Linear equalities analysis

Software correctness, security, and reliability

[CM0476]

Nicolas Pietro Martignon (870034)

Thomas Visentin (869438)

Introduction

the LISA Analyzer was used to implement the analysis described in Michael Karr's paper, entitled "Affine Relationships Among Variables of a Program", the purpose of the analysis is to identify possible affine relationships between the variables and, once found, determine them numerically.

Assumption

An algorithm was actually implemented on the basis of example 2 (Increasing Operator Strength) reported in the paper, in particular, therefore, for the analysis we consider programs in which:

- there is a block of code of arbitrary line length before the while

- one while loop

- for reasons of variable settings, it is necessary to analyse one program at a time

There is also the further assumption that an expression can only consist of the + or - operators, as shown in the example above.

High-level descriptions of the code

The program can be conceptually divided into the following sections also highlighted in the source code:

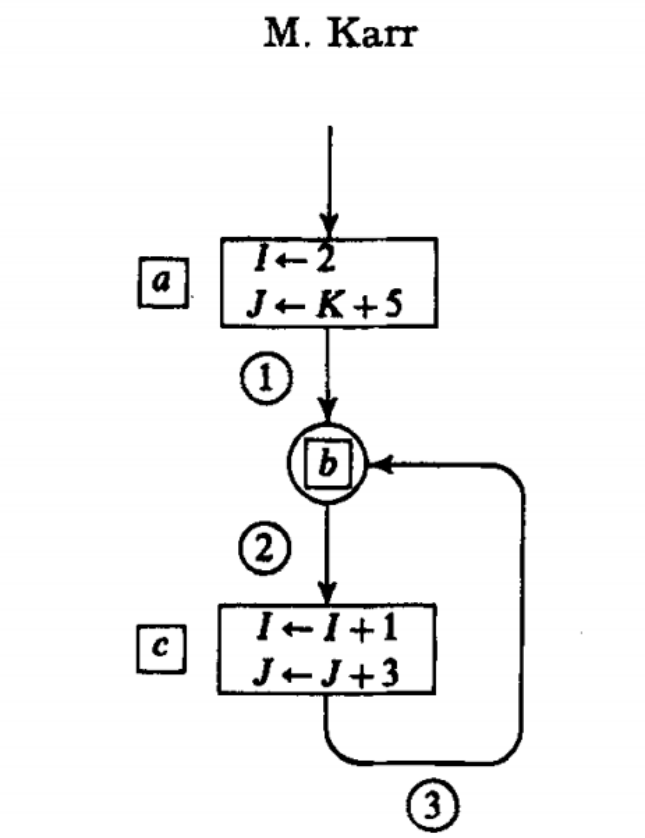
"loop detector": identifies the presence of a while loop inside the program passed as input, by comparing the instructions and their repetition, in particular we check if it is possible to reach an instruction from 2 different points of the program

"line detector and block creation": the "separating line" is identified between the code block before the while and the code inside the while itself.

In the following code blocks the necessary calculations are carried out (refer to Karr's paper), of which at this point we have all the data necessary to apply this algorithm.

Example of application of the algorithm

We will now show an example of application of the algorithm on the real case shown by Karr in his paper.



This visual representation has been converted into imp language, as follows:

```
class LinearEqualities {
   test2() {
       def k = 0;
      def i = 2;
      def j = k +5 ;

        while (i < 100) {
            i = i + 1;
            j = j + 3;

        }
   }
}
```

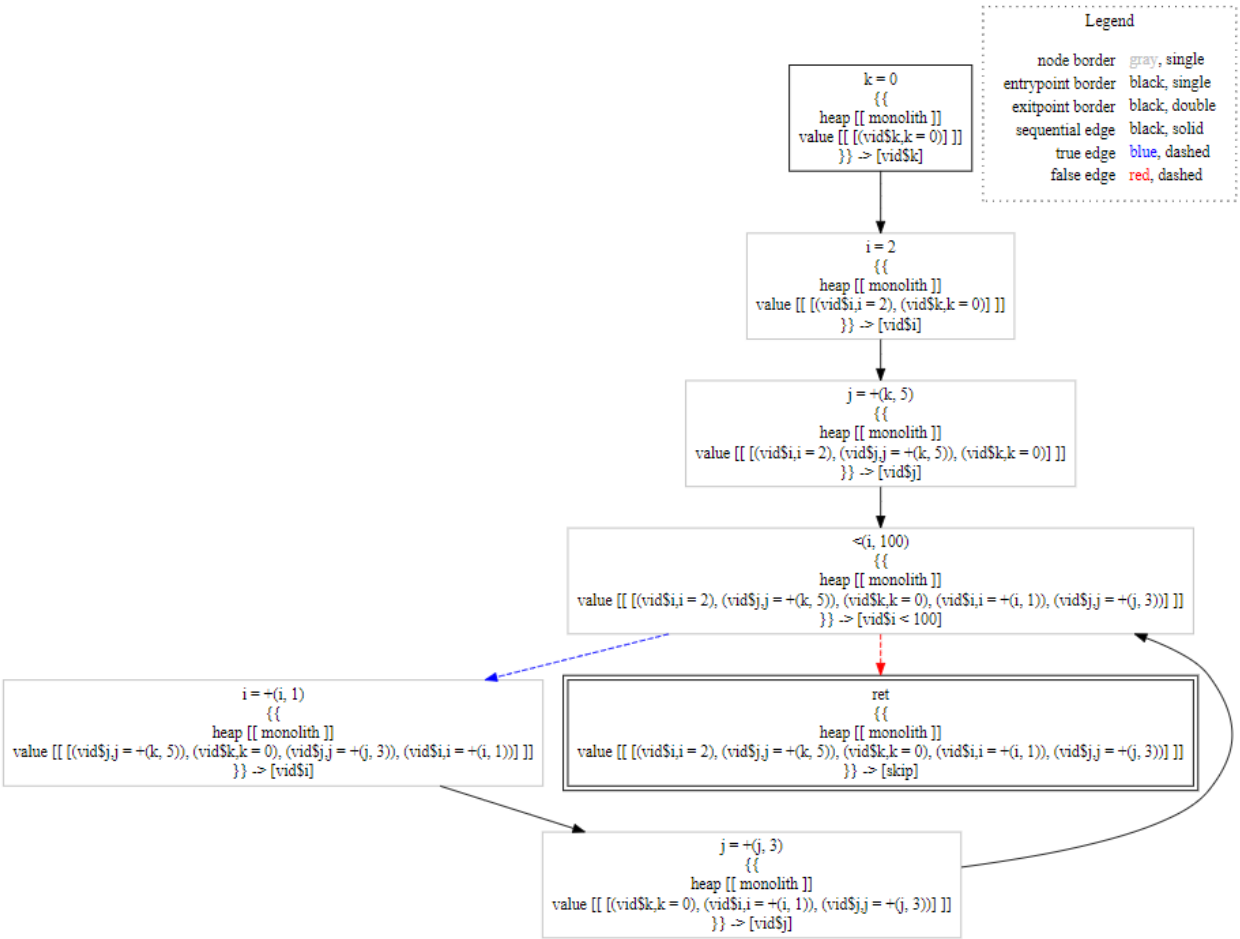
The program returns the result of the processing to the console, in particular:

- the matrices (before and after the loop)

- the test on their comparison, if any of the 2 matrices match, it corresponds to the relationship (affine relationship) between the variables

```
Linear Equalities Analysis
-------------------------------------------------------
First Matrix: {R=0.33333325, vid$i=1.0, vid$j=-0.33333334, vid$k=0.33333334}
Second Matrix: {vid$k=0.33333334, R=0.33333325, vid$i=1.0, vid$j=-0.33333334}
Matrix Match!
-------------------------------------------------------
```

The program also returns a visual representation of the block instructions (file.dot), in the value field there is a simple reaching definition however useful for identifying the conditional jump.



Other example

We also tried other working examples in our application, listed below.

1)

```
Test1() {
            def a = 0;
            def b = a-2;
            def c =  3 ;

            while (a < 100) {
                b = b - 2;
                c = c + 2;

            }
        }
```

In this program the 2 matrices are not equal therefore there is no linear equality.

```
Linear Equalities Analysis
-----------------------------------------------------
First Matrix: {vid$c=1.0, R=2.5, vid$a=-0.25, vid$b=0.25}
Second Matrix: {vid$c=1.0, R=4.0, vid$a=-0.25, vid$b=0.25}
Matrix DON'T Match!
-----------------------------------------------------
```

2)

```
Test3() {
           def k = 0;
           def i = 3;
           def j = k +4 ;

           while (i < 100) {
               i = i + 3;
               j = j + 1;

           }
       }
```

```
Linear Equalities Analysis
-----------------------------------------------------
First Matrix: {R=1.6666666, vid$i=1.0, vid$j=-0.33333334, vid$k=0.33333334}
Second Matrix: {vid$k=0.33333334, R=4.333333, vid$i=1.0, vid$j=-0.33333334}
Matrix DON'T Match!
-----------------------------------------------------
```

Another program where the 2 matrices don’t match, therefore there is no linear equality.

Conclusion

The algorithm developed present some limitations due to the vastness of programs that can be passed in input, through assumptions allows to carry out a significant static analysis that allows us to know any related relationships. If identified, allow to simplify the structure of the code and consequently the computational complexity, allowing to calculate a single variable and to derive the others as a function of the latter.

The assumptions made in the relevant paragraph can be relaxed for example by considering a for loop instead of a while loop, considering that the while loop can be before the code block.

Such program improvements can be achieved with little effort, as the core algorithm has already been implemented.