



Testing OTP Libraries with QuickCheck Mini

John Hughes

Testing Data Structure Libraries



- **dict**—purely functional key-value store
 - `new()`—create empty dict
 - `store(Key,Val,Dict)`
 - `fetch(Key,Dict)`
 - ...
- Complex representation... just test the API
 - “black box testing”
 - In contrast to testing e.g. dict invariants

A Simple Property



- Perhaps the keys are unique...

```
prop_unique_keys() ->  
  ?FORALL(D,dict(),  
    no_duplicates(dict:fetch_keys(D))).
```

- Cannot test this without a *generator for dicts*

Generating dicts



- Black box testing → *use the API to create dicts*

```
dict() ->  
  ?LAZY(  
    oneof([dict:new(),  
          ?LET({K,V,D},{key(),value(),dict()} ,  
              dict:store(K,V,D))])
```

Ac

)

```
oneof([int(),real(),atom()]).
```

recursion!

*Constant
time*

>

Symbolic Test Cases



```
dict() ->
  ?LAZY(oneof([
    {call,dict,new,[]},
    {call,dict,store,[key(),value(),dict()]}
  ])).
```

- $\{call, M, F, Args\}$ *represents* a function call $M:F(...Args...)$ symbolically.

```
prop_unique_keys() ->
  ?FORALL(D,dict(),
    no_duplicates(dict:fetch_keys(eval(D)))).
```

Test case is now *symbolic*

- $eval(X)$ *evaluates* symbolic calls in a term X

Let's test again!



```
Erlang
File Edit Options View Help
[Icons: Print, Save, Bold, Help]
[a,-1.0,{call,dict,new,[],}]}}}}}}}}}}
Shrinking.....(9 times)
{call,dict,store,
 [0,0.0,
  {call,dict,store,
   [0.0,a,
    {call,dict,store,
     [0.0,a,
      {call,dict,store,
       [0.0,0,
        {call,dict,store,
         [0,a,
          {call,dict,store,
           [0.0,0,
            {call,dict,store,
             [a,0.0,{call,dict,new,[],}]}}}}}}}}}}
false
14>
```

Shrinks the *values*, but not the *structure*!

dict() with Shrinking

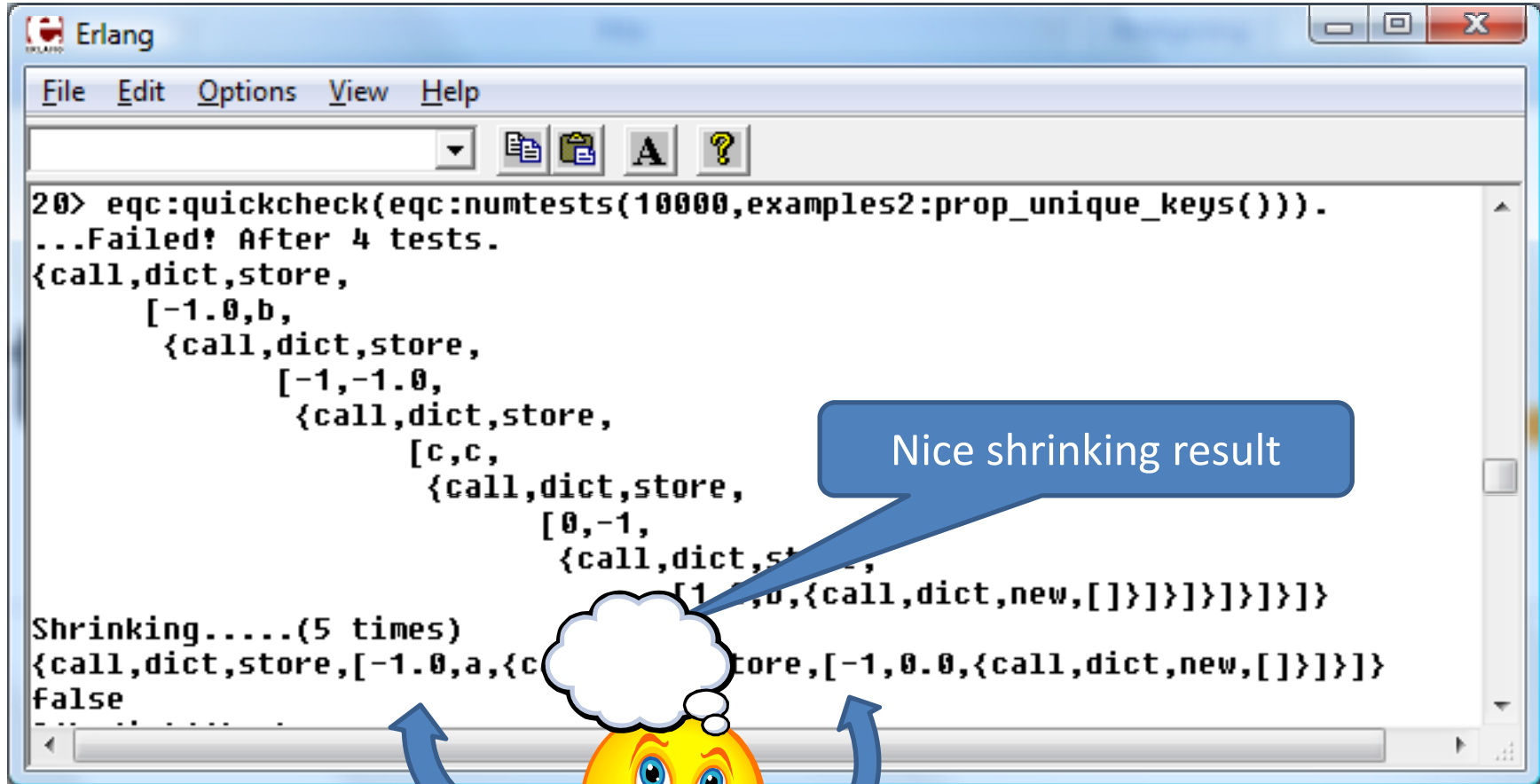


```
dict() ->
  ?LAZY(oneof(
    [{call,dict,new,[]},
      ?LETSHRINK([D],[dict()],
        {call,dict,store,[key(),value(),D]} ) ] ) ).
```

- Like ?LET, binds D to result of dict()
- Can *shrink* to D
- Can bind any number of variables:
 - ?LETSHRINK([X1,X2,...],[G1,G2...],Result)

Let's test and shrink!

Q...



```
20> eqc:quickcheck(eqc:numtests(10000,examples2:prop_unique_keys())).
...Failed! After 4 tests.
{call,dict,store,
  [-1.0,b,
   {call,dict,store,
     [-1,-1.0,
      {call,dict,store,
        [c,c,
         {call,dict,store,
           [0,-1,
            {call,dict,store,
              [1,0,{call,dict,new,[]}]}]}]}]}]}]}
Shrinking.....(5 times)
{call,dict,store,[-1.0,a,{call,dict,store,[-1,0.0,{call,dict,new,[]}]}]}
false
```

Nice shrinking result




-1 and -1.0, eh?

Measuring statistics




```
Erlang
File Edit Options View Help
OK, passed 10000 tests
49% 0
25% 1
12% 2
6% 3
3% 4
1% 5
0% 6
0% 7
0% 8
0% 9
0% 10
0% 11
true
26>
```

A sad orange emoji with a blue arrow pointing to the '25% 1' line in the terminal output.

dict() with Frequencies



```
Erlang
File Edit Options View Help
OK, passed 10000 tests
19% 0
17% 1
13% 2
11% 3
8% 4
7% 5
5% 6
4% 7
3% 8
2% 9
1% 10
1% 11
0% 12
0% 13
```

A yellow thinking face emoji with a hand on its chin, positioned in the center of the terminal window.

Testing vs. an Abstract Model



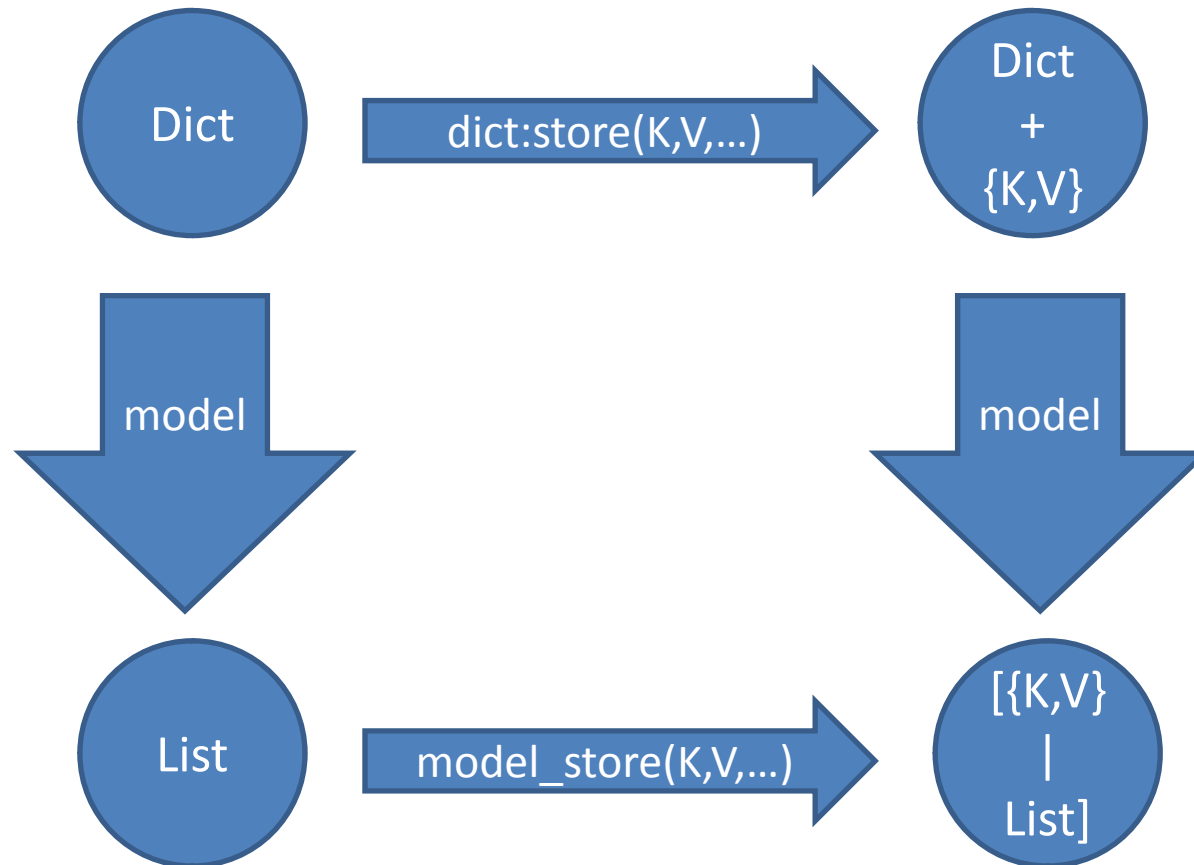
- How *should* `dict` operations behave?
 - The “same” as a list of key-value pairs
- Make comparisons in the “model” world

```
model(Dict) ->  
    dict:to_list(Dict).
```

Returns list of key-
value pairs



Commuting Diagrams



Hoare: *Proof of Correctness of Data Representations*, 1972

Testing store



```
prop_store() ->
  ?FORALL({K,V,D},
          {key(),value(),dict()} ,
  begin
    Dict = eval(D),
    model(dict:store(K,V,Dict)) ==
      model_store(K,V,model(Dict))
```

The screenshot shows an Erlang shell window with the following content:

```
29> eqc:quickcheck(eqc:numtests(10000,examples2:prop_store())).
..Failed! After 3 tests.
{0,0.0,{call,dict,store,[0,0.0,{call,dict,new,[]}]}}
false
30> █
```

Debugging properties



- *Why* is a property false?
 - We need more information!

```
Erlang
File Edit Options View Help
31> eqc:quickcheck(eqc:numtests(10000,examples2:prop_store())).
Failed! After 1 tests.
{a,0,
 {call,dict,store,
  [a,undefined,
   {call,dict,store,
    [0,b,{call,dict,store,[undefined,0,{call,dict,new,[]}]}]}]}]}
[{0,b},{a,0},{undefined,0}] /= [{a,0},{0,b},{a,undefined},{undefined,0}]
Shrinking...(3 times)
{a,0,{call,dict,store,[a,a,{call,dict,new,[]}]}]}
[{a,0}] /= [{a,0},{a,a}]
false
32>
```

Next Steps...



- Write similar properties for *all* the dict operations
- Extend the dict generator to use *all* the dict-returning operations
 - Each property tests *many* operations
- ...and, of course, correct the specification!



Exercises

Exercises



- Download the code from:
 - http://quviq.com/examples/dict_eqc.erl
- Make sure QuickCheck Mini is installed:
 - <http://quviq.com/downloads/eqcmini.zip>
 - `code:addpatha("eqc-1.0.1/ebin")`.
 - `eqc:start()`.
- Start on the problems!