Extension of GCC with a fully manageable reverse engineering front end

Csaba Nagy
University of Szeged, Department of Software Engineering
Nagy.Csaba.5@stud.u-szeged.hu

In the open source community one of the most popular compiler is GNU GCC. It is a very complex and robust compiler but because of its working mechanism it has no ability for special transformations like interprocedural optimizations.

A typical compiler has a three sided (front end, middle end, back end) construction. The front end analyses the source code and builds an abstract internal representation of the program. The middle end executes generic analysis, transformations (eg. optimizations) and prepares the internal representation for final code generation which is realized by the back end. There are smaller but very useful projects for only front/middle/back ends, too. For example Edison Designs Group's C++ front end is used by many modern commercial compilers and Columbus/CAN reverse engineering framework is used for many C++ source analyses. So it seems possible to achieve a more effective compiler by extending GCC with a front end which is capable of running special algorithms.

This paper shows one solution for this extension. It does not solve GCC's interprocedural problems but shows one usable way to extend the compiler with features which solve these problems. The described method is based on using Columbus/CAN instead of GCC's front end and GCC's back end for code generation. The complexity of building this "extended compiler" is in transforming the Columbus's abstract C++ code representation graph to the back end's IR (intermediate representation). As Columbus has a well-structured schema for the representation of C++ sources, by this transformation we will have the ability to execute those special transformations on the code before the compiling phases. Furthermore this technique opens the possibility to link other front ends (like EDG) with GCC to achieve a more powerful compiler, for example in code size optimizations.

This approach has been tested on GCC's official Code-Size Benchmark Environment (CSiBE) as real-world system and for the testing different metrics have been measured on the compilation with this "extended compiler" and with the official GCC.