Review, Bugs as Deviant Behaviour: A General Approach to Inferring Errors in System Code

Robert Hoff

November 2, 2011

1 Summary

An existing technique [1] for finding bugs is based on analysis against a fixed set of specified rules, but defining these rules manually is difficult and error-prone. The technique introduced [2] examines the code by static-analysis to infer such rules automatically. On applying rules against real systems the methodology uncovered many bugs with a low rate of false-positives, that has helped improve existing open-source kernels.

2 The Problem

Finding bugs relies on knowing what is correct, but correctness is not usually specified for a given piece of software. The paper introduces the idea of inferring rules automatically just by analysing the code, without having a-priori knowledge about it's operation.

There are a number of approaches available to finding bugs, for example testing and manual inspection. Built-in approaches include type systems and high level compilation enforce a correctness that eliminates a large class of bugs. Similar type of rules are defined in dynamic invariant inference. For example Daikon and Eraser. The idea of finding rules by templating and static analysis is new in this paper.

3 The Solution

Rules and bugs can be inferred by registering logical inconsistencies or deviations from normal behaviour. For example of pointer inconsistency, if dereferenced and later checked if null this indicates a logical error. Another method monitors the number of times a function was checked in a certain manner, if checking deviates from the normal, statistical analysis determines if it is a possible error.

The analysis is implemented in metal, a system-specific compiler extension to gcc. The rules are applied down each function execution path, and results are recorded for comparison checks.

4 Evaluation

The experiments show that the automated technique of finding bugs reveals one or two orders of magnitude more bugs than previous techniques specifying them manually. But some knowledge of the system is required as one must specify general templates for the rules.

The technique uses static analysis, so it can find bugs without running the code, and can pinpoint their location.

5 My Opinion

The methodology requires writing the compiler extensions, but an opportunity may be to introduce the mechanism in existing compilers. Automation is in some cases superior to human review, at least a useful addition. The reported bugs still need to be rewritten by a human!

6 Questions

1. Are there any type of bugs that the system have problems finding, or cannot find?

2. In what ways could the method be improved by dynamic monitoring?

References

[1] Engler D., Chelf B., and Chou A. Checking system rules using system-specific, programmer-written compiler extensions. *Proceedings of the 4th conference on Symposium on Operating System Design & Implementation*, 2000.

[2] Engler D., Chen D.Y., Seth H., Chou A., and Chelf B. Bugs as deviant behaviour: A general approach to inferring errors in system code. *Proceedings of the eighteenth ACM symposium on Operating systems principles*, Dec 2001.