ETH zürich

P. Müller and D. Basin

Formal Methods and Functional Programming

Exercise Sheet 13: Axiomatic Semantics (Hoare Logic)

Submission deadline: May 30/31, 2023

Viper

Viper (`viper.ethz.ch`) is a verification project, providing a language and tools for verifying functional properties of programs in that language. In particular, Viper supports verification of loops via loop invariants, as you have learned in the course. In this exercise session, you can try out Viper as a tool for helping to practice the important skill of finding loop invariants. There are two simple ways of running Viper: the recommended way is to download the VSCode plugin for Viper, as explained here: `http://www.pm.inf.ethz.ch/research/viper/downloads.html` (make sure to install the "Viper" plugin, and that you have the right dependencies, as explained in the instructions). Alternatively, you can run the Viper tools through the (slower) web interface, available at `http://viper.ethz.ch/examples/blank-example.html`.

The syntax of Viper programs is slightly different to that of IMP, but very similar. Syntax for assignments is the same (although local variables must be declared with an explicit type, using the syntax `var x: Int`. There is no `skip` command, but statement blocks (e.g., branches of conditionals) may be left empty. Viper uses C/Java-like syntax for conditionals and loops. For example, the IMP program

```
if x > 0 then x := x - 1; while x > 0 do x := x - 1 end else skip end
```

could be translated into Viper as follows:

```
if (x > 0) {
  x := x - 1
  while (x > 0) {
    x := x - 1
  }
} else { /* else branch can be omitted */ }
```

Loop invariants can be written between the while-condition and the open brace {, e.g.:

```
while (x > 0)
  invariant x >= 0
{
  x := x - 1
}
```

Assignment 1 (Total Correctness of Exponentiation)

Let s_1 be the following statement:

```
y := 1;
z := 0;
while z < x do
  y := y * 2;
  z := z + 1
end
```

The Viper skeleton can be found on the course website: `program_3_exponentiation.vpr`.

Our goal is to prove that $\vdash \{\ \mathrm{x} = X \wedge X \geq 0\ \}\ s_1\ \{\ \Downarrow \mathrm{y} = 2^X\ \}$.

Task 1.1. Find a suitable loop invariant and loop variant. You may use Viper to do so.

Hint: The invariant should mention all variables used in the loop.

Task 1.2. Give a proof outline for $\vdash \{\ \mathrm{x} = X \wedge X \geq 0\ \}\ s\ \{\ \Downarrow \mathrm{y} = 2^X\ \}$.

Assignment 2

Consider the following IMP program s_2:

```
y := 0;
z := 0;
while y * y < n do
  y := y + 1;
  if y * y <= n then
    z := z + 1
  else
    skip
  end
end
```

The Viper skeleton can be found on the course website: `program_4.vpr`.

Task 2.1. Assume $\texttt{n} \geq 0$ as pre-condition. What does this function compute (*hint:* the result is stored in z)? Write a post-condition to ensure that z has the right value after termination. Also add the pre-condition to the program.

Task 2.2. Find a suitable loop invariant which enables the verifier to prove the post-condition given the pre-condition.

Task 2.3. Prove that

$$\{\, \texttt{n} = N \wedge \texttt{n} \geq 0 \,\}\ s_1\ \{\, \texttt{z}^2 \leq N \wedge N < (\texttt{z}+1)^2 \,\}$$

Task 2.4. Give a suitable variant (i.e., which can be used with the rule $\textsc{WhTot}_{Ax}$) to prove that the program terminates, and explain why it is suitable.