Concrete Semantics with Isabelle/HOL  
Exercise Sheet 12

Exercise 12.1 Hoare Logic: MUL

Define a program $MUL$ that returns the product of $x$ and $y$ in variable $z$. You may assume that $y$ is not negative.

**definition** $MUL :: com$

Prove that $MUL$ does the right thing.

**lemmas** $Seq\_bwd = Seq[\text{rotated}]$

**lemmas** $hoare\_rule[\text{intro}] = Seq\_bwd Assign Assign' If$

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

The specification for $MUL$ has the same problem as $MAX$ in Exercise 11.2. Fix it.

Exercise 12.2 Forward Assignment Rule

Think up and prove a forward assignment rule, i.e., a rule of the form

$$\vdash \{P\} x ::= a \{\ldots\}$$

where $\ldots$ is some suitable postcondition. Hint: To prove this rule, use the completeness property, and prove the rule semantically.

**lemma** $fwd\_Assign$

**lemmas** $fwd\_Assign' = weaken\_post[OF fwd\_Assign]$

Redo the proof for $MAX$ from the previous exercise sheet, this time using your forward assignment rule.

**definition** $MAX :: com$ where

"$MAX \equiv$

$\text{IF} (\text{Less} (V ''a'') (V ''b'')) \text{THEN}$

$''c'' ::= V ''b''$

$\text{ELSE}$

$''c'' ::= V ''a''$

**lemma** $\vdash \{\lambda s. \text{True}\} MAX \{\lambda s. s ''c'' = max (s ''a'') (s ''b'')\}$
Exercise 12.3  Using the VCG (optional but fun!)

For each of the three programs given here, you must prove partial correctness. You should first write an annotated program, and then use the verification condition generator from the VCG theory.

Preliminaries.  Some abbreviations, freeing us from having to write double quotes for concrete variable names:

abbreviation "aa ≡ "'a"" abbreviation "bb ≡ "'b"" abbreviation "cc ≡ "'c""
abbreviation "dd ≡ "'d"" abbreviation "ee ≡ "'e"" abbreviation "ff ≡ "'f""
abbreviation "pp ≡ "'p"" abbreviation "qq ≡ "'q"" abbreviation "rr ≡ "'r"

Some useful simplification rules:

declare algebra_simps [simp]
power2_eq_square [simp]

A convenient loop construct:

abbreviation For :: "vname ⇒ aexp ⇒ aexp ⇒ com ⇒ com"
(FOR / FROM / TO / DO ) [0, 0, 61] 61) where
v ::: a1 ;; WHILE (Less (V v) a2) DO (c ;; v ::: Plus (V v) (N 1))"

abbreviation aFor :: "assn ⇒ vname ⇒ aexp ⇒ aexp ⇒ acom ⇒ acom"
((LET / FOR / FROM / TO / DO ) [0, 0, 0, 0, 61] 61) where
{b} FOR v FROM a1 TO a2 DO c ≡
v ::: a1 ;; {b} WHILE (Less (V v) a2) DO (c ;; v ::: Plus (V v) (N 1))"

Program 1: Multiplication.  Consider the following program MULT for performing multiplication and the following assertions P_MULT and Q_MULT:

definition MULT :: com where "MULT ≡
cc ::= N 0;;
FOR dd FROM (N 0) TO (V aa) DO
cc ::= Plus (V cc) (V bb)"

definition P_MULT :: "int ⇒ int ⇒ assn" where
"P_MULT i j ≡ λs. s aa = i ∧ s bb = j ∧ 0 ≤ i"

definition Q_MULT :: "int ⇒ int ⇒ assn" where
"Q_MULT i j ≡ λs. s cc = i * j ∧ s aa = i ∧ s bb = j"

Define an annotated program AMULT i j, so that when the annotations are stripped away, it yields MULT. (The parameters i and j will appear only in the loop annotations.)

Hint: The program AMULT i j will be essentially MULT with an invariant annotation iMULT i j at the FOR loop, which you have to define:
definition \texttt{iMULT} :: "int \Rightarrow int \Rightarrow \texttt{assn}" where

\[ \texttt{iMULT} i j \equiv 
\{ \texttt{iMULT} i j \} \texttt{FOR dd FROM} (N 0) \texttt{TO} (V aa) \texttt{DO} 
\quad \texttt{cc ::= Plus} (V cc) (V bb) \}
\]

\texttt{lemmas MULT defs = MULT_def P_MULT_def Q_MULT_def iMULT_def AMULT_def}

\texttt{lemma strip_AMULT: "strip (AMULT i j) = MULT"}

Once you have the correct loop annotations, then the partial correctness proof can be done in two steps, with the help of lemma \texttt{vc_sound'}.

\texttt{lemma MULT_correct: "\vdash \{P_MULT i j\} MULT \{Q_MULT i j\}"}

\textbf{Program 2: Division.} Define an annotated version of this division program, which yields the quotient and remainder of \texttt{aa/ bb} in variables "\texttt{q}" and "\texttt{r}", respectively.

\texttt{definition DIV :: \texttt{com \ where} "DIV \equiv 
qq ::= N 0 ;; 
rr ::= N 0 ;; 
\texttt{FOR cc FROM} (N 0) \texttt{TO} (V aa) \texttt{DO} ( 
\quad \texttt{rr ::= Plus} (V rr) (N 1) ;; 
\quad \texttt{IF Less} (V rr) (V bb) \texttt{THEN} 
\quad \quad \texttt{SKIP} 
\quad \texttt{ELSE} ( 
\quad \quad \texttt{rr ::= N 0 ;;} 
\quad \quad \texttt{qq ::= Plus} (V qq) (N 1)) 
\quad ) 
"}

\texttt{definition P_DIV :: "int \Rightarrow int \Rightarrow \texttt{assn}" where
"P_DIV i j \equiv \lambda s. s aa = i \land s bb = j \land 0 \leq i \land 0 < j"}

\texttt{definition Q_DIV :: "int \Rightarrow int \Rightarrow \texttt{assn}" where
"Q_DIV i j \equiv \lambda s. i = s qq \ast j + s rr \land 0 \leq s rr \land s rr < j \land s aa = i \land s bb = j"}

\texttt{definition iDIV :: "int \Rightarrow int \Rightarrow \texttt{assn}" where
"iDIV i j \equiv 
qq ::= N 0 ;; 
rr ::= N 0 ;; 
\{ iDIV i j \} \texttt{FOR cc FROM} (N 0) \texttt{TO} (V aa) \texttt{DO} ( 
\quad \texttt{rr ::= Plus} (V rr) (N 1) ;; 
\quad \texttt{IF Less} (V rr) (V bb) \texttt{THEN} 
\quad \quad \texttt{SKIP} 
\quad \texttt{ELSE} ( 
\quad \quad ) 
\)
\[ rr ::= N \ 0 \ ;; \]
\[ qq ::= \text{Plus} \ (V \ qq) \ (N \ 1) \]
\]

lemma \textit{strip}_{DIV}: \textit{strip} \ (DIV \ i \ j) = DIV

lemma \textit{DIV}\_correct: \textit{⊢} \{P_{DIV} \ i \ j\} \ DIV \ \{Q_{DIV} \ i \ j\}\]

Program 3: Square Roots. Define an annotated version of this square root program, which yields the square root of input \( aa \) (rounded down to the next integer) in output \( bb \).

definition \textit{SQR} :: \textit{com} where \( \textit{SQR} \equiv \)
\[ bb ::= N \ 0 \ ;; \]
\[ cc ::= N \ 1 \ ;; \]
\[ \text{WHILE} \ (\text{Not} \ (\text{Less} \ (V \ aa) \ (V \ cc))) \ \text{DO} \ (\)
\[ bb ::= \text{Plus} \ (V \ bb) \ (N \ 1) \ ;; \]
\[ cc ::= \text{Plus} \ (V \ cc) \ (\text{Plus} \ (V \ bb) \ (\text{Plus} \ (V \ bb) \ (N \ 1))) \]
\]

definition \textit{P}_{SQR} :: \textit{int \ ⇒ \ assn} where 
\( \textit{P}_{SQR} \ i \equiv \ λs. \ s \ aa = i \ ∧ \ 0 \ ≤ \ i \)

definition \textit{Q}_{SQR} :: \textit{int \ ⇒ \ assn} where 
\( \textit{Q}_{SQR} \ i \equiv \ λs. \ s \ aa = i \ ∧ \ (s \ bb)^2 \ ≤ \ i \ ∧ \ i \ < \ (s \ bb + 1)^2 \)