A Very Simple $\LaTeX\,2_\varepsilon$ Template

Vitaly Surazhsky
Department of Computer Science
Technion—Israel Institute of Technology
Technion City, Haifa 32000, Israel

Yossi Gil
Department of Computer Science
Technion—Israel Institute of Technology
Technion City, Haifa 32000, Israel

January 1, 2015

Abstract

This is the paper's abstract ...

1 Introduction

Correctness of computer systems is important in our software dependent society. We depend more and more on this modern computer systems. Such systems consist of complex hardware and software components. So correctness of both hardware and software is important. However, it is much harder to ensure the correctness of the software part of these systems. As the complexity of the software parts are getting more complex than that of the underlying hardware. And manual inspection of complex software is error-prone and costly. Numerous formal tools to find functional design bugs in hardware are available and in wide-spread use. In contrast, the market for software verification tools is still in its infancy. A lot of research in this field is going on. We need highly automated method that provides rigorous guarantee of quality. These methods should be scalable enough to match the enormous complexity of software systems. Bounded model checking (BMC) of programs is one of such methods used for software verification. In this paper we will explore LLBMC, an implementation of the idea of Bounded model checking.

Outline The remainder of this article is organized as follows. Section 4 gives account of previous work. Our new and exciting results are described in Section 5. Finally, Section 6 gives the conclusions.

2 Verification problem

In industry we see the use of testing to ensure software quality and even to find bugs. However such quality guarantees do not reflect to "correctness" of the system. As we know that ensuring the absence of all errors in a design is usually too expensive. So all testing based approaches e.g. random testing and automated test-case generation are incomplete. The other approach depends on the fact that systems can be viewed as mathematical objects with well-specified behaviour. Or we can specify the system (intended behaviour) using mathematical logic. Then one can reason about whether the system meets its specification or not. This field of study has been active and it is often referred to as formal methods. The *verification problem* is: Given program M and specification h determine whether or not the behaviour of M meets the specification h. The specification is formulated in mathematical logic. The model checking problem is an instance of the verification problem. Model checking provides an automated method for verifying finite state systems and checks it for certain properties. The method is algorithmic and often efficient because the system is finite state. If the check fails, that means the design has a bug. And, most of the time, the model checkers generate a counterexample how the bug is produced so the developer can easily figure out what is the mistake.

3 Bounded Model Checking

Model checking is a method for formally verifying finite-state concurrent systems. Specifications about the system are expressed as temporal logic formulas, and efficient symbolic algorithms are used to traverse the model defined by the system and check if the specification holds or not. Extremely large state-spaces can often be traversed in minutes.

Bounded Model Checking is first proposed by Biere et. al. in 1999 [?]. It doesn?t reduce the complexity of model checking however, it can solve many cases which cannot be solved by BDD (binary decision diagram) based techniques. However, BMC has the disadvantage of not being able to prove the program is valid because of its bounded state structure.

Idea of BMC is searching for a counterexample in the given problem until an execution length n. The BMC problem can be efficiently reduced to a propositional satisfiability problem and can be solved by SAT solvers. Modern SAT solvers can handle propositional satisfiability problems with 6-7 digit variables and sometimes more.

However, as stated previously, BMC approach is used for finding bugs, not for proving the program is valid because of the execution length. A greater execution length might show bugs which cannot be found for smaller execution lengths.

In formally, bounded model checking contains: A transition system, a temporal logic formula and a user supplied bound k. Using these properties, we check if the given formula satisfied in the given bound.

In our work, the program flow is described as transition states for the BMC conversion. The easy transform between BMC and SMT formulas is highly beneficial because of the presence of highly optimised SMT solvers.

4 Previous work

A much longer LaTeX 2_ε example was written by Gil [?].

5 Results

In this section we describe the results.

6 Conclusions

We worked hard, and achieved very little.

References

[1] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The LaTeX Companion*. Addison-Wesley, Reading, Massachusetts, 1993.

[2] Albert Einstein. *Zur Elektrodynamik bewegter Körper*. (German) [*On the electrodynamics of moving bodies*]. Annalen der Physik, 322(10):891?921, 1905.

[3] Knuth: Computers and Typesetting,
`http://www-cs-faculty.stanford.edu/~uno/abcde.html`