Generating Test Inputs and Interactions with the Fandango Fuzzer

How to test complex systems automatically and systematically

José Antonio Zamudio Amaya
CISPA Helmholtz Center for Information Security
Saarbrücken, Germany
jose.zamudio@cispa.de

Alexander Liggesmeyer
CISPA Helmholtz Center for Information Security
Saarbrücken, Germany
alexander.liggesmeyer@cispa.de

Marius Smytzek
CISPA Helmholtz Center for Information Security
Saarbrücken, Germany
marius.smytzek@cispa.de

Andreas Zeller
CISPA Helmholtz Center for Information Security
Saarbrücken, Germany
zeller@cispa.de

Abstract

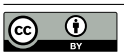
Even in the age of AI, generating comprehensive test inputs for software systems remains a challenge, particularly for *interactive* and *reactive* systems, which require entire *interaction sequences* to reach the desired states or to cover missing behavior. While modern fuzzers are effective at generating random inputs that test the robustness of input processors, they still struggle to produce or mutate complex inputs and interactions. LLM-based systems, on the other hand, can generate small example inputs, but fail to systematically explore the space of inputs and interactions as would be required for comprehensive testing.

In this tutorial, we introduce test generation with *FANDANGO*, a modern specification-based generator of test inputs and interactions. Based on a spec file that defines the structure and properties of the program input and interaction languages, FANDANGO produces inputs that are syntactically and semantically correct, and systematically cover the input space of the program under test.

In three interactive sessions, we (1) cover the basics of specification-based testing, (2) demonstrate how to use constraints to define desired properties, and then precisely (3) delve into full-fledged protocol testing, producing entire interaction sequences between FANDANGO and the network components under test.

Participants are expected to bring a basic understanding of software testing; knowledge of PYTHON is a plus, but not required. At the end of the tutorial, they will (1) be able to specify and test formats for inputs and interactions, (2) produce comprehensive test suites using specification-based techniques, and (3) understand the trade-offs and practical applications of specification-based testing.

ACM Reference Format:
José Antonio Zamudio Amaya, Alexander Liggesmeyer, Marius Smytzek, and Andreas Zeller. 2026. Generating Test Inputs and Interactions with the Fandango Fuzzer: How to test complex systems automatically and systematically. In *34th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (FSE Companion '26), July 05–09, 2026, Montreal, QC, Canada.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3803437.3804899



This work is licensed under a Creative Commons Attribution 4.0 International License.
FSE Companion '26, Montreal, QC, Canada
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2636-1/2026/07
https://doi.org/10.1145/3803437.3804899

1 Aims and Objectives

The primary aim of this tutorial is to equip participants with the knowledge and skills to use FANDANGO [1] effectively for language-based testing. More generally, participants will learn when and how to apply language-based testing techniques, define semantic constraints beyond the capabilities of symbolic constraint solvers, and test complex stateful systems, such as network protocols.

2 Target Audience and Prerequisites

This tutorial is designed for software testing researchers and practitioners interested in advanced techniques for input generation and language-based testing. It is suitable for undergraduate and graduate students, early-career and senior researchers, and industry practitioners looking to enhance their testing methodologies.

Participants should have a basic understanding of software testing concepts and familiarity with programming in PYTHON is a plus. No prior experience with grammar-based testing or search-based testing is required, as the tutorial will cover foundational concepts before delving into FANDANGO's specifics.

3 Relevance

Language-based testing is a key technique for generating valid and diverse inputs for programs that process structured data. However, many existing approaches rely on symbolic constraint solving, which can restrict expressiveness and hinder performance. FANDANGO overcomes these limitations through a search-based approach, addressing a core challenge in modern software engineering: scalable, expressive, and practical test input generation for complex software systems.

This tutorial is highly relevant to the FSE community interested in software testing and quality assurance. It introduces participants to a state-of-the-art approach for testing highly complex, input-driven systems, combining solid theoretical foundations with extensive hands-on exercises. By translating theory into practice, participants gain practical skills applicable in both academic and industrial contexts, designing custom testing workflows that extend beyond FANDANGO.

4 Session Format

The tutorial will be a hands-on session where participants will actively engage in exercises using FANDANGO. The session is built from

individual segments that combine short presentations explaining key concepts and theoretical foundations with practical exercises, allowing participants to apply the concepts they learn immediately.

5 Preferred Duration

The tutorial is designed to be delivered in a 180-minute session.

6 Outline

The tutorial will be structured as follows:

Introduction to Language-Based Testing (10 minutes)

- Overview of language-based testing and its importance.
- Limitations of symbolic constraint solving in existing approaches.

Introduction to FANDANGO (10 minutes)

- Overview of FANDANGO's architecture and key features.
- Comparison with existing language-based testing tools.

Introduction to Grammars (10 minutes)

- Basics of grammar-based input generation.
- Writing grammars for structured input formats.

Hands-on Exercises on Grammar Writing (20 minutes)

- Writing grammars for input generation using FANDANGO.
- Practical exercises to reinforce grammar writing skills.

Short Break (5 minutes)

Defining Semantic Constraints (10 minutes)

- Overview of semantic constraints for input generation.
- Writing constraints in PYTHON for FANDANGO.

Hands-on Exercises on Constraint Definition (25 minutes)

- Writing semantic constraints for input generation.
- Practical exercises to reinforce constraint definition skills.

Introducing Execution Feedback (5 minutes)

- Overview of execution feedback in search-based testing.
- How FANDANGO can leverage execution feedback.

Hands-on Session on Combining Input Constraints and Execution Feedback (20 minutes)

- Using FANDANGO to combine grammars and constraints for coverage-guided input generation.
- Practical exercises to explore FANDANGO's capabilities.

Short Break (5 minutes)

Introducing Protocol Fuzzing (10 minutes)

- Overview of fuzzing protocols using FANDANGO.
- Key challenges in protocol fuzzing.

Hands-on Session on Fuzzing Protocols with FANDANGO (30 minutes)

- Using FANDANGO for protocol fuzzing.
- Practical exercises to show how FANDANGO can participate in stateful interactions.

Discussion and Q&A (20 minutes)

- Open floor for questions and discussions.
- Sharing experiences and insights from the hands-on session.

7 Key Learning Outcomes

By the end of this tutorial, participants will be able to:

(1) Understand the principles of language-based testing and the limitations of symbolic constraint solving.
(2) Write grammars and define semantic constraints to specify complex formats formally.
(3) Use FANDANGO to test complex stateful systems (e.g., protocols), including input generation and output validation.
(4) Showcase the performance and precision of search-based testing approaches compared to symbolic methods.
(5) Apply FANDANGO to their own testing challenges and understand the trade-offs between symbolic and search-based approaches.

8 Tutorial History

The tutorial content has been delivered at the RIO Summer School on Testing and Verification[1], with more than 100 participants.

9 Technical Requirements

Participants will need a laptop with PYTHON 3.11 or higher installed. Before installing, we recommend creating a virtual environment, done as follows:

```
python -m venv .venv
```

To activate the environment, execute:

```
source .venv/bin/activate  # on Unix
.venv\Scripts\activate     # on Windows
```

Installing FANDANGO requires a single `pip` command:

```
pip install fandango-fuzzer
```

10 Recommended Reading

Participants who would like to prepare in advance or continue after the session can consult the original FANDANGO paper [1], which presents the search-based foundations of the tool, its evolutionary solving strategy, and the empirical evaluation. For hands-on work, we recommend the FANDANGO docs at https://fandango-fuzzer.github.io/ [2]. The book is especially useful for attendees because it walks through installation, the basic `fandango fuzz`, `fandango parse`, and `fandango talk` workflow, and the structure of `.fan`: specifications combining grammars, constraints, and PYTHON code.

Acknowledgments

This research was partially funded by the European Union (ERC S3, 101093186). Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

References

[1] José Antonio Zamudio Amaya, Marius Smytzek, and Andreas Zeller. 2025. FANDANGO: Evolving Language-Based Testing. *Proc. ACM Softw. Eng.* 2, ISSTA, Article ISSTA040 (June 2025), 23 pages. https://doi.org/10.1145/3728915
[2] José Antonio Zamudio Amaya, Marius Smytzek, and Andreas Zeller. 2026. Fuzzing with Fandango. https://fandango-fuzzer.github.io.

[1] https://rio2025.dc.exa.unrc.edu.ar