Property-Based Testing in Open-Source Rust Projects

A Case Study of the `proptest` Crate

Antonios Barotsis
Supervisors: Dr. Andreea Costea, Sára Juhošová
EEMCS, Delft University of Technology, The Netherlands

Summary

Testing is a critical part of software development, especially in popular Open Source Software [1] (**OSS**). Property-Based testing (**PBT**) has emerged as an easy yet powerful new testing technique. We aim to gain insights on how the leading PBT framework `proptest` is used in the Rust ecosystem.

1. What is PBT?

Here's an example which verifies that reversing a list *twice* should give us the original list:

```
proptest! {
  #[test]
  fn pbt(list in any::<Vec<i32>>()) {
  ⓘ: Use generator to get random input
    let mut reversed = list.clone();
    reversed.reverse(); // reverse once
    reversed.reverse(); // reverse twice

    // Assert reversing list twice == initial list
    assert_eq!(list, reversed);
    ⓘ: Failing test inputs automatically shrunk
  }
}
```

- **Generators** generate hundreds of random inputs for our test.
- Upon encountering a failing test, the PBT framework tries to **shrink** the failing input. In other words, simplifying it to the smallest form that still reproduces the failure.

2. How is it used in OSS?

Our research questions expand upon the following:

- Properties
  1. What type of properties do PBTs generally check?
  2. What do these properties look like?
  3. What role does PBT play within the correctness guarantees and bug-finding strategies of the project overall?
- Generators and Shrinking:
  1. How and when are generators implemented?
  2. In which cases is shrinking support explicitly added?

3. Our methodology

```
repos = proptest.dependents.sorted_by(total_downloads)
for repository in repos:
    Gather descriptive project metadata (size, amount of tests, amount of PBTs, downloads etc)
    Analyze each PBT individually
```

References

[1] M. Hoffmann, F. Nagle, and Y. Zhou, "The Value of Open Source Software," *SSRN Electronic Journal*, 2024, doi: 10.2139/ssrn.4693148.

4. Results

We explored **16 repositories** using `proptest` and analyzed **143 tests**, here's what we learned:

1. **Property Types**: Most PBTs used TestOracles.

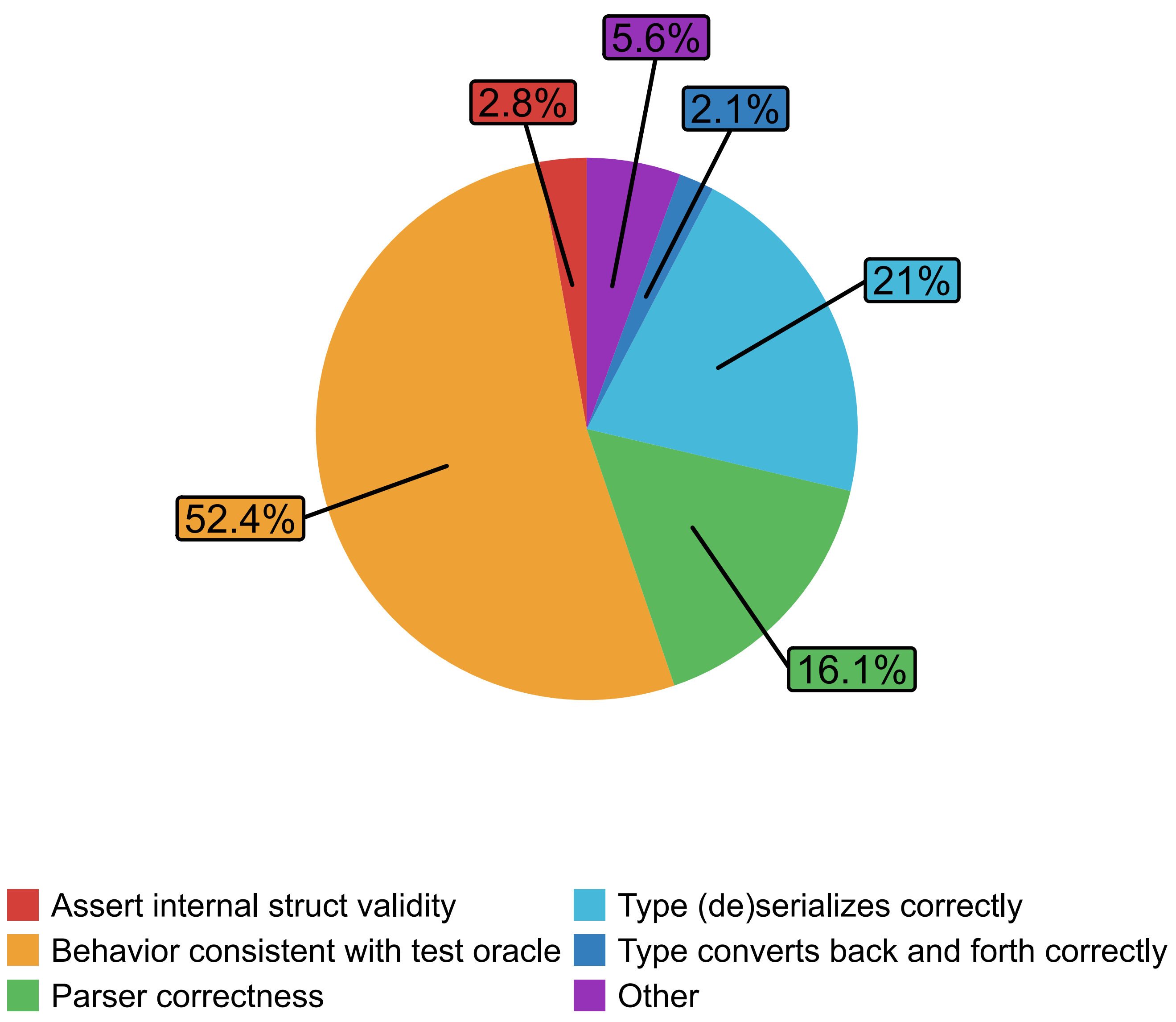
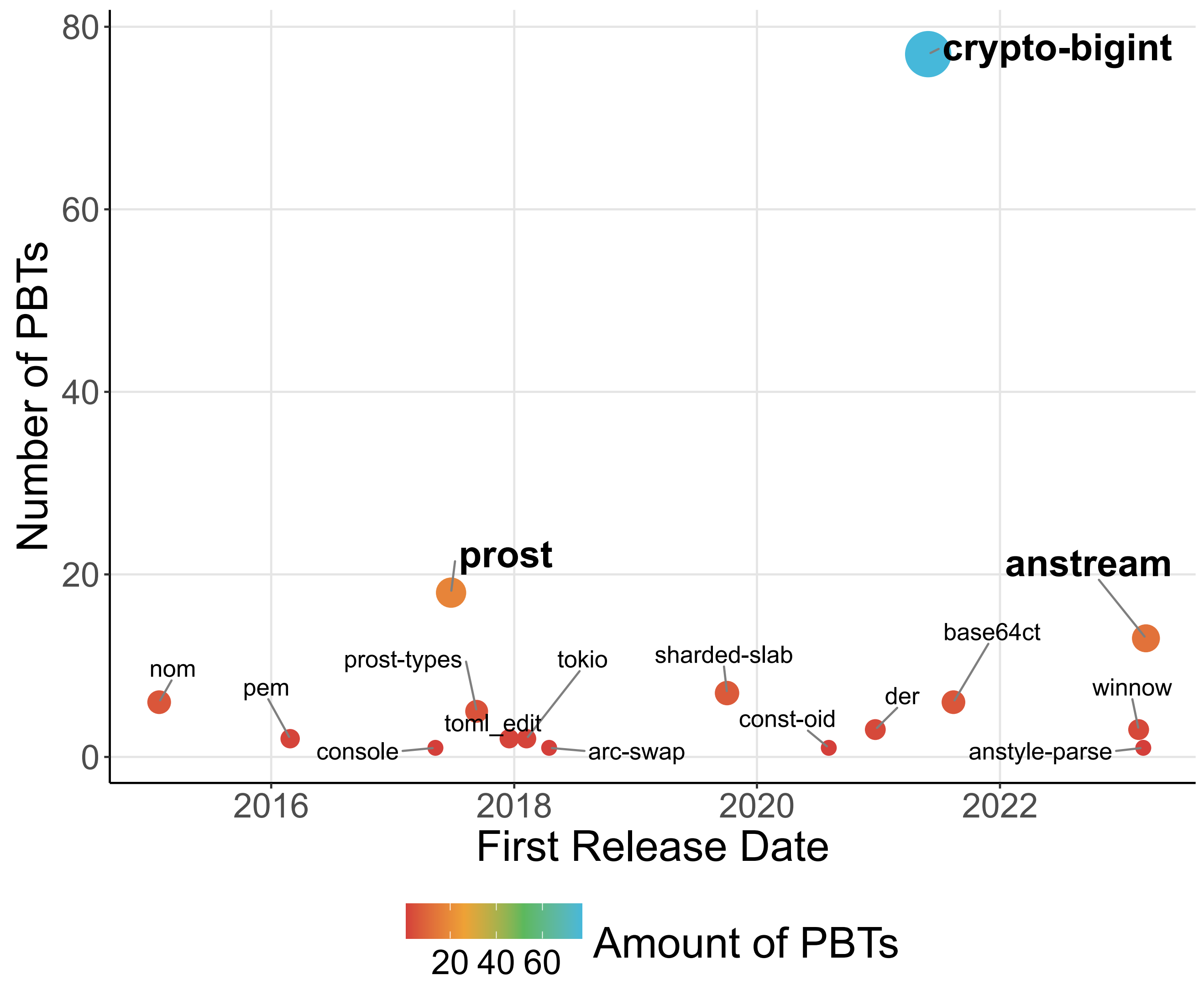


Figure 1: PBT Category Breakdown

2. **Complexity**: tests are kept *simple*. Only two assertions per test, 87% of properties are non-decomposable.
3. **Generators & Shrinkers**: 74% of our examined projects make use of custom generators, yet *none* implement custom shrinkers!

We also gained some insights that apply to the Rust language as a whole:

- Rust's type system largely handles invariants → no need to test for them.
- Specialized tools/frameworks are used to test for undefined behavior and concurrency rather than PBT.
- Rust's "doctests" are used to document code, a role oftentimes filled by PBTs in other languages.



Projects with ≥10 PBTs are in**bold**

Figure 2: First Release Date vs Amount of PBTs