Hazards of Verification

Daniel Jackson

MIT CSAIL

> It is insufficiently considered that men more often require to be reminded than informed.
>
> Samuel Johnson

Abstract. Great progress has been made in software verification. One has only to look, for example, at the growth in power of SAT and its widening class of applications to realize that concerns early on that verification was fundamentally intractable and that it would remain a largely manual activity were mistaken. Nevertheless, despite many research advances, verification is still not widely applied to software. In order to make verification practical, some fundamental obstacles will need to be addressed. This talk outlined a sample of these obstacles, in three categories: technical obstacles in the mechanism of verification itself; engineering obstacles related to the problem of establishing confidence in a system rather than a single component; and social and managerial obstacles related to the process of verification.

These obstacles are known and have been fairly widely discussed. It is useful, however, to remind ourselves of them since otherwise, as a research community, we risk focusing on technical details that may be overwhelmed by these larger factors. In explaining some of the obstacles, I also sought to question some assumptions that are often made in the verification community that make sense in a narrower, mathematical context but are not defensible in the larger system context.

One example of this is the notion of conservatism. It is often assumed that a 'conservative' analysis that never misses bugs, but which may generate false alarms, is necessarily superior to an analysis that is 'unsound' and may fail to report real errors. In practice, however, this superiority is not inevitable, and in many cases, an unsound analysis is preferable. First, the need to be conservative can result in a dramatic increase in false positives, so that the unsound analysis may actually be more accurate (in the sense that the probability of missing a bug is low, whereas the comparable conservative analysis produces a report full of erroneous claims). Second, the presence of false alarms may increase the burden of reading the report so dramatically that bugs may be overlooked in the filtering process, so that the conservative analysis becomes, in its context of use, unsound. Similarly, in a software design context, the assumption that checks should be conservative is often not justified in engineering terms. In air-traffic control, for example, it has long been recognized that a conflict detection tool must not generate false alarms. The catastrophic accident in Guam in 1997, in which a 747 flew into the ground, might not have occurred had a safe-altitude-warning system not been disabled because of its over-conservative error reporting.

H. Chockler and A.J. Hu (Eds.): HVC 2008, LNCS 5394, p. 1, 2009.