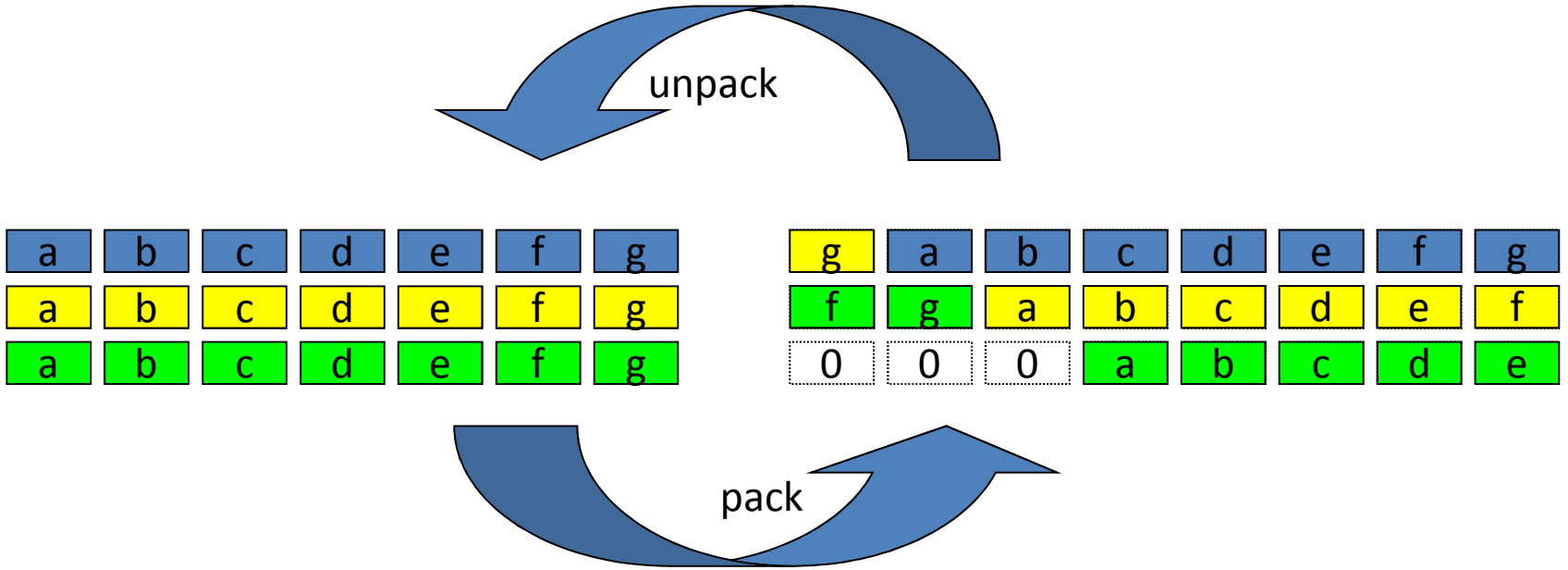
The Joy of Testing

John Hughes
Chalmers University/Quviq AB

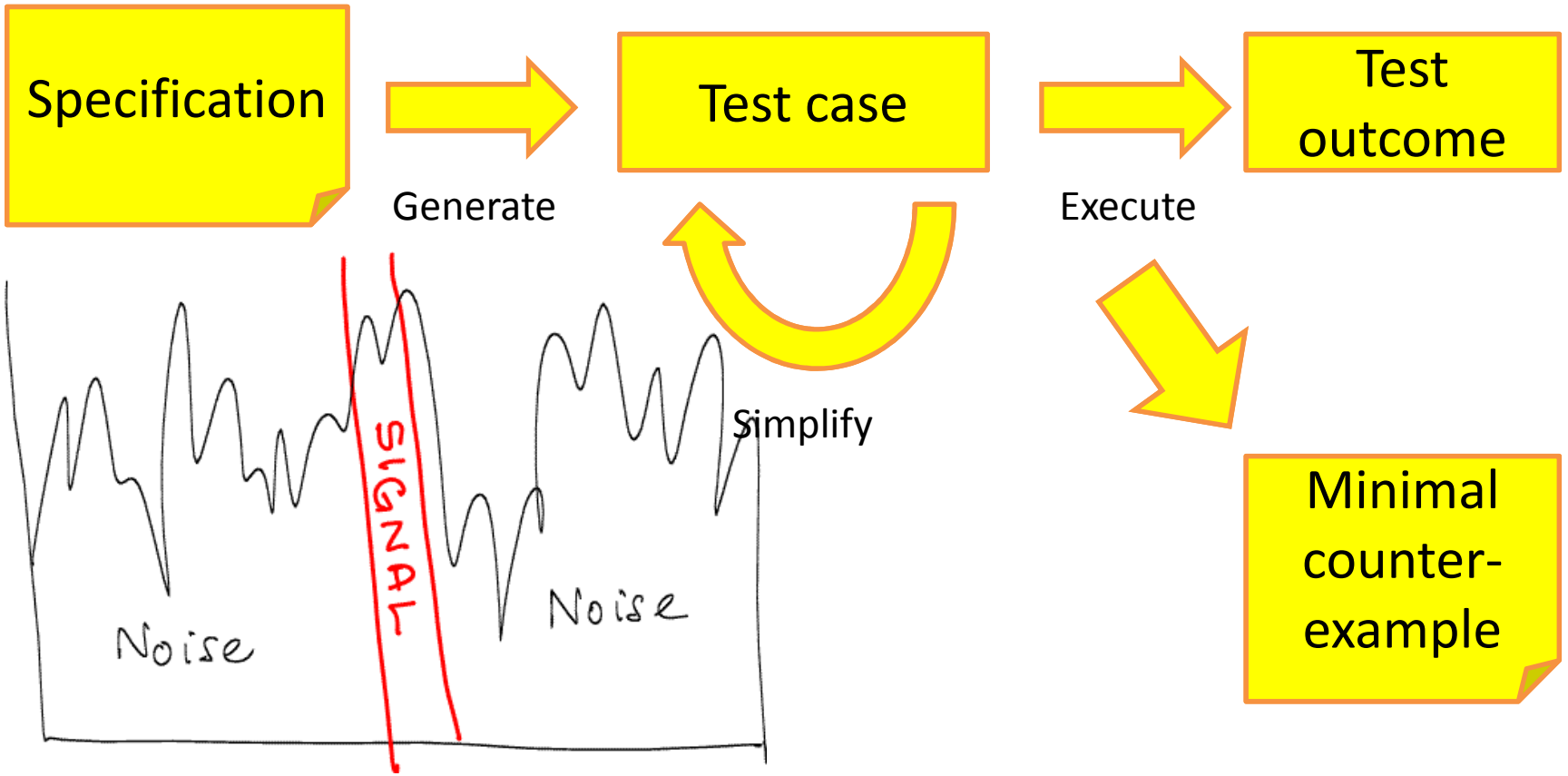
Example: Text Message Encoding

- 7-bit characters packed into 8-bit bytes



DEMO

QuickCheck in Brief



QuickCheck in Context

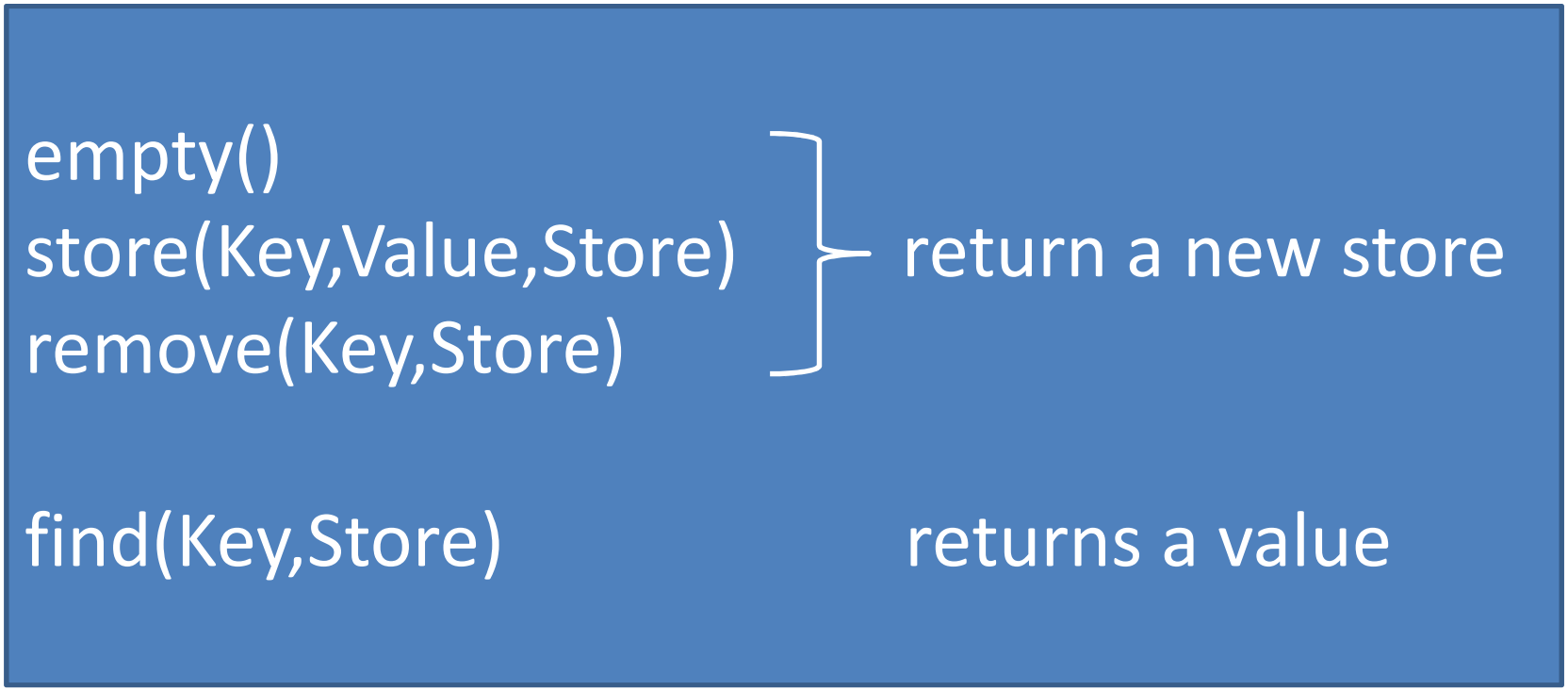
- QuickCheck in Haskell
  - Koen Claessen and yours truly, 1999
- SmallCheck
  - Enumerates small test cases, University of York
- Property Based Testing (ProTest)
  - Since 2008
- Commercial version in Erlang

Key Idea

Write properties, not test cases!

How about TDD?

- **Example:** a key-value store



EUnit Tests for Remove

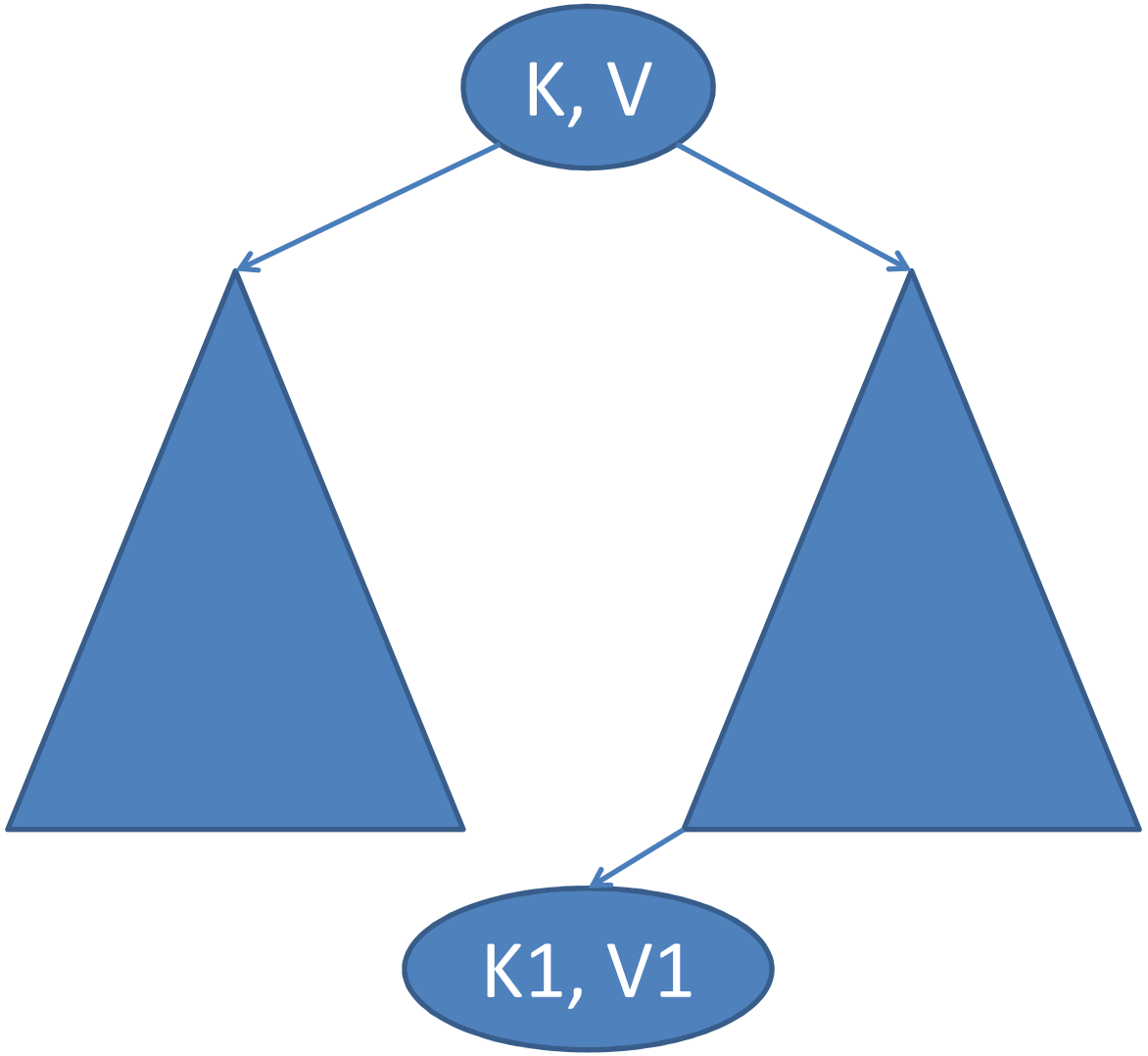
```
remove_empty_test() ->
    ?assertEqual(remove(a,empty()),
                 empty()).

remove_yields_empty_test() ->
    ?assertEqual(remove(a,store(a,1,empty())),
                 empty()).

remove_store_first_test() ->
 ?assertEqual(remove(a,store(a,1,store(b,2,empty()))),
                 store(b,2,empty)).

remove_store_second_test() ->
 ?assertEqual(remove(b,store(a,1,store(b,2,empty()))),
                 store(a,1,empty)).
```

Implementation

- Binary search trees:
  - empty
  - {node,LeftTree,Key,Value,RightTree}

```
store(K,V,empty) ->
    {node,empty,K,V,empty};
store(K,V,{node,L,K1,V1,R}) ->
    if K =<K1 ->
          {node,store(K,V,L),K1,V1,R};
       K > K1 ->
          {node,L,K1,V1,store(K,V,R)}
    end.
```

Removing a Key



Code for Remove

```
remove(K,{node,L,K1,V1,R}) ->
  if K < K1 ->{node,remove(K,L),K1,V1,R};
     K== K1 ->
       case R of
         empty -> L;
         _ -> {K2,V2} = leftmost(R),
              {node,L,K2,V2,remove(K2,R)}
         end;
     K > K1 ->{node,L,K1,V1,remove(K,R)}
  end.
```

So far so good…

- All the tests pass—what about code coverage?

```
remove(K,{node,L,K1,V1,R}) ->
  if K < K1 ->{node,remove(K,L),K1,V1,R};
     K== K1 ->
       case R of
         empty -> L;
         _ -> {K2,V2} = leftmost(R),
              {node,L,K2,V2,remove(K2,R)}
         end;
     K > K1 ->{node,L,K1,V1,remove(K,R)}
  end.
```

Just… one… more… test

- The *order* of keys matters in a binary search tree…

```
remove_store_second_reversed_keys_test() ->
  ?assertEqual(
     remove(a,store(b,2,store(a,1,empty()))),
     store(b,2,empty)).
```

- A few more tests gives 100% code coverage
  - So the code works, right?

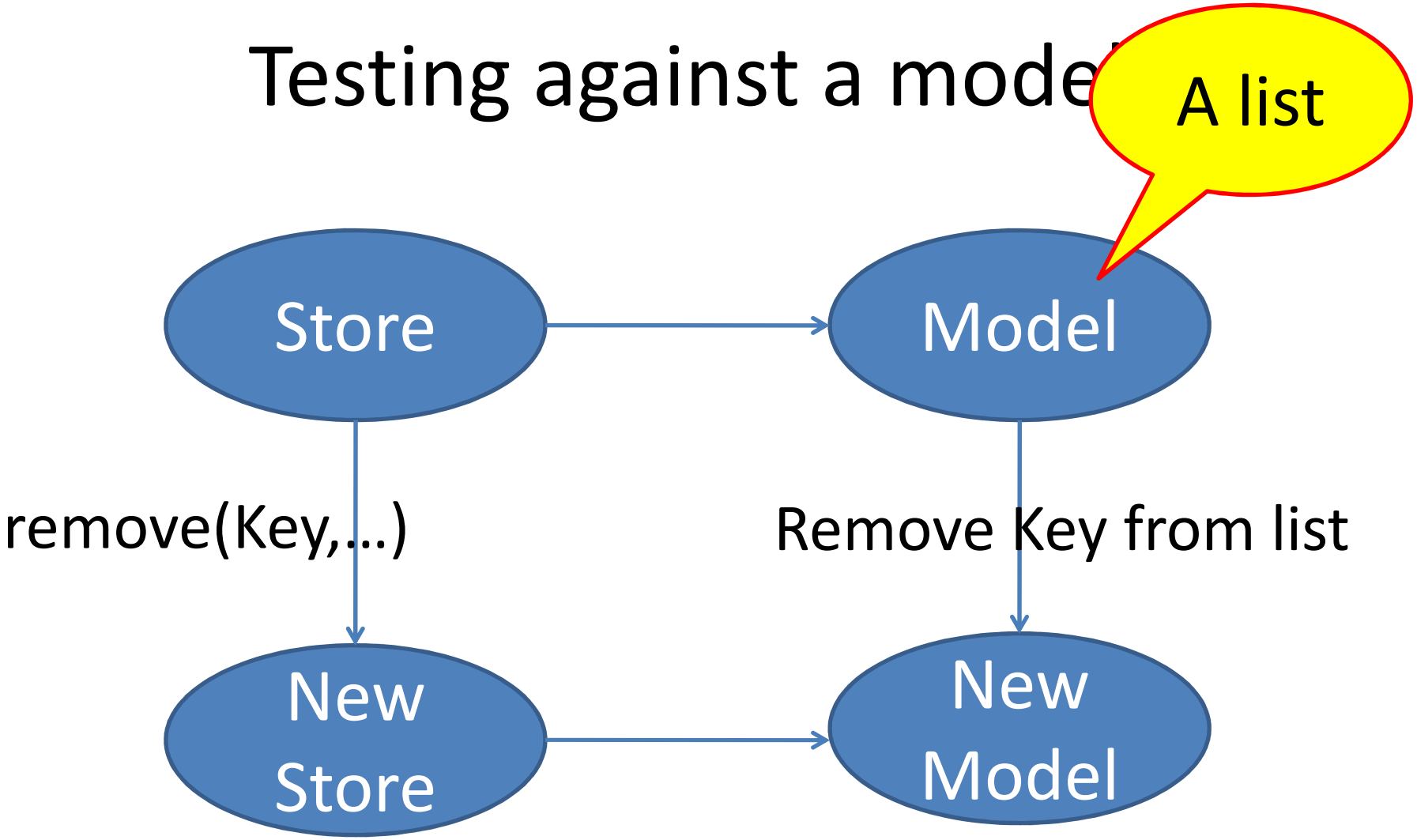
Let's write a property

- Generalise one of the unit tests

```
prop_remove_store() ->
    ?FORALL({K,V,T},{key(),val(),tree()},
                    equals(remove(K,store(K,V,T)),T)).
```

```
...Failed! After 4 tests.
{a,1,{node,empty,a,0,empty}}
{node,empty,a,1,empty} /= {node,empty,a,0,empty}
false
```

Testing against a model



Testing remove

```
prop_remove() ->
    ?FORALL({K,T},{key(),tree()},
                equals(
                    model(remove(K,T)),
                    model(T) -- [{K
```

Removing b…

…corrupted the *values* stored with c!

…from a tree containing c twice…

```
Sh  nking.....(5 t
{b,{node,emp ,b,0,{node,{node,empty,c,0,empty},c,1,empty}}}
[{c,0},{c,0}] /= [{c,0},{c,1}]
```

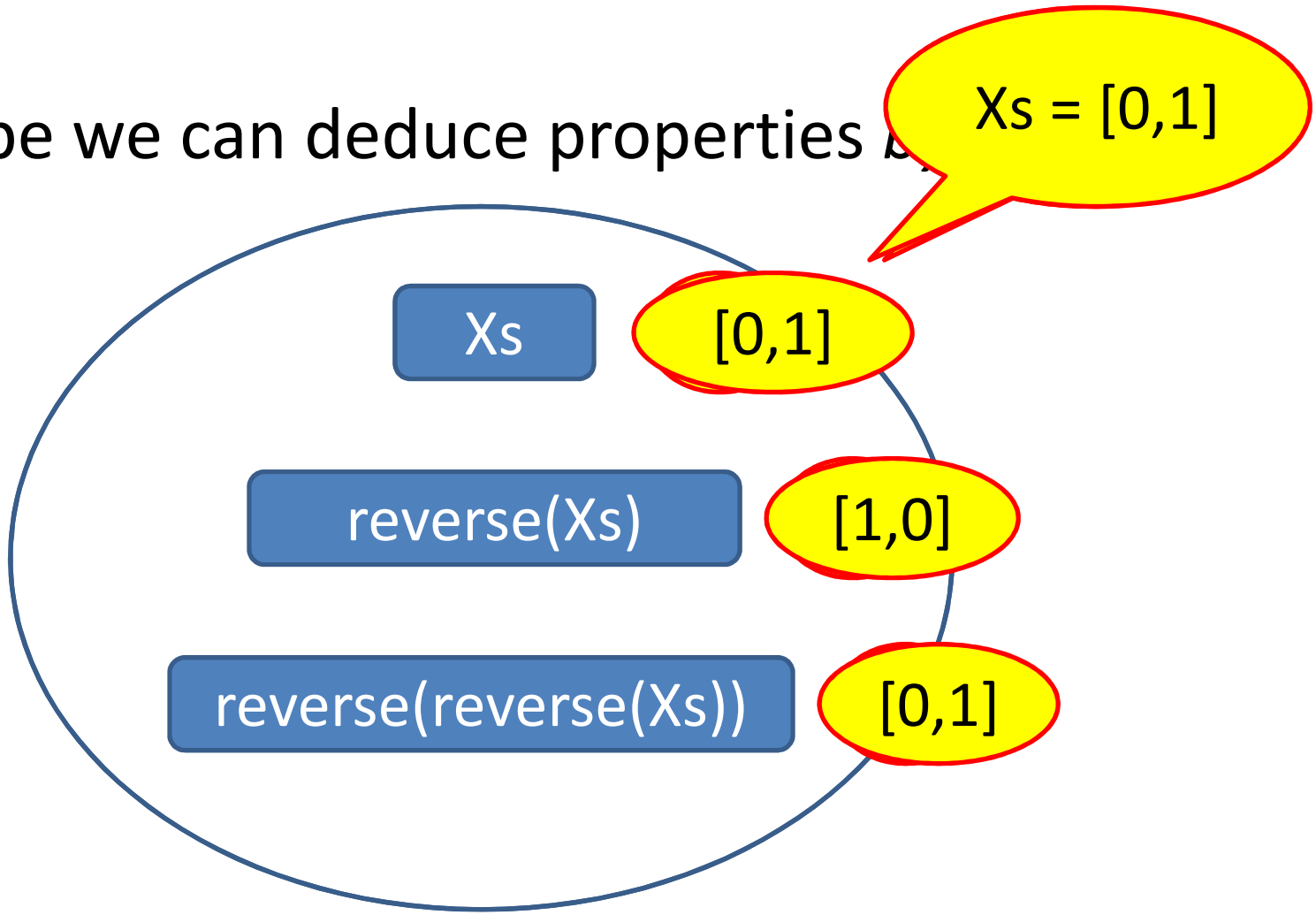
Simple Fix

- **Don't** allow duplicate keys!
  - Modify store to *replace* existing keys, rather than add another copy

- prop_remove() passes!
  - …and so do prop_store(), prop_find(), prop_empty()
  - With 100% code coverage (of course)

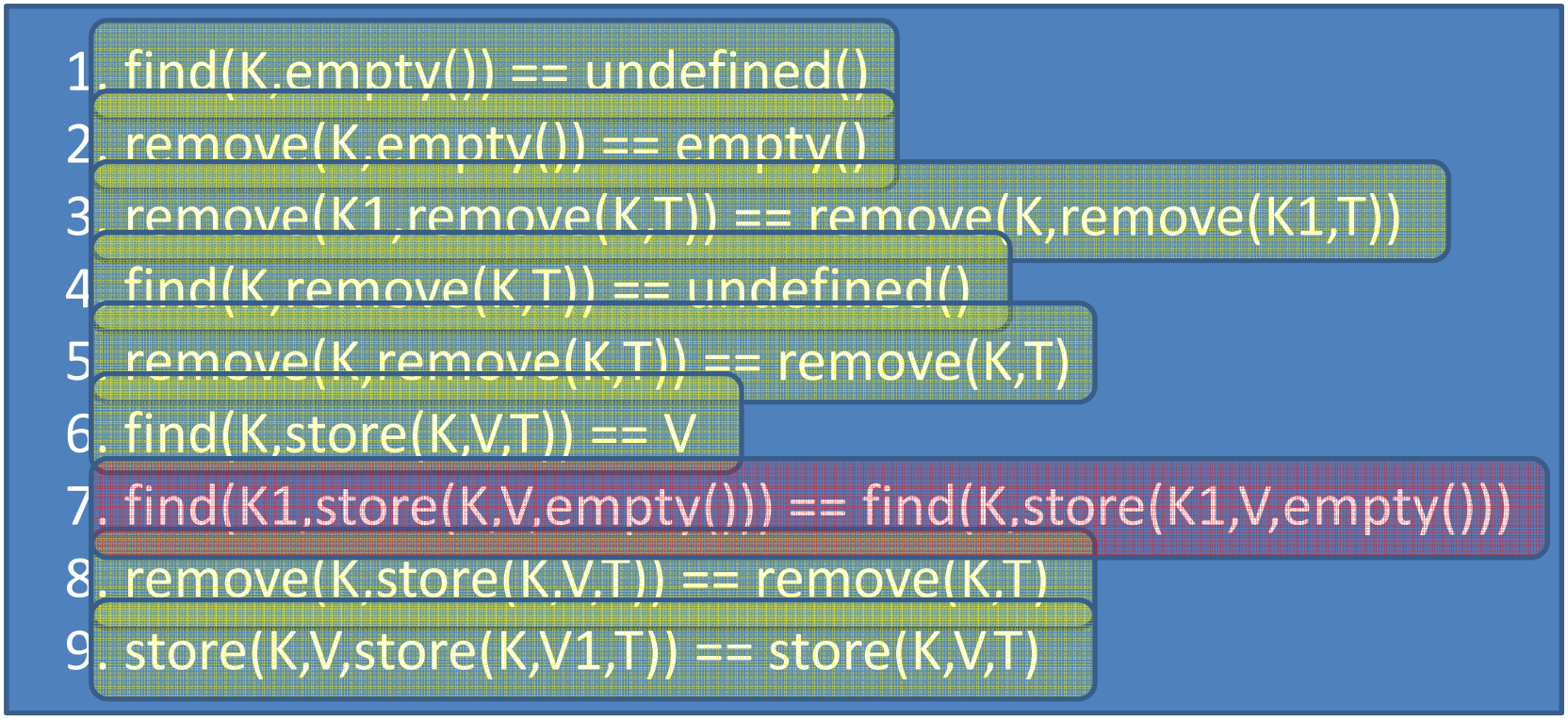
Where do properties come from?

- Maybe we can deduce properties b



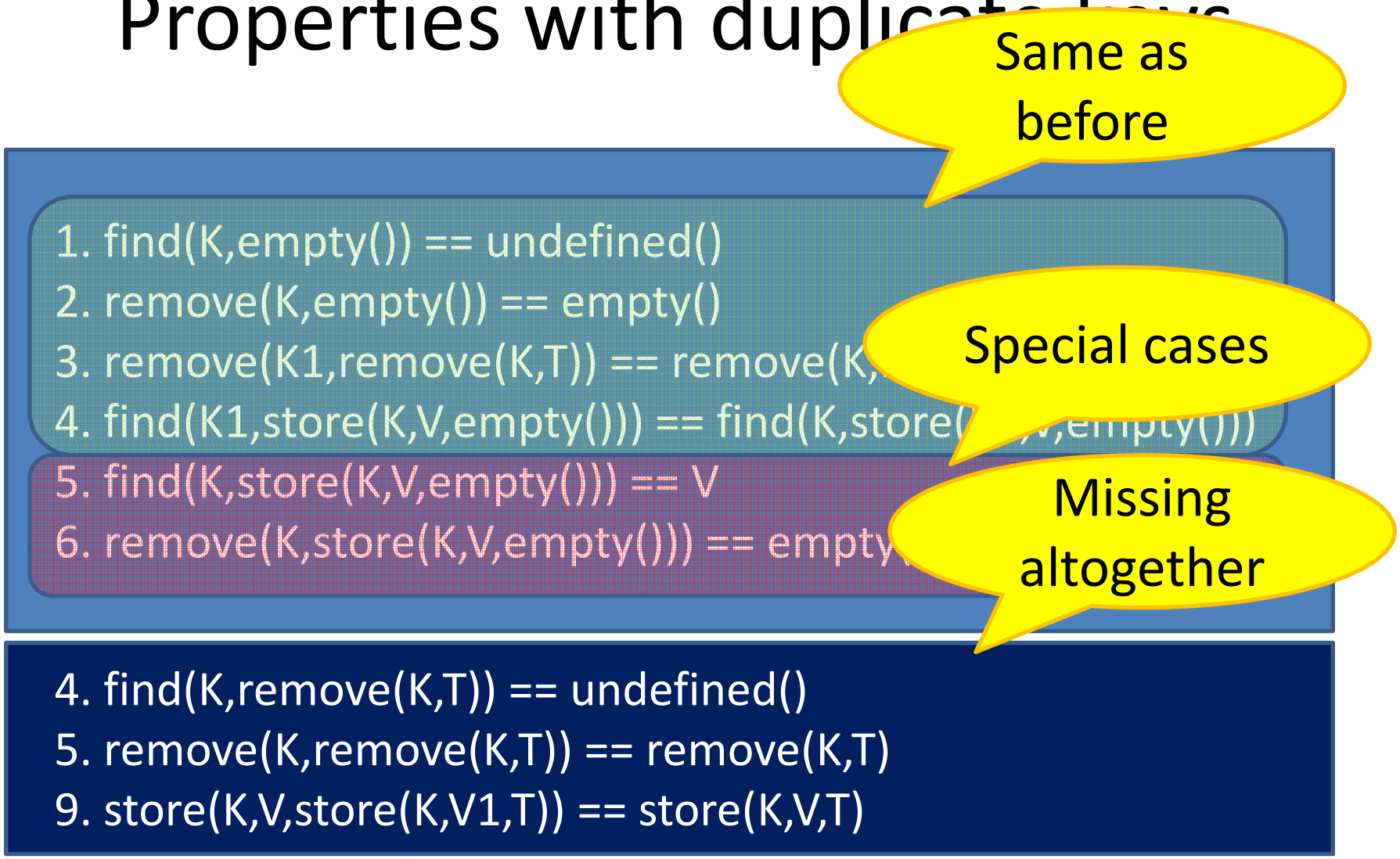
QuickSpec Demo

Properties for trees



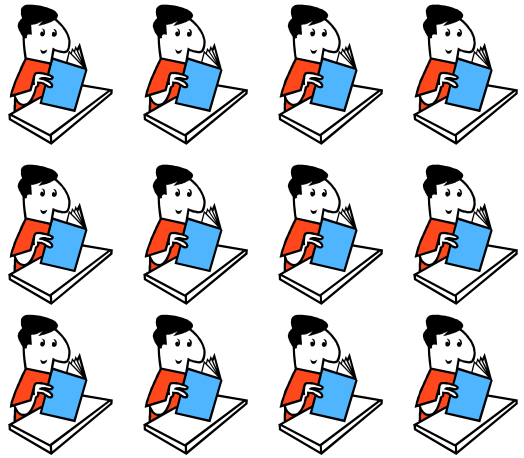
- Ready-made properties for regression testing, testing a different implementation…

Properties with duplicate keys



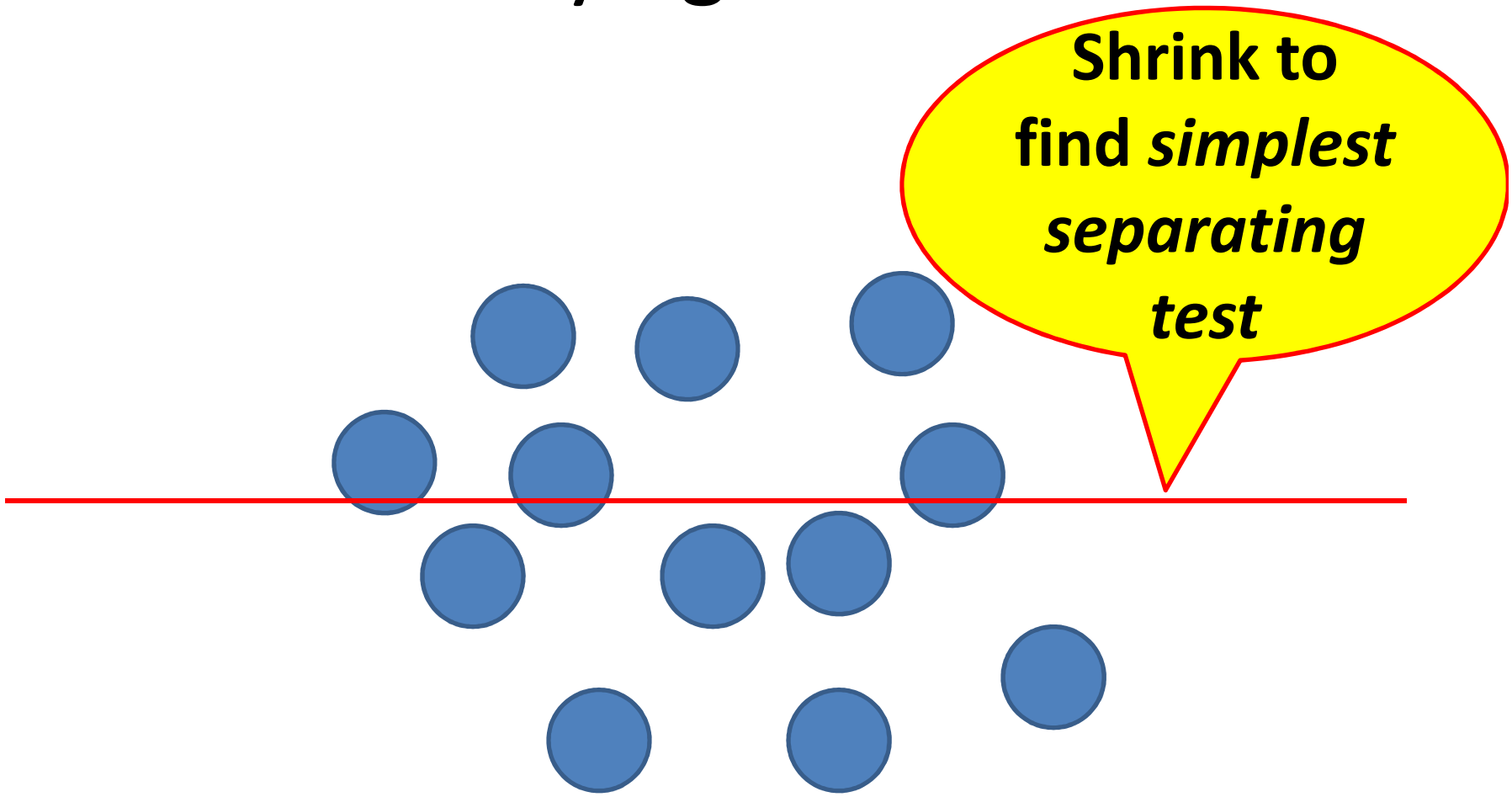
Is PDD really better than TDD?

TDD



PDD

Classifying solutions

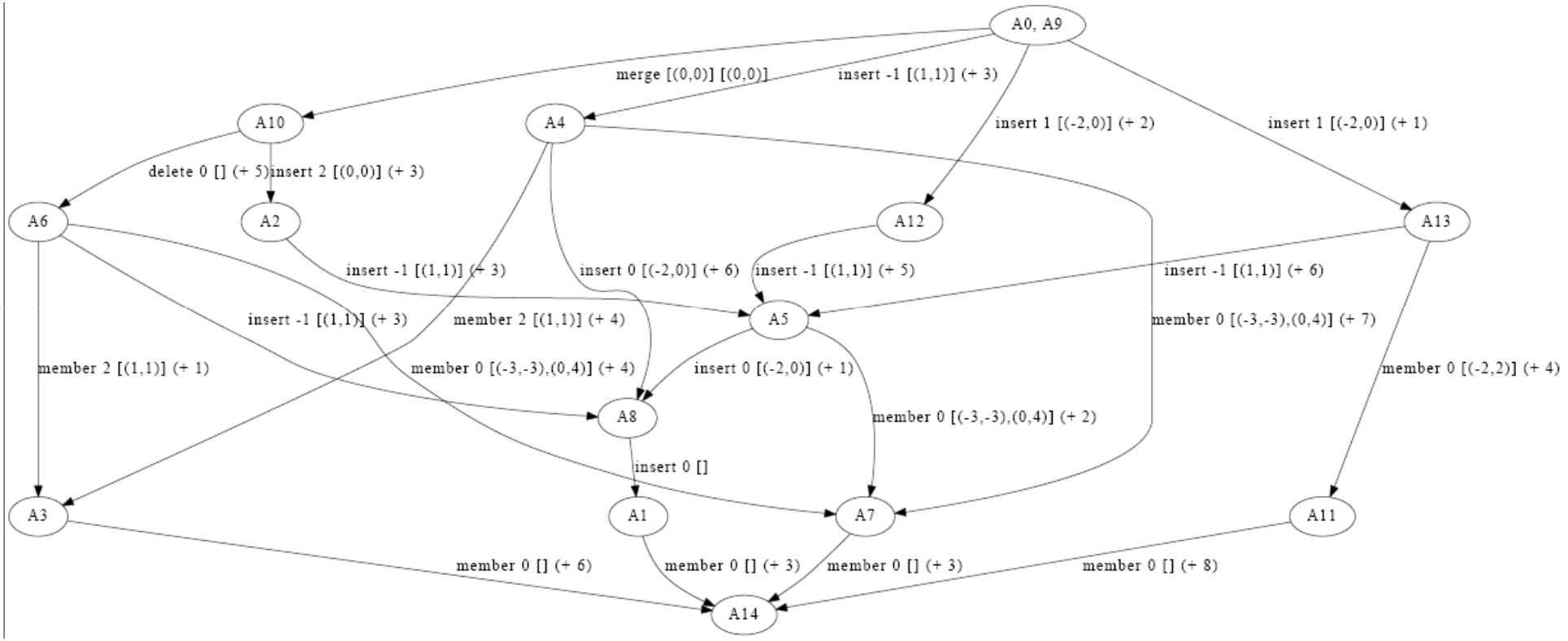


Example: Interval Sets

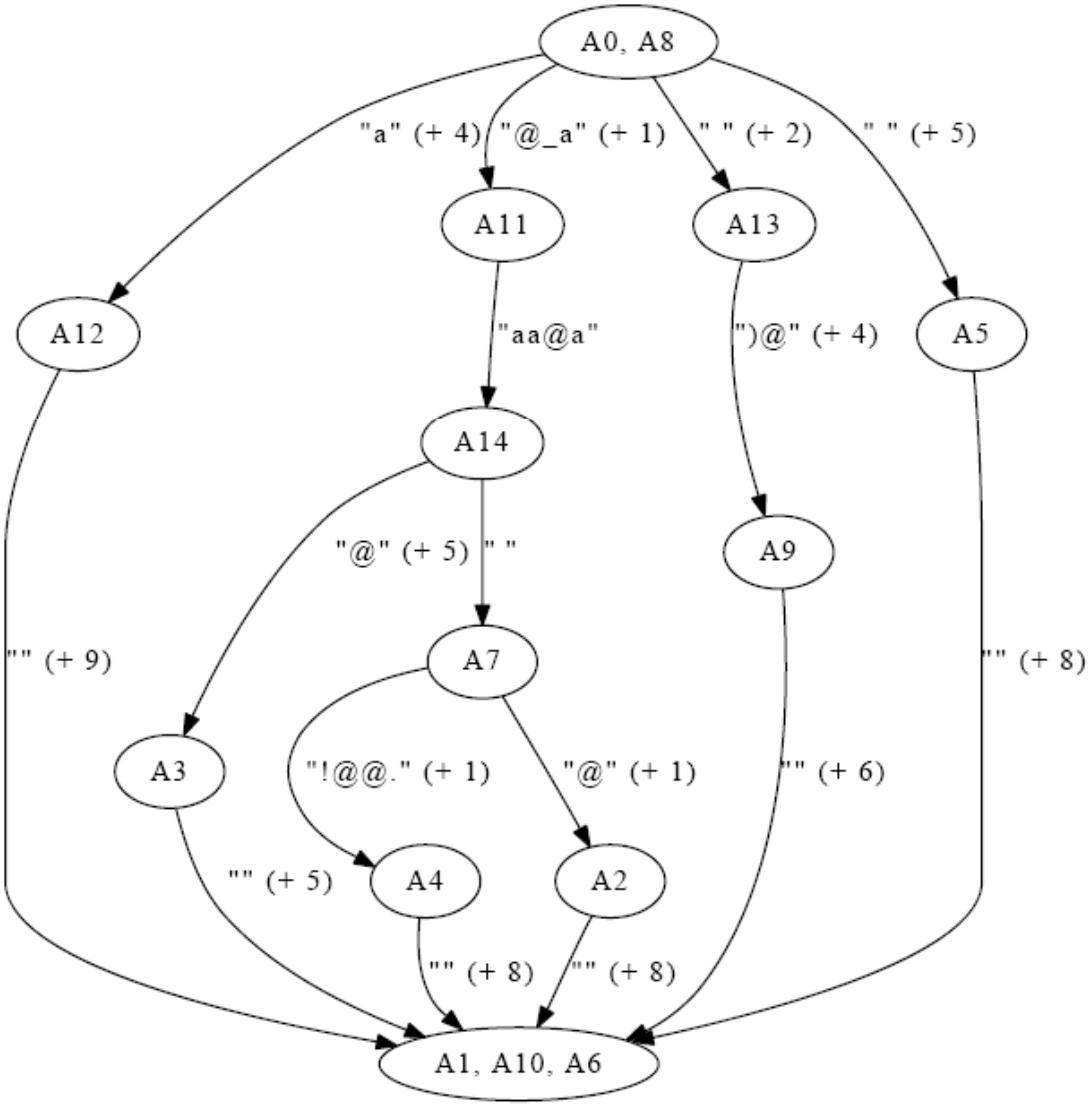
- [(1,3),(6,10)] represents [1,2,3,6,7,8,9,10]

Distinguishing tests for insert
insert 0 []
insert -1 [(1,1)]
insert 0 [(-2,0)]
insert 1 [(-2,0)]
insert 2 [(0,0)]

Ranking Interval Set Solutions



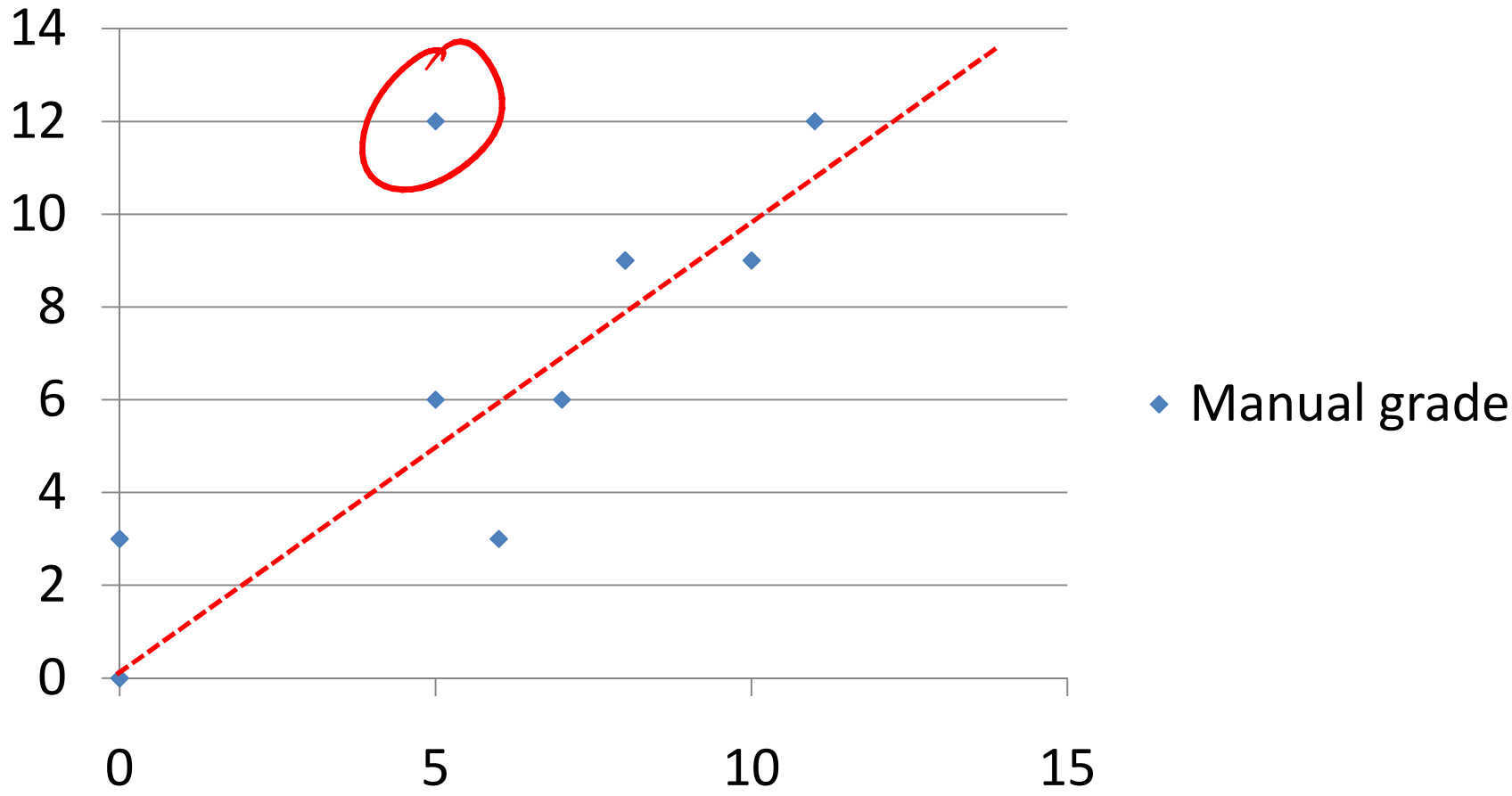
Email Anonymizer Solutions



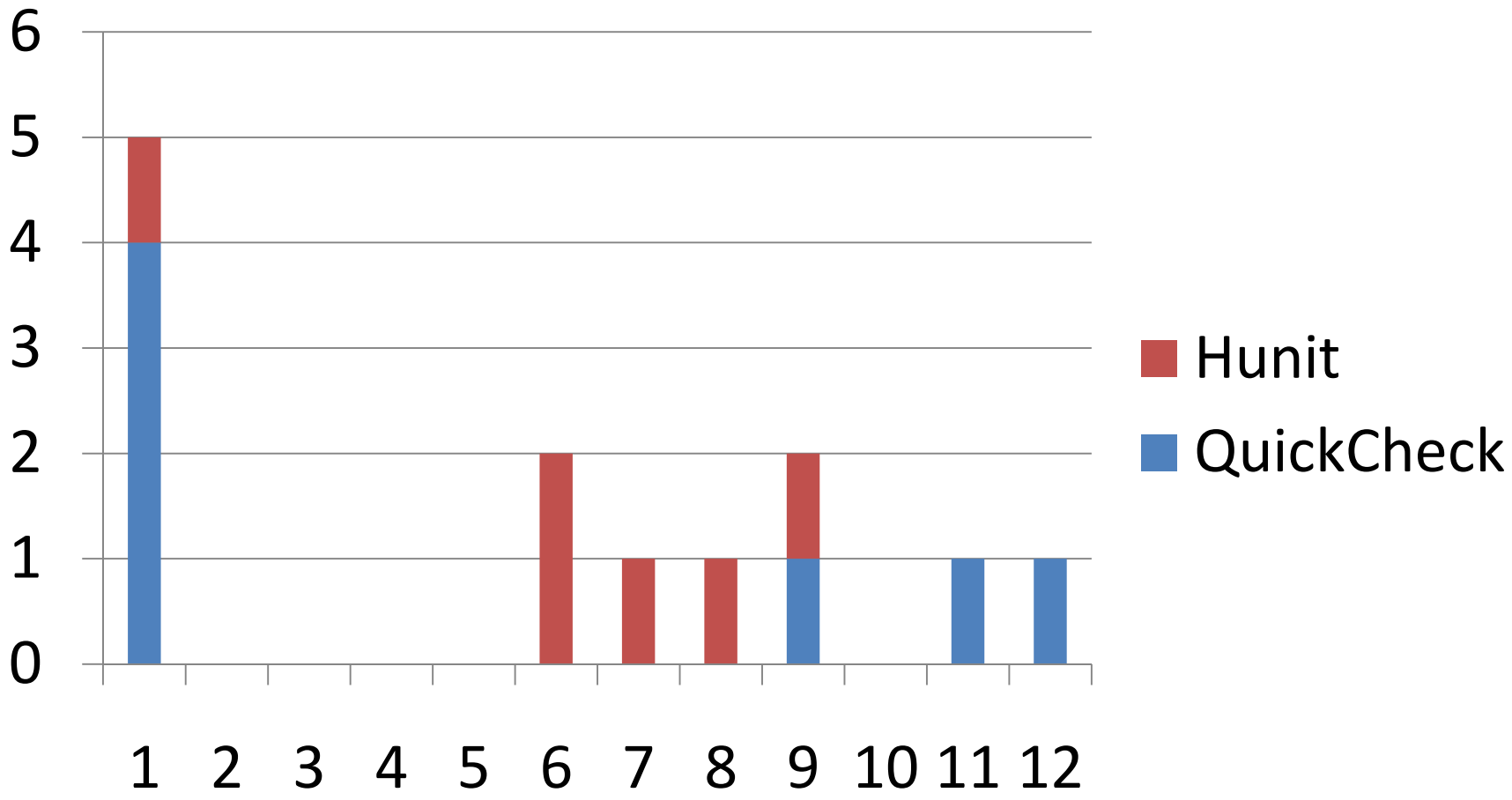
Comparing Test Suites

	Answer 1	Answer 2	Answer 3	Answer 4
Test Suite 1	✗	✓	✗	✓
Test Suite 2	✓	✗	✗	✗
Test Suite 3	✓	✗	✓	✓
Test Suite 4	✗	✓	✗	✓

Manual grading of test suite vs automated scoring



Test Suite Quality



Can we test imperative code this way?

- **YES!** (but with a more complex model)

- **Example:**
  - A circular buffer in C
  - A state machine model using a *list* to model contents

DEMO

Bug found in GCC

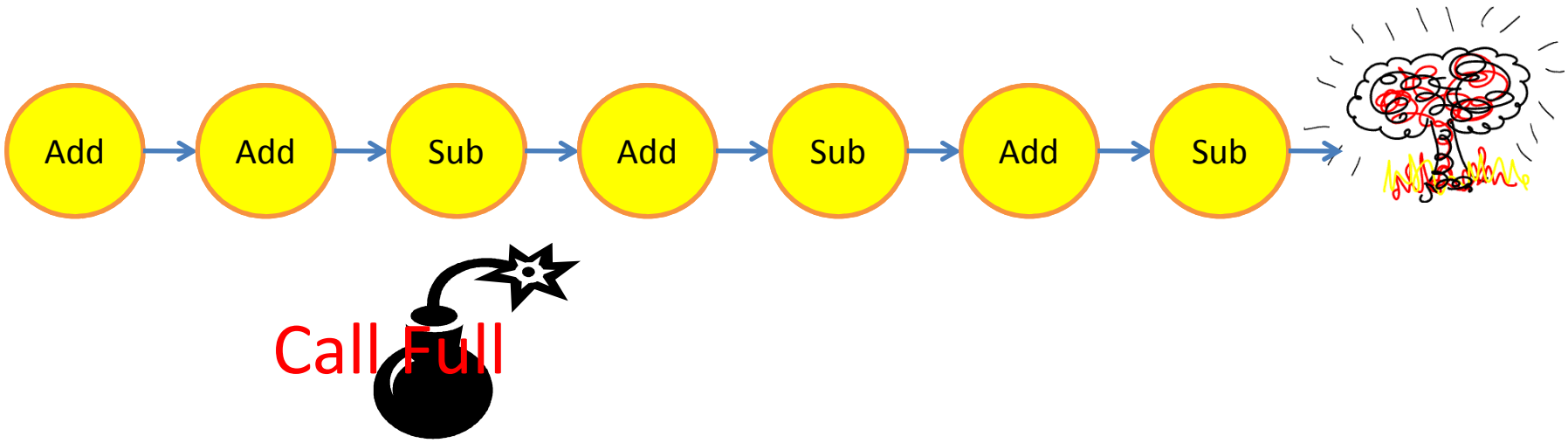
- The type
  - –struct { int n; complex float c; }

  cannot be passed as a parameter!
  - – (imaginary part of c is corrupted)

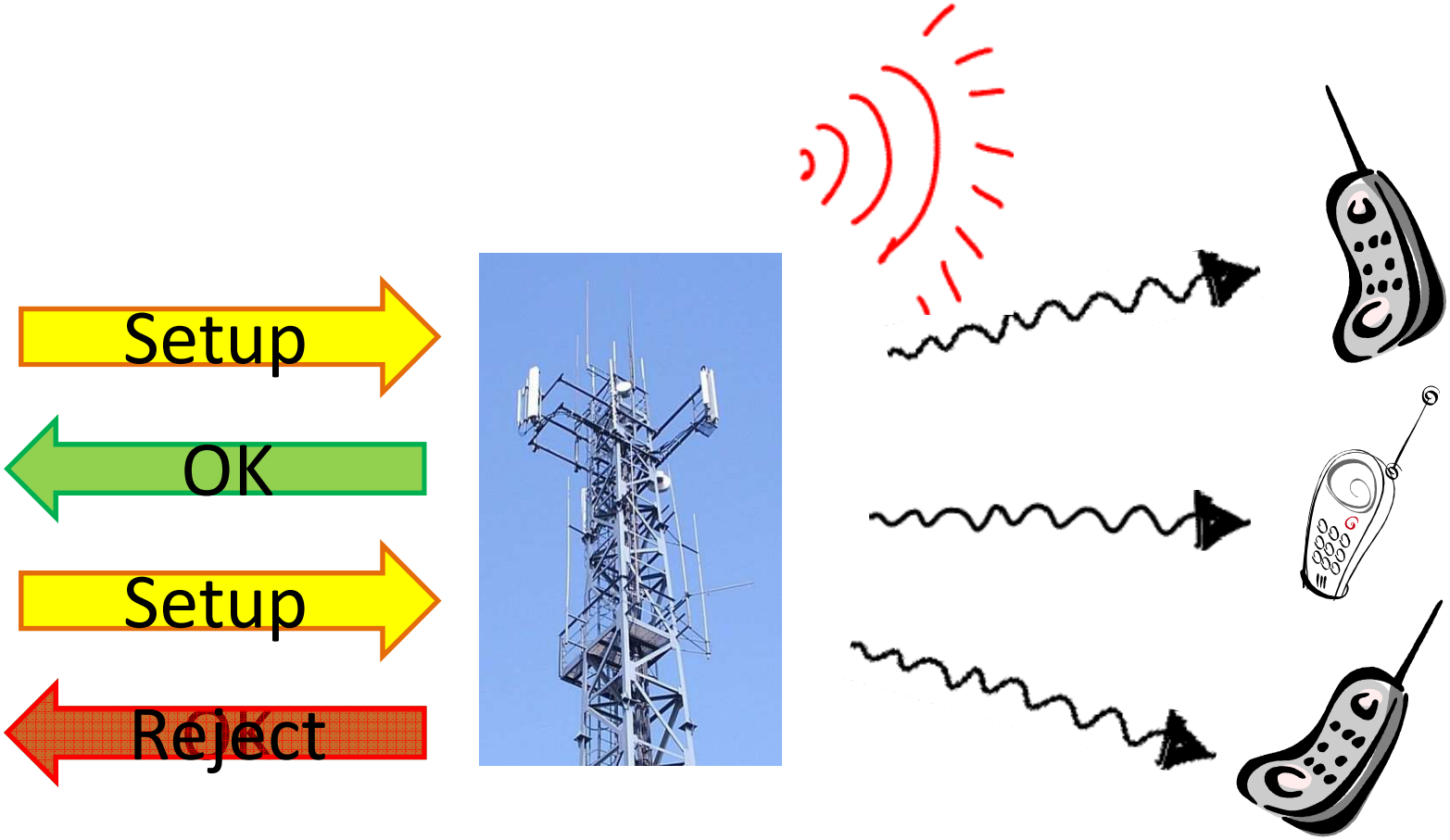
- Found while testing eqc_c
  - – QuickCheck shrank a complicated type to this one

Industrial System Testing: Media Proxy

- Multimedia IP-telephony (IMS)
- Connects calls across a firewall
- Test adding and removing callers from a call



3G Radio Base Station



IM vs IMAP Gateways at Erlang Solutions

- Comparable projects, 3 developers each
- QuickCheck used for *system testing* of IMAP
- **Faults found:**

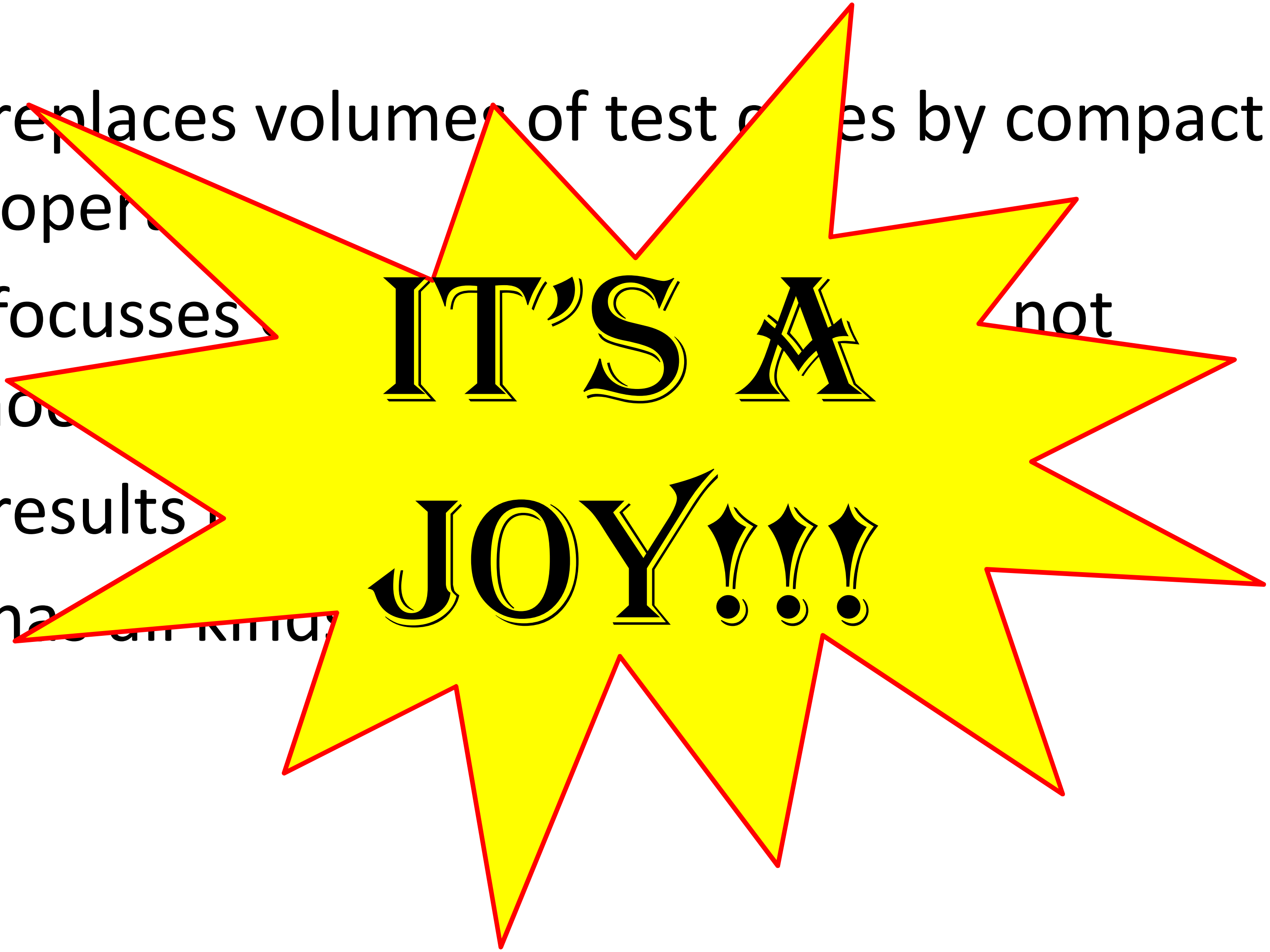
By QuickCheck tests	48
By tests based on studying QuickCheck traces	9
By fully hand-crafted tests	7
By acceptance tests	17

IM Gateway: 86% should have been found earlier

41% should have been found earlier

Property-Based Testing…

- …replaces volumes of test c es by compact
  proper
- …focusses not
  cho
- …results
- …ha



SIP Message Parser (Hans Nilsson, Ericsson)

- SIP messages generated from the grammar
- Property?

```
parse(unparse(parse(S))) == parse(S)
```

- 3,300 LOC—no bugs!
  - (during testing+12 months in service)
- 600 lines *not* tested with QuickCheck
  - Several bugs reported
  - Several *more* bugs found later with QuickCheck