Technische Universität München
Institut für Informatik
Prof. Tobias Nipkow, Ph.D.
Lars Hupel

WS 2017/18
19.12.2017

Semantics of Programming Languages

Exercise Sheet 10

Exercise 10.1 Hoare Logic

In this exercise, you shall prove correct some Hoare triples.

Step 1 Write a program that stores the maximum of the values of variables a and b in variable c.

definition MAX :: com **where**

Step 2 Prove these lemmas about max:

lemma [$simp$]: "$(a{::}int)<b \Longrightarrow max\ a\ b = b$"

lemma [$simp$]: "$\neg(a{::}int)<b \Longrightarrow max\ a\ b = a$"

Show that MAX satisfies the following Hoare triple:

lemma "$\vdash \{\lambda s.\ True\}\ MAX\ \{\lambda s.\ s\ ''c'' = max\ (s\ ''a'')\ (s\ ''b'')\}$"

Step 3 Now define a program MUL that returns the product of x and y in variable z. You may assume that y is not negative.

definition MUL :: com **where**

Step 4 Prove that MUL does the right thing.

lemma "$\vdash \{\lambda s.\ 0 \leq s\ ''y''\}\ MUL\ \{\lambda s.\ s\ ''z'' = s\ ''x'' * s\ ''y''\}$"

Hints:

- You may want to use the lemma *algebra_simps*, containing some useful lemmas like distributivity.

- Note that we use a backward assignment rule. This implies that the best way to do proofs is also backwards, i.e., on a semicolon $c_1;;\ c_2$, you first continue the proof for c_2, thus instantiating the intermediate assertion, and then do the proof for c_1. However, the first premise of the *Seq*-rule is about c_1. In an Isar proof, this is no problem. In an **apply**-style proof, the ordering matters. Hence, you may want to use the [*rotated*] attribute:

lemmas *Seq_bwd* = *Seq*[*rotated*]

lemmas *hoare_rule*[*intro?*] = *Seq_bwd Assign Assign' If*

Step 5 Note that our specifications still have a problem, as programs are allowed to overwrite arbitrary variables.

For example, regard the following (wrong) implementation of *MAX*:

definition "$MAX\_wrong = (''a''::=N\ 0;;''b''::=N\ 0;;''c''::= N\ 0)$"

Prove that *MAX_wrong* also satisfies the specification for *MAX*:

lemma "$\vdash \{\lambda s.\ True\}\ MAX\_wrong\ \{\lambda s.\ s\ ''c'' = max\ (s\ ''a'')\ (s\ ''b'')\}$"

What we really want to specify is, that *MAX* computes the maximum of the values of a and b in the initial state. Moreover, we may require that a and b are not changed.

For this, we can use logical variables in the specification. Prove the following more accurate specification for *MAX*:

lemma "$\vdash \{\lambda s.\ a{=}s\ ''a'' \wedge b{=}s\ ''b''\}$
MAX
$\{\lambda s.\ s\ ''c'' = max\ a\ b \wedge a = s\ ''a'' \wedge b = s\ ''b''\}$"

The specification for *MUL* has the same problem. Fix it!

Exercise 10.2 Forward Assignment Rule

Think up and prove a forward assignment rule, i.e., a rule of the form $\vdash \{P\}\ x ::= a\ \{Q\}$, where Q is some suitable postcondition. Hint: To prove this rule, use the completeness property, and prove the rule semantically.

lemmas *fwd_Assign'* = *weaken_post*[*OF fwd_Assign*]

Redo the proofs for *MAX* and *MUL* from the previous exercise, this time using your forward assignment rule.

lemma "$\vdash \{\lambda s.\ True\}\ MAX\ \{\lambda s.\ s\ ''c'' = max\ (s\ ''a'')\ (s\ ''b'')\}$"
lemma "$\vdash \{\lambda s.\ 0 \le s\ ''y''\}\ MUL\ \{\lambda s.\ s\ ''z'' = s\ ''x'' * s\ ''y''\}$"

Homework 10.1 Fixed Points

Submission until Tuesday, January 9, 2018, 10:00am.

Prove the following fixed point theorem:

definition *gfp* :: *"('a set ⇒ 'a set) ⇒ 'a set"* **where**
"gfp f = $\bigcup\{P.\ P \subseteq f\ P\}$*"*

lemma
  assumes "$\bigwedge x\ y.\ x \subseteq y \Longrightarrow f\ x \subseteq f\ y$"
  shows "$f\ (gfp\ f) = gfp\ f$" "$\bigwedge a.\ f\ a = a \Longrightarrow a \subseteq gfp\ f$"

The theorem proves two properties. The general way to do that is as follows:

lemma
  assumes "$P \wedge Q$"
  shows P Q
proof $-$
  show P
    using *assms* **by** *simp*

  show Q
    using *assms* **by** *simp*
qed

Homework 10.2 Be Original!

Submission until Tuesday, January 9, 2018, 10:00am. (20 regular points, plus bonus points for nice submissions)

Think up a nice formalization yourself, for example

- Prove some interesting result about graph/automata/formal language theory
- Formalize some results from mathematics
- Find interesting modifications of IMP material and prove interesting properties about them
- ...

You should set yourself a time limit before starting your project. Also incomplete/unfinished formalizations are welcome and will be graded!

Please comment your formalization well, such that we can see what it does/is intended to do.

You are welcome to discuss your plans with the tutor before starting your project.