Technische Universität München
Institut für Informatik
Prof. Tobias Nipkow, Ph.D.
Kevin Kappelmann, Lukas Stevens

WS 24/25
19.12.2024

Semantics of Programming Languages

Exercise Sheet 10

Exercise 10.1 Hoare Logic

In this exercise, we will prove the correctness of some concrete programs using Hoare logic.

First, write a program that stores the maximum of the values of variables a and b in variable c.

definition *Max* :: *"com"*

Show that *Max* satisfies the following Hoare triple:

lemma *"⊢ {λs. True} Max {λs. s ''c'' = max (s ''a'') (s ''b'')}"*

Unfortunately, our specifications has a problem. For example, consider the following (wrong) implementation of *Max*:

definition *"Max_wrong = (''a''::=N 0;;''b''::=N 0;;''c''::= N 0)"*

Prove that *Max_wrong* also satisfies the specification for *Max*:

lemma *"⊢ {λs. True} Max_wrong {λs. s ''c'' = max (s ''a'') (s ''b'')}"*

What we really want to specify is that *Max* computes the maximum of the values of a and b in the initial state. Moreover, we may require that a and b are not changed.

For this, we can use logical variables in the specification. Prove the following more accurate specification for *Max*:

lemma *"⊢ {λs. a=s ''a'' ∧ b=s ''b''}*
  Max {λs. s ''c'' = max a b ∧ a = s ''a'' ∧ b = s ''b''}"

Now consider the following program *Mul* that returns the product of x and y in variable z, assuming that y is not negative.

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))"
```

Prove that *Mul* does the right thing using the VCG on an annotated program.

Hint: You may want to use the lemmas *algebra_simps*, containing some useful lemmas like distributivity.

unbundle *ACOM*

definition *Mul_annot* :: *"state ⇒ acom"* **where**
lemma *Mul_annot_strip*: *"strip (Mul_annot s_0) = Mul"*

Homework 10.1 Extended Euclidean Algorithm

Submission until Wednesday, Jan 8, 23:59pm.

The following 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 (partially) correct using the VCG for an extension of IMP containing some more arithmetic primitives. You can find the VCG in the definitions, following the same naming conventions as in the lecture.

Hint: Read the Wikipedia article first to understand the algorithm if you have not encountered it before.

```
''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''))
)
```

unbundle *ACOM*

definition *AEGGT_annot* :: *"state ⇒ acom"* **where**
lemma *AEGGT_annot_strip*: *"strip (AEGGT_annot s_0) = EGGT"*
theorem *EGGT_correct*: "⊢
  $\{\lambda s.\ s = s_0 \wedge s_0\ ''a'' > 0 \wedge s_0\ ''b'' > 0\}$
  EGGT
  $\{\lambda s.\ s\ ''old\_r'' = gcd\ (s_0\ ''a'')\ (s_0\ ''b'')$
    $\wedge\ gcd\ (s_0\ ''a'')\ (s_0\ ''b'') = s_0\ ''a'' * s\ ''old\_s'' + s_0\ ''b'' * s\ ''old\_t''\ \}$"

Homework 10.2 Formalization of Formal Language Theory/Be Creative

Submission until Wednesday, Jan 8, 23:59pm.

Continue your project from sheet 9.

Merry Christmas!