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

WS 23/24
16.01.2024

Semantics of Programming Languages

Exercise Sheet 11

Exercise 11.1 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 11.2 Total Correctness

Prove total correctness of the multiplication and square root program.

Rotated rule for sequential composition:

lemmas *Seq_bwd = Hoare_Total.Seq[rotated]*

lemmas *hoareT_rule[intro?] = Seq_bwd Hoare_Total.Assign Hoare_Total.Assign'*

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

Homework 11.1 Automadness

Submission until Monday, Jan 22, 23:59pm.

In this homework, we consider yet another way to compute the square root of a natural number. For this, we extend *aexp* with some new arithmetic operators and add their expected semantics:

datatype *aexp = N int | V (char list) | Plus aexp aexp | Minus aexp aexp | Mul aexp aexp | Div aexp aexp*

aval (N n) s = n
aval (V x) s = s x
aval (Plus a_1 a_2) s = aval a_1 s + aval a_2 s
aval (Div a_1 a_2) s = aval a_1 s div aval a_2 s
*aval (Mul a_1 a_2) s = aval a_1 s * aval a_2 s*
aval (Minus a_1 a_2) s = aval a_1 s − aval a_2 s

The program is defined as follows:

```
L := 0;
R := x + 1;
WHILE L + 1 < R DO
  M := (L + R) / 2;
  IF x < M^2
  THEN R := M
```

```
    ELSE L := M
```

Your goal is to prove the correctness of the program using the VCG. Define a suitable invariant and complete the proof below. Do not change the given lines that set up the VCG proof!

definition *SQRT_invar* :: *"state ⇒ assn"*
theorem *SQRT_partially_correct*:
"⊢ {λs. s=sorig ∧ 0 ≤ s ''x''}
SQRT
{λs. (s ''L'')^2 ≤ s ''x'' ∧ s ''x'' < (s ''L'' + 1)^2 ∧ (∀ v. v ∉ SQRT_vars ⟶ s v = sorig v)}"

apply (*subst SQRT_annot_strip*[*of SQRT_invar sorig*, *symmetric*])
apply (*rule vc_sound'*)

While you do not have to show total correctness, we want to you prove the key arguments of the termination proof. For this, define a suitable measure function and prove the following lemmas, showing that the measure decreases with each loop-iteration:

definition *SQRT_measure* :: *"state ⇒ nat"*
lemma *SQRT_measure_correct*:
assumes *"s ''L'' + 1 < s ''R''"*
shows *"s ''x'' < ((s ''L'' + s ''R'') div 2)^2 ⟹*
SQRT_measure (s(''M'' := (s ''L'' + s ''R'') div 2, ''R'' := (s ''L'' + s ''R'') div 2))
< SQRT_measure s"
and *"¬(s ''x'' < ((s ''L'' + s ''R'') div 2)^2) ⟹*
SQRT_measure (s(''M'' := (s ''L'' + s ''R'') div 2, ''L'' := (s ''L'' + s ''R'') div 2))
< SQRT_measure s"