FRIEDRICH-SCHILLER-
UNIVERSITÄT
JENA

Prof. Dr. Clemens Grelck
Eric Wintzler, MSc
System Software Group
Compiler Construction
October 28, 2025

Assignment 2

Familiarisation with the CoCoNut Compiler Framework

This assignment series is intended to deepen your understanding of the CoCoNut compiler construction framework to be used for the CiviC compiler project.

Assignment 2.1: CoCoNut Framework: code transformation

Implement a compilation pass (traversal and phase) that implements the *strength reduction* optimisation, introduced in Chapter 2 of the lecture series. Some processor architectures implement addition more efficiently than multiplication, thus giving rise to the following code transformations:

```
2*k -> k+k
k*2 -> k+k
```

for any (integer) variable `k`. Place your implementation of strength reduction in a new subdirectory to the `src` directory aimed at optimisations in general. Furthermore, create a new compilation phase for optimisations. For consistency the demo optimisation on subtraction shall also be moved to both the new source subdirectory and the new compilation phase for optimisations.

Assignment 2.2: CoCoNut Framework: parameterisation

Based on your solution for Assignment 2.1 implement another compilation pass that implements an extended form of strength reduction, say *super strength reduction*, where the maximum value of the constant factor to apply strength reduction to becomes a command line parameter of the compiler.

Note that in real compilers these days strength reduction is not needed because on usual contemporary architectures both addition and multiplication take one cycle. Hence super strength reduction is not only useless but even counterproductive in the sense that it would slow down program execution. So, the purpose of this assignment lies purely with the exploration of the CoCoNut framework.

Assignment 2.3: CoCoNut Framework: collecting information

Implement a compilation pass that counts the number of occurrences of each of the five arithmetic operators in the source code. The `data` facility shall be used to collect the information; the use of global variables is not permitted.

Unlike in the demo pass `sum_ints` coming with the initial framework, the information shall at the end of the traversal be stored in the root node of the syntax tree.

For this purpose define a new node type to be the dedicated root node of the syntax tree of a compilation unit. This root node shall (for now) contain a sequence of CiviC statements (as already given) plus the necessary attributes to store the newly inferred information.

Assignment 2.4: CoCoNut Framework: pretty printing

Extend the pretty print traversal such that it produces appropriate output for the new root node introduced in Assignment 2.3, including the newly inferred information when available.

Note:

Submit all assignments at once as a clean tar-file of the entire directory tree. You can remove all generated files with the following command:

- `cmake --build <build-dir> --target clean`

Assignment due date: Friday, November 7, 2025