Scalable Automated Verification via Expert-System Guided Transformations

Hari Mony[1], Jason Baumgartner[1], Viresh Paruthi[1],
Robert Kanzelman[2], and Andreas Kuehlmann[3]

[1] IBM Systems Group, Austin, TX
[2] IBM Engineering & Technology Services, Rochester, MN
[3] Cadence Berkeley Labs, Berkeley, CA

Abstract. *Transformation-based verification* has been proposed to synergistically leverage various transformations to successively simplify and decompose large problems to ones which may be formally discharged. While powerful, such systems require a fair amount of user sophistication and experimentation to yield greatest benefits – every verification problem is different, hence the most efficient transformation flow differs widely from problem to problem. Finding an efficient proof strategy not only enables exponential reductions in computational resources, it often makes the difference between obtaining a conclusive result or not. In this paper, we propose the use of an *expert system* to automate this proof strategy development process. We discuss the types of rules used by the expert system, and the type of feedback necessary between the algorithms and expert system, all oriented towards yielding a conclusive result with minimal resources. Experimental results are provided to demonstrate that such a system is able to automatically discover efficient proof strategies, even on large and complex problems with more than 100,000 state elements in their respective cones of influence. These results also demonstrate numerous types of algorithmic synergies that are critical to the automation of such complex proofs.

1 Introduction

Despite advances in formal verification technologies, there remains a large gap between the size of many industrial design components and the capacity of fully-automated formal tools. General exhaustive algorithms such as symbolic reachability analysis [1] solve a PSPACE-complete problem and are limited to design slices with significantly fewer than one thousand state elements. Overapproximate proof techniques such as induction [2] solve an NP-complete problem and may be applied to significantly larger designs, though are often prone to inconclusive results in such cases. Consequently, even a piece of an industrial processor execution unit (much less an entire chip) is likely to be too large for a reliable application of automatic proof techniques.

Technologies such as bounded model checking (BMC) [3] and semi-formal verification [4, 5] address the simpler NP-complete problem of exhaustive bounded search, leveraging the bug-finding power of formal algorithms against much larger designs. Though *incomplete*, hence generally unable to provide proofs of correctness, such applications have become prevalent throughout the industry due to their scalability and ability to efficiently flush out most design flaws. Nevertheless, once such approaches exhaust their ability to find bugs, the end-user is often left debating how many resources to expend before giving up and hoping that the lack of falsification ability is as good as a proof.

The concept of *transformation-based verification* (TBV) [6] has been proposed to synergistically apply various transformation algorithms to simplify and decompose large problems into sufficiently small problems that may be formally discharged. While the complexity of a verification problem is not necessarily related to its size, the complexity class of the algorithms indicates an exponential worst-case relationship between these metrics, which is validated by practical experience. By resource-bounding any possibly costly BDD- or SAT-based analysis, it is possible to limit the complexity of most transformations used in a TBV system to polynomial while exploiting their ability to render exponential speedups to the overall verification flow as noted in [6, 7].

The strength of TBV is based upon the availability of a variety of different complementary transformations which are able to successively chip away at the verification problem until it can be handled by a terminal decision procedure. We have found that the power of TBV is often able to yield a proof for large problems which otherwise would be candidates only for falsification techniques. However, in cases, achieving such results requires a fair amount of user sophistication and trial-and-error – every verification problem is different, hence the most efficient transformation flow varies widely from problem to problem. Given a TBV system with a finite number of algorithms, each with a finite number of discretely-valued parameters, there are a countably infinite number of distinct proof strategies that could be attempted. Finding an efficient proof strategy not only entails exponential reductions in overall computational resources, it often makes the difference between obtaining a conclusive result or not.

In this paper, we propose the use of an *expert system* to automatically guide the flow of a transformation-based verification system. We discuss the type of rules used by the expert system to ensure that commonly-useful, low-resource strategies are explored first, then gradually more expensive strategies are attempted. We have found this approach useful for quickly yielding conclusive results for simpler problems, and efficiently obtaining more costly yet conclusive strategies for more difficult problems. We additionally discuss the type of feedback necessary between the TBV system and the expert system, needed to enable the expert system to effectively experiment with proof strategies. Lastly, we discuss the learning procedure used by the expert system to ensure that it leverages the feedback of previous experimentation in its quest for the best-tuned proof strategy for the problem at hand – ultimately seeking a conclusive result. Experimental results are provided to demonstrate that such a system is able to automatically yield proofs of correctness for large designs (with more than 100,000 state elements in their cones of influence) by maximally exploiting the synergy of the transformation and verification algorithms within the system against the problem under consideration.

Mechanizing the application of proof strategies is not a new concept; it is an essential component of most general-purpose theorem provers, e.g., HOL [8], PVS [9], and ACL2 [10]. However, the presented TBV approach is well-tuned for the verification of safety properties of hardware designs, incorporating numerous specialized transformations that are applicable to large systems. Finding a good scheduling and parameter setting for these transformations is non-trivial, though key to full automation.