Paper Review 《Detecting Incorrect Build Rules》

Paper Info

Nandor Licker, Andrew Rice

ICSE 2019

Main Contribution

This paper is aimed at the detection of incorrect build rules. The aim and the approach of this work are both similar to Vemake. The core insight of this work is build fuzzing, which refines the dependency graph which is constructed based on IO information. Mining in the dependency graphs can help them find race conditions. Another shining point of this work is the unbelievably large scale of experiments. They perform the experiments on 500 open-source projects, classify the results by their types and obtain the common patterns of incorrect build rules. They also discuss the causes of this incorrect build rules and whether it is necessary to fix these incorrect rules or not.

Main Work

The work consists of three parts:

- Dependency graph construction: Similar to Vemake, the tool in this paper also makes use of `strace` to monitor the system calls and obtains the file dependency relations. Based on this information, the authors propose algorithm 1 to construct the dependency graph. However, the dependency graph might be over-constraining and under-constraining, hence it needs refinements, which is achieved by build fuzzing in algorithm 2.
- Dependency graph refinement based on build fuzzing: They change the content of each file(add some blank lines at the end of the file). This can trigger the incremental build. According to the dependency graph returned in algorithm 1, the expected changed files can be obtained. Compared the expected changed files with actual changed files, it is easy to get the missing edges and duplicated edges in the dependency graph. The dependency graph can be refined in this way.
- Race condition detection: Based on the original dependency graph returned in algorithm 1 and the refined dependency graph returned in algorithm 2, it is easy to find race condition just by comparing the file dependencies of each file in two graphs. If they are not equal, race condition exists.

In the evaluation part, they describe the experimental settings and perform a large range of experiments on 500 open-source projects. They classify the causes of incorrect build rules and discuss the necessity of fixing them.

Some Criticisms

Although this work is rewarded as the distinguished paper in ICSE 2019, the approach is limited in several aspects. For example, it can not detect all the missing dependencies and race conditions. Only some of them can be found in their approach.

They claim that it involves substantial engineering effort if they parse build definitions by creating custom parsers, but it might be the only way to extract the precise expected dependency relations. In theory, we can find all of the reports of a specific dependency issue.

Moreover, build fuzzing is time-consuming. They change the content of each file and try to trigger the incremental build. Almost all the files need to be tested one by one, so this process is quite time-consuming. However, Vemake only builds several times, and the number of builds is determined by the number of top targets.