C de analysis
and
transf rmation

Homework H6

1 Description

Write an LLVM pass starting from the code you have developed for H5.

The goal of this assignment is to reduce the conservativeness of H5 third assignment for CAT variables stored in memory. In more detail, instead of skipping such variables in the constant propagation as you did in H5, you now need to rely on the dependences identified by LLVM to properly handle them both in the reaching definition analysis and in your constant propagation algorithm.

2 Code example

For example, consider the following C code:

```
void generic_C_function (void){
  CATData x;
  void **ref;
  ref = malloc(sizeof(void *));

  x = CAT_create_signed_value(8);
  (*ref) = x;

  printf("%ld", CAT_get_signed_value(x));
  free(ref);
}
```

Your H5 pass does not perform the constant propagation for the CAT variable `x`. Instead, for your H6 homework, you need to be able to perform the propagation even if the variable `x` get stored in memory.

You can find more examples of transformations you need to be able to perform in the `tests` directory (by comparing program.ll with output/program_optimized.ll).

3 API

This section describes the set of LLVM APIs I have used in my H6 solution that I did not used in prior assignments. You can choose whether or not using these APIs.

- To get the output of the dependence analysis :

  ```
  DependenceAnalysis &deps = getAnalysis<DependenceAnalysis>();
  ```

  you also need to include the appropriate header:

  ```
  #include "llvm/Analysis/DependenceAnalysis.h"
  ```

- To tell LLVM that we depend on dependence analysis:

  ```
  AU.addRequiredTransitive<DependenceAnalysis>();
  ```

- To check if there is a direct dependence from an instruction `i1` to another instruction `i2`:

  ```
  deps.depends(i1, i2, false)
  ```

  where `i1,i2` are instances of `Instruction`.

4 Testing your work

Your pass has to pass all tests distributed in H6.tar.bz2. To perform this check, set the path of your pass in `LLVMPASSPATH` of `tests/misc/Makefile`. Then go to the directory `tests` and run your pass:

```
make ;
```

The output tells you how many tests you have passed and the list of the ones you didn't pass.
To check why your pass didn't pass a specific test, go to that test:

```
cd H6/tests/test0
```

Now, invoke your pass

```
make clean ; make ;
```

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

```
make check
```

The output message tells you what went wrong. Output differences are stored in the `diff` directory .
Good luck with your work!

5 What to submit

Submit via Canvas

- The C++ file you've implemented (CatPass.cpp)

For your information: my H6 solution added 56 lines of C++ code to H5 third assignment (computed by `sloccount`).

6 Homework due

11/23 at 2am