Semantics of Programming Languages
Exercise Sheet 10

**Exercise 10.1** Denotational Semantics

Define a denotational semantics for REPEAT-loops, and show its equivalence to the bigstep semantics.

Use the exercise template that we provide on the course web page.

**Exercise 10.2** Hoare Logic

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

First, write a program that stores the maximum of the values of variables \( a \) and \( b \) in variable \( c \).

**definition** MAX :: com where

For the next task, you will need the following lemmas. Hint: Sledgehammering may be a good idea.

**lemma** [simp]: \((a::\text{int})<b \implies \text{max } a b = b\)

**lemma** [simp]: \(\neg(a::\text{int})<b \implies \text{max } a b = a\)

by auto

Show that MAX satisfies the following Hoare-triple:

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

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

Prove that MUL does the right thing.

**lemma** \(\vdash \{ \lambda s. 0 \leq s \ "y" \} \text{MUL} \{ \lambda s. s \ "z" = s \ "x" \ast s \ "y" \} \)
Hints You may want to use the lemma algebra_simps, that contains 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 $S_1; S_2$, you first continue the proof for $S_2$, thus instantiating the intermediate assertion, and then do the proof for $S_1$. However, the first premise of the Seq-rule is about $S_1$. Hence, you may want to use the rotated-attribute, that rotates the premises of a lemma:

```
lemmas Seq_bwd = Seq[rotated]
```

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

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 $MAX_{\text{wrong}}$ also satisfies the specification for $MAX$:

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 “{\lambda s. a=s "a"' \land b=s "b"} 
MAX 
{\lambda s "c"' = max \ a \ b \land a = s "a"' \land b = s "b"}”
```

The specification for $MUL$ has the same problem. Fix it!
Homework 10  Be Original!

Submission until Tuesday, 13 January 2012, 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
- Prove some results from Program Optimization
- ...

In case you don’t have a good idea, here are some further inspirations:

- Add some new language features to IMP, and redo some proofs (e.g., compiler, typing, Hoare-Logic).
- Do Floyd-style verification on control flow graphs.
- Compile commands to a register machine, and show correctness.
- Prove correct some non-trivial program, e.g., square roots using the bisection method. Hint: A modular approach of writing and proving programs may help, e.g., you may try to reuse a program for multiplication and its correctness proof, rather than inlining the program and the proof.

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 one of the tutors before starting your project.