Automating unit testing

Roman Cheplyaka

May 12, 2012
YAPC::Russia «May Perl + Perl Mova»

Example: prime numbers

Task: write a function that generates only prime numbers

$$p(n) : \forall n \geq 0 \; f(n) \text{ is prime}$$

Example: prime numbers

Task: write a function that generates only prime numbers

$p(n) : \forall n \geq 0$ $f(n)$ is prime

$$p(n) = n^2 + n + 41$$

Testing prime numbers

```
use Test::Simple;

sub p($) { my $n = shift; return $n**2+$n+41; }

ok(isPrime(p(1)), "p(1) is prime");
```

Testing prime numbers

```
use Test::Simple;

sub p($) { my $n = shift; return $n**2+$n+41; }

ok(isPrime(p(1)), "p(1) is prime");
ok(isPrime(p(7)), "p(7) is prime");
```

Testing prime numbers

```
use Test::Simple;

sub p($) { my $n = shift; return $n**2+$n+41; }

ok(isPrime(p(1)), "p(1) is prime");
ok(isPrime(p(7)), "p(7) is prime");
ok(isPrime(p(24)), "p(24) is prime");
```

Testing prime numbers

```
use Test::Simple;

sub p($) { my $n = shift; return $n**2+$n+41; }

ok(isPrime(p(1)), "p(1) is prime");
ok(isPrime(p(7)), "p(7) is prime");
ok(isPrime(p(24)), "p(24) is prime");
ok(isPrime(p(37)), "p(37) is prime");
```

Testing prime numbers

```
use Test::Simple;

sub p($) { my $n = shift; return $n**2+$n+41; }

ok(isPrime(p(1)), "p(1) is prime");
ok(isPrime(p(7)), "p(7) is prime");
ok(isPrime(p(24)), "p(24) is prime");
ok(isPrime(p(37)), "p(37) is prime");
ok(isPrime(p(50)), "p(50) is prime");
```

Better testing

```
for $i (1..100) {
    ok(isPrime(p($i)),"p($i) is prime");
}
```

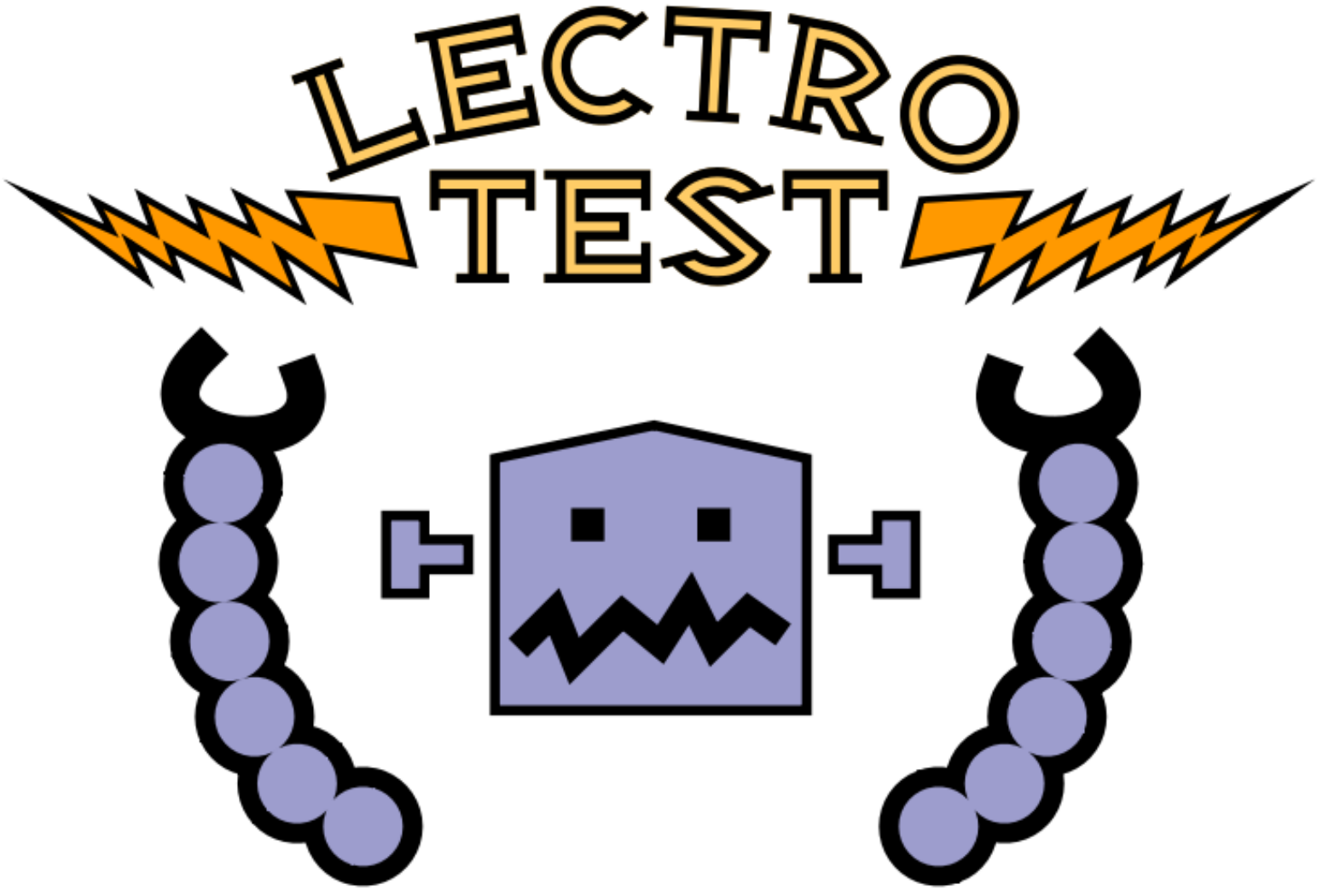
Sorting

```
sub mysort($) {
    my @arr = @{$_[0]};
    return \@arr unless @arr;
    my $first = shift @arr;
    my @less = @{mysort([grep { $_ < $first } @arr])};
    my @greater = @{mysort([grep { $_ >= $first } @arr])};
    push @less, $first, @greater;
    return \@less;
}
```

Testing sorting

```
ok(mysort [3,1,4,2] ~~ [1,2,3,4], "sort(3,1,4,2)");
```

Introducing LectroTest



Primes

```
use Test::LectroTest;

sub p { my $n = shift; return $n**2+$n+41; }

Property {
    ##[ n <- Int(range => [0,1000]) ]##
    isPrime(p($n));
}, name => "Always prime";
```

Example

```
not ok 1 - 'Always prime' falsified in 80 attempts
# Counterexample:
# $n = 40;
```

Sorting

```
Property {
    ##[ x <- List(Int) ]##
    mysort($x) ~~ [sort {$a <=> $b} @$x];
}, name => "my sort";
```

No reference implementation?

Check necessary properties
Check sufficient properties

Sorting

Check necessary properties

Sorting

Check necessary properties

- The result is a sorted array

Sorting

Check necessary properties

- The result is a sorted array
- The result has the same length as the original array

Sorting

Check sufficient properties

1. The result is a sorted array
2. Source and result consist of the same elements

Advice on testable code

```
sub nextPerlEvent {
    my ($mday,$mon,$year) = (localtime())[3,4,5];
    # ... compute the next date
}
```

Advice on testable code

```
sub nextPerlEvent {
    my ($mday,$mon,$year) = @_;
    # ... compute the next date
}
```

Conclusions

- Seek to automate test case generation
- `use Test::LectroTest;`
- Compare to reference implementation
- Check sufficient properties
- If not possible, check necessary properties
- Stay functional!