Anne Koziolek, Ina Schaefer, Christoph Seidl (Hrsg.): Software Engineering 2021,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2021 97

Probabilistic Grammar-based Test Generation

Ezekiel Soremekun[1,2] Esteban Pavese[3] Nikolas Havrikov[4] Lars Grunske[5] Andreas Zeller[6]

Abstract: Given a program that has been tested on some sample input(s), what does one test next? To further test the program, one needs to construct inputs that cover (new) input features, in a manner that is different from the initial samples. This talk presents an approach that learns from past test inputs to generate *new but different* inputs.

To achieve this, we present an approach called *inputs from hell* which employs *probabilistic context-free grammars* to learn the distribution of input elements from sample inputs. In this work, we employ probabilistic grammars as input parsers and producers. Applying probabilistic grammars as *input parsers*, we learn the statistical distribution of input features in sample inputs. As a *producer*, probabilistic grammars ensure that generated inputs are syntactically correct by construction, and it controls the distribution of input elements by assigning probabilities to individual production rules. Thus, we create inputs that are dissimilar to the sample by inverting learned probabilities.

In addition, we generate *failure-inducing inputs* by learning from inputs that caused failures in the past, this gives us inputs that share similar features and thus also have a high chance of triggering bugs. This approach is useful for bug reproduction and testing for patch completeness.

Keywords: Grammar; Test Case Generation; Probabilistic Grammars; Input Samples

1 Summary

This article is an abridged version of our paper titled "Inputs from Hell: Learning Input Distributions for Grammar-Based Test Generation" which is published in the proceedings of the IEEE Transactions on Software Engineering (TSE) [So20].

Grammar-based test generation techniques automatically produce thousands of valid inputs for software testing [HZ19]. However, it is also important to test programs on diverse inputs, in order to explore different features. In this work, we address the problem of generating *syntactically valid inputs that are (dis)similar to seen inputs.* Specifically, given a program that has been tested on some sample inputs, we ask the following: Which inputs should one test next? How can one generate inputs that are (dis)similar to the initial samples? To further

[1] SnT, University of Luxembourg, Luxembourg ezekiel.soremekun@uni.lu
[2] This work was done while working at CISPA Helmholtz Center for Information Security, Saarbrücken
[3] Department of Computer Science, Humboldt-Universität zu Berlin pavesees@informatik.hu-berlin.de
[4] CISPA Helmholtz Center for Information Security, Saarbrücken nikolas.havrikov@cispa.de
[5] Department of Computer Science, Humboldt-Universität zu Berlin grunske@informatik.hu-berlin.de
[6] CISPA Helmholtz Center for Information Security, Saarbrücken zeller@cispa.de

doi:10.18420/SE2021_36

test the program, one needs to construct inputs that cover new input features. Hence, the developer is tasked with the generation of *syntactically valid but different inputs*.

To tackle this challenge, we present a probabilistic grammar-based test generation approach called *inputs from hell*. The main idea of this technique is to apply *probabilistic context-free grammars* (PCFG) as input parsers to learn the distribution of input elements from sample inputs, then apply the learned grammar as a producer. Specifically, applying probabilistic grammars as parsers, we learn the frequency of occurrence of production rules in sample inputs. Then, we apply the learned PCFG as a producer to serve two major purposes, (1) it ensures that generated inputs are syntactically correct by construction, and (2) it controls the distribution of input elements by assigning probabilities to individual production rules.

This approach allows for three test generation strategies: 1) `Common inputs` – by generating inputs using the learned probabilistic grammar, we can create inputs that are similar to the sample; this is useful for regression testing. 2) `Uncommon inputs` – inverting the learned probabilities in the grammar yields inputs that are strongly dissimilar to the sample; this is useful for completing a test suite with (different) inputs that test uncommon features, yet are syntactically valid. 3) `Failure-inducing inputs` – learning from inputs that caused failures in the past gives us inputs that share similar features and thus also have a high chance of triggering bugs; this is useful for testing the completeness of fixes.

We examined the effectiveness of our approach using 20 subject programs and three input formats. Our experimental results show that "common inputs" reproduced 96% of the program features (i.e. methods) induced by the samples. In contrast, for almost all subjects (95%), the "uncommon inputs" covered significantly different methods from the samples. By learning from failure-inducing samples, our approach reproduced all failures triggered by the sample inputs and also reveals new failures.

We have presented a technique that applies PCFG to generate (dis)similar test inputs. This approach provides a general and cost-effective means to generate test cases that can then be targeted to the commonly used portions of the software, or to the rarely used features.

Bibliography

[HZ19] Havrikov, Nikolas; Zeller, Andreas: Systematically covering input structure. In: 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE). IEEE, pp. 189–199, 2019.

[So20] Soremekun, Ezekiel; Pavese, Esteban; Havrikov, Nikolas; Grunske, Lars; Zeller, Andreas: Inputs from Hell: Learning Input Distributions for Grammar-Based Test Generation. In: IEEE Transactions on Software Engineering. 2020.