Understanding Property-Based Testing by Talking to People

Joseph Cutler, Harrison Goldstein, and Adam Stein

University of Pennsylvania
CIS 700-003
Prof. Andrew Head

OVERVIEW

Property-based testing (PBT) is a powerful tool for software validation. It is more rigorous than unit testing but not nearly as labor-intensive as formal verification. We spoke to users of *Hypothesis*, a popular Python library for doing PBT, to determine where current PBT methods succeed and where they fall short.

This preliminary study consisted of seven interviews over the course of a few weeks. In that time we learned a lot about PBT:

- Properties that users test vary, from simple to very complex.
- Users with a math background tend to find PBT more natural; others less so.
- Only very clean, well-abstracted code is amenable to PBT.
- Users often need to "prove" to management that PBT is a worthy investment.

We hope that these findings inform future research into PBT.

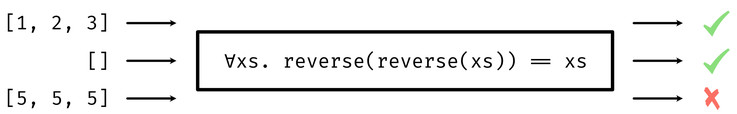
BACKGROUND

In PBT, a tester gains confidence in their code's correctness by coming up with *properties* that they expect to be true. A property for a `reverse` function might be something like:

```
∀xs. reverse(reverse(xs)) == xs
```

This property states that for any list `xs`, calling `reverse` twice on `xs` is the same as doing nothing.

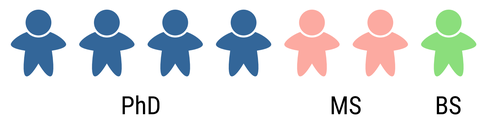
Once a user has stated a property, they can validate it with hundreds or thousands of *random test inputs*. Unlike in classical unit testing, the tester is not limited by their ability to come up with good examples.



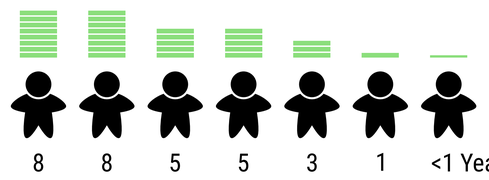
Hypothesis is a Python library that provides users with the tools to test properties of their own programs. Random inputs are produced by *generators* or *strategies*, and may need to satisfy certain preconditions.

RECRUITING

We recruited seven informants from within our academic networks. Our informants all had at least a bachelor's degree:



The informants had experience using Hypothesis in an industry setting, though experience varied:



This sample was sufficient for a preliminary study, but follow-on work will require more diversity, especially with respect to educational background.

INTERVIEWS

Our interviews lasted for 30 minutes each. We conducted each interview on Zoom and transcribed each recording using Otter.ai. In the course of the interview we asked three main questions:

1. **Tell us about your most memorable experience using property-based testing.**
   This got the informant thinking about a *specific* experience that was also interesting.

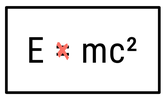
2. **How did you come up with the properties that you tested?**
   This helped establish what conditions the informant hoped to validate with testing.

3. **Did you need custom generators for your testing, and if so what did they do?**
   This gave us a better sense of how current research addresses the informant's needs.

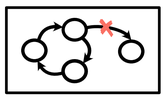
WHAT WE LEARNED

USERS TEST A VARIETY OF PROPERTIES

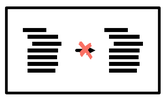
We found that our informants used PBT to test a number of different kinds of properties:



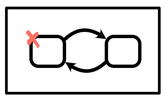
Math and Science
Some "obvious" properties are due to constraints in a concrete domain.



State Machine Invariants
API invariants can be tested by calling functions at random.



Regression Testing
As code changes, PBT can help ensure that behavior stays the same.



"Round Trip"
Ensuring that two functions are inverses is a classic use of PBT.

PBT DOES NOT COME NATURALLY TO EVERYONE

A recurring theme of our interviews was the "mental model" of PBT. We found that, in general, users with a background in mathematics or science found PBT natural, whereas users with a more traditional software engineering background struggled to find properties to test.

PBT IS NOT YET "PROVEN" IN INDUSTRY

Our informants reported that preparing code for PBT, ensuring that testable components are pure and well-abstracted, is a major pain-point.

WELL-ABSTRACTED CODE IS KEY

When discussing PBT in the broader context of an engineering team or company, most of our informants talked about needing to convince others that PBT is "worth it." In general, the value that PBT adds is not well-understood by those who do not use it.