Exercise 10.1 Using the VCG

Use the VCG to prove correct a multiplication and a square root program:

*Note: With the template, use \texttt{unbundle COM/unbundle ACOM} before writing down a \texttt{com/acom} definition, otherwise parsing will be slow.

\textbf{definition} \texttt{MUL :: com where}
\begin{verbatim}
"MUL =
  "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))"
\end{verbatim}

\textbf{theorem} \texttt{MUL_partially_correct}:
\begin{verbatim}
\vdash \{ \lambda s. 0 \leq s "y"' \land s=sorig \} MUL
  \{ \lambda s. s "x"'* s "y"' \land (\forall v. v\notin\{"z","c"\} \rightarrow s v = sorig v) \}''
\end{verbatim}

\textbf{definition} \texttt{SQRT :: com where}
\begin{verbatim}
"SQRT =
  "r"::= N 0;;
  "s"::= N 1;;
  WHILE (Not (Less (V "x") (V "s"))) DO (
  "r"::= Plus (V "r") (N 1));;
  "s"::= Plus (V "s") (V "r"));;
  "s"::= Plus (V "s") (V "r"));;
  "s"::= Plus (V "s") (N 1)
)"
\end{verbatim}

\textbf{theorem} \texttt{SQRT_partially_correct}:
\begin{verbatim}
\vdash \{ \lambda s. s=sorig \land s "x"' \geq 0 \} SQRT
  \{ \lambda s. (s "r")' \leq s "x"' \land s "x"' < (s "r")'+1 \land (\forall v. v\notin\{"s","r"\} \rightarrow s v = sorig v) \}''
\end{verbatim}
Exercise 10.2  Total Correctness

Prove total correctness of the multiplication and square root program.

Rotated rule for sequential composition:
lemma Seq.bwd = Hoare_Total.Seq[rotated]

Prove the following syntax-directed conditional rule (for total correctness):
lemma IfT:
  assumes “Γ t {P1} c_1 {Q}” and “Γ t {P2} c_2 {Q}”
  shows “Γ t {λ s. (bval b s → P1 s) ∧ (¬ bval b s → P2 s)} IF b THEN c_1 ELSE c_2 {Q}”

lemmas hoareT_rule[!intro?] = Seq.bwd Hoare_Total.Assign Hoare_Total.Assign’ IfT

theorem MUL_totally_correct:
  “Γ t {λ s. 0 ≤ s "y" ∧ s=sorig} MUL
   {λ s. "z" = s "x" + s "y" ∧ (∀ v. v¢{"z","c"} → s v = sorig v)}”

theorem SQRT_totally_correct:
  “Γ t {λ s=sorig ∧ s "x" ≥ 0} SQRT
   {λ s. (s "r") ≠ 2 ≤ s "x" ∧ s "x" < (s "r"+1) ≠ 2 ∧ (∀ v. v¢{"s","r"} → s v = sorig v)}”

Homework 10.1  Division

Submission until Sunday, Jan 24, 23:59. Write a simple IMP program that divides (positive) inputs p by q and returns the results n and r such that q * n + r = p and 0 ≤ r < q, i.e. n is the result and r the reminder.

One of the difficulties here is that there is no minus operation in IMP - so you’ll likely have to use two loops.

definition DIV :: com

Prove correctness using the VCG.

The actual proof should be easy once you got the invariants right.

theorem DIV_correct:
  “Γ {λ s=s_0 ∧ s "p" > 0 ∧ s "q" > 0} DIV
   {λ s_0 "q" * s "n" + s "r" = s_0 "p" ∧ 0 ≤ s "r" ∧ s "r" < s_0 "q"}”

Now, prove total correctness.
**theorem** \( \text{DIV}.t_{\text{correct}}: \)

\[ \vdash t \{ \lambda s. (s = s_0 \land s \"p\" > 0 \land s \"q\" > 0) \}

\text{DIV}

\{ \lambda s. (s_0 \"q\" \ast s \"r\" + s \"r\" = s_0 \"p\" \land s \"r\" < s_0 \"q\") \}

---

**Homework 10.2  Collecting Semantics**

*Submission until Sunday, Jan 24, 23:59.*

**Note:** This is a typical exam exercise.

Show the iterative computation of the collecting semantics of the following program in a table like the one on page 228 of the concrete semantics book (submit as text/comment in your submission file or via email).

\[
\begin{align*}
&x := 2; y := 1 \{A_0\}; \\
&\{A_1\} \\
&\text{WHILE } 0 < x \\
&\text{DO } \{A_2\} (y := y \ast x; x := x - 1 \{A_3\}) \\
&\{A_4\}
\end{align*}
\]

Note that two annotations have been suppressed to make the task less tedious. You do not need to show steps where only the suppressed annotations change.

Because the program contains two variables, the state sets in the table should be represented as sets of pairs \((x, y)\). In order to keep the table compact, you can also just write \(xy\), e.g. \(02\) instead of \((0, 2)\) — the values of the variables do not exceed single digits.