CS 6620

14 points **Homework 3** assigned: Mar. 5, 2020 due: ≤23:59 EST Apr. 3, 2020

This assignment should be completed individually.

This project involves extending and improving an existing LLVM pass that performs a basic interval range analysis on TIP programs compiled to bitcode. To get started clone the repository https://github.com/matthewbdwyer/tipc-passes and follow the installation instructions.

A number of starter passes are included in this repo. Feel free to play with them to gain familiarity with LLVM and bitcode processing. Your project will involve the "interval range pass" which you will find in the `.../src/intervalrangepass` directory. This pass exploits the SSA structure of bitcode produced after running the `-mem2reg` pass to realize a *sparse interval analysis*. There are a few test cases in the `test` subdirectory that you can look at to get started.

Part 1 (5 points)

The interval range pass is **imprecise**.

You need to define more precise versions of the `interval::mul()` and `interval::div()` operations. You can use any resource you like to determine the specification for interval arithmetic, for example, the Wiki page for interval arithmetic https://en.wikipedia.org/wiki/Interval_arithmetic.

The comparison operations, e.g., `interval::lt()`, are also very imprecise. These are harder to address, but if you do I will award some extra credit.

You need to define some test cases to demonstrate that your solution produces correct results. Note that `interval4.tip` is an existing example that illustrates this problem.

Part 2 (5 points)

The interval range pass **does not include widening**.

Without widening the interval analysis will diverge when loops are present. You need to add a widening strategy to the pass. We considered a few concepts in class that project the infinite interval lattice onto a finite sub-lattice. They do this by transforming a value in the original lattice to the *nearest* overapproximating value in the finite sub-lattice. This guarantees finite ascending chains and termination.

You need to define some test cases to demonstrate that your solution produces correct results. Note that `interval5.tip` is an existing example that illustrates this problem.

Part 3 (4 points)

The interval range pass is **unsound**.

The current implementation uses the min/max integer values to represent negative/positive infinity. The problem is that if one performs the following operation: *[0, max-1] + [2, 2]* the upper bound will wrap around. There are multiple ways to handle this, e.g., use an arbitrary precision arithmetic package, with different tradeoffs. Select an approach and implement it.

You need to define some test cases to demonstrate that your solution produces correct results. Note that `interval5.tip` is an existing example that illustrates this problem.

Submission

Upload to the course collab site a zip of the `.../src/intervalrangepass` directory that includes all of your solutions above and your test cases. Please also include a PDF file describing your solution.

A Note On Grading This is more work than the prior two assignments. It is stated to count as 14 points in the course, but I will adjust the weighting of the final grades to reflect the fact that this is a more challenging assignment.