Software maintenance of a C++ “linter” tool

TECHNISCHE UNIVERSITÄT DARMSTADT

Applicable for students as HiWi

Keywords: *Clang, Source Analysis, Source Transformation, Testing, LLVM-Lit*

SCIENTIFIC COMPUTING

Alexander Hück
alexander.hueck@tu-darmstadt.de

Office: S1|22 Room 412
Alexanderstraße 2
64287 Darmstadt
Tel. 06151 16-7 55 77

Date: 18th June, 2020

Introduction

In C++, operator overloading can be used to replace built-in types (e.g., `double`) with user-defined ones. This is done to introduce new semantics to an existing code base, e.g., multi-precision data types (Boost.Multiprecision).

```cpp

double foo(double a) {
  double phi = a * a * .5;
  return phi;
}
```

```cpp
#include "adouble.h"
adouble foo(adouble a) {
  adouble phi_s = a * a * .5;
  return phi_s;
}
```

Figure 1: *Left:* Input function using built-in **`double`**. *Right:* Type change: **`double`** is replace by the user-defined type **`adouble`** providing required operator overloads.

The C++ standard treats these user-defined types differently than built-in ones in certain contexts. Hence, the code can become illegal and the compiler will produce an error after the type change. As a result, the tool OO-Lint [1] was developed. It is based on the Clang compiler infrastructure [2], to find such problematic code constructs *before* the type change. This enables the developer to fix the problems without interpreting thousands of lines of compiler error output.

Tasks

The OO-Lint tool needs to be made compatible with the recent Clang version 10. You will fix any API breakage and modernize the code if necessary.

Source code analysis

(a) Improve the matchers of the static analyser to reduce false positive rates.

(b) Unit tests for the various matchers based on the LLVM-Lit testing tool.

Source code transformation

(a) Existing source transformation capabilities need to be tested and refactored.

(b) Development of new source transformation capabilities.

Qualifications

- Experience with modern C++ and the CMake build system.
- Knowledge of the Clang tooling library [2].

References

[1] `https://github.com/ahueck/opovlint/tree/clang6.0`

[2] `https://clang.llvm.org/docs/LibTooling.html`