Exercise 10.1  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.

Redo the proofs for \( \text{MAX} \) and \( \text{MUL} \) from the previous exercise sheet, this time using your forward assignment rule.

definition \( \text{MAX} :: \text{com} \) where

\[
\text{"MAX} \equiv \text{IF (Less (V "a") (V "b")) THEN ("c"::=V "b") ELSE ("c"::=V "a")"
\]

definition \( \text{MUL} :: \text{com} \) where

\[
\text{"MUL} \equiv \text{"z"::=N 0;; ("c")::=N 0;; (WHILE (Less (V "c") (V "y")) DO ("z"::=Plus (V "z") (V "x")); ("c"::=Plus (V "c") (N 1)))"
\]

lemma \( \vdash \{ \lambda s. 0 \leq s "y" \} \text{MUL} \{ \lambda s. s "z" = s "x" \ast s "y" \} \)

Exercise 10.2  Using the VCG

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 \( VCG.thy \).
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 ≡ "d"" abbreviation "ff ≡ "f""
abbreviation "pp ≡ "p"" abbreviation "qq ≡ "q"" abbreviation "rr ≡ "r"

Some useful simplification rules:

declare algebra_simps[rint] declare power2.eq_square[rint]

Rotated rule for sequential composition:

A convenient loop construct:

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

abbreviation Afor :: "assn ⇒ vname ⇒ aexp ⇒ aexp ⇒ acom ⇒ acom"
("(\{\} / FOR / FROM / TO / DO .)" [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))"

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 iMULT :: "int ⇒ int ⇒ assn" where
"iMULT i j ≡
 (cc ::= N 0) ;;
\{iMULT i j\} FOR dd FROM (N 0) TO (V aa) DO
lemmas MULT_defs = MULT_def P_MULT_def Q_MULT_def iMULT_def AMULT_def

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 vc_sound'.

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

**Division.** Define an annotated version of this division program, which yields the quotient and remainder of \( aa/\)bb in variables "q" and "r", respectively.

definition DIV :: com where "DIV ≡

\[
\begin{align*}
qq & := N 0 :: \\
rr & := N 0 :: \\
& \text{FOR } cc \text{ FROM } (N 0) \text{ TO } (V aa) \text{ DO } ( \\
& \quad rr ::= Plus (V rr) (N 1) :: \\
& \quad \text{IF } Less (V rr) (V bb) \text{ THEN} \\
& \quad \quad \text{Com.SKIP} \\
& \quad \text{ELSE} ( \\
& \quad \quad rr ::= N 0 :: \\
& \quad \quad qq ::= Plus (V qq) (N 1))
\end{align*}
\]

definition P_DIV :: \"int ⇒ int ⇒ assn\" where

"P_DIV i j ≡ \(λs. \ s \ aa = i \land s \ bb = j \land 0 \leq i \land 0 < j\)"

definition Q_DIV :: \"int ⇒ int ⇒ assn\" where

"Q_DIV i j ≡ \(λs. \ s \ i = s \ qq \cdot s \ j + s \ rr \land 0 \leq s \ rr \land s \ rr < j \land s \ aa = i \land s \ bb = j\)"

definition iDIV :: \"int ⇒ int ⇒ assn\" where

\[
\begin{align*}
\{iDIV i j\} \text{ FOR } cc \text{ FROM } (N 0) \text{ TO } (V aa) \text{ DO } ( \\
& \quad rr ::= Plus (V rr) (N 1) :: \\
& \quad \text{IF } Less (V rr) (V bb) \text{ THEN} \\
& \quad \quad \text{SKIP} \\
& \quad \text{ELSE} ( \\
& \quad \quad rr ::= N 0 :: \\
& \quad \quad qq ::= Plus (V qq) (N 1))
\end{align*}
\]

lemma strip_ADIV: "strip (ADIV i j) = DIV"

lemma DIV_correct: "\{ P_DIV i j \} DIV \{ Q_DIV i j \}"

\[3\]
**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** $SQR :: \text{com where}$ “$SQR \equiv$

$bb ::= N \ 0 ;;$

$cc ::= N \ 1 ;;$

$WHILE (\text{Not} \ (\text{Less} \ (V \ aa) \ (V \ cc))) \ DO (\$

$bb ::= \text{Plus} \ (V \ bb) \ (N \ 1)); ;$

$cc ::= \text{Plus} \ (V \ cc) \ (\text{Plus} \ (V \ bb) \ (\text{Plus} \ (V \ bb) \ (N \ 1)))$

) “

**definition** $P_{SQR} :: \text{int} \Rightarrow \text{assn}$ where

“$P_{SQR} i \equiv \lambda s. \ s \ aa = i \land 0 \leq i$”

**definition** $Q_{SQR} :: \text{int} \Rightarrow \text{assn}$ where

“$Q_{SQR} i \equiv \lambda s. \ s \ aa = i \land (s \ bb)^2 \leq i \land i < (s \ bb + 1)^2$”

**Homework 10** Be Original!

*Submission until Tuesday, 14 January 2014, 10:00am.*

Use the second week to polish your formalization a bit.