CS292C Homework 1

Deepak Nathani

April 8, 2024

1 Self-Grade

Problem	Self-Grade
1: Dafny Installation	2
2: Minimum of 2D Array	0
3: Binary Search	0
4: Buggy Binary Search	0
5: Bubble Sort	0

2 Problems

Problem 1. Dafny Installation

Install Dafny by following these instructions. We recommend simply installing the Dafny VSCode extension. If you're on macOS or Linux, you will be prompted in VSCode to install .NET 6.0 once you open a .dfy file. Simply follow the link in the prompt to install it.

You should install Dafny 4. If you're prompted by VSCode to upgrade from Dafny 3 to Dafny 4, you should let it automatically upgrade.

The whole process should be fairly quick, taking no more than 5 minutes. Once you're done, run Dafny on the demo file we covered during the tutorial, and attach the output to your submission PDF.

Solution

```
tutorials/01-dafny/demo.dfy(19,4): Error: a postcondition could not be proved
on this return path
   |
19 |     return b; // BUG!
   |     ^^^^^^

tutorials/01-dafny/demo.dfy(16,10): Related location: this is the postcondition
that could not be proved
   |
16 |   ensures c >= a && c >= b && (c == a || c == b)
   |           ^^^^^^

Dafny program verifier finished with 4 verified, 1 error
```

Problem 2. Minimum of 2D Array

In min.dfy, you will find a Dafny method that finds the minimum of a 2D array. Your tasks are:

(a) State the correctness property of this method in terms of pre-condition(s) (using the requires keyword) and post-condition(s) (using the ensures keyword).

Your pre-condition(s) should allow Dafny to prove that there is no out-of-bound array access.

(b) Annotate each of the two loops with appropriate invariants that allow Dafny to verify the overall correctness of the method.

Hint

> The invariant for the outer should be very similar to your post-condition. For the inner loop, you may need to provide two different invariants: the first invariant talks about all elements from sub-arrays a[0] up to a[i-1] inclusive, and the second invariant specifically looks at the sub-array a[i] and talks about its elements between 0 and j-1 inclusive. It may help draw a dummy 2D array on paper, and mark the elements that have been processed so far when the loop is at (i,j)-th position. Those processed elements will be the ones (and the only ones) that are mentioned in the invariants.

(c) Attach your code with the annotations in your submission. Also, informally explain what your pre- and post-conditions, and loop invariants mean in plain English.

You should not change the implementation code in any way.

Solution

Problem 3. Binary Search

The file binary_search.dfy contains a method BinarySearch that performs binary search on a sorted array. The sorted-ness is guaranteed by the $Ordered(a)$ predicate, which is just a formula that asserts $a[i] <= a[j]$ for all $i <= j$. The post-condition is provided, which says that the method returns true if and only if the key x is contained in the array. Also, note that the while-loop is annotated with decreases $hi - lo$, which enables Dafny to prove that the loop terminates because the quantifier $(hi - lo)$ monotonically decreases in each iteration of the loop.

Your task is to replace invariant true in the while-loop of BinarySearch with an appropriate loop invariant that allows Dafny to verify the correctness of the method.

Hint:

> Your invariant needs to talk about where x cannot be found in the array based on the current values of lo and hi.

Attach your code and a brief explanation of your invariant in your submission.

Solution

Problem 4. Buggy Binary Search

Immediately below BinarySearch, we duplicated its implementation into another method called BuggyBinarySearch. The only change we made is to make the index variables (lo, mid, hi, etc.) into 4-bit integers (unsigned), whereas in Dafny int is the default infinite-precision integer.

Copy and paste the (working) invariant you wrote for BinarySearch into BuggyBinarySearch and see if Dafny can verify the correctness of the method. Note that to make the invariant well-typed, you may need to do some conversion from int to bv4 using the syntax x as $bv4$ which converts a variable of type int to bv4.

After that, you shall see that Dafny cannot verify the correctness of BuggyBV4BinarySearch. Your tasks are to:

(a) locate the exact line where Dafny reports an error,

(b) explain why Dafny cannot verify the correctness of the method

(c) patch only one line of implementation code in the loop body to make verification succeed.

Include your answers to the above questions in your submission. You should also attach the modified code with the fix in your submission.

Hint

> The answers to the above questions can be found in the Wikipedia page for binary search, under the section Implementation Issues

Solution

Problem 5. Bubble Sort

Implement bubble sort in Dafny, and prove its correctness. There are multiple levels you can aim for in terms of the complexity of your implementation and the strength of your correctness proof:

Your bubble sort outputs an ordered list Your bubble sort outputs an ordered list whose elements form the same set as the input list Your bubble sort outputs an ordered list whose elements form the same multiset as the input list You may choose to aim for any of the above levels, but you should clearly state which level you are aiming for in your submission.

You may find this tutorial on verifying selection sort in Dafny helpful.

Attach your implementation, including all necessary annotations, in your submission. You should also informally explain in plain English the meaning of your pre- and post-conditions, and the loop invariants you used.

Solution