Technische Universität München
Institut für Informatik
Prof. Tobias Nipkow, Ph.D.
Fabian Huch

WS 2021/22
11.01.2022

Semantics of Programming Lectures

Exercise Sheet 10

Exercise 10.1 Using the VCG

Use the VCG to prove correct a multiplication and a square root program:
*Note: With the template, use *unbundle COM/unbundle ACOM* before writing down a *com/acom* definition, otherwise parsing will be slow.

```
definition MUL :: com where
"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))"

theorem MUL_partially_correct:
"⊢ {λs. 0 ≤ s ''y'' ∧ s=sorig}
     MUL
   {λs. s ''z'' = s ''x'' * s ''y'' ∧ (∀ v. v∉{''z'',''c''} ⟶ s v = sorig v)}"

definition SQRT :: com where
"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)
  )"

theorem SQRT_partially_correct:
"⊢ {λs. s=sorig ∧ s ''x'' ≥ 0}
     SQRT
   {λs. (s ''r'')^2 ≤ s ''x'' ∧ s ''x'' < (s ''r''+1)^2 ∧ (∀ v. v∉{''s'',''r''} ⟶ s v = sorig v)}"
```

Exercise 10.2 Total Correctness

Prove total correctness of the multiplication and square root program.

Rotated rule for sequential composition:

lemmas *Seq_bwd = Hoare_Total.Seq[rotated]*

Prove the following syntax-directed conditional rule (for total correctness):

lemma *IfT*:
assumes "$\vdash_t$ {*P1*} c_1 {*Q*}"
and "$\vdash_t$ {*P2*} c_2 {*Q*}"
shows "$\vdash_t$ {$\lambda s.$ (*bval b s* $\longrightarrow$ *P1 s*) $\wedge$ ($\neg$ *bval b s* $\longrightarrow$ *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*:
"$\vdash_t$ {$\lambda s.$ $0 \leq s\ ''y'' \wedge s{=}sorig$}
MUL
{$\lambda s.$ $s\ ''z'' = s\ ''x'' * s\ ''y'' \wedge (\forall v.\ v \notin \{''z'', ''c''\} \longrightarrow s\ v = sorig\ v)$}"

theorem *SQRT_totally_correct*:
"$\vdash_t$ {$\lambda s.$ $s{=}sorig \wedge s\ ''x'' \geq 0$}
SQRT
{$\lambda s.$ $(s\ ''r'')\hat{}2 \leq s\ ''x'' \wedge s\ ''x'' < (s\ ''r''+1)\hat{}2 \wedge (\forall v.\ v \notin \{''s'', ''r''\} \longrightarrow s\ v = sorig\ v)$}"

Homework 10.1 VCG for Total Correctness

Submission until Sunday, Jan 16, 23:59pm.

We want to build a verification condition generator for annotated programs with explicit variants.

Annotated commands are already defined for you in the definitions:

datatype *acom = Askip | Aassign (char list) aexp | Aseq acom acom | Aif bexp acom acom | Awhile ((char list ⇒ int) ⇒ nat) ((char list ⇒ int) ⇒ bool) bexp acom*

This version of IMP also contains more arithmetic expressions, total and partial Hoare logics, as well as bundles for *COM*/*ACOM*. Define the VCG for total Hoare logic, and show soundness.

unbundle *ACOM*

fun *pre* :: "*acom* ⇒ *assn* ⇒ *assn*"

fun *vc* :: "*acom* ⇒ *assn* ⇒ *bool*"

lemma *vc_sound*: *"vc C Q* $\Longrightarrow \vdash_t$ *{pre C Q} strip C {Q}"*

corollary *vc_sound'*:
"$[\![$ *vc C Q;* $\forall$ *s. P s* $\longrightarrow$ *pre C Q s* $]\!]$ $\Longrightarrow \vdash_t$ *{P} strip C {Q}"*
by (*metis strengthen_pre vc_sound*)

Homework 10.2 Extended Euclidean Algorithm

Submission until Sunday, Jan 16, 23:59pm.

The following IMP program (from the English Wikipedia on extended euclidean algorithm) computes the greatest common divisor of two numbers a and b, as well as some coefficients s and t such that $gcd\ a\ b = a * s + b * t$.

Prove it totally correct! Use the VCG from the previous exercise.

In case you did not manage to show *vc_sound*, show partial correctness instead. There is a VCG for partial Hoare logic in the definitions (all names prefixed with "partial").

Hint: Read the Wikipedia article first to understand the algorithm.

```
''old_r'' ::= V ''a'';;
''r'' ::= V ''b'';;

''old_s'' ::= N 1;;
''s'' ::= N 0;;

''old_t'' ::= N 0;;
''t'' ::= N 1;;

WHILE neq (V ''r'') (N 0) DO (
  ''quotient'' ::= Div (V ''old_r'') (V ''r'');;

  ''tmp'' ::= V ''old_r'';;
  ''old_r'' ::= V ''r'';;
  ''r'' ::= Minus (V ''tmp'') (Mul (V ''quotient'') (V ''r''));;

  ''tmp'' ::= V ''old_s'';;
  ''old_s'' ::= V ''s'';;
  ''s'' ::= Minus (V ''tmp'') (Mul (V ''quotient'') (V ''s''));;

  ''tmp'' ::= V ''old_t'';;
  ''old_t'' ::= V ''t'';;
  ''t'' ::= Minus (V ''tmp'') (Mul (V ''quotient'') (V ''t''))
)
```

theorem *EGGT_correct*: "$\vdash_t$
$\{\lambda s.\ s = s_0 \wedge s_0\ ''a'' > 0 \wedge s_0\ ''b'' > 0\}$
EGGT
$\{\lambda s.\ s\ ''old\_r'' = gcd\ (s_0\ ''a'')\ (s_0\ ''b'') \wedge gcd\ (s_0\ ''a'')\ (s_0\ ''b'') = s_0\ ''a'' * s\ ''old\_s'' + s_0\ ''b'' * s\ ''old\_t''\}$"

corollary *EGGT_partial*: "$\vdash \{\lambda s.\ s = s_0 \wedge s_0\ ''a'' > 0 \wedge s_0\ ''b'' > 0\}$
EGGT
$\{\lambda s.\ s\ ''old\_r'' = gcd\ (s_0\ ''a'')\ (s_0\ ''b'') \wedge gcd\ (s_0\ ''a'')\ (s_0\ ''b'') = s_0\ ''a'' * s\ ''old\_s'' + s_0\ ''b'' * s\ ''old\_t''\}$"
by (*rule total_imp_partial*[*OF EGGT_correct*])