Automated Penetration Testing with QuickCheck

Agustín Mista
Chalmers University of Technology
mista@chalmers.se

Alejandro Russo
Chalmers University of Technology
russo@chalmers.se

Abstract

Random property-based testing is an increasingly popular approach to finding bugs [2, 7, 8]. In the Haskell community, *QuickCheck* [3] is the dominant tool of this sort. *QuickCheck* requires developers to specify *testing properties* describing the expected software behavior. Then, it generates a large number of random *test cases* and reports those violating the testing properties. *QuickCheck* generates random data by employing *random test data generators* or *QuickCheck* generators for short. The generation of test cases is guided by the *types* involved in the testing properties. It defines default generators for many built-in types like booleans, integers, and lists. However, when it comes to user-defined ADTs, developers are usually required to specify the generation process. The difficulty is, however, that it might become intricate to define generators so that they result in a suitable distribution or enforce data invariants.

Rather than manually writing generators, libraries *derive* [11] and *MegaDeTH* [5, 6] allow us to automatically synthesize generators for a given user-defined ADT. The library *derive* provides no guarantees that the generation process terminates, while *MegaDeTH* pays almost no attention to the distribution of values. In contrast, *Feat* [4] provides a mechanism to uniformly sample values from a given ADT. It enumerates all the possible values of a given ADT so that sampling uniformly from ADTs becomes sampling uniformly from the set of natural numbers.

In this work, we consider the scenario where developers are not fully aware of the properties and invariants that input data must fulfill. This constitutes a valid assumption for *penetration testing* [1], where testers often apply fuzzers in an attempt to make programs crash—an anomaly which might lead to a vulnerability.

Our realization is that *branching processes* [12], a relatively simple stochastic model conceived to study the evolution of populations, can be applied to predict the generation distribution of ADTs' constructors in a simple and automatable manner. To the best of our knowledge, this stochastic model has not yet been applied to this field, and we believe it may be a promising foundation to develop future extensions.

Using our probabilistic formulas, we design heuristics capable of automatically adjusting probabilities in order to synthesize generators which distributions are aligned with users' demands [10].

Furthermore, we provide a Haskell implementation of our mechanism in a tool called DRAGEN and perform case studies with real-world applications. When generating random values, our synthesized *QuickCheck* generators show improvements in code coverage when compared with those automatically derived by state-of-the-art tools.

Recently, we extended our framework to consider addiotional sources of structural information from the users' codebase [9]. In this light, our random generators can produce complex patterns of values, as well as calls to functions in abstract interfaces, obtaining remarkable improvements in terms of code coverage. All of this while still being able to reason about the distributions of generated values in terms of branching processes.

SWITS, June 3–4, 2019, Karlstad, Sweden
2019.

References

[1] B. Arkin, S. Stender, and G. McGraw. 2005. Software penetration testing. *IEEE Security Privacy* (2005).
[2] T. Arts, J. Hughes, U. Norell, and H. Svensson. 2015. Testing AUTOSAR software with QuickCheck. In *In Proc. of IEEE International Conference on Software Testing, Verification and Validation, ICST Workshops.*
[3] K. Claessen and J. Hughes. 2000. QuickCheck: A Lightweight Tool for Random Testing of Haskell Programs. In *Proc. of the ACM SIGPLAN International Conference on Functional Programming (ICFP).*
[4] J. Duregård, P. Jansson, and M. Wang. 2012. Feat: Functional enumeration of algebraic types. In *Proc. of the ACM SIGPLAN Int. Symp. on Haskell.*
[5] G. Grieco, M. Ceresa, and P. Buiras. 2016. QuickFuzz: An automatic random fuzzer for common file formats. In *Proc. of the ACM SIGPLAN International Symposium on Haskell.*
[6] G. Grieco, M. Ceresa, A. Mista, and P. Buiras. 2017. QuickFuzz testing for fun and profit. *Journal of Systems and Software* 134 (2017).
[7] J. Hughes, C. Pierce B, T. Arts, and U. Norell. 2016. Mysteries of DropBox: Property-Based Testing of a Distributed Synchronization Service. In *Proc. of the Int. Conf. on Software Testing, Verification and Validation.*
[8] J. Hughes, U. Norell, N. Smallbone, and T. Arts. 2016. Find more bugs with QuickCheck!. In *The IEEE/ACM International Workshop on Automation of Software Test (AST).*
[9] Agustín Mista and Alejandro Russo. 2019. Generating Random Structurally Rich Algebraic Data Type Values. In *Proceedings of the 14th International Workshop on Automation of Software Test.*
[10] Agustín Mista, Alejandro Russo, and John Hughes. 2018. Branching processes for QuickCheck generators. In *Proc. of the ACM SIGPLAN Int. Symp. on Haskell.*
[11] N. Mitchell. 2007. Deriving Generic Functions by Example. In *Proc. of the 1st York Doctoral Syposium.* Tech. Report YCS-2007-421, Department of Computer Science, University of York, UK, 55–62.
[12] H. W. Watson and F. Galton. 1875. On the probability of the extinction of families. *The Journal of the Anthropological Institute of Great Britain and Ireland* (1875).