Formal Methods in Software Engineering
Laboratory 1

1. Install Dafny and familiriaze with its use.
2. Read first the blog *Clear Separation of Specification and Implementation in Dafny* [8].
3. Define a predicate $IsOddNat(x)$ that holds when the integer x is an odd natural number.
4. Define a subsort Odd, defining the odd natural numbers.
5. Define a predicate $IsEvenNat(x)$ that holds when the integer x is an even natural number.
6. Define a subsort $Even$, defining the even natural numbers.
7. Define a lemma showing that the addition of two odd numbers is an even number.
8. Define a predicate $IsPrime(x)$, which holds when the integer x is a prime number.
9. Define a lemma showing that any prime number $x > 2$ is an odd number.
10. Define a type *int32* that represents signed integers with values that range from negative 2,147,483,648 (which is represented by the Int32.MinValue constant) through positive 2,147,483,647 (which is represented by the Int32.MaxValue constant). Define functions that implements the operations of this type in the same way your preferred programming language does it.

References

[1] Dafny Site. `https://dafny.org/dafny/`

[2] Dafny: Visual Studio Code Extension. `https://marketplace.visualstudio.com/items?itemName=dafny-lang.ide-vscode`

[3] Dafny: Github Repository.
`https://github.com/dafny-lang/dafny`

[4] Dafny: Emacs Mode.
`https://github.com/boogie-org/boogie-friends`

[5] Dafny: Command Line Options.
`https://manpages.debian.org/testing/dafny/dafny.1.en.html`

[6] DAFNY POWER USER.
http://leino.science/dafny-power-user/

[7] Verier Luke Herbert, K. Rustan M. Leino, and Jose Quaresma. Using Dafny, an Automatic Program. LNCS, volume 7682

[8] Aaron Tomb. Clear Separation of Specification and Implementation in Dafny
`https://dafny.org/blog/2023/08/14/clear-specification-and-implementation/`

[9] K. Rustan M. Leino. Program Proofs. The MIT Press, 2023
`https://mitpress.mit.edu/9780262546232/program-proofs/`

A Many Sorted Logic in Dafny

A.1 Builtin theories

Each builtin type defines in fact many sorted logic theory. For instance, the type `int` defines the following theory:

Sorts: `int`

Function symbols: $\Sigma_{[],\texttt{int}} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$ (all integers), $\Sigma_{\texttt{int int},\texttt{int}} = \{+, -, +, *, /, \%\}$, $\Sigma_{\texttt{int},\texttt{int}} = \{-\}$, ...

Predicate symbols: $\Pi_{\texttt{int int}} = \{<, <=, >, >=\}$

Formulas expressing, e.g., that the binary operators are left associative.

A.2 User defined sorts

Example:

```
newtype int8 = x: int | -128 <= x < 128
```

A.3 User defined functions

Example:

```
 function addInt8(a: int8, b: int8): int8
{
  var res := a as int + b as int;
  if (-128 <= res < 128) then res as int8 else 0
}
```

A.4 User defined predicates

Example:

```
predicate evenInt8(x: int8)
{
  (x as int) % 2 == 0
}
```