CITS5501 Software Testing and Quality Assurance
Semester 1, 2022
Workshop 9 (week 10) – Formal methods

Reading

It is strongly suggested you complete the recommended readings for weeks 1-9 *before* attempting this lab/workshop.

A. Applicability of formal methods

Discuss the following scenarios. Would you use formal methods for any of the following systems? If so, which systems, and why?

a. The next version of *Confectionery Crush Story*, a web– and mobile-app–based puzzle video game your company develops. The game is available for both Android and iOS mobile devices, and the previous version grossed over $US 1 billion in revenue.

b. *Exemplarex*, software produced for Windows and MacOS operating systems and licensed to educational institutions. The software semi-automatically invigilates exams set by the institutions: machine-learning techniques are used to analyse audio and video of exam candidates to identify possible academic misconduct.

c. The online banking website provided by a major Australian bank, EastPac. Over 5 million customers use the website to perform banking transactions on personal or business bank accounts.

d. A radiation therapy system used to treat cancer patients. The system has two principal components:

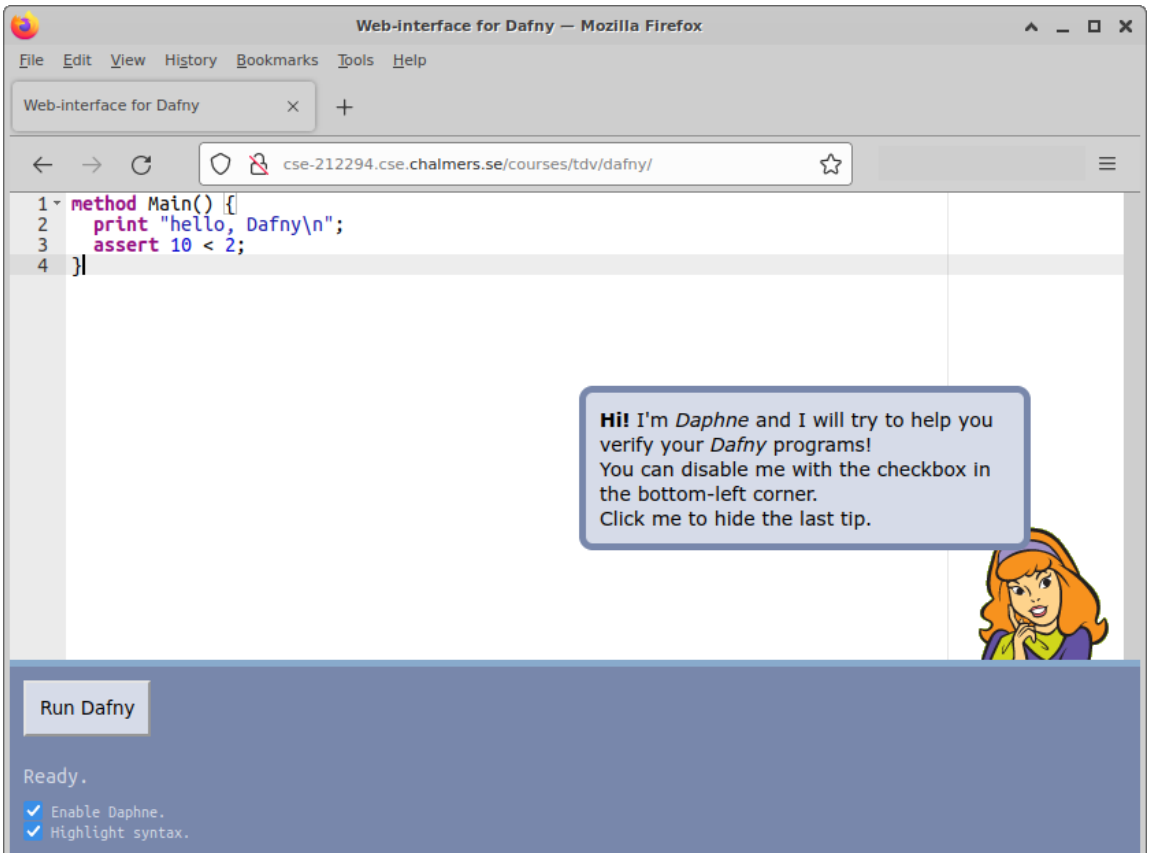
   - A radiation therapy machine that delivers controlled doses of radiation to tumour sites, controlled by an embedded software system.
   - A treatment database that includes details of the treatment given to each patient.

B. Introduction to Dafny

Chalmers university web interface

Dafny is a "verification-aware" programming language. The compiler for the language incorporates a program verifier – as you type in a program, the verifier constantly checks to see whether what you have written can be proved correct, and flags any errors.

A simple web-based interface (only verifies a single file) for working with Dafny is provided at http://cse-212294.cse.chalmers.se/courses/tdv/dafny/.



If you press the "Run Dafny" button, your Dafny code will be checked and compiled. Try compiling the code that is initially contained in the code box:

```
1 method Main() {
2   print "hello, Dafny\n";
3   assert 10 < 2;
4 }
```

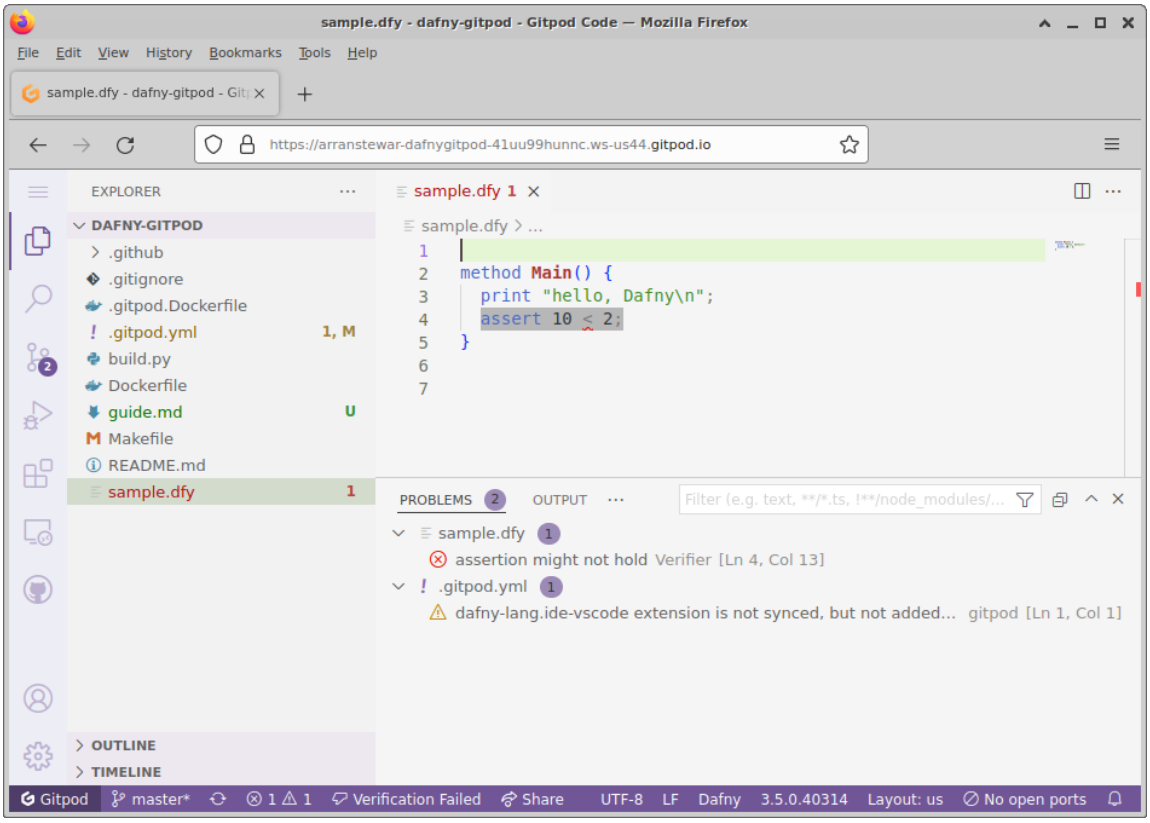
You should see that the Dafny compiler reports an *assertion violation* in line 3; the assertion "$10 < 2$" is not true.

Now comment out the assertion (using double forward-slashes – "`//`" – the same as Java.) If you try re-compiling, you should see the code now compiles and runs.

Gitpod

You can also compile (as well as run) Dafny programs using Gitpod, which provides you with an online development environment based on Microsoft VS Code.

Visit https://gitpod.io/#https://github.com/arranstewart-dev/dafny-gitpod in your browser. Gitpod will download a Docker image containing the Dafny compiler, and create an online IDE environment (this may take a couple of minutes).



Once it appears, open the "sample.dfy" file in the editor, and an "editor extension" will be installed which tells the editor how to highlight Dafny code and show errors.

(You can ignore any error messages saying the "dafny-lang.ide-cscode extension is not synced" – those are a limitation of the VS Code setup, and can't easily be got rid of.)

Toward the bottom of the screen is also a "Terminal" tab, which gives you command-line access to the virtual machine in which Microsoft VS Code is running. Typing `dafny /help` in the terminal will display the compiler online help (warning: it's very long).

Dafny resources

In case you are interested, the paper introducing Dafny is available here.

Documentation for the language (including detailed tutorials and reference material) is available at:

https://dafny-lang.github.io/dafny/

C. Preconditions

Try compiling the following code:

```
method Main() {
  PrintInt(-1);
}

```

```
method PrintInt(x : int)
{
  print "the int is: \n";
  print x;
}
```

The int -1 should be printed with a message. Now add a precondition to PrintInt, "`requires x >= 0`":

```
method Main() {
  PrintInt(-1);
}

method PrintInt(x : int)
  requires x >= 0
{
  print "the int is: \n";
  print x;
}
```

If you try compiling again, you will see that Dafny won't permit you to call the `PrintInt` method, unless it can verify all the preconditions hold. Change the `int` being passed from -1 to 1, and you should see the code now compiles.

D. Using Dafny features

The following Dafny code is intended to find the position of the largest element of an array. It is only guaranteed to produce a result if the array is *non-empty*, however. We won't compile this (yet), but look at the following code:

```
method FindMax(arr: array<int>) returns (r: int)
{
  var max_val : int := arr[0];
  var max_idx : int := 0;

  var i : int       := 1;

  while (i < arr.Length)
  {
    if arr[i] > max_val
      {
        max_idx := i;
        max_val := arr[i];
      }
    i := i + 1;
  }
  return max_idx;
}
```

At what points in the code might we insert the following, and what Dafny keywords would be used?

- preconditions
- postconditions
- loop invariants
- assertions

See if you can state what the preconditions and postconditions are (in English is fine).

Challenge exercise: If you have some experience with formal logic and would like a challenge, try verifying the above code using the online Dafny verifier. You will probably want to try the rest of the exercises in this worksheet first, and work through the online Dafny tutorial. This will explain how to use the `forall` keyword, which we have not covered, but which is needed for the challenge.

E. Writing Dafny methods

The following code does not yet compile:

```
method Max(a: int, b:int) returns (c: int)
  ensures c >= a
{
```

```
4   return 0;
5 }
```

Write the body of the method Max. It should take two integer parameters and return their maximum. Add appropriate annotations (one is already provided) and make sure your code verifies. Refer to the documentation and tutorials, if necessary.