P57 - Automatic Annotation Inserter

Massimo Fioravanti

Polimi

June 21, 2019

Abstract

Taffo compiler is a collection of llvm passes used to perform precision tuning. The user is allowed to annotate variables and functions with bounds and properties, so that Taffo can make optimizations that are otherwise impossibile.

```
float lerp(
    float start,
    float end,
    float __attribute(annotate("scalar(range(0, 1))")) i)
{
    return ((1-i) * start) + (i * end);
}
```

The floating point variable i will be assumed to be in the range [0,1].

Abstract

The objective is to decouple the internal reppresentation of taffo annotations from the one used by end users, allowing the external one to be simpler and more usable.

In particular we wish to contain the complexity araising from structs. When a variable is a struct it is neccessary to specify the annotation of each member of the struct, and for each member of each included struct.

```
struct Internal float v1, float v2, float v3;
struct External float u1, struct Internal u2;

struct External __attribute(annotate("struct[
   scalar(...), struct[
      scalar(...),
      scalar(...),
      scalar(...)]]")) var;
```

Solution

We requre the ability to express 2 concepts, list of named structs, and dictionaries of final annotations.

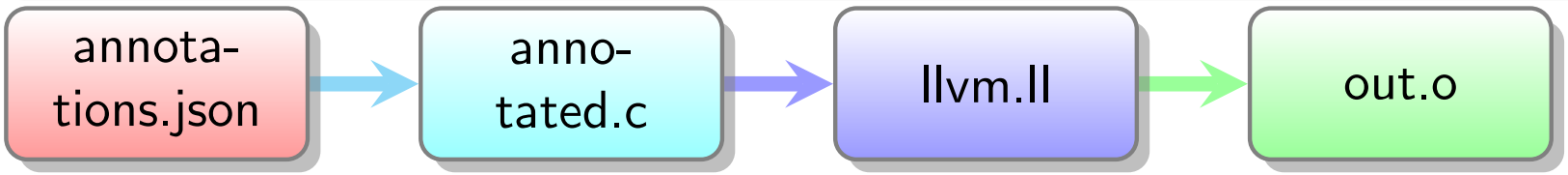
```
{   "struct": "internal",
   "content": [{...}, {...}, {...}]},
{   "struct": "external",
   "content": [{...}, "internal"]},
{   "globalVar":  "ext", "struct:": "external"},
{   "globalVar":  "i", "rangeMin": 0, "rangeMax": 1}
```

This allow us to express a hierarchical structure without having to utilize it with syntax, and provides us all the expressive power we need for this project.

Insertion

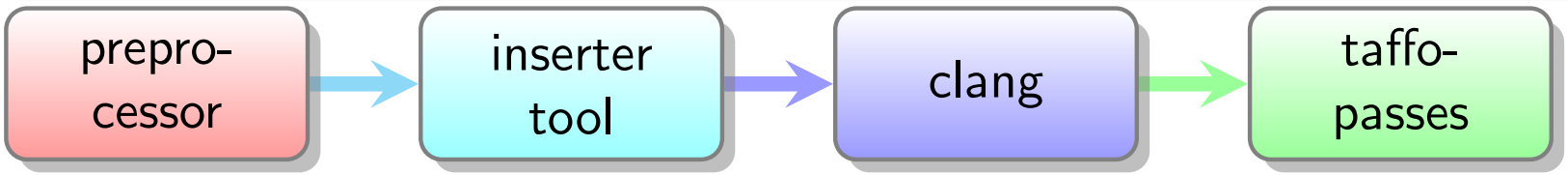
We decided to use a subset of the json language to rappresent our user annotation language. This allow us reuse a well known library to load the user defined annotations from a json file.

Since the user will often tune the annotations we wish to make sure that the user does not need to change the source files every time.



Insertion

The c/c++ compilation pipeline will look as follow, we run the preprocessor and save the output in a temporary file, parse the content temporary, locate the declarations, insert the annotation, and then compile as normal.



The inserter tool only need to be able to parse c and cpp files. Clang libtooling is the standard solution for this operation.

Clang tools

Libtooling is a clang library that is part of the llvm project. It allows to parse the Abstract Syntax Tree of a c/c++ program and to rewrite the source code, in this case by inserting annotations.

Since the c++ is complex many edge cases are present, but most of them are solved by running the preprocessor before the tool. Still other kinds of code generations, such as c++ templates, can elude the insertion.

Results

This tool https://www.github.com/HEAPLab/TAFFO is compatible with the most complex benchmarchs supported by taffo. TAFFO has been integrated with can use the inserter to apply annotations without changing the benchmarchs source code at all.

The overall pipeline is easy to use and only require to provide the annotation file to the TAFFO compiler. Users do not need to be exposed to the taffo internals.

Future Works

The user language is small and easy to use and there is no particular need to extend it.

As is always the case with c++ syntax there is plenty of room to cover insertion edge cases, as well as detecting impossible insertion and alert the user that such edge cases are taking place.

The compilation pipeline is still similar to a standard one, and integration with build tools can lead to a great improvement in compilation times, since currently it is neccessary recompile every annotated source file at each iteration.

Both this expansions are time consuming but not particulary complex, they are could be implemented when they are needed, or as a introductory project for someone that wishes to learn about clang, c++ syntax or build systems.

References

- TAFFO: Tuning Assistant for Floating point to Fixed point Optimization S. Cherubin, D. Cattaneo, A. Di Bello, M. Chiari, G. Agosta IEEE Embedded Systems Letters. Apr 2019. DOI: 10.1109/LES.2019.2913774
- A. Yazdanbakhsh, D. Mahajan, P. Lotfi-Kamran, H. Esmaeilzadeh, "AXBENCH: A Multi-Platform Benchmark Suite for Approximate Computing", IEEE Design and Test, special issue on Computing in the Dark Silicon Era 2016.
- Clang, libtootling https://clang.llvm.org/