# CS5218: Assignment 2 – Abstract Interpretation Analysis 'with and without' Path Sensitivity on LLVM IR

## Introduction

The purpose of this assignment is to perform an abstract interpretation analysis 'with and without' path sensitivity. Your task would be mainly defining an abstract domain and performing an analysis which requires widening. This assignment is not a group assignment.

### Difference Analysis on LLVM IR

Consider a special kind of "difference" analysis. We want to know, at each program point, what is the difference in value between any *pair* of variables, eg. for variables x and y, what is their *separation*? We now define the (symmetric) function:

$$sep(x,y) = \begin{cases} x-y & if \quad x \geq y \\ y-x & otherwise \end{cases}$$

The analysis serves to determine, for each program point, and each pair of variables, the *largest* separation.

**Example 1:** For example, consider the program below:

```
int main() {
    int w,x,y,z = 0;
    if (w < -10)
        x = 2;
    else
        x = -4;
    x+=10;
    if (y != 10)
        z = 50;
}
```

One difference analysis, for the pair of variables $x$ and $z$, could return for $sep(x,z)$, at the terminal program point, the value 44.

### Interval Analysis with Some Path Sensitivity

The purpose of this part of the assignment is to perform an abstract interpretation analysis with some path sensitivity.

Interval analysis is a famous analysis developed in 1950s and 1960s. In this approach, an interval for a variable $x$ is represented by $x : [a, b]$ where $a$ is a lower bound and $b$ is an upper bound on the possible values that $x$ can contain. For example, in the program below, the interval for $x$ before the assertion would be: $[0, 10]$. Since the value of $x$ is at most 10, it can be inferred that the assertion will never be violated.

```
int main() {
    int x, y, z=10;
    if (y == 10)
        x = z + 0;
    else
        x = z - 10;
    assert (x < 11);
    return x;
}
```

Note that the intervals for the other variables in this program would be: $y : [-\infty, \infty]$ and $z : [10, 10]$.

Interval analysis is a light-weight analysis that is widely used to infer the possible values for variables at different program points. For this part of the assignment, you will be implementing an Interval Analysis with some path sensitivity for all the variables in the input program. All the algorithms you need are available from the lecture material.

Here, the abstract interpretation framework considers the effect of equality, non-equality, and disequalities on the abstract domain too. For this, you need to consider `icmp` and conditional `br` instructions too. For example, consider the program below. A non-path sensitive interval analysis would infer that the interval of the values of $x$ before the assertions will be: $[0, 5]$. As a result, it would infer that the assertion will be violated. However, considering the interval for $y : [10, 10]$, we can see that the true branch of the if-statement is infeasible. Considering this point, the update on the value of $x$ in the true branch should be ignored. As a result, the interval for $x$ would change to $x : [0, 0]$ which does not violate the assertion at the end of the program.

```
int main() {
    int x, y=10, z = 5;
    if (y != 10)
        x = z + 5;
    else
        x = z - 5;
    assert (x < 5);
    return x;
}
```

Finally, the analysis will return the following intervals for the other variables in this program: $y : [10, 10]$ and $z : [5, 5]$.

**Note**: Please add path sensitivity for Interval Analysis only. The information carried by the abstract domain using path sensitivity doesn't help much in Difference Analysis.

## Task 1: Designing Difference and Interval Analysis

Define your analysis using the framework of *abstract interpretation* (Finite Lattice, Galois connection, abstract semantics, etc) separately for:

1. Difference Analysis
2. Interval Analysis

**Note**: For the following two tasks, you have to implement the difference analysis for the following instructions: `Alloca`, `Add`, `Sub`, `Div`, `Mul`, `Rem`, and `Load/Store`. For interval analysis, the program should be able to consider arithmetic expressions (two or more variables, constants, `add` and `sub`, e.g. y = x + 5) and boolean expressions (exp X const where X is `<`, `<=`, `>`, `>=`, $\neq$ and ==), for example: $y - 12 <= 15$ or $x + 2 \neq 12$).

## Task 2: Implementing the Difference and Interval Analysis in LLVM

LLVM is a set of compiler infrastructure tools. You have seen `clang`, and LLVM C compilers in the demo sessions. In this task, implement LLVM pass (program) to perform the analysis on loop-free programs for *Difference Analysis* as well as *Interval Analysis*.

## Task 3: Adding Support for Loops to the Analysis

Extend your implementation to accommodate cycles in the CFG. This time you need to be concerned about the termination of your analysis. In order to guarantee termination, the analysis may now not be able to produce a finite bound at all times.

For difference analysis, you, therefore, need to perform some kind of *abstraction*.

**Example 2:** Consider the program below:

```
int main() {
  int a,b,x,y,z = 0 ;
  // assume N is an input value
  int i = 0;
  while (i < N) {
     x = -((x + 2*y * 3*z) % 3);
     y = (3*x + 2*y + z) % 11;
     z++;
  }
}
```

One difference analysis, for the pair of variables $x$ and $y$, could return for $sep(x,y)$ the value 12 at the end of the loop body. For the pair of variables $x$ and $z$, on the other hand, it may not be possible to obtain a finite bound.

For interval analysis, you, therefore, need to perform some kind of *widening*.

**Example 3:** Consider the program below:

```
int main() {
    int a = -2,b = 5,x = 0, y;
    // assume N is an input value
    int i = 0;
    while (i++ < N) {
        if (a > 0){
            x = x + 7;
            y = 5;
        }else{
            x = x - 2;
            y = 1;
        }
        if (b > 0){
            a = 6;
        }else{
            a = -5;
        }
    }
}
```

A non path-sensitive interval analysis, would return $[-\infty,\infty]$ for $x$. A path-sensitive interval analysis, however, would return the following intervals for the variables (assuming loop iterates at least once):

$a : [6,6]$
$b : [5,5]$
$x : [-2,\infty]$
$y : [1,5]$
$i : [0,\infty]$
$N : [-\infty,\infty]$

**Note:** One sample abstract domain can be $[-\infty,-20,...,-1,0,1,...,20,\infty]$ which can represent all the values that z can have in an abstract way. Explain the abstraction that you are use in the report. You may want to first attempt a simpler version of Task 3 by ignoring the abstraction and widening aspects. This means that your algorithm may not termintate on some examples. Getting this part correct (ie. the analysis is correct *assuming termination*) will provide some credit.

## Submission - Deadline: 23:59 Sunday 24$^{th}$ March 2024

Please submit the following in a single archive file (`zip` or `tgz`):

1. A report in PDF (`asg2.pdf`). It should describe, your design for task 1 and the implementation of tasks 2 and 3, and the algorithm used. How to build and run. The output of the example programs tested.
2. Your source code.
3. Your test C files with corresponding `ll` files.

For technical support on LLVM, you can contact Arpita Dutta (`arpita@comp.nus.edu.sg`) or Chew Wei Ze Alvin (`a.chew@nus.edu.sg`). Please have your subject line begin with "CS5218".

Also, please ensure your reports and source files contain information about your name, matric number and email. Your zip files should have the format *Surname-Matric*-asg2.zip (or `tgz`). Submit all files above to the appropriate Canvas workbin.