WEIYILEI A0276571W E1132304@u.nus.edu

Task 1: Designing Difference and Interval Analysis

1. Difference Analysis

1) Finite Lattice: The interval lattice will contains each variable occurred during the analysis and its interval(p(variable->interval)), each interval is in the format [lower bound, upper bound]. The bottom lattice is an empty set and the top lattice is all variables with an interval. Then the difference lattice will contains each variable pair occurred during the analysis and their difference(p([variable1, variable2]->difference)), each difference is an integer. The bottom lattice is an empty set and the top lattice is all possible combinations of variables and their corresponding difference. The lattice uses the partial order of the set inclusion relation, i.e., if D1 is a subset of D2, then D1 $\subseteq$ D2.

2) Galois connection: $\alpha(ZZ)=\{|z_1-z_2| \mid (z_1, z_2)\in ZZ\}$

$\gamma(Z) = \{(z_1, z_2) \mid |z_1-z_2|\in Z\}$

C = all combinations of every two variables

$A = \{\perp\} \cup \{x \mid x\in Z\}$ for difference analysis

$A = \{\perp\} \cup \{[p, q] \mid p\in\{-\infty\} \cup Z, q\in Z\cup\{\infty\}, p <= q\}$ for intervals

3) Abstract semantics: If we want to get the difference between each variable in each block at the end, then we have to maintain the interval range of each variable in each block synchronously during analysis. The abstract semantics are as follows.

[a,b] => [a,b]

-[a,b] => [-b,-a]

[a,b]+[c,d] => [a+c,b+d]

[a,b]-[c,d] => [a-d,b-c]

[a,b]*[c,d] => [min(ac,ad,bc,bd), max(ac,ad,bc,bd)]

[a,b]/[c,d] =>

$$\begin{cases} \perp, \text{if } c = d = 0 \\ [\min(a/1, a/d, b/1, b/d), \max(a/1, a/d, b/1, b/d)], \; if\ c = 0\ and\ d \neq 0 \\ [\min(a/c, a/(-1), b/c, b/(-1)), \max(a/c, a/(-1), b/c, b/(-1))], \; if\ c \neq 0\ and\ d = 0 \\ [\min(a/c, a/d, b/c, b/d), \max(a/c, a/d, b/c, b/d)], \text{otherwise} \end{cases}$$

$$[a,b]\%[c,d] => \begin{cases} \perp, \text{if } c = d = 0 \\ [\min(a\%c, a\%d, b\%c, b\%d), 0], \text{if } d \leq 0 \\ [0, max(a\%c, a\%d, b\%c, b\%d)], \; if\ c \geq 0 \\ [\min(a\%c, a\%d, b\%c, b\%d), max(a\%c, a\%d, b\%c, b\%d)], \; otherwise \end{cases}$$

When the analysis reaches a fixpoint, the final differences will be calculated based on these intervals.

$X^{\#}: L \to D$ any solution of $\begin{cases} x_e^{\#} \text{ such that } x_e^{\#} \leq \gamma(x_e^{\#}) \\ x_{s+e}^{\#} \geq^{\#} \cup C^{\#}[c] x_s^{\#} \end{cases}$

$\forall z \in L: \gamma(x_e^{\#}) \geq x_z$

4) Forward, may and widen: Difference analysis is a forward analysis, meaning that the analysis follows the direction of the control flow of the program. It starts at the entry point of

the program and progressively updates the intervals of all variables, as instructions are executed. As we are going to find the maximum difference between any two variables, so we need to widen the intervals of each variable, from this perspective it's a may. But as the true difference of pair (x,y) must satisfy that it is equal to or less than the analyzed difference, so from this perspective it can be regarded as a must.

5) Monotonicity: If there is a path from A to B and A≤B, then A must be a subset of B.

6) Transfer functions: $f_l(l) = (l \backslash kill([B]^l)) \cup gen([B]^l)$, where $[B]^l \in blocks(S^*)$

$Kill(a = exp)^l$ = {interval of a}

$Gen(a = exp)^l$ = {new interval of a}

2. Interval Analysis

1) Finite Lattice: The lattice will contains each variable occurred during the analysis and its interval(p(variable->interval)), each interval is in the format [lower bound, upper bound]. The bottom lattice is an empty set, the top lattice is all variables with an interval. The lattice uses the partial order of the set inclusion relation, i.e., if I1 is a subset of I2, then I1 ⊆ I2.

2) Galois connection: $\gamma([p,q]) = \{x \in Z | p <= x <= q\}$, where $\gamma(\perp)$ = empty set.

$\alpha(c)=[inf(c), sup(c)]$, if $c \neq 0$, where $\alpha(\text{empty set}) = \perp$

C = all variables

$A = \{\perp\} \cup \{[p, q] \mid p \in \{-\infty\} \cup Z, q \in Z \cup \{\infty\}, p <= q\}$

3) Abstract semantics: All the interval part is the same as difference analysis, but as interval analysis is path sensitive, so we won't union all the predecessor's intervals, instead, we need some more abstract semantics to implement path sensitive, so that we can judge whether the branch condition is true or not.

$$J(X = Y) = \begin{cases} false, if\ b < c\ or\ a > d \\ true, otherwise \end{cases}$$

$$J(X \neq Y) = \begin{cases} false, if\ max(a, c) \leq min(b, d) \\ true, otherwise \end{cases}$$

$$J(X < Y) = \begin{cases} false, if\ a \geq d \\ true, otherwise \end{cases}$$

$$J(X > Y) = \begin{cases} false, if\ b \leq c \\ true, otherwise \end{cases}$$

$$J(X \leq Y) = \begin{cases} false, if\ a > d \\ true, otherwise \end{cases}$$

$$J(X \geq Y) = \begin{cases} false, if\ b < c \\ true, otherwise \end{cases}$$

4) Forward and may: Interval analysis is also forward analysis, following the direction of program control flow. However, since it is path-sensitive, at conditional branches, the analysis follows the reachable branching paths, respectively. This means that unreachable code paths are not considered by the analysis. It's also a may.

5) Monotonicity: Interval analysis also satisfies the monotonicity condition, if the mapping I1 of a variable to an interval contains I2, then for any transfer function F, there is F(I1) contained in F(I2)

6) Transfer functions: Same as difference analysis, only need to consider branch condition.

Task 2: Implementing the Difference and Interval Analysis

1. Difference Analysis

We use a set *blocks* to record all the updated blocks during each iteration, and a map called *lastAnalysisMap* to judge whether a fixpoint is reached. First of all, all blocks are added to *blocks* and the analysis program will execute them one by one in order. All variables will be initiated with interval$[-\infty, \infty]$. Before we deal with a complete block, we need to union the predecessors' interval as long as they are updated, using the widen method. After each iteration, we look at the analysisMap and whenever it is updated, we need to do a new iteration. Only from the second iteration onwards, the entry block no longer needs to be analyzed. Finally when fixpoint arrives, we calculate the maximum difference between the variables pair-wise for each blocks based on the intervals.

The results are as follows.

1) Example1_1.cpp

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

```
yileiwei@tracerx:~/Desktop/cpp_files$ ./DifferenceAnalysis example1_1.ll
PRINTING Difference Analysis:
Analysis for block %0:
The difference between w and x is INFINITE
The difference between w and y is INFINITE
The difference between w and z is INFINITE
The difference between x and y is INFINITE
The difference between x and z is INFINITE
The difference between y and z is INFINITE
Analysis for block %11:
The difference between w and x is INFINITE
The difference between w and y is INFINITE
The difference between w and z is INFINITE
The difference between x and y is INFINITE
The difference between x and z is 44
The difference between y and z is INFINITE
Analysis for block %12:
The difference between w and x is INFINITE
The difference between w and y is INFINITE
The difference between w and z is INFINITE
The difference between x and y is INFINITE
The difference between x and z is 44
The difference between y and z is INFINITE
Analysis for block %4:
The difference between w and x is INFINITE
The difference between w and y is INFINITE
The difference between w and z is INFINITE
The difference between x and y is INFINITE
The difference between x and z is 2
The difference between y and z is INFINITE
Analysis for block %5:
The difference between w and x is INFINITE
The difference between w and y is INFINITE
The difference between w and z is INFINITE
The difference between x and y is INFINITE
The difference between x and z is 4
The difference between y and z is INFINITE
Analysis for block %6:
The difference between w and x is INFINITE
The difference between w and y is INFINITE
The difference between w and z is INFINITE
The difference between x and y is INFINITE
The difference between x and z is 12
The difference between y and z is INFINITE
yileiwei@tracerx:~/Desktop/cpp_files$
```

2. Interval Analysis

The implementation of interval updating is the same as Difference Analysis, but as mentioned before, we need to take branch condition into consideration. So if the branch instruction is unconditional, inheritance of intervals from the parent block remains as usual. But if the branch is conditional, for those branches that can't satisfy the condition, we'll give them an empty set instead of the parent set, so that the program won't make any progress on the unreachable branches, which serves the purpose of path sensitive and narrow down the intervals.

1) Example1_2.cpp

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

```
yileiwei@tracerx:~/Desktop/cpp_files$ ./IntervalAnalysis example1_2.ll
PRINTING Interval Analysis:
Analysis for block %0:
The interval of x is [NEGATIVE_INF, POSITIVE_INF]
The interval of y is [NEGATIVE_INF, POSITIVE_INF]
The interval of z is [10, 10]
Analysis for block %10:
The interval of x is [0, 0]
The interval of y is [NEGATIVE_INF, POSITIVE_INF]
The interval of z is [10, 10]
Analysis for block %13:
The interval of x is [0, 0]
The interval of y is [NEGATIVE_INF, POSITIVE_INF]
The interval of z is [10, 10]
Analysis for block %14:
Analysis for block %15:
Analysis for block %16:
The interval of x is [0, 0]
The interval of y is [NEGATIVE_INF, POSITIVE_INF]
The interval of z is [10, 10]
Analysis for block %4:
Analysis for block %7:
The interval of x is [0, 0]
The interval of y is [NEGATIVE_INF, POSITIVE_INF]
The interval of z is [10, 10]
yileiwei@tracerx:~/Desktop/cpp_files$
```

Since we need to reach path sensitive, *y==10* is also unreachable here, as I understand.

2) Example1_3.cpp

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

```
yileiwei@tracerx:~/Desktop/cpp_files$ ./IntervalAnalysis example1_3.ll
PRINTING Interval Analysis:
Analysis for block %0:
The interval of x is [NEGATIVE_INF, POSITIVE_INF]
The interval of y is [10, 10]
The interval of z is [5, 5]
Analysis for block %10:
The interval of x is [0, 0]
The interval of y is [10, 10]
The interval of z is [5, 5]
Analysis for block %13:
The interval of x is [0, 0]
The interval of y is [10, 10]
The interval of z is [5, 5]
Analysis for block %14:
Analysis for block %15:
Analysis for block %16:
The interval of x is [0, 0]
The interval of y is [10, 10]
The interval of z is [5, 5]
Analysis for block %4:
Analysis for block %7:
The interval of x is [0, 0]
The interval of y is [10, 10]
The interval of z is [5, 5]
```

Task 3: Adding Support for Loops to the Analysis

For difference analysis, we set a large enough value for POSITIVE_INF, and a small enough value for NEGATIVE_INF (9999 and -9999, respectively), so that when any variable is updated outside of any of these ranges, that portion of the variable interval is displayed as either POSITIVE_INF or NEGATIVE_INF, in order to help an infinitely looping program to reach the fixpoint, and to prevent never-ending runs in loops.

For interval analysis, based on the support for loops in difference analysis, the program will additionally keep running in the order in which it was originally designed, and as long as the loop conditions are no longer satisfied, analysisMap will not be updated, and the fixpoint will be reached.

1. Difference Analysis

1) Example2.cpp

```cpp
int main(){
    int a,b,x,y,z = 0;
    int N = 10;
    int i = 0;
    while(i < N){
        x = -((x + 2*y * 3*z) % 3);
        y = (3*x + 2*y + z) % 11;
        z++;
    }
}
```

```
yileiwei@tracerx:~/Desktop/cpp_files$ ./DifferenceAnalysis example2.ll
PRINTING Difference Analysis:
Analysis for block %0:
The difference between N and a is INFINITE
The difference between N and b is INFINITE
The difference between N and i is 10
The difference between N and x is INFINITE
The difference between N and y is INFINITE
The difference between N and z is 10
The difference between a and b is INFINITE
The difference between a and i is INFINITE
The difference between a and x is INFINITE
The difference between a and y is INFINITE
The difference between a and z is INFINITE
The difference between b and i is INFINITE
The difference between b and x is INFINITE
The difference between b and y is INFINITE
The difference between b and z is INFINITE
The difference between i and x is INFINITE
The difference between i and y is INFINITE
The difference between i and z is 0
The difference between x and y is INFINITE
The difference between x and z is INFINITE
The difference between y and z is INFINITE
Analysis for block %2:
The difference between N and a is INFINITE
The difference between N and b is INFINITE
The difference between N and i is 10
The difference between N and x is INFINITE
The difference between N and y is INFINITE
The difference between N and z is INFINITE
The difference between a and b is INFINITE
The difference between a and i is INFINITE
The difference between a and x is INFINITE
The difference between a and y is INFINITE
The difference between a and z is INFINITE
The difference between b and i is INFINITE
The difference between b and x is INFINITE
The difference between b and y is INFINITE
The difference between b and z is INFINITE
The difference between i and x is INFINITE
The difference between i and y is INFINITE
The difference between i and z is INFINITE
The difference between x and y is INFINITE
The difference between x and z is INFINITE
The difference between y and z is INFINITE
Analysis for block %26:
```

```
Analysis for block %26:
The difference between N and a is INFINITE
The difference between N and b is INFINITE
The difference between N and i is 10
The difference between N and x is INFINITE
The difference between N and y is INFINITE
The difference between N and z is INFINITE
The difference between a and b is INFINITE
The difference between a and i is INFINITE
The difference between a and x is INFINITE
The difference between a and y is INFINITE
The difference between a and z is INFINITE
The difference between b and i is INFINITE
The difference between b and x is INFINITE
The difference between b and y is INFINITE
The difference between b and z is INFINITE
The difference between i and x is INFINITE
The difference between i and y is INFINITE
The difference between i and z is INFINITE
The difference between x and y is INFINITE
The difference between x and z is INFINITE
The difference between y and z is INFINITE
Analysis for block %6:
The difference between N and a is INFINITE
The difference between N and b is INFINITE
The difference between N and i is 10
The difference between N and x is 12
The difference between N and y is 10
The difference between N and z is INFINITE
The difference between a and b is INFINITE
The difference between a and i is INFINITE
The difference between a and x is INFINITE
The difference between a and y is INFINITE
The difference between a and z is INFINITE
The difference between b and i is INFINITE
The difference between b and x is INFINITE
The difference between b and y is INFINITE
The difference between b and z is INFINITE
The difference between i and x is 2
The difference between i and y is 10
The difference between i and z is INFINITE
The difference between x and y is 12
The difference between x and z is INFINITE
The difference between y and z is INFINITE
yileiwei@tracerx:~/Desktop/cpp_files$
```

As mentioned in *"asg2-2024.pdf"*, "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."

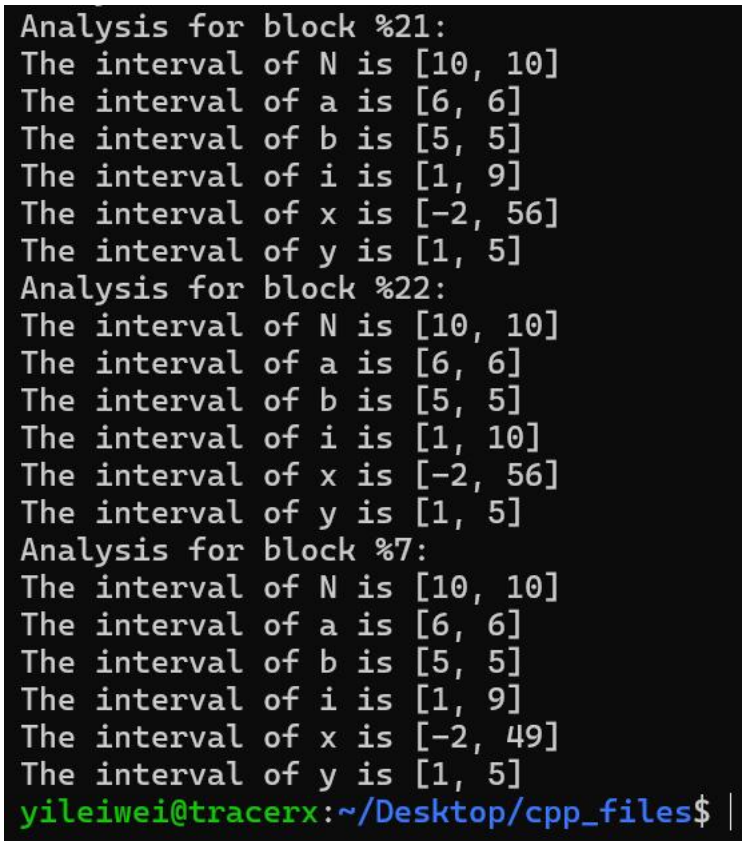
In this output, the block of loop end is %6, we can see that in block %6 sep(x,y) is 12 and sep(x,z) is INFINITE, which meets the expectation.

2. Interval Analysis

1) Example3.cpp

```
int main() {
    int a = -2,b = 5,x = 0, y;
    // assume N is an input value
    int N = 10;
    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;
        }
    }
}
```

```
yileiwei@tracerx:~/Desktop/cpp_files$ ./IntervalAnalysis example3.ll
PRINTING Interval Analysis:
Analysis for block %0:
The interval of N is [10, 10]
The interval of a is [-2, -2]
The interval of b is [5, 5]
The interval of i is [0, 0]
The interval of x is [0, 0]
The interval of y is [NEGATIVE_INF, POSITIVE_INF]
Analysis for block %10:
The interval of N is [10, 10]
The interval of a is [6, 6]
The interval of b is [5, 5]
The interval of i is [1, 9]
The interval of x is [5, 56]
The interval of y is [5, 5]
Analysis for block %13:
The interval of N is [10, 10]
The interval of a is [-2, -2]
The interval of b is [5, 5]
The interval of i is [1, 1]
The interval of x is [-2, -2]
The interval of y is [1, 1]
Analysis for block %16:
The interval of N is [10, 10]
The interval of a is [-2, -2]
The interval of b is [5, 5]
The interval of i is [1, 9]
The interval of x is [-2, 56]
The interval of y is [1, 5]
Analysis for block %19:
The interval of N is [10, 10]
The interval of a is [6, 6]
The interval of b is [5, 5]
The interval of i is [1, 9]
The interval of x is [-2, 56]
The interval of y is [1, 5]
Analysis for block %2:
The interval of N is [10, 10]
The interval of a is [6, 6]
The interval of b is [5, 5]
The interval of i is [1, 10]
The interval of x is [-2, 56]
The interval of y is [1, 5]
Analysis for block %20:
Analysis for block %21:
```



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

Meet the expectation.