Verifying Functional Equivalence Between C and Fortran Programs

Author: Wenhao Wu, Advisor: Stephen F. Siegel

Introduction

The proposed prototype is an innovative equivalence verifier for Fortran and C programs by extending a Concurrent Intermediate Verification Language framework (CIVL). This prototype contains a Fortran front-end for parsing, a C front-end for converting C programs into CIVL abstract syntax tree (AST), and an equivalence verifier. Additionally, a set of extensive experiments has been conducted, and the results show that the performance of this prototype is satisfactory.

Motivation

- Complex HPC projects involve both Fortran and C programs. - HPC code evolution from the legacy code to the new code. - Comparing a sequential Fortran program with a concurrent C program. - Completely implemented Fortran front-end: Develop the methods for transforming each Fortran behavior into CIVL-IR. - Completely support external APIs (e.g. OpenMP): Based on the current Fortran grammar, extend the front-end to process external APIs - Integrate CIVL primitives into Fortran grammar - Extend the current verification system for newly introduced Fortran Features: Some Fortran behaviors are not in C standards, but they can be presented as a CIVL-C program. - Both C and Fortran programs are transformed into CIVL-IR, which can be presented as a CIVL-C program.

Objective

- Parsing Fortran programs correctly according to the Fortran 2008 standard. - An extension of Fortran grammar to process OpenMP pragmas and CIVL primitives. - A set of test cases for each language feature (i.e., grammar rules and specific behaviors).

Approach

The implementation is an extension of CIVL, which is a verification platform for concurrent C programs that use the MPI, OpenMP, CUDA, and PThreads APIs. The CIVL-IRs are represented as sets of Abstract Syntax Trees constructed from each input C or Fortran source file and linked into one result AST by a Java-based C Front-IR. CIVL traverses the state space in terms of the required properties (such as deadlock) to search for violations in all reachable states. A Fortran parser has been integrated, which is derived from Open Fortran Parser (OPP), and implements interfaces provided by the CIVL verifier. Furthermore, to adapt the AST structure used by CIVL, an additional procedure is applied to transform a Fortran parsing tree into CIVL-IR structure has been applied. The program MAY NOT be correct. See CIVLREP/ERROR for more information!

Experimentation

Equivalence verification:

1. In the first set, CIVL computes two small pieces of Fortran and C programs. Both of these calculate the modulo of B for random bounded input integers but with different methods. (i.e., one program uses a bit-wise AND operation with 7, and the other program uses integer division.)

2. In the second set, both Fortran and C programs will calculate a harmonic value for a given integer in a same mathematical procedure. The interesting point is that those math functions are statements in Fortran but function calls requiring "math." in C.

3. The last but the most interesting experiment compares a sequential Fortran program with two parallel OpenMP implementations - a correct one and an incorrect one. The functionality is to multiply matrices. Also, this piece of code is truncated from a real-world Fortran program - Nek5000.

Some related statistics and results are shown in this section.

Results:

Those correct C implementations can be verified successfully without violations detected. And incorrect implementations will result in an assertion violation (i.e., the assertion declares both outputs from the specification and actual outputs are exactly equivalent).

Most verification operations can be finished in one minute. For the last experiment of multiplying matrices, the cost increases exponentially. All results are satisfactory and possible to be improved in the future.

Conclusion

Goals achieved by current system are listed below:

- Verifying basic sequential Fortran programs - Program Units (Program, Subroutine, Function, Data Block) - Structures (DO-Loop, IF-branch, and Computed-GOTO) - Expressions, Operators, Identifiers.

- Comparing a sequential Fortran program with a concurrent C program. - Equivalence verification performed by analyzing the ranges of the inputs and the outputs.

- Transform Fortran programs into CIVL-C programs - Both C and Fortran programs are transformed into CIVL-IR, which can be presented as a CIVL-C program.

Future Work

- Completely implemented Fortran front-end: Develop the methods for transforming each Fortran behavior into CIVL-IR.

- Completely support external APIs (e.g., OpenMP) - Based on the current Fortran grammar, extend the front-end to process external APIs - Integrate CIVL primitives into Fortran grammar - Extend the current verification system for newly introduced Fortran Features: Some Fortran behaviors are not in C standards, but they can be presented as a CIVL-C program. - Both C and Fortran programs are transformed into CIVL-IR, which can be presented as a CIVL-C program.

Further Info

Scan the square code to visit the website of CIVL for more information!

References


Acknowledgement

The whole project is funded by: National Science Foundation (CCF-1319571)