Technische Universität München
Institut für Informatik
Prof. Tobias Nipkow, Ph.D.
Dr. Peter Lammich

WS 2015/16
22. 12. 2015

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:

```
definition MUL :: com
lemma "⊢
  {λs. 0 ≤ s ''y'' ∧ s=sorig}
  MUL
  {λs. s ''z'' = s ''x'' ∗ s ''y'' ∧ (∀ v. v∉{''z'',''c''} ⟶ s v = sorig v)}"
```

```
definition "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)
  )"
```

```
lemma "⊢
  {λs. s=sorig ∧ s ''x'' ≥ 0}
  SQRT
  {λs. (s ''r'') ˆ2 ≤ s ''x'' ∧ s ''x'' < (s ''r''+1) ˆ2 ∧ (∀ 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]
```

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

```
lemma IfT:
  assumes "⊢t {P1} c1 {Q}" and "⊢t {P2} c2 {Q}"
  shows "⊢t {λs. (bval b s ⟶ P1 s) ∧ (¬ bval b s ⟶ P2 s)} IF b THEN c1 ELSE c2 {Q}"

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


lemma "⊢t
  {λs. 0 ≤ s ''y'' ∧ s=sorig}
  MUL
  {λs. s ''z'' = s ''x'' ∗ s ''y'' ∧ (∀ v. v∉{''z'',''c''} ⟶ s v = sorig v)}"
lemma "⊢t
  {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)}"
```

Homework 11 Be Original!

Submission until Tuesday, 12 January 2016, 10:00am. (20 regular points, plus bonus points for nice submissions)

Think up a nice formalization yourself, for example

- Prove some interesting result about graph/automata/formal language theory
- Formalize some results from mathematics
- Prove some results from program optimization
- ...

In case you don't have a good idea, here are some further inspirations: Register machines, register allocation, non-trivial IMP-programs, IMP + { procedures, arrays, etc }

You should set yourself a time limit before starting your project. Also incomplete/unfinished formalizations are welcome and will be graded!

Please comment your formalization well, such that we can see what it does/is intended to do.

You are welcome to discuss your plans with the tutor before starting your project.

Merry Christmas!