The Soundness of Bugs is What Matters (Position Statement)

Patrice Godefroid
Bell Laboratories, Lucent Technologies

There is one thing stronger than all the armies in the world; and that is an idea whose time has come. – Victor Hugo

In this short note[1], I argue that most program analysis and verification research seems confused about the ultimate goal of software defect detection.

The Goal is to Find Bugs. The main practical usefulness of software defect detection is the ability to *find bugs*, not to report that "no bugs have been found". Unfortunately, the latter is sometimes confused for a *correctness proof*. In practice, there is no such thing as a *complete* correctness proof, since even a *sound analysis* implemented flawlessly in a bug-free tool is bound to check only a specific set of properties.

So, Why May-Analysis? Yet, most defect detection tools are surprisingly based on program verification ideas and make use of *conservative abstractions*. By design, such tools detect bugs that *may* happen. The price to pay for this questionable design decision is enormous: such tools are *doomed* to report (many) false alarms, i.e., *unsound bugs*. Despite progress on limiting false alarms (e.g., by using more precise symbolic execution or alarm classification techniques), any *may* program analysis is bound to generate false alarms and hence to require (significant) human effort.

The Importance of Testing. This may explain why *testing* has been a multi-billion dollar industry for many years, while automated code-inspection is merely emerging as a multi-million dollar business. Today, several orders of magnitude more effort (people, money, time) is spent on testing than on code inspection (manual and automated). The reason is simple: *testing finds sound bugs* while *may-analysis does not*.

A Paradox. *If defect detection is the goal, why are so many defect detection tools based on may-analysis?*

Sources of Imprecision. It is important to distinguish *two distinct* sources of imprecision in program analysis: (a) since we want to analyze *open* programs, we need realistic *environment assumptions*; (b) using abstraction implies approximate reasoning. While (a) is a hard problem (i.e., often requires user assistance), (b) is simply an *engineering issue* (see below).

Alternatives. After realizing that *"The Soundness of Bugs is What Matters"*, alternatives emerge. Here are three concrete examples, drawn from my own work:
1. VeriSoft is a software model checker for languages like C and C++ which uses a run-time scheduler for systematically driving the executions of a software implementation through its state space (no abstraction is used). Since this search is typically incomplete, VeriSoft sacrifices, by design, soundness of correctness proofs ("soundness") for soundness of bugs.
2. Must abstractions are abstractions geared towards finding errors, which dualize may abstractions geared towards proving correctness. With combined may/must abstractions, *both correctness proofs and bugs found* are guaranteed to be sound.
3. DART (Directed Automated Random Testing) is a new approach and tool (see PLDI'2005) for *automatically* testing software (i.e., no test driver needed) that combines (1) *automated* interface extraction from source code, (2) *random* testing at that interface, and (3) dynamic test generation to *direct* executions along alternative program paths. Although DART uses imprecise abstraction techniques, all bugs found by DART are guaranteed to be sound, by design.

Role of "Soundness". In my opinion, "soundness" (to find *all* potential bugs of a particular type) is important not for exhaustiveness reasons but only for efficiency reasons: the ability to prove the absence of bugs can be used to stop a dual search for sound bugs.

Conclusion. The *May* and the *Must* are the Yin and the Yan of program analysis. Yet, past research in program analysis and verification has tilted the balance towards the *May* and hence prevented a wider adoption of program analysis tools. I suggest to repair this imbalance by restoring the proper role of *Must*, i.e., to put *the soundness of bugs first*.

[1] Written in a provocative style for entertainment purposes – please take this note with the grain of salt it deserves.