Assignment 2

Task 1

We shall design a difference analysis with the Monotone Framework defined in [1] as follows.

$$Analysis_{\circ}(\ell) = \begin{cases} \iota & if\ \ell \in E \\ \bigsqcup\{Analysis_{\bullet}(\ell') \mid (\ell', \ell) \in F\} & otherwise \end{cases}$$

$$Analysis_{\bullet}(\ell) = f_{\ell}\big(Analysis_{\circ}(\ell)\big)$$

The goal of the difference analysis is to compute the absolute difference between each pair of variables at each program point ℓ. We can base the analysis on a separation function $\boldsymbol{sep}: \boldsymbol{Var}_{\star} \times \boldsymbol{Var}_{\star} \to \mathbb{N}$, for the pairs that appear in a basic block B^{ℓ}. In addition, the ordering of two variables in $\boldsymbol{Var}_{\star} \times \boldsymbol{Var}_{\star}$ does not affect the outcome of difference calculation.

Consider the following example program written in the While language.

$$[d := 0]^1; if\ [a > 0]^2\ then\ [c := 5]^3\ else\ [c := 10]^4; if\ [b > 0]^5\ then\ [d := -11]^6$$

With the Monotone Framework we have defined above, the difference analysis has the complete lattice $(\mathbb{P}(sep), \subseteq)$, the instance of the framework, and the transfer function below.

$$F = flow(S_{\star})$$

$$E = \{init(S_{\star})\}$$

$$\iota = \emptyset$$

$$f_{\ell}(l) = l \cup gen\big(B^{\ell}\big)\ where\ B^{\ell} \in blocks(S_{\star})$$

$$gen\big(B^{\ell}\big): Blocks_{\star} \to \mathbb{P}(sep)$$

We shall now list all of the associated equations based on the instance for each ℓ as follows.

$$DF_{entry}(1) = \emptyset$$
$$DF_{entry}(2) = DF_{exit}(1)$$
$$DF_{entry}(3) = DF_{exit}(2)$$
$$DF_{entry}(4) = DF_{exit}(2)$$
$$DF_{entry}(5) = DF_{exit}(3) \cup DF_{exit}(4)$$
$$DF_{exit}(1) = DF_{entry}(1) \cup \{(d, ?): ?\}$$
$$DF_{exit}(2) = DF_{entry}(2)$$
$$DF_{exit}(3) = DF_{entry}(3) \cup \{(d, c): 5\}$$
$$DF_{exit}(4) = DF_{entry}(4) \cup \{(d, c): 10\}$$
$$DF_{exit}(5) = DF_{entry}(5)$$
$$DF_{exit}(6) = DF_{entry}(6) \cup \{(d, c): 16, (d, c): 21\}$$

By solving the equations, the greatest difference among those pairs of variables is $\boldsymbol{sep(d, c) = 21}$. Note that we ignore the unknown variables denoted by question marks, which could be unassigned, since analyzing them is likely to be not significant in this case.

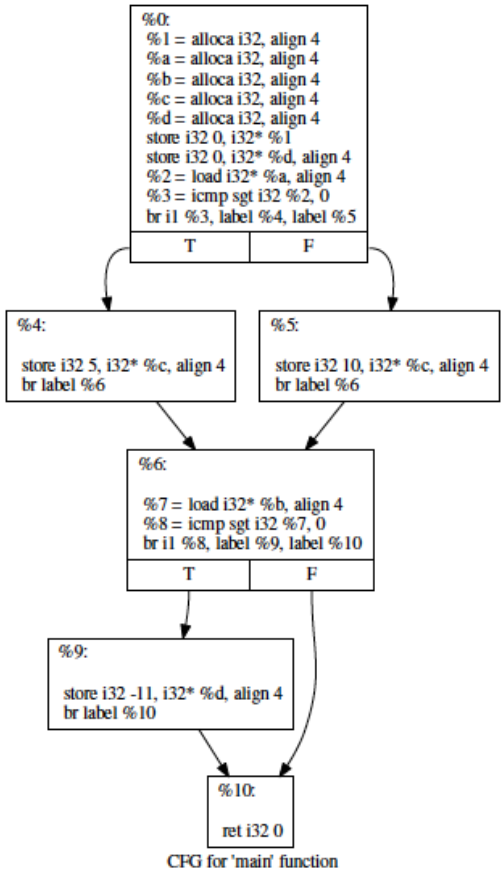
Task 2

Based on the framework from Task 1, the implementation is shown as the following algorithm for loop-free program analyses.

Algorithm

varMap: a mapping of a variable to its corresponding value for each basic block
initVars: save all the allocated variables into the **varMap**
updateVars: traverse over each basic block **BB**, update the **varMap** according to each **Instruction**
printResult: after each of such blocks, calculate the separation between distinct values, and print them out.
First, we initialize **varMap** by invoking **initVars**. Then, in each **BB** the **updateVars** function will be called to update all the variables in the map. Immediately before the end of each **BB**, we use **printResult** to show the differences among the variables from the map.

Using the aforementioned algorithm, we shall take the same example program from Task 1, and traverse over each block of the program. The program analysis is explained with the CFG below. For brevity, it is possible to ignore the virtual registers generated by LLVM as we will do for Task 3 as well. Note that this algorithm does not support to calculate any absolute difference of uninitialized variables.



CFG for 'main' function

Block 1: // %0

First path
Block 2: sep(c, d) = 5 // %4
Block 3: sep(c, d) = 5 // %6
Block 4: sep(c, d) = 16 // %9
Block 5: sep(c, d) = 16 // %10

Second path
Block 6: sep(c, d) = 5 // %10

Third path
Block 7: sep(c, d) = 10 // %5
Block 8: sep(c, d) = 10 // %6
Block 9: sep(c, d) = 21 // %9
Block 10: sep(c, d) = 21 // %10
Forth path
Block 11 : sep(c, d) = 10 // %10

Hence, among these evaluations the greatest separation is **sep(c, d) = 21**.

Task 3

In order to support the difference analysis for loop programs, we shall modify our implementation based on part of the work from Task 1 and Task 2, incorporate the interval concept from the abstract interpretation, and define an algorithm that reaches a fixed point.

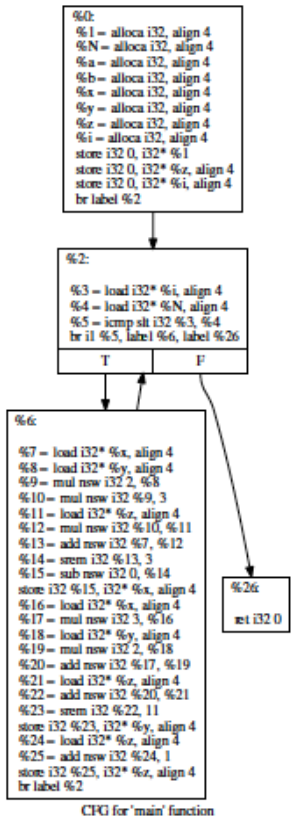
Algorithm

intervalMap: a global mapping of a variable to its corresponding interval
initVars: insert all the allocated variables and their infinity intervals into the **intervalMap**
updateVars: update **intervalMap** according to each instruction, with interval arithmetic
printResult: calculate the separation between distinct variables, and print the result out for each **BB**
reachFixedPoint: compare the old **intervalMap** and the new **intervalMap**
First, every variable is initialized to an interval of negative infinity and positive infinity. When traversing over each **BB**, we update the variables with **updateVars,** print the separation result by **printResult**, and check the interval change for each variable with **reachFixedPoint**. If all the intervals have not changed for some times, we claim that the analysis reaches a fixed point, and exit the analysis program.

In the example program

$$[i := 0]^1; while\ [i < N]^2\ ;$$
$$do\ ([x := (-(x + 2y3z))\ \%\ 3\ ]^3; [y := (3x + 2y + z)\ \%\ 11]^4;$$
$$[z := z + 1]^5)$$

we shall notice that the analysis is bounded by modulo 3 and 11, respectively, such that $|x| \leq 2$ and $|y| \leq 10$. However, since we cannot guarantee the choice of N, z could be infinite. To clearly show the result, only finite values of the variables are presented below.



CFG for 'main' function

Block 1: sep(z, i) = 0 // %0
Block 2: sep(z, i) = 0 // %2
Block 3: sep(x, y) = 12, sep(x, i) = 2, sep(y, i) = 10 // %6
Block 4: sep(x, y) = 12, sep(x, i) = 2, sep(y, i) = 10 // %2
Block 5: sep(x, y) = 12, sep(x, i) = 2, sep(y, i) = 10 // %6
Block 6: sep(x, y) = 12, sep(x, i) = 2, sep(y, i) = 10 // %2
Block 7: sep(x, y) = 12, sep(x, i) = 2, sep(y, i) = 10 // %6
Block 8: sep(x, y) = 12, sep(x, i) = 2, sep(y, i) = 10 // %26
Block 9: sep(x, y) = 12, sep(x, i) = 2, sep(y, i) = 10 // %26
Block 10: sep(x, y) = 12, sep(x, i) = 2, sep(y, i) = 10 // %26

Hence, among these evaluations the greatest separation is **sep(x, y) = 12**.

Reference

[1] Flemming Nielson, Hanne Riis Nielson, and Chris Hankin. 1999. Data Flow Analysis. In *Principles of Program Analysis*. Springer Berlin Heidelberg, Berlin, Heidelberg, 35–139. DOI:https://doi.org/10.1007/978-3-662-03811-6_2