Submission for Debugging and Fuzzing Exercise 8

Christian Klempau

February 5, 2023

1 Design

The fuzzer itself is programed in **C lang**. The choice was based on a number of factors:

- Access to low level Linux / OS APIs (discussed later)
- Speed and small memory footprint
- Connection with Python (DD was made in Python)

The main strategy consisted on trying to generate interesting WCNF problems, try to parse them on a buggy parser, and if an invalid return code occurs, send that file to the Python Delta Debugger to try to minimize it.

To generate interesting cases, the approach consisted on:

- Have different generator functions. Some more aggressive on expanding the problem, others more naive, and others more conservative to keep the problem small, but while adding noise.
- Prioritize small problems, but still expand onto larger ones if we do not find bugs.
- Use a uniform probability distribution, to ensure edge cases were met. We sample floats from $[0, 1]$ uniformly, to gate the rate of hard v/s soft clauses, and the rate of present / negated variables (separately sampled).

As I mentioned before, the program uses low level OS syscalls, for some of it's features. The main one, being opening a file pointer in RAM (specifically, $/tmp$, using $tmpnam(NULL)$), instead of disk. This allows us to quickly read and write to this file pointer, to execute a big magnitude of test cases without a problem. We **ONLY** write to disk when we find a bug, or when we want to delta debug it.

The fuzzer finds around 5 different invalid return codes between 30 and 39, in less than one minute. To get more codes, especially the ones >= 40, it needs a lot of execution time, and better strategies. It even found a segmentation fault in the parser, but I forgot to save the seed and lost it.

For a basic execution:

```
$ make && ./fuzzer
```

Workflow

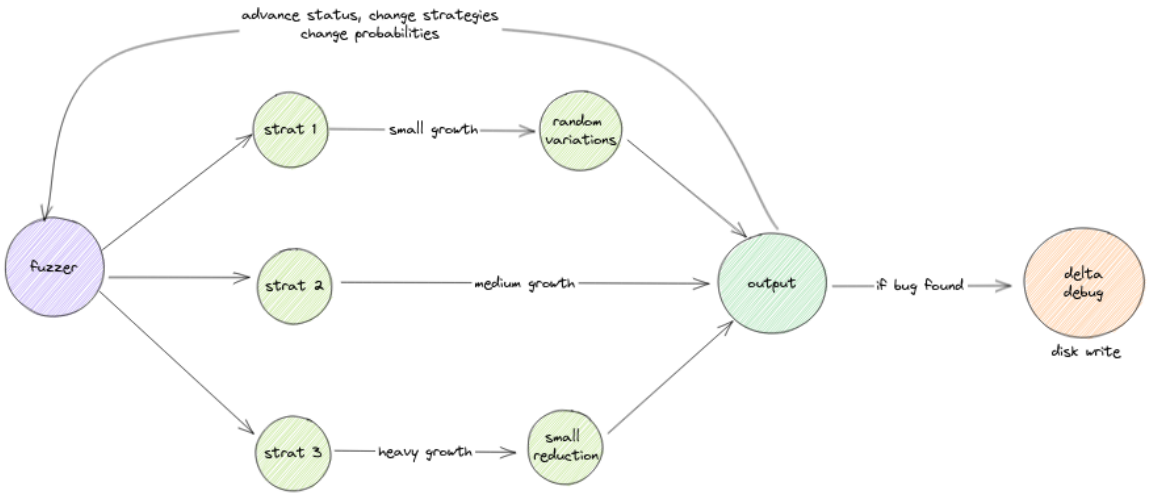

Figure 1: Program's workflow

2 Output

Example input / outputs:

```
➜ fuzzer git:(main) ✗ ./fuzzer
file name: /tmp/fileEAvWpg
Seed: 1675611439

-------------- Type: 1 - Growth: 0 - N_ITERS: 1000 --------------

CODE: 0


-------------- Type: 1 - Growth: 1 - N_ITERS: 5000 --------------

CODE: 34

file name: delta_debugger_output/dd_original_34.wcnf
DONE! Minimal file is at delta_debugger_output/dd_original_34.wcnf_minimal.wcnf
Reduced to 1 lines

CODE: 33

file name: delta_debugger_output/dd_original_33.wcnf
DONE! Minimal file is at delta_debugger_output/dd_original_33.wcnf_minimal.wcnf
Reduced to 1 lines

CODE: 31

file name: delta_debugger_output/dd_original_31.wcnf
DONE! Minimal file is at delta_debugger_output/dd_original_31.wcnf_minimal.wcnf
Reduced to 1 lines

-------------- Type: 1 - Growth: 2 - N_ITERS: 9000 --------------

-------------- Type: 2 - Growth: 0 - N_ITERS: 1000 --------------

-------------- Type: 2 - Growth: 1 - N_ITERS: 5000 --------------
```

Figure 2: Execution example

```
CODE: 31

file name: delta_debugger_output/dd_original_31.wcnf
DONE! Minimal file is at delta_debugger_output/dd_original_31.wcnf_minimal.wcnf
Reduced to 1 lines

-------------- Type: 1 - Growth: 2 - N_ITERS: 9000 --------------

-------------- Type: 2 - Growth: 0 - N_ITERS: 1000 --------------

-------------- Type: 2 - Growth: 1 - N_ITERS: 5000 --------------

-------------- Type: 2 - Growth: 2 - N_ITERS: 9000 --------------

CODE: 32

file name: delta_debugger_output/dd_original_32.wcnf
No reduction found

-------------- Type: 3 - Growth: 0 - N_ITERS: 1000 --------------

-------------- Type: 3 - Growth: 1 - N_ITERS: 5000 --------------

CODE: 38

file name: delta_debugger_output/dd_original_38.wcnf
No reduction found

-------------- Type: 3 - Growth: 2 - N_ITERS: 9000 --------------
➜  fuzzer git:(main) ✗
```

Figure 3: More examples of executions

3 About

Time taken: around 25 hours in total. I implemented a basic fuzzer in around 6 hours, but to make it interoperable with the delta debugger, to find better strategies and to optimize everything, took another 20 hours.

Difficulties: modelling the problem correctly, to ensure that a wide range of SAT problems ocurred.