Fuzz & Property Based Testing

Hello!

@fernandezpablo

bit.ly/fuzz_talk

Agenda

Fuzz Testing Origins → What Can Fuzz Testing Do? → Fuzz Applied: Property Based Testing

Code Demo → Tips And Closing Remarks → Q&A

1.
"On a dark and stormy night"

Fuzz Testing Origins

“It started on a dark and stormy night. One of the authors was logged on to his workstation on a dial-up line from home and the rain had affected the phone lines; there were frequent spurious characters on the line”

B. Miller '89

An Empirical Study of The Reliability of UNIX utilities

- Generation of random input
- "Random walk through program state"
- Not a replacement for formal methods
- Not a replacement for conventional testing
- Test on UNIX tools, see what happens.

B. Miller '89

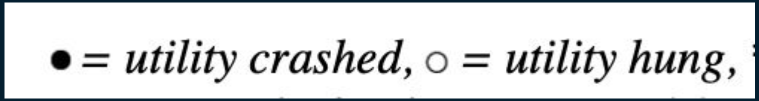
An Empirical Study of The Reliability of UNIX utilities

Crash

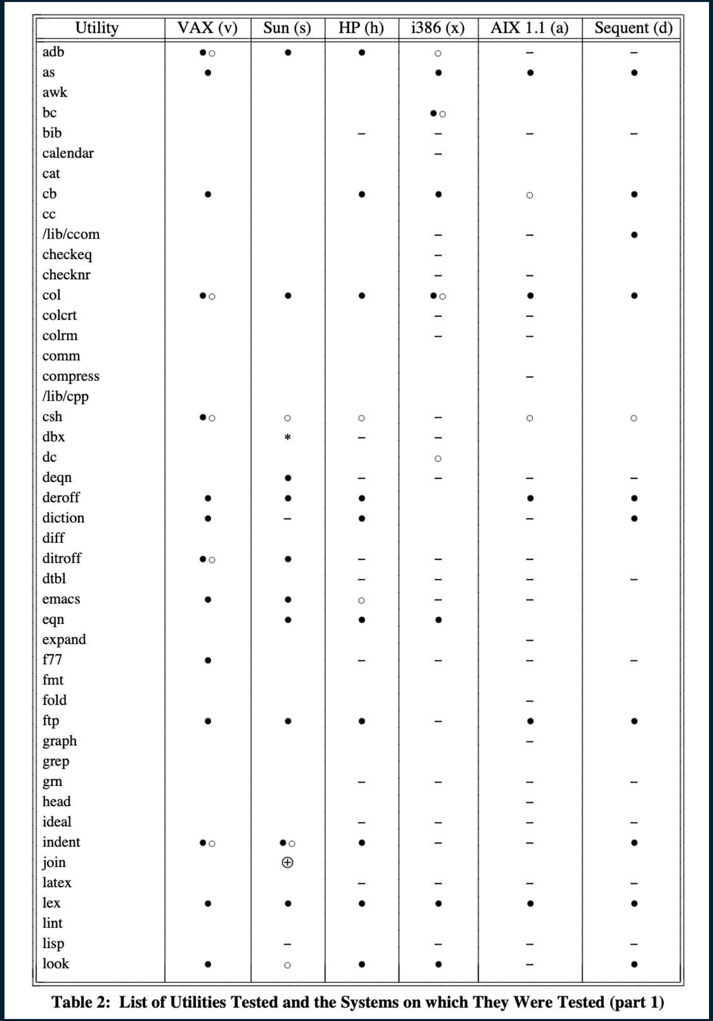
A program was considered crashed if it terminated producing a core (state dump) file.

Hang

A program was considered hung if it continued executing producing no output while having available input

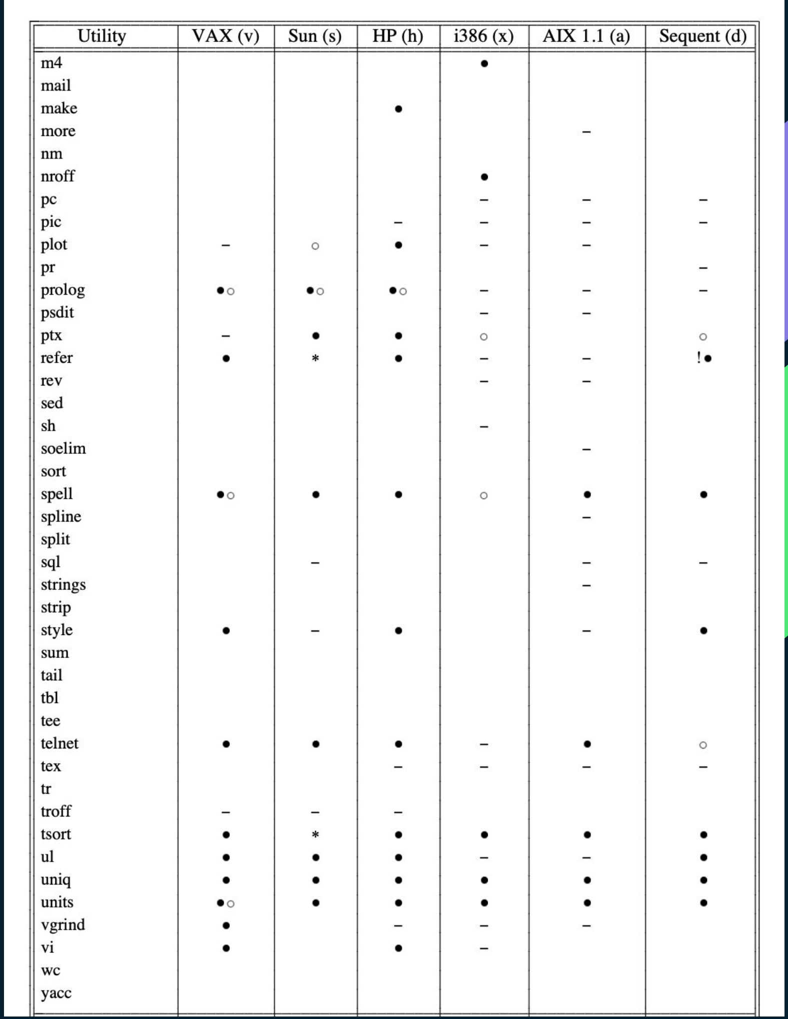


● = *utility crashed,* ○ = *utility hung,*

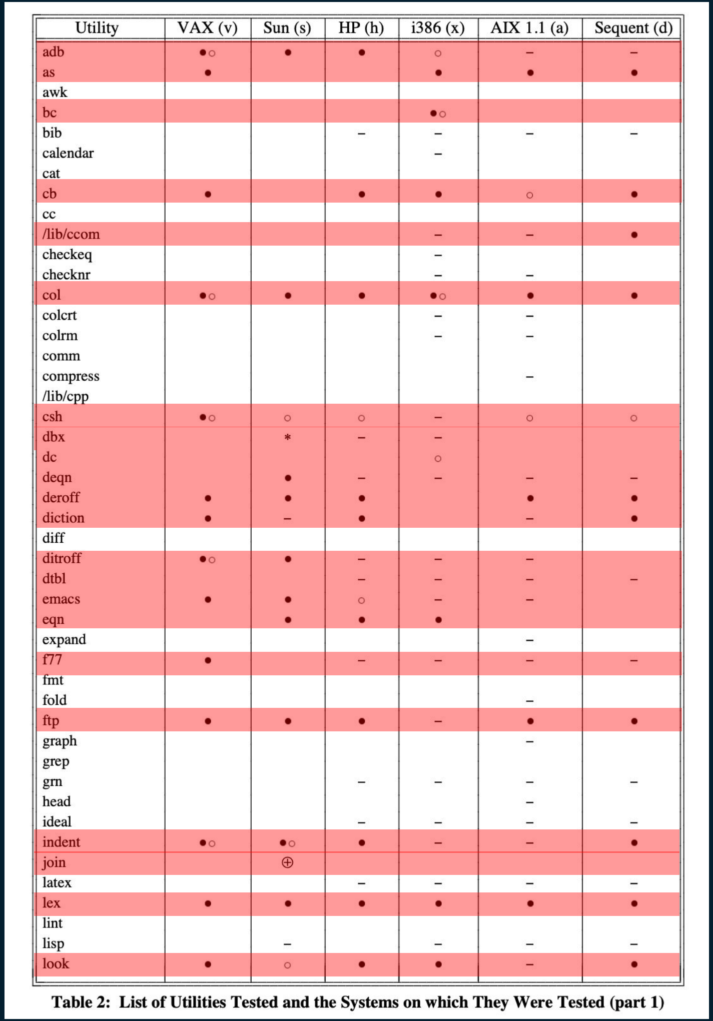


Utility	VAX (v)	Sun (s)	HP (h)	i386 (x)	AIX 1.1 (a)	Sequent (d)
adb	●○	●	●	○	–	–
as	●			●	●	●
awk						
bc				●○		
bib			–	–	–	–
calendar				–		
cat						
cb	●		●	●	○	●
cc						
/lib/ccom				–	–	●
checkeq				–		
checknr				–	–	
col	●○	●	●	●○	●	●
colcrt				–	–	
colrm				–	–	
comm						
compress					–	
/lib/cpp						
csh	●○	○	○	–	○	○
dbx		*	–	–		
dc				○		
deqn		●	–	–	–	–
deroff	●	●	●		●	●
diction	●	–	●		–	●
diff						
ditroff	●○	●	–	–	–	
dtbl			–	–	–	–
emacs	●	●	○	–	–	
eqn		●	●	●		
expand					–	
f77	●		–	–	–	–
fmt						
fold					–	
ftp	●	●	●	–	●	●
graph					–	
grep						
grn			–	–	–	–
head					–	
ideal			–	–	–	–
indent	●○	●○	●	–	–	●
join		⊕				
latex			–	–	–	–
lex	●	●	●	●	●	●
lint						
lisp		–		–	–	–
look	●	○	●	●	–	●

Table 2: List of Utilities Tested and the Systems on which They Were Tested (part 1)

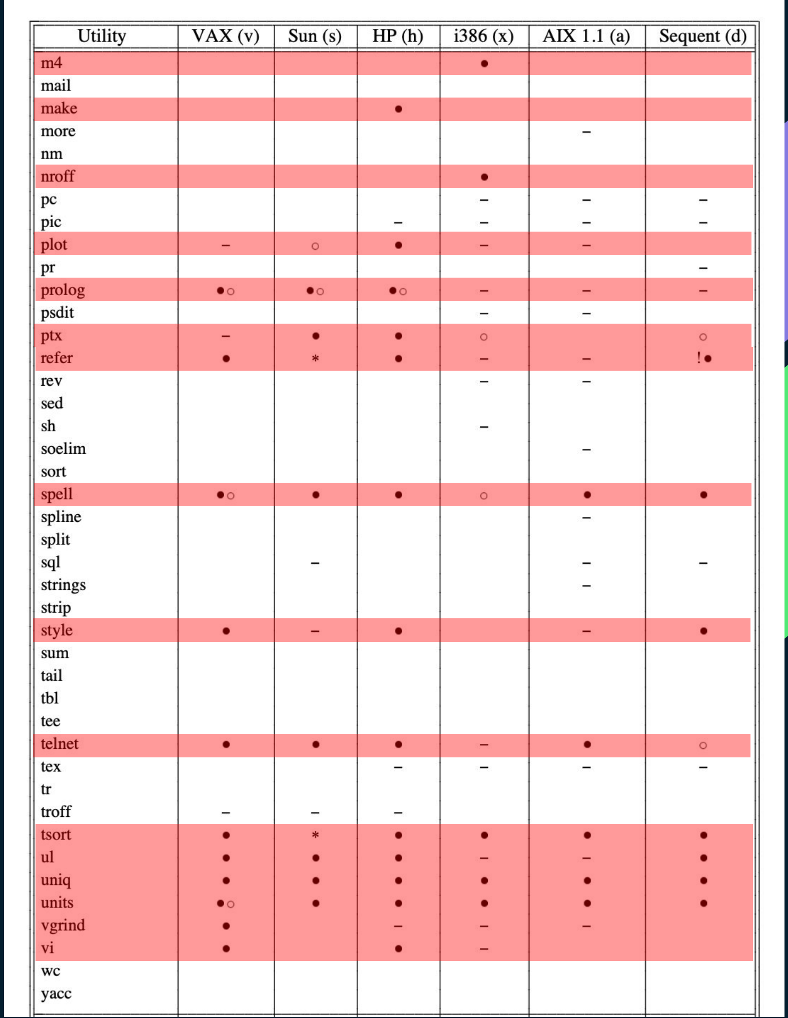


Utility	VAX (v)	Sun (s)	HP (h)	i386 (x)	AIX 1.1 (a)	Sequent (d)
m4				●		
mail						
make			●			
more					–	
nm						
nroff				●		
pc				–	–	–
pic			–	–	–	–
plot	–	○	●	–	–	
pr						–
prolog	●○	●○	●○	–	–	–
psdit				–	–	
ptx	–	●	●	○		○
refer	●	*	●	–	–	!●
rev				–	–	
sed						
sh				–		
soelim					–	
sort						
spell	●○	●	●	○	●	●
spline					–	
split						
sql		–			–	–
strings					–	
strip						
style	●	–	●		–	●
sum						
tail						
tbl						
tee						
telnet	●	●	●	–	●	○
tex			–	–	–	–
tr						
troff	–	–	–			
tsort	●	*	●	●	●	●
ul	●	●	●	–	–	●
uniq	●	●	●	●	●	●
units	●○	●	●	●	●	●
vgrind	●		–	–	–	
vi	●		●	–		
wc						
yacc						

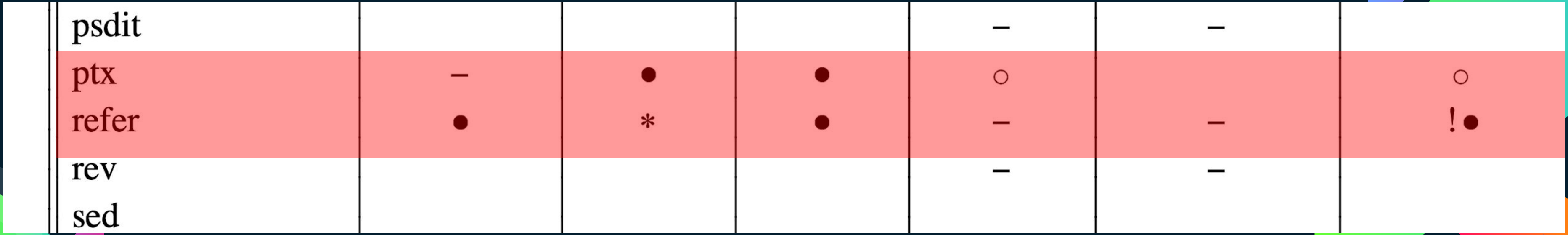


Utility	VAX (v)	Sun (s)	HP (h)	i386 (x)	AIX 1.1 (a)	Sequent (d)
adb	●○	●	●	○	–	–
as	●			●	●	●
awk						
bc				●○		
bib			–	–	–	–
calendar				–		
cat						
cb	●		●	●	○	●
cc						
/lib/ccom				–	–	●
checkeq				–		
checknr				–	–	
col	●○	●	●	●○	●	●
colcrt				–	–	
colrm				–	–	
comm						
compress					–	
/lib/cpp						
csh	●○	○	○	–	○	○
dbx		*	–	–		
dc				○		
deqn		●	–	–	–	–
deroff	●	●	●		●	●
diction	●	–	●		–	●
diff						
ditroff	●○	●	–	–	–	
dtbl			–	–	–	–
emacs	●	●	○	–	–	
eqn		●	●	●		
expand					–	
f77	●		–	–	–	–
fmt						
fold					–	
ftp	●	●	●	–	●	●
graph					–	
grep						
grn			–	–	–	–
head					–	
ideal			–	–	–	–
indent	●○	●○	●	–	–	●
join		⊕				
latex			–	–	–	–
lex	●	●	●	●	●	●
lint						
lisp		–		–	–	–
look	●	○	●	●	–	●

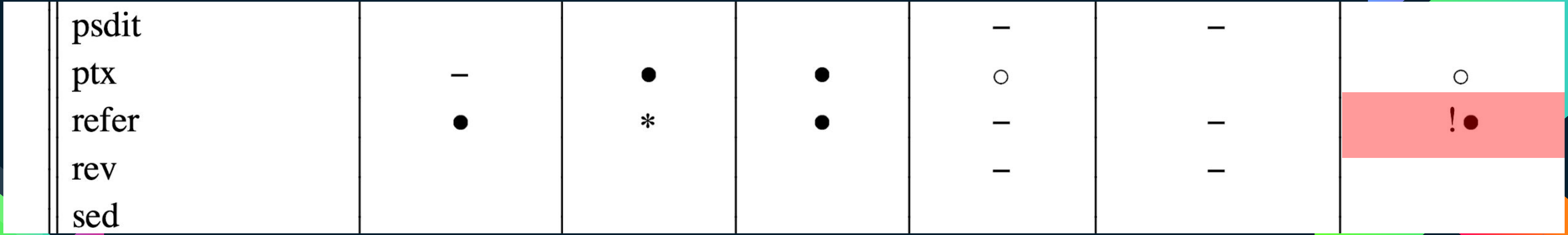
Table 2: List of Utilities Tested and the Systems on which They Were Tested (part 1)



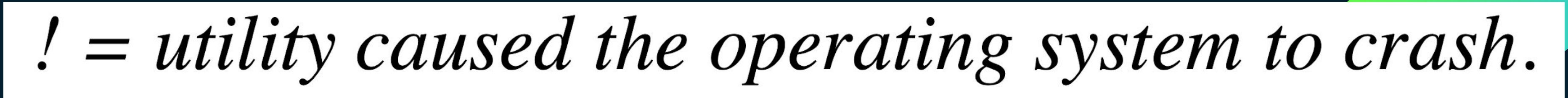
Utility	VAX (v)	Sun (s)	HP (h)	i386 (x)	AIX 1.1 (a)	Sequent (d)
m4				●		
mail						
make			●			
more					–	
nm						
nroff				●		
pc				–	–	–
pic			–	–	–	–
plot	–	○	●	–	–	
pr						–
prolog	●○	●○	●○	–	–	–
psdit				–	–	
ptx	–	●	●	○		○
refer	●	*	●	–	–	!●
rev				–	–	
sed						
sh				–		
soelim					–	
sort						
spell	●○	●	●	○	●	●
spline					–	
split						
sql		–			–	–
strings					–	
strip						
style	●	–	●		–	●
sum						
tail						
tbl						
tee						
telnet	●	●	●	–	●	○
tex			–	–	–	–
tr						
troff	–	–	–			
tsort	●	*	●	●	●	●
ul	●	●	●	–	–	●
uniq	●	●	●	●	●	●
units	●○	●	●	●	●	●
vgrind	●		–	–	–	
vi	●		●	–		
wc						
yacc						



psdit				–	–	
ptx	–	●	●	○		○
refer	●	*	●	–	–	!●
rev				–	–	
sed						



psdit				–	–	
ptx	–	●	●	○		○
refer	●	*	●	–	–	!●
rev				–	–	
sed						



! = utility caused the operating system to crash.

Results and Analysis

High Error Rates

About 25-30% on all tested OSes.

Common Tools

emacs, vi, ftp, lex, make, telnet, uniq, etc.

Error Taxonomy

The authors grouped errors by their cause.

2.
Further Studies on Fuzz Testing

Finding bugs on more tools and OSes

B. Miller '95

A Re-Examination of The Reliability of UNIX Utilities and Services

- Re-run fuzz tests on UNIX tools, include GNU and Linux
- About the same failure rate (~25%)
- GNU and Linux have less errors (6% and 9%)
- Other tools (malloc, x-window)
- Revisiting '89 results

“many of the bugs discovered…, are still present in their exact form in 1995. The 1990 study was widely published in at least two languages. The code was made freely available via anonymous ftp. The exact random data streams used in our testing were freely available via ftp”

“Several of the bugs found in the 1995 study were likely present in the 1990 study, but were masked by the original bugs. Fixing the original bugs and re-testing should have exposed these new ones.”

Forrester & Miller '00

An Empirical Study of the Robustness of Windows NT Applications Using Random Testing

- Same study on Windows NT
- Only Win32 apps
- About the same failure rate (45%)
- No source code so can't tell the error causes

B. Miller '06

An Empirical Study of the Robustness of MacOS Applications Using Random Testing

- Same study on Mac OSX
- Command Line and Cocoa Apps
- Command Line failure rate: 7%
- Cocoa Apps failure rate: 70%

"...software packages are providing more features and therefore are getting more complex. In such a view of the world, it is not surprising that the reliability of applications is not improving, but instead seems to be getting worse."

~30% OF APPS HAVE BUGS

Which can be found using Fuzz Testing

FUZZ TESTING IS JUST ANOTHER TOOL

Complementary to what you already use

3.
Property Based Testing

Appling Fuzz Testing concepts to everyday code

We want Fuzz testing now what?

Lessons Learned from Millers' Work

- Random input seems to work fine for detecting bugs
- What level of abstraction shall we use?
- Using HTTP requests would be cumbersome
- What about random input for functions?

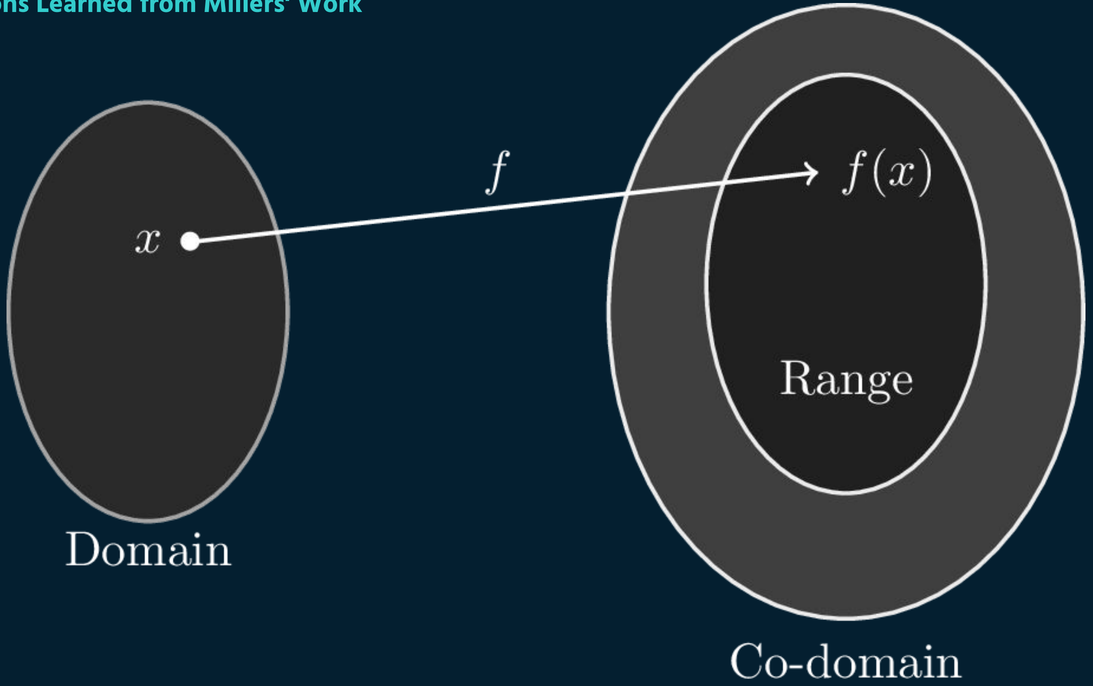
Function Domains

Lessons Learned from Millers' Work

- Functional programming / Math concept
- Domain: the set of input values for which the function is defined

Function Range and Domain

Lessons Learned from Millers' Work



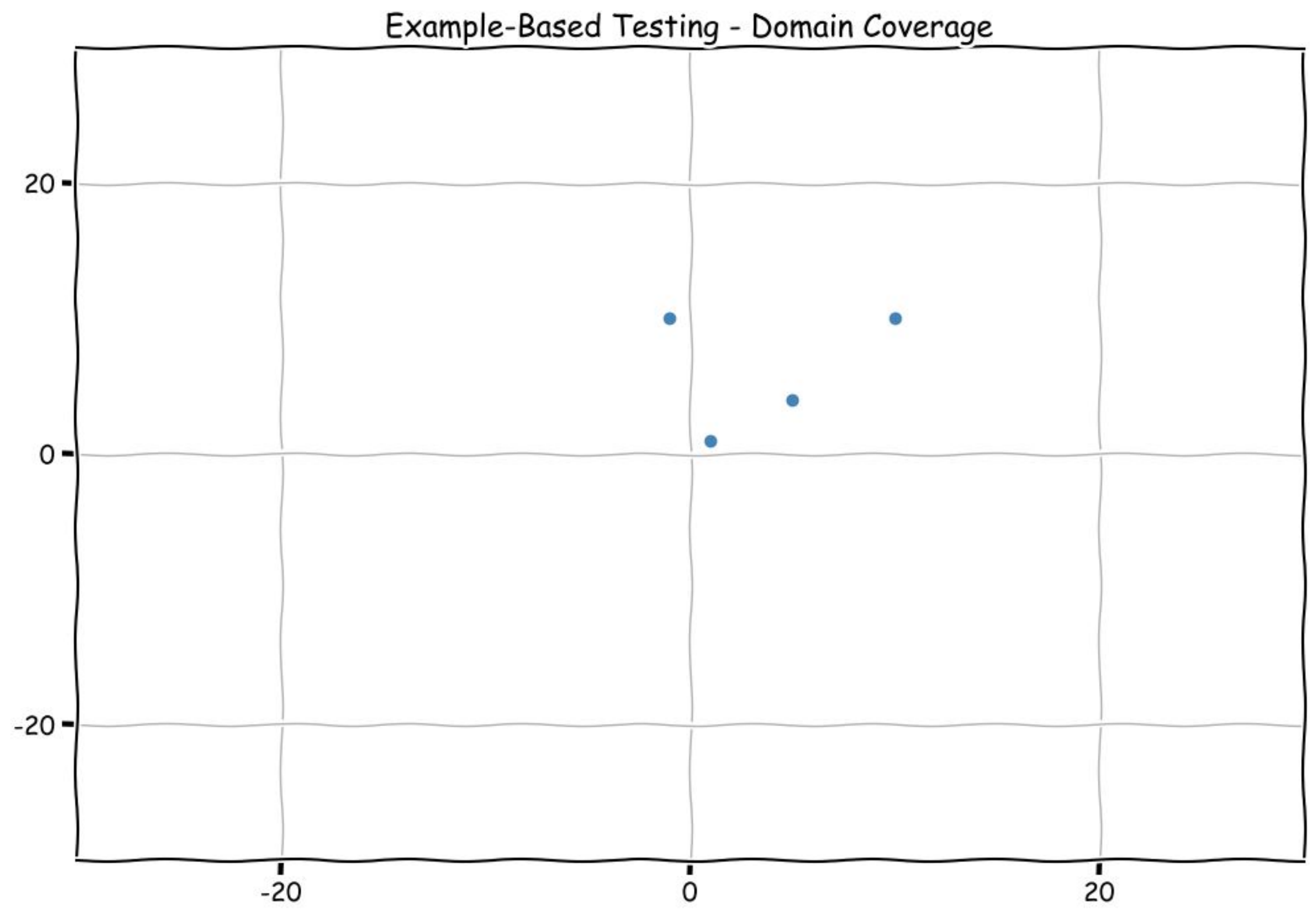
Domain of an add function

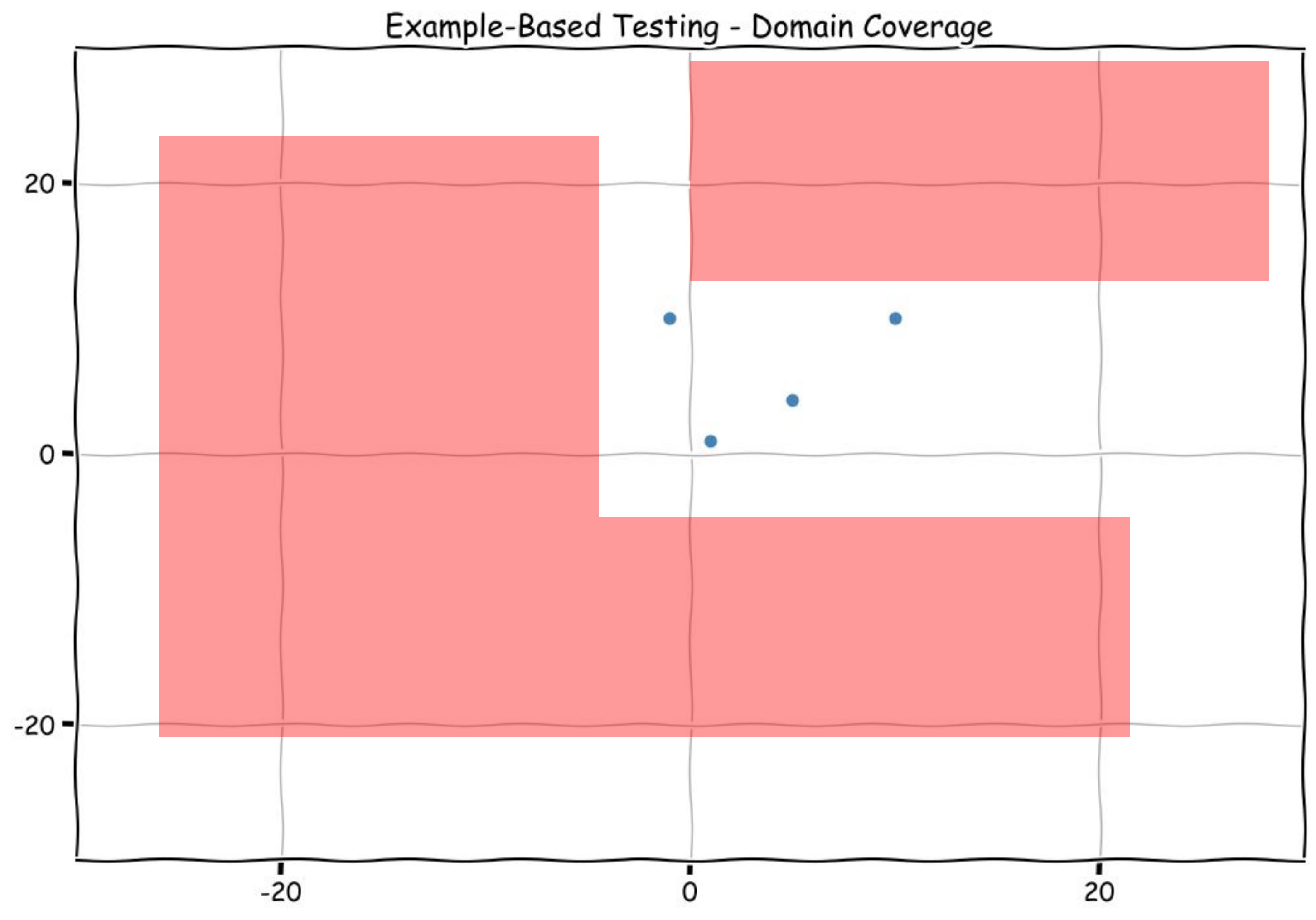
- Given an add function that sums two integers

- The Domain of add would be a cartesian plane (X, Y)

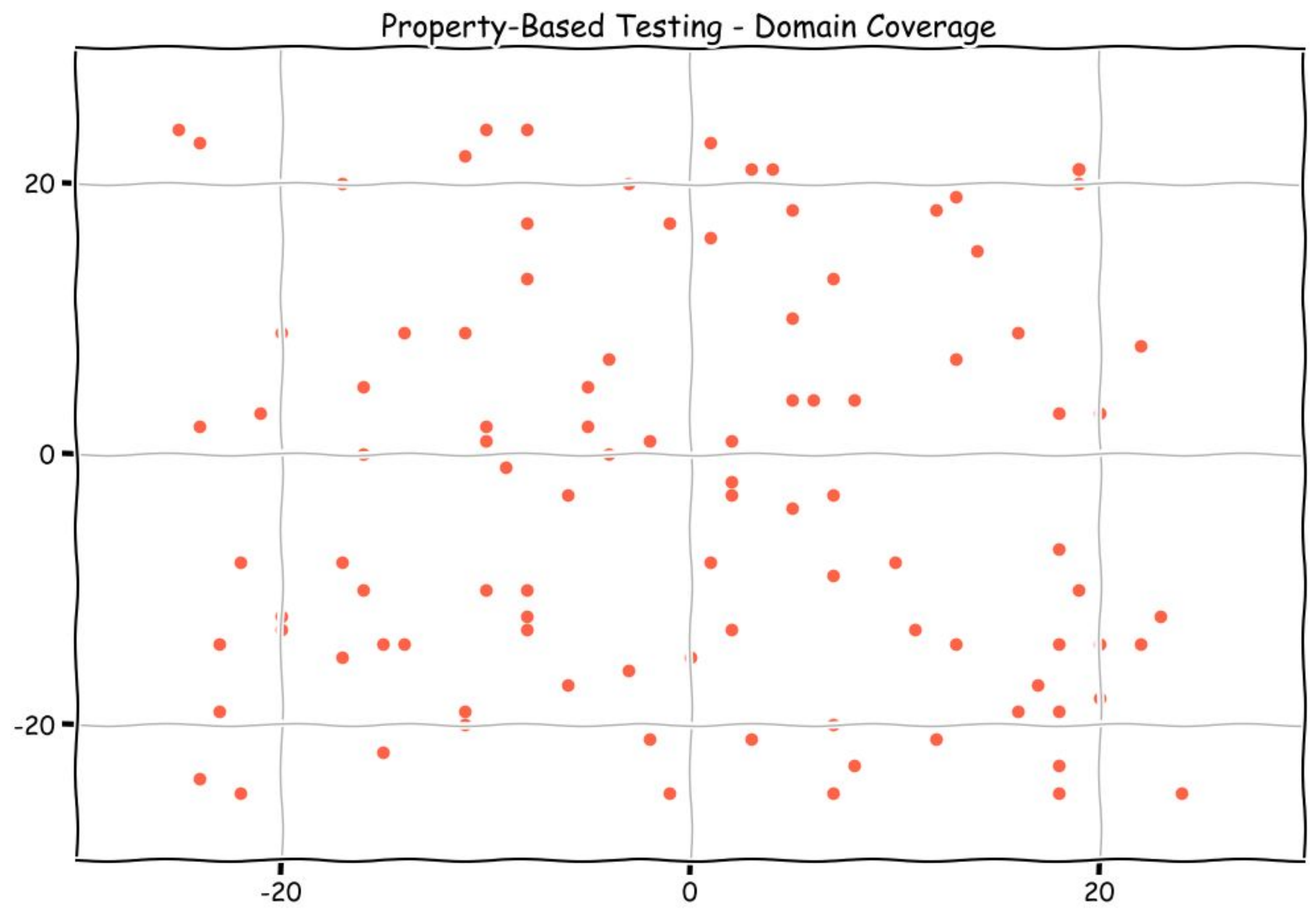
Tests explore a function domain

- assert add(1, 1) == 2
- assert add(1, 0) == 1
- assert add(5, 4) == 9
- assert add(-1, 10) == 9
- assert add(10, 10) == 20


Example-Based Testing - Domain Coverage
20
0
-20
-20
0
20


Example-Based Testing - Domain Coverage
20
0
-20
-20
0
20

RANDOM INPUT GENERATION FOR DOMAIN


Property-Based Testing - Domain Coverage
20
0
-20
-20
0
20

BUT WHAT ABOUT ASSERTIONS? 😕

We used to test:

```
add(1, 2) == 3
```

But with random input:

```
add(a, b) == ???
```

But with random input:

```
add(a, b) == a + b
```

But with random input:

```
add(a, b) == add(b, a)
```

We need to use properties

- Also known as "invariants", "behavior", "contracts", etc.
- Things that will always be true no matter what a and b are
- Let's get to it!

4.
Code Demo

5.
Tips & Closing Remarks

Other languages and More

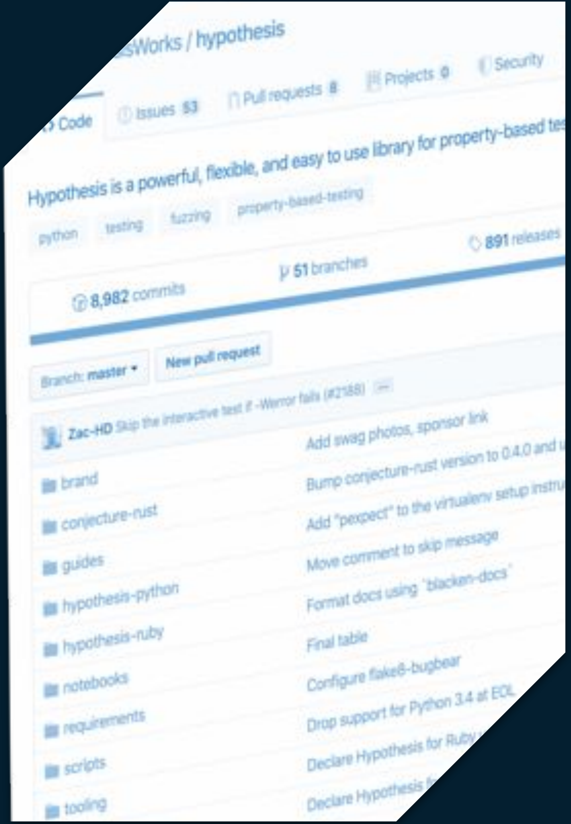
- Available in all languages (QuickCheck)
- Go-Fuzz (code coverage instrumentation)
- Metamorphic Testing by Tsong Chen
- "Choosing properties for property-based testing" by Scott Wlaschin (F#)
- These and more on repository "resources" folder

About 30% of Apps have errors

Many of those can be found with simple assertions and property based testing

Many good quality **libraries** exist

Hypothesis is great! a lot of functionality not covered in this talk for time reasons



sWorks / hypothesis

Hypothesis is a powerful, flexible, and easy to use library for property-based tes

8,982 commits · 51 branches · 891 releases

Zac-HD Skip the interactive test if -Werror fails (#2188)

brand	Add swag photos, sponsor link
conjecture-rust	Bump conjecture-rust version to 0.4.0 and u
guides	Add "pexpect" to the virtualenv setup instru
hypothesis-python	Move comment to skip message
hypothesis-ruby	Format docs using `blacken-docs`
notebooks	Final table
requirements	Configure flake8-bugbear
scripts	Drop support for Python 3.4 at EOL
tooling	Declare Hypothesis for Ruby
	Declare Hypothesis f

Property Based Testing is complementary

Think of it as an additional tool, not a replacement

Thanks!

@fernandezpablo

Questions?

@fernandezpablo

bit.ly/fuzz_talk