Bachelor's Thesis

Expression Evaluation for Debugging with Sulong

Student: Christoph Pichler
SKZ/Matr.Nr.: 521 / k01605053
Email: pichler.christoph@outlook.com

Advisor: DI Jacob Kreindl, Bsc.

Start Date: Feb. 5, 2019

DI Jacob Kreindl, Bsc.
Institute for System Software

P +43 732 2468 4370
F +43 732 2468 4345
jacob.kreindl@jku.at

Office:
Karin Gusenbauer
Ext. 4342
karin.gusenbauer@jku.at

Sulong [1,2,3] is an interpreter for LLVM IR, an intermediate representation of source code that can be produced by the Clang [4] compiler for the C family of programming languages. It is based on the Truffle [5,6] framework for implementing interpreters for programming languages and part of the GraalVM [7] project. In addition to executing programs that were compiled to LLVM IR, Sulong also supports Truffle's framework for program instrumentation and debugging [8,9] to allow users to debug these programs at source-level. At the moment, this debugging support is limited to basic actions such as inspecting symbols in and stepping through the original source code. Sulong does, however, not include a parser for the original source language (i.e., C or C++ code) and is therefore unable to evaluate snippets of code in those languages in the context of a currently executing function during debugging.

The goal of this thesis is to implement a parser for a subset of the C and C++ programming languages that will allow Sulong to execute snippets of code in the context of a function currently halted in the debugger. This should allow users to more effectively inspect and understand the current program state.

The concrete goals of the thesis are as follows:

- Design a Domain-Specific Language (DSL) for expressions whose syntax and semantics are similar to that of the C and C++ programming languages and supports at least the following features:
  - Accessing symbols defined in the program context
  - Literals of different primitive types
  - Common C data types (char, short, int, long, float, double, long double, etc.)
  - Basic arithmetic and logic operations (addition, subtraction, multiplication, and, or, shifting in range of a defined data type with and without sign extension, etc.) for values of primitive and pointer types

JOHANNES KEPLER UNIVERSITY LINZ
Altenberger Straße 69
4040 Linz, Austria
jku.at
DVR 0093696

    - Explicit type casts
        - Between primitive types, including sign extension or truncation where required
        - Between integer and floating point type of the same size
        - Between integer and pointer types of the same size
        - Between different pointer types
    - Resolving pointers
    - Resolving members of structured values
- Develop a parser for the expression DSL using the Coco/R parser generator [10].
    - The parser should notify the user if a syntax error occurs without crashing the application.
- Use Sulong's existing infrastructure to execute the parsed code snippets in a provided execution context that includes both source-level and IR-level symbols.
- Provide an extensive test-suite to demonstrate and verify the functionality of the expression evaluation engine.

Optional goals for this project are as follows:

- Supporting the following features in the expression DSL:
    - Calling functions from the executing program.
    - Calling member functions of complex objects.

Explicit non-goals of this thesis are:

- Supporting the following features in the custom language:
    - Changing the values of symbols or fields in the running program.
    - Feature completeness with respect to the C/C++ programming languages.
- Advanced error handling when executing code of the custom language
    - e.g. executing a user-defined snippet is allowed to throw a SEGFAULT
- Fixing bugs in Sulong, Truffle or GraalVM.
    - However, any bugs that are found need to be reported.

The code written as part of this thesis is intended to be merged into the Sulong project, therefore it must meet the project's standards with respect to code quality, documentation and test coverage. The code must also be able to pass the Sulong project's requirements for being merged [11].

The progress of the thesis should be discussed with the adviser on at least a bi-weekly basis. The final version of the written thesis should be submitted before September 30, 2019.

[1] M. Rigger, R. Schatz, R. Mayrhofer, M. Grimmer, and H. Mössenböck, “Sulong, and Thanks for All the Bugs: Finding Errors in C Programs by Abstracting from the Native Execution Model,” in Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems, New York, NY, USA, 2018, pp. 377–391.

[2] https://github.com/oracle/graal/tree/master/sulong

[3] https://www.graalvm.org/docs/reference-manual/languages/llvm/

[4] https://clang.llvm.org/

[5] T. Würthinger et al., “One VM to Rule Them All,” in Proceedings of the 2013 ACM International Symposium on New Ideas, New Paradigms, and Reflections on Programming & Software, New York, NY, USA, 2013, pp. 187–204.

[6] https://github.com/oracle/graal/tree/master/truffle

[7] https://www.graalvm.org/

[8] M. Van De Vanter, C. Seaton, M. Haupt, C. Humer, and T. Würthinger, “Fast, Flexible, Polyglot Instrumentation Support for Debuggers and other Tools,” The Art, Science, and Engineering of Programming, vol. 2, no. 3, pp. 14:1-14:30, Mar. 2018.

[9] https://www.graalvm.org/docs/graalvm-as-a-platform/implement-instrument/

[10] http://www.ssw.uni-linz.ac.at/Coco/

[11] https://github.com/oracle/graal/blob/master/sulong/docs/CONTRIBUTING.md