COMPILER-RELATED PROJECTS

A COLLECTION OF THESIS PROPOSALS FOR MSC STUDENTS

LAST UPDATE: 2018-12-17

SCOPE OF THIS PRESENTATION

We present in this document a selection of research projects related to compilers that are currently active within our department. They can be easily be targeted by MSc thesis.

We kindly invite you to contact the project reference person if you want to know more and/or get involved with these projects.

MAIN RESEARCH FOCUS

- Compiler-Level Precision Tuning Optimizations (TAFFO)
- Compiler Front-end Design & Development (MODELICA)
- Reverse Engineering of Binary Code (REV.NG)

TAFFO

TUNING **A**SSISTANT FOR **F**LOATING POINT TO **F**IXED POINT **O**PTIMIZATION

CONTACT: STEFANO.CHERUBIN@POLIMI.IT

WHAT IS TAFFO

- Framework that converts floating point code into integer (fixed point) code
- Structured as a set of LLVM compiler plugins
- Do not require compiler customizations

CURRENT STRUCTURE OF TAFFO

1. User Annotation Propagation
2. Value Range Analysis
3. Data Type Allocation
4. Code Conversion
5. Performance Estimation

TAFFO RESEARCH DIRECTIONS

- Avoid user annotations and perform automatic profiling to understand the range of possible input values
- Improve the effectiveness of the Value Range Analysis
- Improve the TAFFO static Performance Estimation analysis by leveraging new LLVM-based performance estimation tools such llvm-mca
- Minimization of type cast instructions via graph-based algorithms

TAFFO ADDITIONAL NOTES

- One joint MSc thesis (2 students) is going to be discussed in December 2018
  - Objective: design and implementation of most of the TAFFO components
- Large range of use cases
  - From Embedded Systems to High Performance Computing
- Precision tuning is a hot trend in the scientific community
  - Good opportunities to publish papers on this topic

TAFFO REQUIREMENTS

- Good C++ Proficiency
- Familiarity with the LLVM Framework

MODELICA

DESIGN OF A COMPILER FOR A MODELLING AND SIMULATION LANGUAGE

CONTACT: STEFANO.CHERUBIN@POLIMI.IT

WHAT IS MODELICA

- Modelica is a widely-used modelling language for physical systems
- Models are based on differential equations
- Simulation of a Modelica model entails the conversion of Modelica code into C code, the compilation of C code, linking with an external equation solver, and one single execution of the compiled code to produce results
- Traditionally Modelica compilers are designed by mathematicians instead of compiler engineers, and do not preserve object-oriented structure of the models
- Nowadays such compilers are experiencing scalability issues

MODELICA RESEARCH DIRECTIONS

Currently we have a prototype of a modern Modelica compiler that demonstrates effectiveness of the modern compiler approach over the Modelica traditional approach. We want to extend it to a full-featured compiler front-end for LLVM. Thus we need to:

- Avoid the C code generation stage and emit directly LLVM-IR
- Extend the support of the prototype to a wider subset of the Modelica language
- Implement system/equation simplification algorithms from the state-of-the-art using a structured representation of the model

MODELICA ADDITIONAL NOTES

- One MSc thesis is going to be discussed in December 2018
  - Objective: demonstrate feasibility and effectiveness of modern approach to Modelica compilation
- Ongoing collaboration with Automation & Control System Engineering @ DEIB
- HOT topic
  - More than one thesis can be assigned on this topic.

MODELICA REQUIREMENTS

- Good C++ Proficiency
- Familiarity with the LLVM Framework
- Familiarity with Systems of Differential Equations

REV.NG

REVERSE ENGINEERING FRAMEWORK

CONTACT: ALESSANDRO.DIFEDERICO@POLIMI.IT

WHAT IS REV.NG

- A reverse engineering framework
- Obtain a pseudo-C as close as possible to the original one
- rev.ng can also do static binary translation
  - e.g., translate `ls` for ARM in `ls` for x86-64

REV.NG PROJECTS

- Automated vulnerability discover through fuzzing/symbolic execution
- Patching localized portions of a program with custom code
- Distinguish code from data
- Introduce support for new input architectures/platforms (easy)

SKILLS YOU WILL ACQUIRE

- Knowledge of C++ development
- Knowledge of the LLVM framework
- Knowledge of low-level internals of CPUs/operating systems
- Ninja reverse engineering abilities