C.P.A. Paper Review
Lei Yang
2005-5-18

****

Title

Bugs as deviant behavior:
A general approach to inferring errors in systems code

Author

Dawson Angler, David U Cohen, Set hall em, Andy Hour, and Benjamin Chef

Summary

This paper demonstrates bug-finding techniques that extract checking information from the source code instead of programmers, requiring no priori knowledge of system rules.

Most important ideas

This paper shows how to automatically find bugs in a system without having a priori knowledge of the system correctness rules. This approach tried to determine programmer's intention, which they express as "beliefs". They collects sets of programmer beliefs, and check them for contradictions. "Belief" is the key concept in this paper: beliefs are facts about the system implied by the code. They examined two types of beliefs: MUST beliefs and MAY beliefs. For MUST beliefs, they just need to look for contradictions; for MAY beliefs, they must first separate valid beliefs from coincidences (they used statistical analysis to rank each error by the probability of its beliefs.) The essential part of this work is that it eliminates the need to understand the system in any deep way: a contradicted belief must be an error, although you may not know what is correct. So you can find a bug without knowing the truth, where contradiction is the key. They implemented six checkers that follow this conceptual framework. They found hundreds of bugs in real system such as Linux and OpenBSD.

Their discussion of related work, the discussion of difference between their approach and the other approaches is very nice.

Flaws/Questions

(1). Not a criticism, but I am not clear of the description on "building block hypotheses". Does that have the same meaning of the "template" used in this paper. (2). As to the implementation of checkers, why they used xgcc, instead of gcc? Section 3.5 is just not clear to me. (3). They've stated that static analysis is better than dynamic analysis in bug-finding. Because dynamic approach only sees executed paths. This is convincing. But it seemed to me that they didn't explain explicitly how they explore paths in a static way. Did they search through every possible way.

Relevance/Potential Future Research

Machine learning and graphical model should be useful for this work. But on a Google search and the author's home page I didn't see any follow-up paper on this. Belief Networks might also be useful on this work in that it could help predict the probability of a future error, based on a history of errors.