Formal Method Project
Formal verification of a sorting function
ISEP
ISEP Cycle Ingénieur A3

Mykola CHOBAN, Pierre VERBE

Monday 18th January 2021

Introduction

As part of our training as a digital engineer specializing in software development, we must know how to perform proofs on C programs. In fact, this language is still widely used, especially due to its performance and maturity. It is available on many platforms. It is often chosen as the language of choice for the realization of systems requiring a critical level of safety. We can cite as an example the aeronautics, armament, etc... In order to ensure that these applications do not contain any unexpected behavior, it is possible to prove mathematically that no problem will appear at runtime. This is where frama-c takes all its importance. As part of our end-of-semester formal approach project we will prove a sort function.

Study of the sorting function

For this project we have chosen the sort function "sort_1.c" which is in the "plain" directory. We chose this sort function since we recognized it quite quickly. This sort is a bubble sort or also known as sinking sort. It sorts the numbers in an array from the smallest to the largest. We have already had the opportunity to implement and use this sort in other languages like java and python. It seemed logical to us to choose it since we as rather familiar with it.

Before launching into the proof phase with frama-c we had a bubble-sort analysis phase. This phase allows us to better understand the proof phase.

When entering the sort function we have a pointer “t” which will be our array and an integer ”l” which will be the size of our array. The size of the pointer array “t” should be similar to “l”. If “l” is too small the array will only be partially sorted. In case “l” is too big the program will plant since it will try to get a number to sort that does not exist.

The implementation of the sorting function is based on 2 loops for which are used to browse the array and compare the elements consecutively. The first loop using the variable “i” allows to position the progress of the sorting of the array. This means that at the beginning of the loop the numbers of the “i”-1 th previous cells of the array are sorted. The second loop using the variable “j” allows to compare the remaining numbers of the array to be sorted. If the “i” th cell of the array is larger than the “j” th cell of the array, it will be reversed. The aim is to keep the new smaller number in the “i” th cell of the array. It is the swap function which takes care of reversing the cells of the array.

For the swap function you have to enter a “t” pointer which is the array, an integer “l” which is the size of the array, an integer “I" which is the number of the cell of the array to be changed, and an integer “j” which is the number of the cell containing the new smallest number. The implementation of the sort function consists in temporarily storing the number of cell “i” in a temporary variable named “tmp”. Then it assigns the new smallest number (cell “j”) to the cell “i” of the array. And to finish, cell “j” of the array receives the number of the temporary variable.

Prove the Sorting function in Frama-C

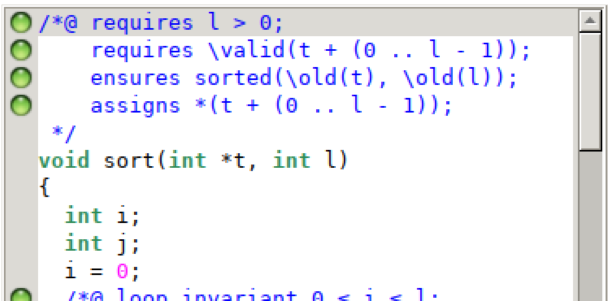
After analysing the bubble-sorting function in its entirety, we will be able to move on to the proof side. For this we will use a virtual machine on Lubuntu on which is installed frama-c.

For our proof we tried to understand what the result after each execution of code is, not less, not more, at each moment, that helped us to figure out the invariant in loops.

We have limited what the input functions expect, detected when the variables are subject to modifications. We have understood that after each loop of j we have t[i] smaller than all other and start after the first iteration, when i is different from 0.

For the loop in j, we tried to figure out what happened before j and what is for sure correct after.

After writing all the rules in the file for the sort and swap function, Frama-c confirmed that the proof is valid and complete.



```
/*@ requires l > 0;
    requires \valid(t + (0 .. l - 1));
    ensures sorted(\old(t), \old(l));
    assigns *(t + (0 .. l - 1));
  */
void sort(int *t, int l)
{
  int i;
  int j;
  i = 0;
  /*@ loop invariant 0 ≤ i ≤ l;
```

Figure 1. Result of the proof

Conclusion

Through the tp dedicated to frama-c and this tp at the end of the semester we were able to discover the concepts of frama-c and perform our first proofs of c programs. Our formal approach project consisted in mathematically proving the proper functioning of the bubble sort function. We finally managed to fully prove it, which means that the function has no unexpected behaviors.