Review 20

4/3/12

CS6V81.502

Seungtack Baek

**Bugs as Deviant Behavior:**

**A General Approach to Inferring Errors in Systems Code**

In this Paper, Engler et al. demonstrates how to find errors in System code (such as Linux and OpenBSD). To find errors, they had to infer rules (what they called, *beliefs –* not *truth*) from the source code without a priori knowledge of the code itself. In the process of inferring rules, they took the source code and inferred beliefs from each statement.

Now they have developed a tool to infer errors in Systems Code, they applied this tool to Linux and OpenBSD. While applying this tool, they took four different approaches to find different types of errors.

1. Internal null consistency: to find all pointer errors
2. Security Checker: to check for a rule “no dereference on null pointer”
3. Inferring failure: to check if a function, *f*, is checked for failure
4. Deriving temporal rules: to check for two rules “no ‘a’ after ‘b’” and “‘b’ must follow ‘a’”

As a result, they found a number of yet-to-be-fixed bugs and “new” bugs traditional methods would not have found. For an example, they found an unanticipated error in IS\_ERR, where a function is checked against a null pointer, even if the function does not return null pointer on failure, but a bogus pointer. The authors claim that this bug would have not been found by other tools/methods, yet this tool found it.

Yet, this finding makes me wonder, “Why wasn’t it caught before?” If there has been an issue with Linux, originated in this bug, this bug should have been caught before. Or, maybe this piece is never included in the execution slice in any of the testing cases. Yet, the fact that this tool found unforeseen error seems very attractive. Towards the end of this paper, they believed that “this will be a profitable approach.”

As I was reading this paper, I was afraid that they will only do two types of checks, null pointer dereference and unprotected variable by lock. Yet, they also included statistical approach to find more errors through machine learning. Aforementioned, this tool found couple of very interesting bugs. Overall, I am convinced that this tool is very promising.

**Question:**

1. What will be a disadvantage of using machine learning approach for this type of problem?