Exploiting Partial Success in Applying Automated Formal Methods

Matthew B. Dwyer

Department of Computer Science and Engineering, University of Nebraska,
Lincoln NE 68588-0115, USA
dwyer@cse.unl.edu

Abstract. The past decades have produced a wide-range of formal techniques for developing and assessing the correctness of software systems. Techniques, such as various forms of static analysis, automated verification, and test generation, can provide valuable information about how a system satisfies its specified correctness properties. In practice, when applied to large modern software systems all existing automated formal methods come up short. They might produce false error reports, exhaust available human or computational resources, or be incapable of reasoning about some set of important properties. Whatever their shortcoming, the goal of *proving* a system correct remains elusive.

Despite this somewhat dire outlook, there have been enormous gains in the effectiveness of a range of automated formal methods. Rather than looking for a *silver bullet* of a formal method, we ought to admit that no one method will be effective for all properties on all software systems. We should embrace the wealth of existing techniques by trying to characterize their relative strengths and weaknesses across a range of properties and software domains.

Moreover, we should exploit the conventional wisdom that software systems are *mostly correct* – systems have much more correct behavior than incorrect behavior. Given this we should shift from focusing on proving correctness, to developing automated formal methods that calculate the set of system behaviors that are consistent with system specifications. Clearly if the specification-consistent set of behaviors is the set of all behaviors, then the property is proved, but that will rarely be the case.

It is likely, however, that methods will be able to demonstrate that large sets of behaviors are specification-consistent. This type of *partial evidence* of correctness will be most valuable if evidence from multiple techniques can be combined. Equipped with a rich suite of evidence-producing formal methods, where the weakness of each method is masked by the strength of another, and a means for combining their partial evidence we will be well positioned to target the verification and validation of modern software systems.

Acknowledgments

We thank Elena Sherman and Sebastian Elbaum for discussions related to this work. This work was supported in part by the National Science Foundation through awards CCF-0747009 and CCF-0915526, the National Aeronautics and Space Administration under grant number NNX08AV20A, and the Air Force Office of Scientific Research under award FA9550-09-1-0129.