《Fault Localization for Make-Based Build Crashes》Review

Paper Info

《Fault Localization for Make-Based Build Crashes》 Jafar Al-Kofahi, Hung Viet Nguyen, and Tien N. Nguyen

Major Contribution

The paper introduces an automatic tool to localize faults in build code that cause run-time build failures. GNU make tool can not capture the fault localization and can only report the location of build crashes. This is difficult for developers to localize the fault only based on the error message provided by the GNU make tool. To localize a fault, a tool needs to analyze not only the faulty concrete rules but also their originating code in Makefiles.

The authors propose some models to track the generation of concrete rules and the dependencies, including E-trace model, CDG(Concrete Dependency Graph). When a crash occurs, fault can be localized based on CDG. Based on E-trace model, the lines of code in Makefile which generates the fault can be found. This is the major contribution of their work.

In their work, they use a novel Bayesian-like rating algorithm to give suspiciousness scores to the original statements in the Makefile. It is the core insight of their work.

Main Work

As explained above, the main contribution of this work is the fault localization of Make system crashes. Because Make processes a Makefile in two distinct phases including evaluation phase and execution phase, the authors try to record the intermediate result of these two phases.

CDG(Concrete Dependency Graph) is a tree model representing the dependency relations between targets and prerequisites. This is the most common used model in Make system analysis.

E-trace model represents how the concrete build rules in a CDG are computed and manipulated through Makefile's program elements. A node in E-trace model refers to an expression at a code location in the Makefile. The edges represent the evaluation flows among those expressions.

When a crash is reported by GNU make tool, we can get the crash point based on the message provided by GNU make tool. Then they distribute fault probabilities over different nodes in the CDG. The incorrect state at the crash point can be caused by the concrete rule at the crash point being incorrect itself, or by the incorrect state of one of the rules for the preceding target or prerequisites. Firstly, assign the probability of 100% to the crash point. Then the distribution of probabilities continues for the nodes at the preceding target and prerequisites. These probabilities decrease multiplicatively with the number of nodes as the process descends in the CDG further away from the crashing node.

Given the fault probability of a concrete rule, the tool distributes that probality over all the code locations in the Makefile that are responsible for generating the conrete rule based on the E-tace model of that rule. Finally, they can get a score list recording suspiciousness scores of original statements in Makefile, and display them in a descending order of the score.

Some Criticism

This paper is a good work of fault localization for Make system crashes. However, it has several limitations and needs to be discussed and improved.

- The paper is based on GNU make tool, and only reports the fault when a crash occurs in the process of make. This is often not enough in the real-world practice. Sometimes GNU make tool does not report any error messages and can build correctly, but the faults still exist in Makefiles, such as some duplicated prerequisites. This kind of faults can not be detected by their tool.
- Moreover, their Bayesian-like rating algorithm needs to be improved. They suppose the fault probabilities decrease multiplicatively with the number of nodes, and the child of the current node in CDG has the same probability of crash. This assumption is lack of evidence.
- In this paper, the authors use E-trace to distribute fault probability over all the code locations in the Makefile that are responsible for generating the concrete rule. Although E-trace model is proposed in other works of Make system analysis, it is not necessary in this work. GNU make tool has a option `-profile`, which can track the generation of concrete rules.