Formal Methods in Software Development
Course 14. Practical Dafny

Mădălina Eraşcu

Content based on the book Leino, K. Rustan M. Program Proofs. MIT Press, 2023; Dafny resources
https://dafny.org/latest/toc

Thanks to Costel Anghel, 3rd year Bachelor student, Applied Informatics

June 5, 2024

Dafny Useful Resources

- Useful Dafny documentation: https://dafny.org/latest/toc
- Understanding the usefulness of formal verification in software as well as familiarisation with Dafny IDE and pre/post-conditions: https://www.youtube.com/watch?v=oLS_y842fMc
- Tutorial for loop invariants: https://www.youtube.com/watch?v=J0FGb6PyO_k
- Tutorial on arrays: https://www.youtube.com/watch?v=-_tx3lk7yn4

Recalling from previous lecture

Sum of first n natural numbers. Verify it in Dafny.

Exercises

1. Similar to the ideas from sum, write an imperative program which computes `Fibonacci(n)`, i.e. the sequence `0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...`

Exercises

1. Similar to the ideas from sum, write an imperative program which computes `Fibonacci(n)`, i.e. the sequence `0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...`
2. Write an algorithm with complexity $\mathcal{O}(\log n)$ which searches an element in a sorted array of integers.

Ensuring Modularity with Predicates

In the BinSearch algorithm, we could write a logical formula inplace to express the preconditon that the array must be sorted:

```
1  requires forall i,j :: 0 <= i < j < a.Length ==> a[i] <= a[j]
```

Modularity can be ensured by defining a predicate.

```
1  predicate sorted(a: array<int>)
2  reads a
3  {
4    forall j, k :: 0 <= j < k < a.Length ==> a[j] <= a[k]
5  }
6
7  requires sorted(a)
8  ensures ...
9  method BinSearch (...)
10 {
11 ...
12 }
```

Reads Clauses

```
1 predicate sorted(a: array<int>)
2 reads a
3 {
4   forall j, k :: 0 <= j < k < a.Length ==> a[j] <= a[k]
5 }
```

Remark

The predicate `sorted` depends on the values of the given array `a`, so it must include the `reads` specification.

Remark

If a method/function/predicate accesses the elements of an array `a`, then its specification must include `reads a`.

Remark

A method/function/predicate cannot modify anything, so it does not have a `write` frame (We don't want the values of the array `a` given as argument to be changed in another place!), but a `read` frame exists which announces the method/function/predicate dependencies on the heap (where memory is dynamically allocated which is the case for arrays in Dafny). This dependency information is used to determine if various mutations of the heap affect the value of the function (which is especially important if the body of the method/function/predicate is not available or if it is recursive).