Paper summary

Identifying automatic bugs in system code is a significant challenge as defining a correctness rule is an arduous task. This paper finds bugs automatically by identifying behavior that is "deviant"(departing from the usual standards) from the source code itself. The system automatically generates beliefs by noticing patterns and comparing them to contradictions which can be an error or a coincidence. There are two sets of beliefs: MUST belief and MAY belief. A MUST belief implies that it should always hold. E.g., Division by z means z is non-zero, or a pointer dereferences indicates that a pointer is non-null. It uses internal consistency for finding MUST beliefs. A MAY belief suggests a belief that may be a coincidence. We assume that MAY beliefs are MUST beliefs and look for a violation of these beliefs and use statistical analysis to rank each error by the probability of its belief. If a particular belief is persistent, then it's probably a reasonable belief. E.g., Enclosing in locks may mean locking is required, or a() followed by b() may mean functions a and b must be paired. It specifies a general template for a rule. Once we know a rule, we can check it with the help of an extended compiler. Therefore, there is an exponential reduction in manual effort and finding rule instances automatically.

Strengths

1. Lead to many kernel patches in the Linux and OpenBSD systems.
2. Able to find unforeseen errors.
3. No prior knowledge is required.
4. The code examples gave a much better clarity of the system.
5. It possibly covers more rules than dynamic analysis like Eraser.
6. It tells what the error is with the help of an extended compiler.
7. It reduces manual efforts.

Weaknesses

1. Probably slower than dynamic analysis.
2. Eraser didn't require the source code. It works on the application binary. However, here the access to source code is required. For closed source programs like windows, this can be an issue.

Comments for author

An exciting idea to infer the rules from the source code itself! Structured and easy-to-follow language. The system looks highly scalable as well! However, some empirical data supporting that evidence would be helpful. Also, some statistical comparison with dynamic analysis would bring more clarity.