Fully Automated Functional Fuzzing of Android Apps for Detecting Non-crashing Logic Bugs

Evaluation and Development of New Strategy

Caleb Hendrix

Introduction

The Need: Automatically test apps

- Find bugs, Prevent crashes
- Retain users

Existing Tools: Monkey, Stoat, Time-Machine

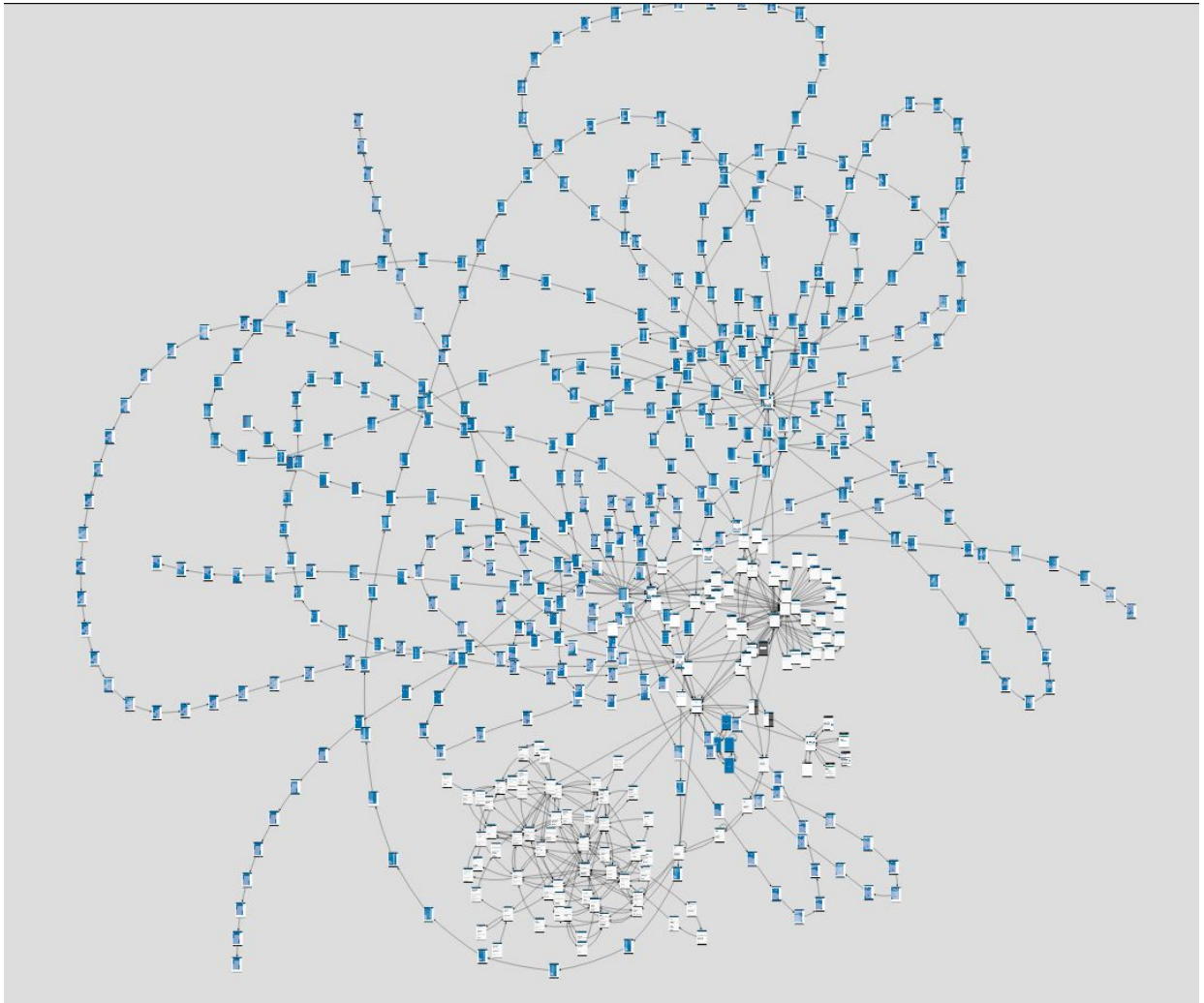
- Good at finding crashes
- Cannot find logic/semantic bugs

Motivation: Finding semantic bugs and crash bugs

- Genie tool can do both

Intro cont.

Genie uses DroidBot to generate a UTG: UI Transition Graph

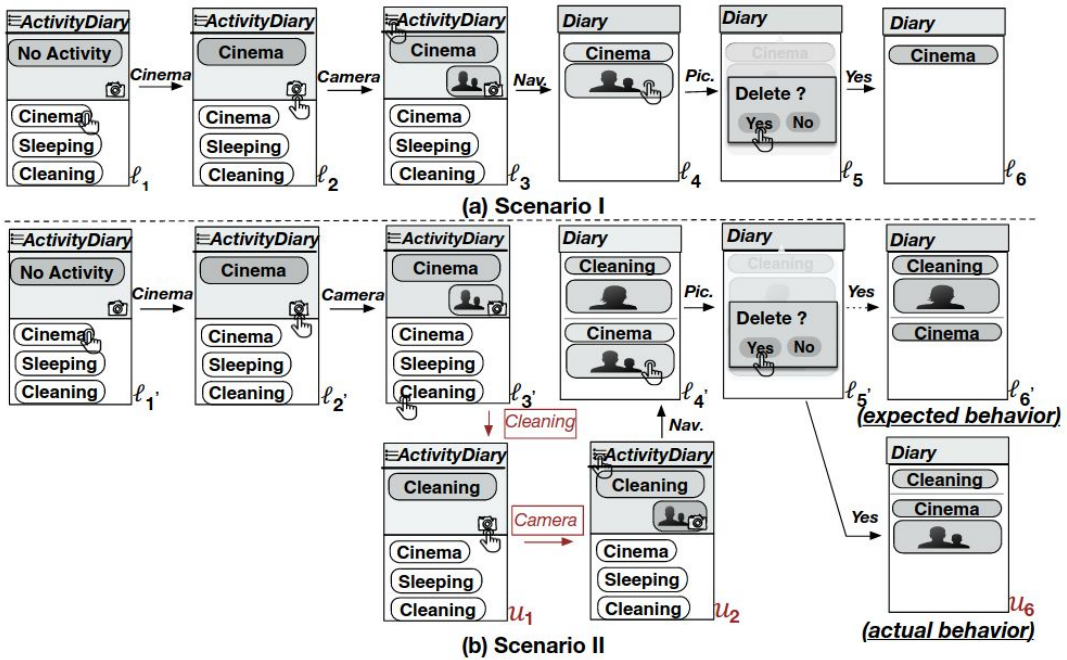


Intro cont.

The UTG is used to generate Seed and Mutant Tests:

Seed:

Mutant:



Intro cont.

The Problem:

- Genie uses a random seed test generation strategy based on Stoat
- The paper cites this as an open challenge

The Goal:

- Implement a systematic method for generating seed tests
- Test apps using the random and systematic methods

Implementation

How to improve on existing strategy:

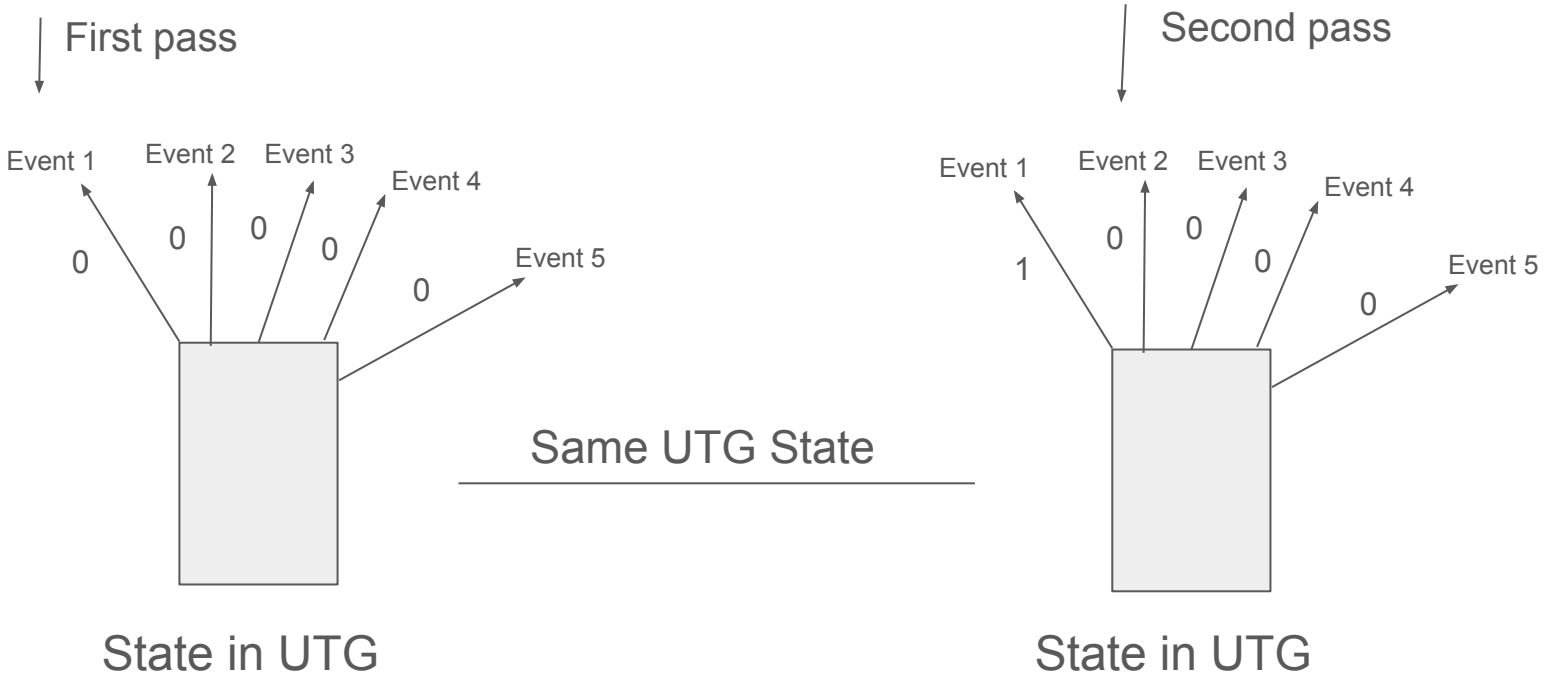
- Genie is currently using Stoat
- Time travel testing has been shown to outperform Stoat

Use the principles of Time Travel testing and systematic generation

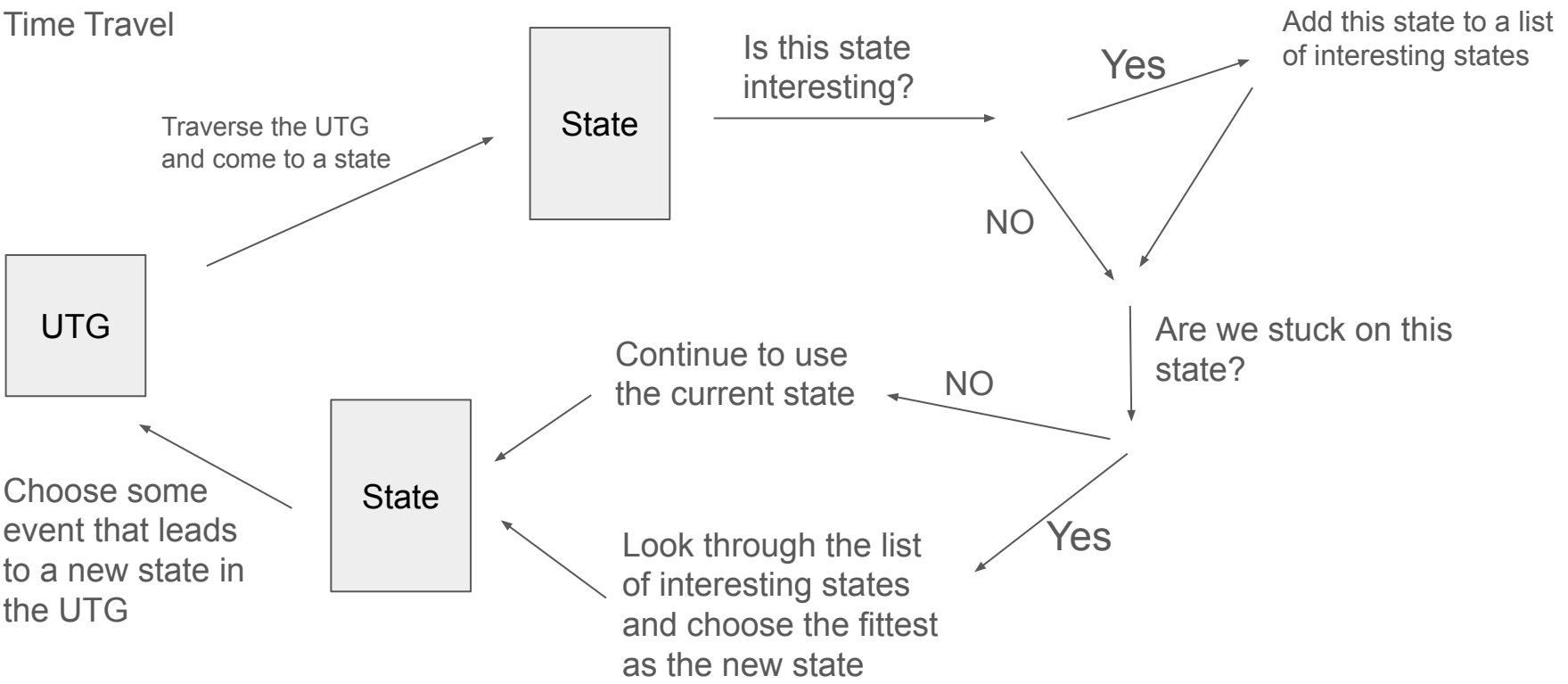
- Find interesting states, know if stuck, go back to interesting states
- Keep track of when we take certain paths
- Traverse paths that have not been taken

Implementation cont.

Systematic Algorithm



Implementation cont.



Evaluation

Identified 2 open source apps for testing

- Simple Accounting
- Privacy Friendly Minesweeper

Different goals for both apps

- Identified 3 known bugs for Simple Accounting
- See how Genie tool will respond to Minesweeper

Both Tested using the default test settings/command arguments

Both Re-tested using my new systematic testing option

Evaluation cont.

Simple Accounting Baseline:

- Evaluated over 10,000 mutant tests
- Identified 2 semantic bugs

Simple Accounting Systematic:

- Evaluated over 4,000 mutant tests
- No semantic or crash bugs identified

Minesweeper Baseline and Systematic:

- Identified no semantic or crash bugs

Evaluation cont.

Improvements to Systematic testing method:

- Longer GUI exploration
- Longer individual seed tests
- Include some user scripts

Questions?

References

Dong, Z., Böhme, M., Cojocaru, L., & Roychoudhury, A. (2020b). Time-travel testing of Android apps. *IEEE/ACM 42nd International Conference on Software Engineering*. https://doi.org/10.1145/3377811.3380402

Li, Y., Yang, Z., Guo, Y., & Chen, X. (2017). DroidBot: a lightweight UI-Guided test input generator for android. *IEEE/ACM 39th IEEE International Conference on Software Engineering Companion*. https://doi.org/10.1109/icse-c.2017.8

Su, T., Meng, G., Chen, Y., Wu, K., Yang, W., Yao, Y., Pu, G., Liu, Y., & Su, Z. (2017). Guided, stochastic model-based GUI testing of Android apps. *Association for Computing Machinery*. https://doi.org/10.1145/3106237.3106298

Su, T., Yan, Y., Wang, J., Sun, J., Xiong, Y., Pu, G., Wang, K., & Su, Z. (2021). Fully automated functional fuzzing of Android apps for detecting non-crashing logic bugs. *Proceedings of the ACM on Programming Languages*, *5*(OOPSLA), 1–31. https://doi.org/10.1145/3485533