C de analysis
and
transf rmation

Homework H4

1 Description

Write an LLVM pass starting from your H3 code.

The goal of this new pass is to implement both the constant propagation and the constant folding by using the IN and OUT reaching definition sets you have computed for H3.

Constant folding is the transformation that translates an operation (e.g., binary add like v3=v1+v2) into a definition (e.g., v3=5) by performing at compile-time the target operation. This transformation can be performed only when the operands of the operator are constants that are known at compile-time.

As it was the case for H3, the only variables you need to consider are the CAT variables.

2 Assumptions

You can make the same code assumptions that you had for the H3 homework.

3 Run all tests

Go to `H4/tests` and run

```
make
```

to test your work.

4 LLVM API and Friends

TYou can choose whether or not using these APIs.

These APIs are the following:

- Checking whether or not an instance of `Value` is an integer constant:

  ```
  isa<ConstantInt>(v)
  ```

  where `v` is an instance of `Value`.

- To fetch the actual constant value from an instance of `Value`:

  ```
  int64_t c = v->getSExtValue();
  ```

  where `v` is an instance of `Value`.

- To substitute all uses of a variable defined by an instruction with a constant:

  ```
  ReplaceInstWithValue(bb->getInstList(), ii, constValue)
  ```

  where `bb` is an instance of `BasicBlock`, `ii` is an instance of `BasicBlock::iterator` , and `constValue` is an instance of `Value`.

- To create an instance of `BasicBlock::iterator`:

  ```
  BasicBlock::iterator ii(i);
  ```

  where `i` is an instance of `Instruction`.

I've also used the following new header:

```
#include "llvm/IR/Constants.h"
```

5 What to submit

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

For your information: my solution for H4 added 157 lines of C++ code to H3 (computed by `sloccount`).

Good luck with your work!