Using clang as a Frontend on a Formal Verification Tool

Mikhail R. Gadelha[1], Jeremy Morse[2], Lucas Cordeiro[3,4], and Denis Nicole[1]

[1] University of Southampton, UK
[2] University of Bristol, UK
[3] Federal University of Amazonas, Brazil
[4] University of Oxford, UK
esbmc@googlegroups.com

One of the major challenges in software verification is the development and maintenance of an infrastructure that can handle real-world programs; an issue that intensifies given the ever evolving programming language standards. The C and C++ programming languages are problematic in this sense, with the standards now being updated every 3 years, after a 13 years gap between the C/C++98 and C/C++11 major releases.

The increasing number of features introduces both semantic and syntactic changes, which researchers must support in their tools, reducing the amount of effort that can be spent addressing research questions. We develop ESBMC[1], an SMT-based context-bounded model checker that aims to provide bit-precise verification of both C and C++ programs. Earlier releases of ESBMC used a modified C parser written by James Roskind and a C++ parser based on OpenC++, together spanning more than 62KLOC; maintaining these was a substantial task, and did not meet 100% standard conformance.

In this poster presentation we will introduce ESBMC's new clang-based frontend. Using clang as a frontend brings a number of advantages:

- We address the problem of maintaining a frontend for C and C++ in a simple and elegant way: by using clangs API to access and traverse the program AST, without having details of the input program compiled away.
- ESBMC now provides compilation error messages as expected from a compiler.
- ESBMC leverages clang's powerful static analyser to provide meaningful warnings when parsing the program.
- clang is able to simplify some expressions, e.g., calculate `sizeof`/`alignof` expressions, evaluate `static_asserts`, evaluate if a `dynamic_cast` is always null, etc., which eases the analysis of the input program.

The frontend was developed using libTooling and we will also present the challenges faced during development, including bugs found in clang (and patches submitted to fix them).

Simply put, the clang-based frontend is a converter, that reads clang's AST and converts it into a format understandable by our tool; given the simplicity of the solution, it was implemented using only 11KLOC, a fraction of the original frontends.

Finally, we will present a short summary of ESBMC's features, and our future goal of fully supporting the C++ language, and the remaining work for attaining that goal.

[1] `https://github.com/esbmc/esbmc`