Verifying String Safety Properties in AWS C99 Package with CBMC

Muhammad Osama and Anton Wijs
Department of Mathematics and Computer Science
Eindhoven University of Technology, Eindhoven, The Netherlands
{o.m.m.muhammad, a.j.wijs}@tue.nl

***Abstract*—In this paper, state-of-the-art proofs are generated with harness using the CBMC bounded model checker for the Amazon Web Services C99 core package. We check, in particular, the safety properties of the *String compare* routine with various *loop unwinding* settings. The generated proof has proven to be reasonably hard to solve using modern SAT solvers. It has many variable-clause redundancies which are not only challenging for a SAT solver but also useful to assess the performance of different simplification techniques.**

I. Introduction

Bounded Model Checking (BMC) [1], [2] determines whether a model M satisfies a certain property φ expressed in temporal logic, by translating the model checking problem to a propositional satisfiability (SAT) problem or a Satisfiability Modulo Theories (SMT) problem. The term *bounded* refers to the fact that the BMC procedure searches for a counterexample to the property, i.e., an execution trace, which is bounded in length by an integer k. If no counterexample up to this length exists, k can be increased and BMC can be applied again. This process can continue until a counterexample has been found, a user-defined threshold has been reached, or it can be concluded (via k-induction [2]) that increasing k further will not result in finding a counterexample. CBMC [3], [4] is an example of a successful BMC model checker that uses SAT solving. CBMC can check ANSI-C programs. The verification is performed by *unwinding* the loops in the program under verification a finite number of times, and checking whether the bounded executions of the program satisfy a particular safety property [5]. These properties may address common program errors, such as null-pointer exceptions and array out-of-bound accesses, and user-provided assertions.

II. Benchmarks

In this paper, we are interested in verifying the safety properties of the *compare* routine implemented in the String data structure of the Amazon Web Services (AWS) C99 core package. The proof covers the following:

- Memory allocation failure and access violations
- Pointer/floating-point overflow
- Data types conversion

We generated 41 different formulas using a *loop unwinding* upper-bound in the range $[600, 1000]$, with an increasing step of 10. These bounds make the SAT formulas achieve 100% coverage of all functionalities. All problems are written in this format:

`string_compare_safety_cbmc_unwinding_<x>`

where `x` denotes the unwinding value. The first and the last formulas are solved via MiniSat [6] within 470 and 3000 seconds respectively on a machine with Intel Core i5-7600 operating at 3.5 GHz. The solving time of the rest of the benchmarks are expected to be monotonically increasing.

III. Acknowledgment

We would like to thank Daniel Kroening and Natasha Jebbo for referring us to the AWS C99 package and helping with the configuration of the proof environment.

References

[1] A. Biere, A. Cimatti, E. Clarke, and Y. Zhu, "Symbolic model checking without BDDs," in *TACAS*. Springer, 1999, pp. 193–207.
[2] A. Biere, A. Cimatti, E. Clarke, O. Strichman, and Y. Zhu, "Bounded Model Checking," *Advances in Computers*, vol. 58, pp. 117–148, 2003.
[3] E. Clarke, D. Kroening, and F. Lerda, "A Tool for Checking ANSI-C Programs," in *TACAS*, ser. LNCS, vol. 2988. Springer, 2004, pp. 168–176.
[4] D. Kroening and M. Tautschnig, "CBMC – C Bounded Model Checker," in *TACAS*. Springer, 2014, pp. 389–391.
[5] D. Kroening and O. Strichman, *Decision Procedures*. Springer, 2016.
[6] N. Eén and N. Sörensson, "An Extensible SAT-solver," in *SAT*, ser. LNCS, vol. 2919. Springer, 2004, pp. 502–518.