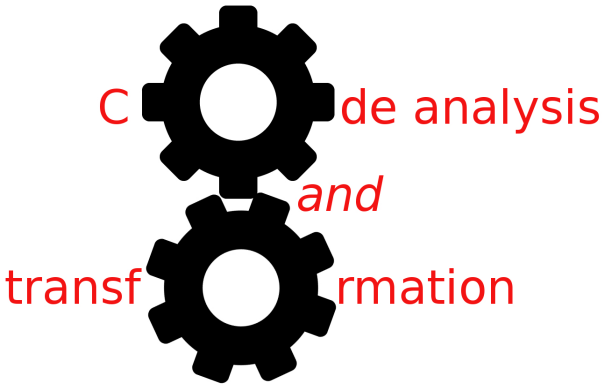

Homework H1

1 Description

Write an LLVM pass starting from the code you have developed for H0. The goal of this pass is to print statistics about invocations of functions included in the CAT API described next. Specifically, for each bitcode function, you must print

- the name of a CAT function that is invoked in the current bitcode function, and
- the number of instructions of that function that invoke it

The order of CAT functions to print is

1. `CAT_binary_add`
2. `CAT_binary_sub`
3. `CAT_create_signed_value`
4. `CAT_get_signed_value`

Finally, CAT functions that are not invoked by a bitcode function are not printed.

2 CAT sources

You can find the CAT API in CAT.h available in the distributed tests.

3 Example

Consider the following program:

```
#include <CAT.h>

void CAT_execution (void){
  CATData d1;
```

```
  CATData d2;
  CATData d3;

  d1  = CAT_create_signed_value(5);
  d2  = CAT_create_signed_value(8);
  d3  = CAT_create_signed_value(0);

  CAT_binary_add(d3, d1, d2);

  return CAT_get_signed_value(d3);
}

int main (int argc, char *argv[]){
  return CAT_execution();
}
```

your pass must generate the following output (stored in `compiler_output`):

```
H1: "CAT_execution": CAT_binary_add: 1
H1: "CAT_execution": CAT_create_signed_value: 3
H1: "CAT_execution": CAT_get_signed_value: 1
```

`H1.tar.bz2` includes a few programs you can use to test your work.

Run all tests Go to `H1/tests` and run `make` to test your work.
The following output means you passed all tests:

```
./misc/run_tests.sh
SUMMARY: 4 tests passed out of 4
```

If you didn't pass a test, then the output will include all tests that have failed.

Run a test You probably want to figure out why you failed a test. To do so, go to such test (let's assume test0 failed):

```
$ cd H1/tests/test0
```

Now, compile a program

```
$ make clean ; make
```

Check the output generated by your pass against the oracle output:

```
$ make check
```

and follow the instructions printed in the output to debug your work.

4 LLVM API and Friends

This section lists the set of LLVM APIs I have used in my H1 solution that I did not use for the past assignment H0. You can choose whether or not using these APIs.

- Method `getFunction` of the class `Module`

- Method `isa<LLVM CLASS>(LLVM OBJECT)`. For example, `isa<CallInst>(i)` where `i` is an instance of the class `Instruction`
- Method `cast<LLVM CLASS>(LLVM OBJECT)`. For example, `CallInst *callInst = cast<CallInst>(i)` where `i` is an instance of the class `Instruction`
- Method `getCalledFunction` of the class `CallInst`
- Method `write_escaped` of the class `raw_ostream`

Next are some headers you might find useful.

```
#include "llvm/Pass.h"
#include "llvm/IR/Module.h"
#include "llvm/IR/Function.h"
#include "llvm/IR/Instructions.h"
#include "llvm/Support/raw_ostream.h"
#include "llvm/Transforms/IPO/PassManagerBuilder.h"
#include <iostream>
#include <map>
#include <vector>
```

5 What to submit

Submit via Canvas the C++ file you've implemented (CatPass.cpp).

For your information: my solution for H1 added 59 lines of C++ code to H0 (computed by `sloccount`).

6 Homework due

10/5 at noon

Good luck with your work!