UNIVERSITÄT ZU LÜBECK
INSTITUTE FOR SOFTWARE ENGINEERING
AND PROGRAMMING LANGUAGES

isp

Development of Safety Critical Systems (Sichere Software) Winter term 18
Martin Leucker, Karam Kharraz, Felix Lange March 25, 2019

Exercise Sheet 3

Deadline:

Frama-C is a platform dedicated to the analysis of source code written in the C programming language. The Frama-C platform gathers several analysis techniques into a single collaborative extensible framework. The goal is to combine the results computed by different analyses techniques organized as Plug-ins. The tool rely on a formal specification language called ACSL[1] to express properties, assumptions *etc.*
The Eva[2] (Evolved value analysis) plug-in computes variation domains for program variables using abstract interpretation techniques. The installation steps of Frama-C are available on the official website [3] . The source code of the lab tasks is available in Gitlab [4] .

Task 3.1 Hands on Eva with fixed-width integer types overflow

```
1  int average(int a,int b){
2    int z = (a+b)/2;
3    return z ;
4  }
```

The function *average* computes the average values of two numbers. In this implementation, the "developer" used the type *int*. Eva is able to detect possible *RTE* (RunTime Errors) without specifying the related properties in ACSL. Try the following command:

```
1  frama-c-gui -val average1.c -main average
```

the tool detects 2 possibilities of overflow with an orange flag. The average of two values in $[a, b]$ is always in $[a, b]$, why is the tool raising those two alarms. Is it a false positive?

Suppose that this function was designed for positive values only, we can specify this by adding an ACSL precondition by adding this lines before the function:

```
1  /*@ requires a > 0;
2      requires b > 0; */
```

Start the analysis again, what is the difference and why ?
Modify the code of function average to avoid the alarm(overflow) without modifying the output of the oracle tests $(12, 14), (11, 14), (11, 13)$.
Now, suppose that this function is designed for positive and negative values. Remove the preconditions, what is the result found by Eva for the modified function average? Why?
Modify the function code again to avoid the possible overflows and getting the same result for the aforementioned test oracles.

[1] https://frama-c.com/download/acsl-implementation-Chlorine-20180501.pdf
[2] https://frama-c.com/download/frama-c-value-analysis.pdf
[3] https://frama-c.com/install-aluminium-20160501.html
[4] https://gitlab.isp.uni-luebeck.de/kharraz/lab_Eva

Task 3.2 Eva: widening, precision, termination

Consider the following program:

```
1 int x, y = 50;
2 void main(){
3         while(y<100){
4                 y = y + (100 - y)/2;
5                 x= y / (y-124);
6                 }
7 }
```

There appears to be a risk of division by zero in line 5. Use the following command:

```
1 frama-c-gui -val divsion.c
```

What is the result ?
To unroll the loop execution execute the following command:

```
1 frama-c-gui -val divsion.c -ulevel 10
```

Is the value of y reaching a fixpoint? is the result of the previous analysis a true positive then?

Execute the analysis without widening, run the following command:

```
1 frama-c-gui -val divsion.c -slevel 100
```

Is there still an alarm, what is the new information given by the tool? What is the differences between the tool with option -slevel and CTF^I seen in the class?