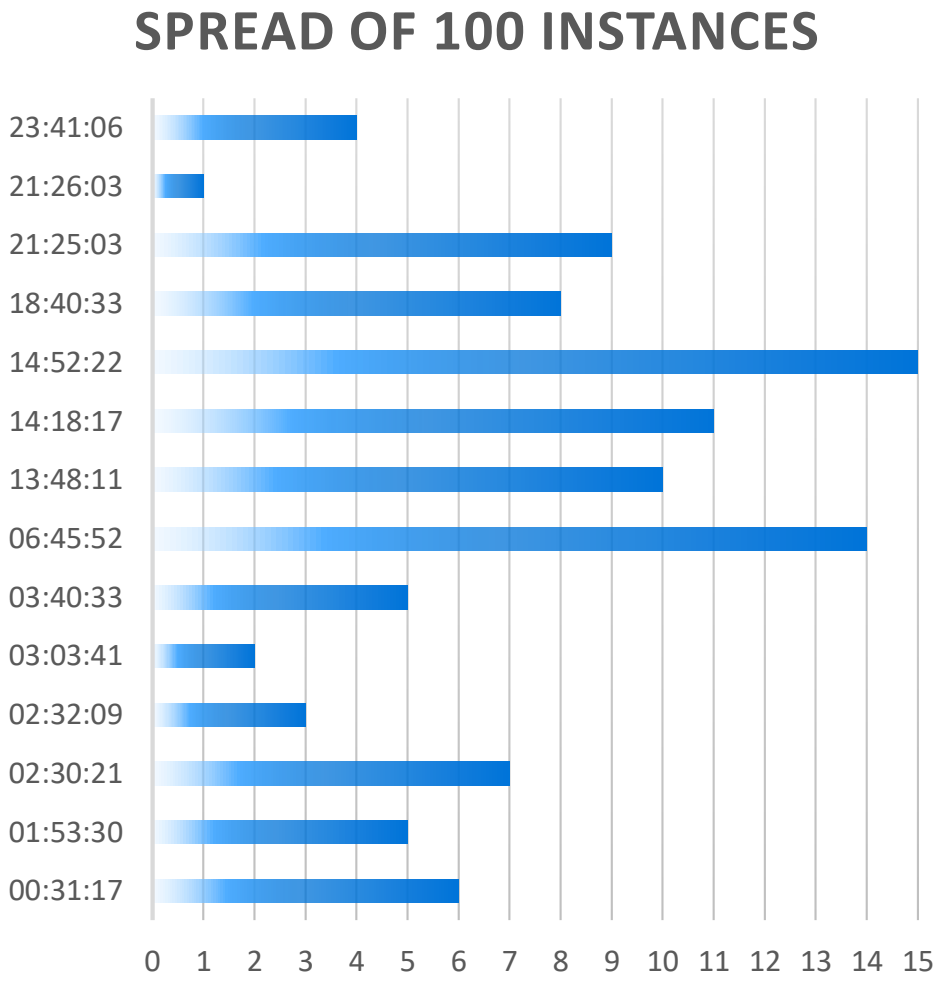
Universal Quantifiers

```
// instead of a conditional property,
// we can use an Arb and a universal quantifier
[Property]
public static Property RoundTripSerialization ()
{
  // define arbitrary generator
  var zones = Gen.Elements(
    from   z in zone.GetSystemTimeZones()
    select z)
    .ToArbitrary();

  // "for all" zones, run a test...
  return Prop.ForAll(zones, z =>
  {
    var str = z.ToSerializedString();
    var obj = zone.FromSerializedString(str);
    return z.Equals(obj);
  });
```

DATA GENERATION

Arb with Shrinker



CUSTOM DATA GENERATION

```
''' encapsulates several IArbitrary instances
Friend Module Generator
  ''' generates PositiveTime instances by leveraging FsCheck's built-in
  ''' support for generating and shrinking TimeSpan instances
  Public Function PositiveTime () As Arbitrary(Of PositiveTime)
    Dim generator =
      From    t In Arb.Generate(Of Time)()
      Where   t > Time.Zero
      Select  New PositiveTime(t)

    Dim shrinker As Func(Of PositiveTime,IEnumerable(Of PositiveTime)) =
      Function (posTime)
          Return From   t In Arb.Shrink(posTime.Value)
                 Where  t > Time.Zero
                 Select New PositiveTime(t)
      End Function

    Return Arb.From(generator,shrinker)
  End Function
```

DIAGNOSTICS: LABELLING PROPERTIES

```
[<Property (Arbitrary=[| typeof<Generator> |])>]
let ``conversion ignores detours`` (civil :date) (zone1 :zone) (zone2 :zone) =
  let viaZone1 = zone.ConvertTime (zone.ConvertTime (civil,zone1),zone2)
  let directly = zone.ConvertTime (civil,zone2)

  (viaZone1 = directly) && (directly.Offset = zone2.BaseUtcOffset)
```

TEST EXECUTION SUMMARY

Tests run: 1, Errors: 0, Failed: 1, Ignored: 0

✗ **conversion ignores detours**
Falsifiable, after 4 tests (5 shrinks), (StdGen (199662269,296213481)):

Original: (1948-04-19 16:18:52 +04:59, (UTC+04:00), (UTC-05:00))
Shrunk: (1948-04-19 00:00:00 +00:00, (UTC+04:00), (UTC-05:00))

DIAGNOSTICS: LABELLING PROPERTIES

```
[<Property (Arbitrary=[| typeof<Generator> |])>]
let ``conversion ignores detours`` (civil :date) (zone1 :zone) (zone2 :zone) =
  let viaZone1 = zone.ConvertTime (zone.ConvertTime (civil,zone1),zone2)
  let directly = zone.ConvertTime (civil,zone2)

  (viaZone1 = directly) |@ sprintf "Not the same date!"
  .&.
  (directly.Offset = zone2.BaseUtcOffset) |@ sprintf "Not the same zone!"
```

TEST EXECUTION SUMMARY

Tests run: 1, Errors: 0, Failed: 1, Ignored: 0

✗ **conversion ignores detours**

Falsifiable, after 4 tests (5 shrinks), (StdGen (199662269,296213481)):

Label of failing property: Not the same zone!

Original: (1948-04-19 16:18:52 +04:59, (UTC+04:00), (UTC-05:00))

Shrunk: (1948-04-19 00:00:00 +00:00, (UTC+04:00), (UTC-05:00))

DIAGNOSTICS: LABELLING PROPERTIES

```
[<Property (Arbitrary=[| typeof<Generator> |])>]
let ``conversion ignores detours`` (civil :date) (zone1 :zone) (zone2 :zone) =
  let viaZone1 = zone.ConvertTime (zone.ConvertTime (civil,zone1),zone2)
  let directly = zone.ConvertTime (civil,zone2)

  (viaZone1 = directly) |@ sprintf "Not the same date!"
  .&.
  (directly.Offset = zone2.GetUtcOffset directly) |@ sprintf "Not the same zone!"
```

TEST EXECUTION SUMMARY

Tests run: 1, Errors: 0, Failed: 0, Ignored: 0

✓ **conversion ignores detours**
OK, passed 100 tests

DIAGNOSTICS

Gathering Observations

TEST EXECUTION SUMMARY

Tests run: 1, Errors: 0, Failed: 0, Ignored: 0

✓ **add and change can be reorderd**

OK, passed 100 tests

26% Odd, East of Greenwich, trivial.
21% Even, East of Greenwich, trivial.
14% Even, West of Greenwich, trivial.
13% Odd, West of Greenwich, trivial.
8% Odd, West of Greenwich.
8% Even, West of Greenwich.
5% Odd, East of Greenwich.
5% Even, East of Greenwich.

DIAGNOSTICS: GATHERING OBSERVATIONS

```
// a trival observation partions data into one of two buckets
[Property (Arbitrary = new []{ typeof(Generator) })]
public static Property Interchange (date civil, zone target, NonNegativeInt total)
{
  var days = time.FromDays(total.Item);

  var addThenShift = zone.ConvertTime(civil + days, target);
  var shiftThenAdd = zone.ConvertTime(civil, target) + days;

  // Does target zone support DST? Yes or no
  return (addThenShift == shiftThenAdd).Trivial(target.SupportsDaylightSavingTime);
}
```

DIAGNOSTICS: GATHERING OBSERVATIONS

```
// a classification partions data into one of N, labelled buckets
[Property (Arbitrary = new []{ typeof(Generator) })]
public static Property Interchange (date civil, zone target, NonNegativeInt total)
{
  var days = time.FromDays(total.Item);

  var addThenShift = zone.ConvertTime(civil + days, target);
  var shiftThenAdd = zone.ConvertTime(civil, target) + days;

  // What is civil date's longitude, relative to GMT?
  return (addThenShift == shiftThenAdd)
        .Classify(civil.Offset <  time.Zero, "West of Greenwich")
        .Classify(civil.Offset == time.Zero, "Within Greenwich" )
        .Classify(civil.Offset  > time.Zero, "East of Greenwich");
}
```

DIAGNOSTICS: GATHERING OBSERVATIONS

```
' rather than using a boolean observation, collect reports any value
<[Property] (Arbitrary := { GetType(Generator) }) >
Public Function Interchange (civil  As Dated,
                             target As Zone,
                             total  As NonNegativeInt) As [Property]
  Dim days = Time.FromDays(total.Item)

  Dim addThenShift = Zone.ConvertTime(civil + days, target)
  Dim shiftThenAdd = Zone.ConvertTime(civil, target) + days

  ‘ Is total increment evenly divisible by 2?
  Return (addThenShift = shiftThenAdd)
        .Collect(IIf(total.Item Mod 2 = 0, Even, Odd))
End Function
```

DIAGNOSTICS: GATHERING OBSERVATIONS

```
' observations may be combined as mush as is desired
<[Property] (Arbitrary := { GetType(Generator) })>
Public Function Interchange (civil  As Dated,
                             target As Zone,
                             total  As NonNegativeInt) As [Property]
  Dim days = Time.FromDays(total.Item)

  Dim addThenShift = Zone.ConvertTime(civil + days, target)
  Dim shiftThenAdd = Zone.ConvertTime(civil, target) + days

  Return (addThenShift = shiftThenAdd)
         .Trivial(target.SupportsDaylightSavingTime)                _
         .Classify(civil.Offset < Time.Zero, "West of Greenwich")   _
         .Classify(civil.Offset = Time.Zero, "Within Greenwich" )   _
         .Classify(civil.Offset > Time.Zero, "East of Greenwich")   _
         .Collect(IIf(total.Item Mod 2 = 0, Even, Odd))
  End Function
```

RANDOM TESTING

"One of the major advantages… is that it **encourages** us to formulate **formal specifications, thus improving** our **understanding**…"

from ICFP'00 – Claessen, Hughes

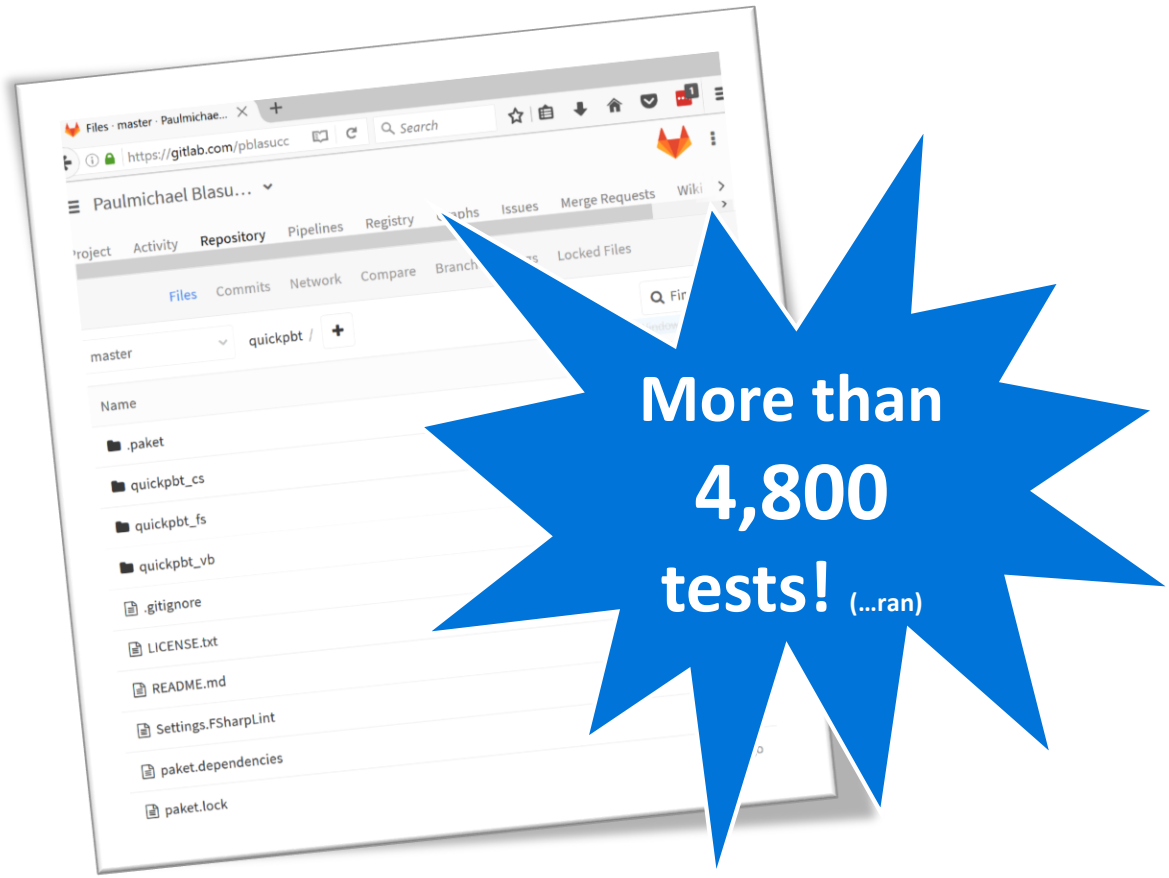
FURTHER INFORMATION

About F# and FsCheck

- fsharp.org
- fscheck.github.io/FsCheck
- fsharpforfunandprofit.com
- www.fssnip.net

About Paulmichael Blasucci

- twitter.com/pblasucci
- github.com/pblasucci
- pblasucci.wordpress.com
- linkedin.com/in/pblasucci



gitlab.com/pblasucci/quickpbt

My code is your code