CS 6110 Project Report

Project Title: VSComp Verification
By: Aisha Syed

Introduction:

In this project I took challenge problems from the Verified Software Competition [1] and tried to verify them using the Dafny programming language from Microsoft Research [2]. VSComp is an yearly competition with different variants (VSComp, VSTTE) and consists of about four or five verification problems. The solution then includes both coming up with the algorithm that solves the problem and then verifying it using any verification tools or languages like Dafny. Often times, the pseudocode is also provided to the contestents. Since the problems can be of arbitrary complexity and my goal was to learn Dafny verification, so I selected 4 problems of low to average complexity.

In the following four sections, I give a brief overview of each of the four selected problems and mention which of their properties I found most challenging or interesting. I also include references to the competition websites from which the problems were taken and the competition reports which decribe the results. My code can be found in the Git repository for the course.

1. Sum-Max:

Problem: Given an N-element array A, compute sum of all elements, and the maximum element.
Verify: Given N >= 0 and A[i] >= 0 *for 0* <= *i* < *N*, prove the post-condition that sum <= N * Max.

This problem turned out to be the easiest since the property they had asked to verify was simple. I used it to get more familiar with Dafny.

The problem was taken from VSComp 2010 [3]

2. Inversion:

Problem: Invert an injective array A on N elements in the subrange from 0 to N − 1. You may assume A is surjective.
Verify: Verify output array B is also injective:
B [A[i]] = i *(0 <= i < N)*

Example:
Input A = 1 3 0 2
Output B = 2 0 3 1

This problem was little bit more complicated in that it required thinking about writing appropriate invariants and tweaking them to prove the post-condition so it took a while but it did not require usage of any advanced Dafny constructs.

The problem was taken from VSComp 2010 [3]

3. Two-Duplets:

Problem: Find two duplets in given array. Assume at least two unique duplets exist in array. Assume input array a of length N+2 with N >= 2. May assume array values between 0 and N-1.
Verify: Verify the post-condition that your program finds 2 duplets from given array.

Similar to the previous problem, this problem also required a lot of time to be spent tweaking the loop invariants to get the post-condition to be verified. I could get it to verify one duplet but not both together and I tried writing the invariants in many different ways. Finally, I looked at one solution from the contest, they had divided the two-duplets problem so that first one duplet was found and verified and then the same function was called for getting the second duplet and verifying that. So, I decided to follow this idea (and just the idea), and divided my 'duplets' function into a 'duplet' function that returned only one duplet. The I used a main function that called this duplet function twice and verified that two duplets (such that duplet1 != duplet2) have been found.

The problem was taken from COST Verification Competition 2011 which was attached to the International Conference on Formal Verification of Object-Oriented (FoVeOOS) [6]. According to their report [7], most teams solved this problem within or shortly after the deadline, one of the teams was able to provide a complete solution after the competition finished but before the end of the conference.

4. TwoWay Sort:

Problem: Sort a boolean array using two pointers moving from the beginning and end of array, swapping values whenever a **TRUE** is seen before a **FALSE.**
Verify:

Safety:	Every array access within bounds
Termination:	Prove function always terminates
Behavior:	After execution, following should hold: (a) array is a permutation of initial contents (b) array is sorted in increasing order.

This problem turned out to be hardest of all for me. While it was a simple sort but it took the most time and outside help and it had some interesting properties that introduced me to some new Dafny constructs too so it was still worthwhile doing it.

The first different thing here was to write the post-condition for sortedness of booleans, which was a little different than the usual checks we use for all our integer sorts. The second interesting thing was ensuring the returned array was a permutation of the original array. I wrote a simple total method for doing that but when I ran Dafny, it threw errors which turned out to be because you can't use methods in writing post-conditions. This was a new thing for me. I did some research and found out about the usage of one line functions and predicates but after many days of trying out different implementations of total function, I couldn't get it to work in just one line. So, I finally Googled how to write permutations in Dafny and found some course slides that explained it using recursion. It was written in terms of Dafny 'sequences' but since this was also a new construct for me so to give myself a little exercise, I converted it to use arrays instead of immutable sequences. The implementation of total and the predicate 'compare' that calls it is given below.

```
method TwoWaySort(a:array<bool>)
  ensures compare(a, old(a)); //ensuring permutation
  ensures forall k:: forall l:: 0 <= k < l < a.Length
                      ==> !a[k]||a[l] //sortedness
{
      i := 0; j := a.Length - 1;
      while (i <= j)
      invariant 0 <= i <= (j+1);
      invariant (i-1) <= j < a.Length;
      decreases (j+1) - i; decreases j - (i-1);
      invariant forall k:: 0 <= k < i ==> !a[k];
      invariant forall l:: j < l < a.Length ==> a[l];
      {
         ... //move pointers i,j and swap when TRUE before FALSE
      }
}
predicate compare (oldArray: array<bool>, newArray: array<bool>)
reads a; reads a1;
requires a != null; requires a1 != null;
{
   forall key:bool :: total(oldArray, oldArray.Length, key)
                        ==
                      total(newArray, newArray.Length, key)
}

function total(a: array<bool>, i:int, key: bool): int
requires a != null; reads a;
{
      if i < 0
            then 0
      else if (a.Length == 0)
            then 0
      else if 0<=i<a.Length && a[i] == key
            then total(a, i-1, key) + 1
      else total(a, i-1, key)
}
```

The problem was taken from VSComp 2012 [4]. According to the results, total of 25 contestants submitted a solution for this problem that was able to attempt at least one task from the problem, 20 of those were able to attempt all tasks for the problem, and 19 of those were found to be perfect solutions. [5]

Conclusion:

The first three problems helped me practice writing programs in Dafny, learning the meaning of its various errors, and gave me a good exercise in writing post-conditions and tweaking loop invariants to fit the specific post-condition. The final fourth problem, while it required help from Google with regards to writing permutation verification but since my main goal was to learn Dafny so I still feel that I may not have invented anything but it did help me exercise with some new Dafny constructs: sequences and their usage compared to arrays; functions and predicates and their difference with normal methods. In short, I think VSComp verification is a good place

for anyone who is starting to learn verification and while it was little frustrating at times when Dafny compiler didn't give any good explanations for the errors but it was still fun to do the challenges.

References:

[1] VSComp. http://vscomp.org
[2] Microsoft Dafny. http://dafny.codeplex.com
[3] VSComp 2010. http://www.macs.hw.ac.uk/vstte10/Competition_files/Competition.pdf
[4] VSComp 2012. https://docs.google.com/viewer?a=v&pid=sites&srcid=ZGVmYXVsdGRvbWFpbnx2c3R0ZTIwMTJ8Z3g6OWE2MjQ3YzA5ZmRjOWQ4
[5] VSComp 2012 Experience Report. https://www.lri.fr/~filliatr/pub/compare2012.pdf
[6] COST IC0701 Verification Competition 2011. http://foveoos2011.cost-ic0701.org/verification-competition
[7] COST IC0701 Verification Competition 2011 Report. https://docs.google.com/viewer?a=v&pid=sites&srcid=Y29zdC1pYzA3MDEub3JnfGZvdmVvb3MyMDExfGd4Ojc1YzdjOWE1Yjk2Yjc4YjM