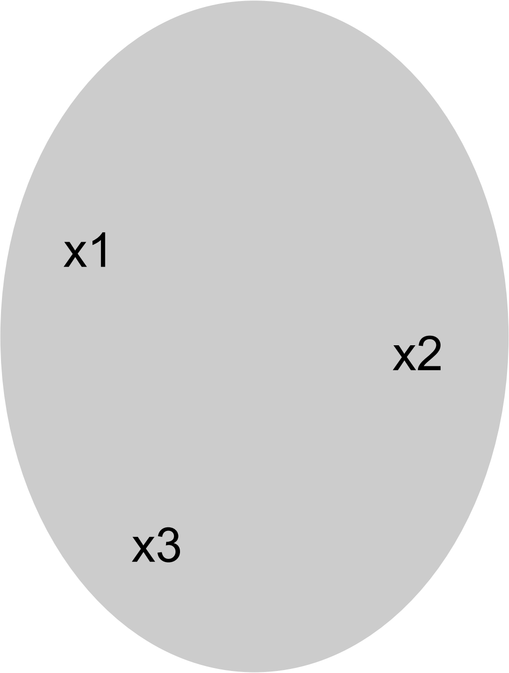
Property-Based Testing

(QuickCheck)

Unit Tests



"Program testing can be used to show the presence of bugs, but never to show their absence!"

Unit Tests

- Usually, only a tiny part of the domain is tested
- Are you sure x1, x2, and x3 the right choices ?

What is Property-Based Testing ?

tl;dr: Put the machine to work !

A way to:

- specify a domain to pick from *at random*
- express *properties* of programs
- run *arbitrary* numbers of tests

Example

1. Define how to generate test data

   ```
   X : integer, L : list(integer)
   # instead of: X = 2, L = [1,2,3]
   ```

2. Define a property that should hold

   ```
   prop : not_member(X, delete(X, L))
   # instead of: 2 in delete(2, [1,2,3])
   ```

3. Run as many tests as you wish

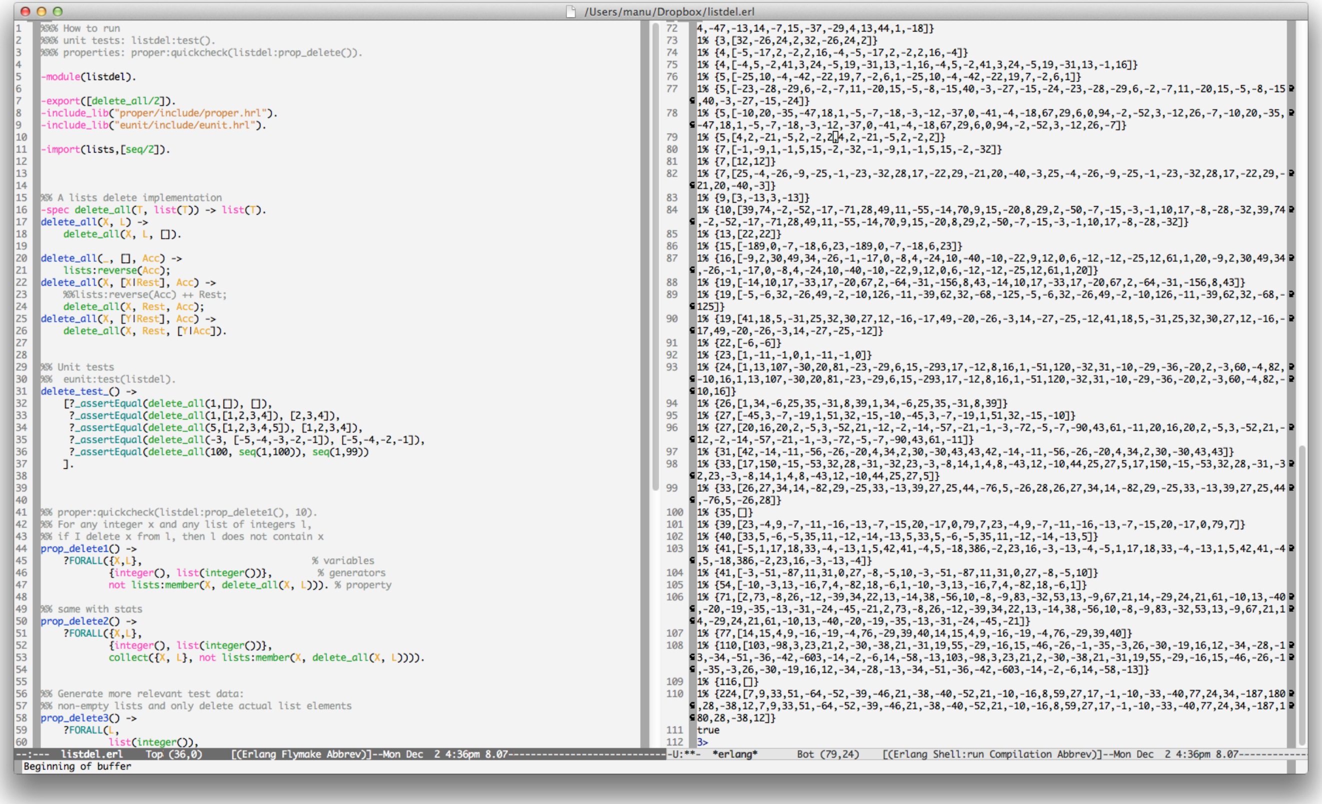
   ```
   check(prop, 100000)
   ```

Shrinking

When a test case fails, QuickCheck will search for a simpler but similar test case that also fails

Useful for debugging !

Demo



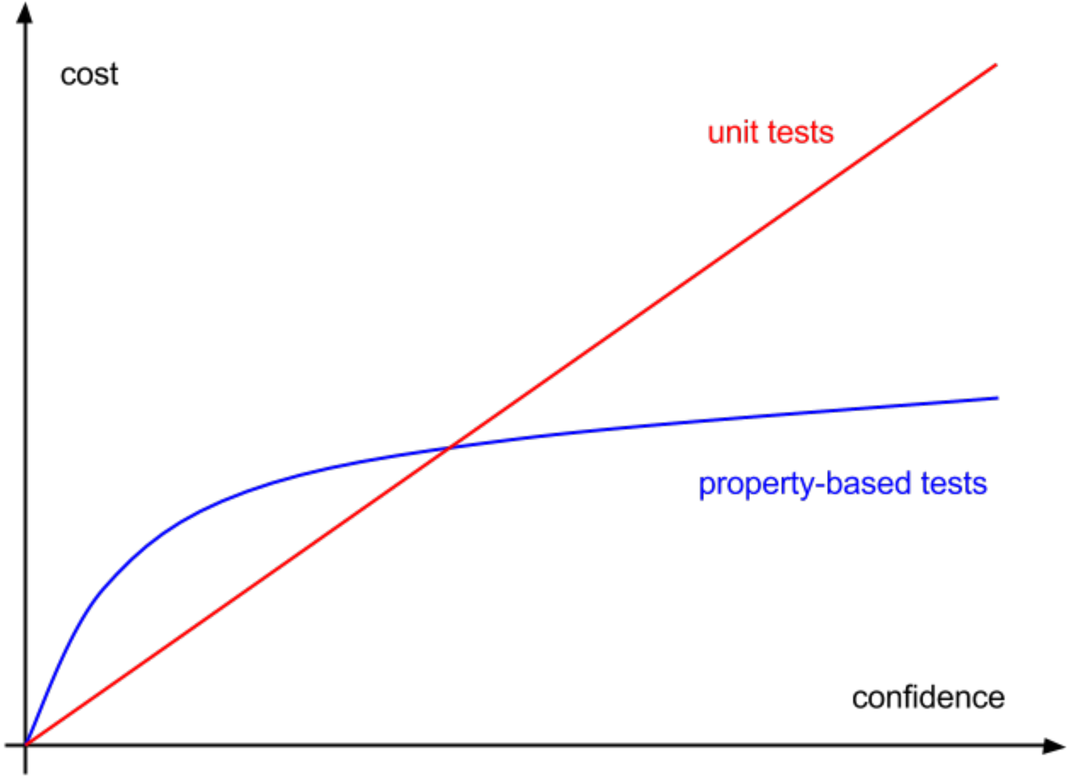
```erlang
%%% How to run
%%% unit tests: listdel:test().
%%% properties: proper:quickcheck(listdel:prop_delete()).

-module(listdel).

-export([delete_all/2]).
-include_lib("proper/include/proper.hrl").
-include_lib("eunit/include/eunit.hrl").

-import(lists,[seq/2]).



%% A lists delete implementation
-spec delete_all(T, list(T)) -> list(T).
delete_all(X, L) ->
    delete_all(X, L, []).

delete_all(_, [], Acc) ->
    lists:reverse(Acc);
delete_all(X, [X|Rest], Acc) ->
    %%lists:reverse(Acc) ++ Rest;
    delete_all(X, Rest, Acc);
delete_all(X, [Y|Rest], Acc) ->
    delete_all(X, Rest, [Y|Acc]).


%% Unit tests
%%  eunit:test(listdel).
delete_test_() ->
    [?_assertEqual(delete_all(1,[]), []),
     ?_assertEqual(delete_all(1,[1,2,3,4]), [2,3,4]),
     ?_assertEqual(delete_all(5,[1,2,3,4,5]), [1,2,3,4]),
     ?_assertEqual(delete_all(-3, [-5,-4,-3,-2,-1]), [-5,-4,-2,-1]),
     ?_assertEqual(delete_all(100, seq(1,100)), seq(1,99))
    ].



%% proper:quickcheck(listdel:prop_delete1(), 10).
%% For any integer x and any list of integers l,
%% if I delete x from l, then l does not contain x
prop_delete1() ->
    ?FORALL({X,L},                                  % variables
            {integer(), list(integer())},            % generators
            not lists:member(X, delete_all(X, L))). % property

%% same with stats
prop_delete2() ->
    ?FORALL({X,L},
            {integer(), list(integer())},
            collect({X, L}, not lists:member(X, delete_all(X, L)))).


%% Generate more relevant test data:
%% non-empty lists and only delete actual list elements
prop_delete3() ->
    ?FORALL(L,
            list(integer()),
```

```
--:---  listdel.erl    Top (36,0)    [(Erlang Flymake Abbrev)]--Mon Dec  2 4:36pm 8.07----
Beginning of buffer
```

```
4,-47,-13,14,-7,15,-37,-29,4,13,44,1,-18]}
1% {3,[32,-26,24,2,32,-26,24,2]}
1% {4,[-5,-17,2,-2,2,16,-4,-5,-17,2,-2,2,16,-4]}
1% {4,[-4,5,-2,41,3,24,-5,19,-31,13,-1,16,-4,5,-2,41,3,24,-5,19,-31,13,-1,16]}
1% {5,[-25,10,-4,-42,-22,19,7,-2,6,1,-25,10,-4,-42,-22,19,7,-2,6,1]}
1% {5,[-23,-28,-29,6,-2,-7,11,-20,15,-5,-8,-15,40,-3,-27,-15,-24,-23,-28,-29,6,-2,-7,11,-20,15,-5,-8,-15
,40,-3,-27,-15,-24]}
1% {5,[-10,20,-35,-47,18,1,-5,-7,-18,-3,-12,-37,0,-41,-4,-18,67,29,6,0,94,-2,-52,3,-12,26,-7,-10,20,-35,
-47,18,1,-5,-7,-18,-3,-12,-37,0,-41,-4,-18,67,29,6,0,94,-2,-52,3,-12,26,-7]}
1% {5,[4,2,-21,-5,2,-2,2,4,2,-21,-5,2,-2,2]}
1% {7,[-1,-9,1,-1,5,15,-2,-32,-1,-9,1,-1,5,15,-2,-32]}
1% {7,[12,12]}
1% {7,[25,-4,-26,-9,-25,-1,-23,-32,28,17,-22,29,-21,20,-40,-3,25,-4,-26,-9,-25,-1,-23,-32,28,17,-22,29,-
21,20,-40,-3]}
1% {9,[3,-13,3,-13]}
1% {10,[39,74,-2,-52,-17,-71,28,49,11,-55,-14,70,9,15,-20,8,29,2,-50,-7,-15,-3,-1,10,17,-8,-28,-32,39,74
,-2,-52,-17,-71,28,49,11,-55,-14,70,9,15,-20,8,29,2,-50,-7,-15,-3,-1,10,17,-8,-28,-32]}
1% {13,[22,22]}
1% {15,[-189,0,-7,-18,6,23,-189,0,-7,-18,6,23]}
1% {16,[-9,2,30,49,34,-26,-1,-17,0,-8,4,-24,10,-40,-10,-22,9,12,0,6,-12,-12,-25,12,61,1,20,-9,2,30,49,34
,-26,-1,-17,0,-8,4,-24,10,-40,-10,-22,9,12,0,6,-12,-12,-25,12,61,1,20]}
1% {19,[-14,10,17,-33,17,-20,67,2,-64,-31,-156,8,43,-14,10,17,-33,17,-20,67,2,-64,-31,-156,8,43]}
1% {19,[-5,-6,32,-26,49,-2,-10,126,-11,-39,62,32,-68,-125,-5,-6,32,-26,49,-2,-10,126,-11,-39,62,32,-68,-
125]}
1% {19,[41,18,5,-31,25,32,30,27,12,-16,-17,49,-20,-26,-3,14,-27,-25,-12,41,18,5,-31,25,32,30,27,12,-16,-
17,49,-20,-26,-3,14,-27,-25,-12]}
1% {22,[-6,-6]}
1% {23,[1,-11,-1,0,1,-11,-1,0]}
1% {24,[1,13,107,-30,20,81,-23,-29,6,15,-293,17,-12,8,16,1,-51,120,-32,31,-10,-29,-36,-20,2,-3,60,-4,82,
-10,16,1,13,107,-30,20,81,-23,-29,6,15,-293,17,-12,8,16,1,-51,120,-32,31,-10,-29,-36,-20,2,-3,60,-4,82,-
10,16]}
1% {26,[1,34,-6,25,35,-31,8,39,1,34,-6,25,35,-31,8,39]}
1% {27,[-45,3,-7,-19,1,51,32,-15,-10,-45,3,-7,-19,1,51,32,-15,-10]}
1% {27,[20,16,20,2,-5,3,-52,21,-12,-2,-14,-57,-21,-1,-3,-72,-5,-7,-90,43,61,-11,20,16,20,2,-5,3,-52,21,-
12,-2,-14,-57,-21,-1,-3,-72,-5,-7,-90,43,61,-11]}
1% {31,[42,-14,-11,-56,-26,-20,4,34,2,30,-30,43,43,42,-14,-11,-56,-26,-20,4,34,2,30,-30,43,43]}
1% {33,[17,150,-15,-53,32,28,-31,-32,23,-3,-8,14,1,4,8,-43,12,-10,44,25,27,5,17,150,-15,-53,32,28,-31,-3
2,23,-3,-8,14,1,4,8,-43,12,-10,44,25,27,5]}
1% {33,[26,27,34,14,-82,29,-25,33,-13,39,27,25,44,-76,5,-26,28,26,27,34,14,-82,29,-25,33,-13,39,27,25,44
,-76,5,-26,28]}
1% {35,[]}
1% {39,[23,-4,9,-7,-11,-16,-13,-7,-15,20,-17,0,79,7,23,-4,9,-7,-11,-16,-13,-7,-15,20,-17,0,79,7]}
1% {40,[33,5,-6,-5,35,11,-12,-14,-13,5,33,5,-6,-5,35,11,-12,-14,-13,5]}
1% {41,[-5,1,17,18,33,-4,-13,1,5,42,41,-4,5,-18,386,-2,23,16,-3,-13,-4,-5,1,17,18,33,-4,-13,1,5,42,41,-4
,5,-18,386,-2,23,16,-3,-13,-4]}
1% {41,[-3,-51,-87,11,31,0,27,-8,-5,10,-3,-51,-87,11,31,0,27,-8,-5,10]}
1% {54,[-10,-3,13,-16,7,4,-82,18,-6,1,-10,-3,13,-16,7,4,-82,18,-6,1]}
1% {71,[2,73,-8,26,-12,-39,34,22,13,-14,38,-56,10,-8,-9,83,-32,53,13,-9,67,21,14,-29,24,21,61,-10,13,-40
,-20,-19,-35,-13,-31,-24,-45,-21,2,73,-8,26,-12,-39,34,22,13,-14,38,-56,10,-8,-9,83,-32,53,13,-9,67,21,1
4,-29,24,21,61,-10,13,-40,-20,-19,-35,-13,-31,-24,-45,-21]}
1% {77,[14,15,4,9,-16,-19,-4,76,-29,39,40,14,15,4,9,-16,-19,-4,76,-29,39,40]}
1% {110,[103,-98,3,23,21,2,-30,-38,21,-31,19,55,-29,-16,15,-46,-26,-1,-35,-3,26,-30,-19,16,12,-34,-28,-1
3,-34,-51,-36,-42,-603,-14,-2,-6,14,-58,-13,103,-98,3,23,21,2,-30,-38,21,-31,19,55,-29,-16,15,-46,-26,-1
,-35,-3,26,-30,-19,16,12,-34,-28,-13,-34,-51,-36,-42,-603,-14,-2,-6,14,-58,-13]}
1% {116,[]}
1% {224,[7,9,33,51,-64,-52,-39,-46,21,-38,-40,-52,21,-10,-16,8,59,27,17,-1,-10,-33,-40,77,24,34,-187,180
,28,-38,12,7,9,33,51,-64,-52,-39,-46,21,-38,-40,-52,21,-10,-16,8,59,27,17,-1,-10,-33,-40,77,24,34,-187,1
80,28,-38,12]}
true
3>
```

```
-U:**-  *erlang*    Bot (79,24)    [(Erlang Shell:run Compilation Abbrev)]--Mon Dec  2 4:36pm 8.07-----------
```

Why ?



The Economy of Testing

The Catch ?

- We test against a specification: sometimes it's the specification that is wrong !
- It can be difficult to come up with "good" properties and they can be tricky to express
- How relevant is the randomly generated test data ?
  (use stats to ensure a good test case distribution !)

Commercial Users

- Ericsson
- Motorola
- Volvo
- Basho (Riak)

From QuviQ’s website: http://www.quviq.com/

The End...

Find a clone in your favorite language…

C, C++, Chicken Scheme, Clojure, Common Lisp, D, F#, Factor, Haskell, Io, Java, JavaScript, Node.js, Obj-C, OCaml, Perl, Python, Ruby, Scala, Scheme, Smalltalk, SML.

http://en.wikipedia.org/wiki/QuickCheck