McMaster University
Department of Computing and Software
Dr. W. Kahl

CAS 706
Fall 2010
Exercise Sheet 3

CAS 706 — Programming Languages

30 November 2010

Exercise 3.1 — Program Correctness Concepts

(a) Define precisely what the phrase “Program **P** is partially correct with respect to the precondition Q and the postcondition R” means.

(b) Define precisely what the phrase “Program **P** is totally correct with respect to the precondition Q and the postcondition R” means.

Exercise 3.2 — Partial Correctness — Simple Hoare Triples

For each of the following Hoare triples, determine whether it holds; if yes, prove it using the rules of axiomatic semantics, and if no, prove a counter-example using the rules of operational semantics (**you may abbreviate each *expression* evaluation into a single step**).

(a) $\{x \geq -5\}\ z := 5 - x\ \{z \leq 11\}$

(b) $\{x \geq -5\}\ z := 5 - x\ \{z \leq 11 \wedge x \geq -7\}$

(c) $\{x \geq -5\}\ z := 5 - x\ ;\ x := x + z\ \{z \leq 11 \wedge x = 2\}$

(d) $\{z = 0\}$ **if** $x = 0$ **then** $w :=$ True **else** $z := 1/x$ **fi** $\{\neg w \rightarrow xz = 1\}$

(e) $\{z = abs(x)\}$ **if** $x \geq 0$ **then** $z := -z$ **fi** $\{xz = -x^2\}$

Exercise 3.3 Partial Correctness Proof

Consider the following program in a language providing a Java-like printing statement:

```
s
:= 1 ;
r
:= 0 ;
while s ≤ n do
    r
    := r + 1 ;
    s
    := s + 2 * r + 1 ;
    println(r + " " + s)
od
```

(a) What is the output of this program for $n = 30$?

(b) Give an equation relating the values of r and s in each **println** statement.

(c) For this program **without the println** statement, **prove partial correctness** with respect to the **precondition** $\{\ n \geq 0\ \}$ and the **postcondition** $\{\ r^2 \leq n\ \wedge\ n < (r+1)^2\ \}$.

Hint: ***Use the equation*** from (b) **as part of the invariant!**

Exercise 3.4 Correctness Proof for Gödel Numbering

Let int variables a, b, p, and i, and the following program fragment **P** in a Pascal-like programming language be given:

```
(a,i) := (p,0);
while  a > i do
   i:=i+1;
   a:=a–i
od;
b:=i–a;
```

Prove (doumenting all intermediate steps, and showing also the implications used) partial correctness of **P** with respect to the precondition $\{p \geq 0\}$ and the postcondition

$$p = \frac{(a+b)(a+b+1)}{2} + a \quad \wedge \quad a \geq 0 \quad \wedge \quad b \geq 0$$

Hint: For producing this proof, you need no creativity at all, but a high degree of diligence.

Background: **P** decodes the natural number stored in p as a pair (a, b) of two natural numbers; this encoding is a simple kind of *Gödel numbering*.

Exerceise 3.5 Correctness Proof for Bisection

The following program fragment implements the bisection method for finding a root of the function $f : \mathbb{R} \rightarrow \mathbb{R}$, where e, m, s, x, u are all variables of type $\mathbb{R}$:

```
s
:= signum(f(x)) ;
while u > x + e do
      m
      := u − (u − x)/2 ;
      if signum(f(m)) ≡ s then x
      := m else u
      := m fi
 od
```

signum is the usual signum (sign) function:

$$signum(z) = \begin{cases} 1 & \text{if} \quad z > 0 \\ 0 & \text{if} \quad z = 0 \\ -1 & \text{if} \quad z < 0 \end{cases}$$

(a) Using the ***global assumptions*** that f is a ***continuous function*** and that $e > 0$ (i.e., no need to carry this explicitly through every proof step; just ***mention where you use it***), **prove partial correctness** of the above program with respect to

- the **precondition** $u > x \wedge signum(f(x)) \neq signum(f(u))$
- and the **postcondition** $\exists\, w \in [x, x+e] \bullet f(w) = 0$.

  (The bracket notation $[x, x+e]$ here denotes the **interval** containing exactly those real numbers z with $x \leq z$ and $z \leq x + e$.)

Hint: Induce the invariant from the **precondition** in this case!
(I.e., **not** from the postcondition as in most previous examples.)

(b) (***independent from the solution of (a)!***)

The postcondition given in (a) asserts that the resulting x is an approximation to **an arbitrary** root. However, the root produced by this program fragment will actually be between the **starting values** of x and u. Provide a precondition-postcondition specification that includes this fact.

Exerceise 3.6 — Correctness Proof for Horner’s Rule

Written in the simple imperative programming language for which the axiomatic semantics rules are available on the distributed rule sheet (**with simultaneous assignment**), and using a type real for real numbers for which you are allowed to assume precise arithmetic operations (i.e., operations satisfying the laws of the field of real numbers), let the following program fragment **P**, with a real **array** variable a, with int variables k and n, and with real variables s and x be given:

```
$(k,\ s) := (n,\ 0)$;
while $k \neq 0$ do
    $(k, s) := (k - 1,\ s * x + a[k{-}1])$
od
```

This program fragment is intended to use Horner’s rule to evaluate a polynomial with coefficients stored in the array a.

Formally prove that **P** is partially correct with respect to the precondition $n \geq 0$ and the following postcondition:

$$s = \sum_{i=0}^{n-1} a[i] * x^i$$

Include all intermediate steps of the proof, and **show** also the **implications** used.

Explicitly state the **invariant!**

Exercise 3.7 — Operational and Axiomatic Semantics: Base-2-Logarithm — 30% of SE3E03 Final 2005

(a) $\boxed{\approx 3\%}$ Define what the phrase "Statement **S** is partially correct with respect to the precondition Q and the postcondition R" means in terms of operational semantics.

Written in the simple imperative programming language for which operational and axiomatic semantics rules are available on the distributed rule sheet, let the following program fragment **P**, with `int` variables i, k, and n, be given in two variants, one with simultaneous assignments, and one without::

```
(k, i) := (1, −1);                       k := 1;
while k ≤ n do                           i := −1;
   (k, i) := (k ∗ 2, i + 1)              while k ≤ n do
od;                                         k := k ∗ 2;
                                            i := i + 1
                                         od;
```

This program fragment is intended to calculate the base-2-logarithm of n, as expressed by the following post-condition *Post*:

$$2^i \leq n \quad \wedge \quad n < 2^{i+1}$$

(b) $\boxed{\approx 6\%}$ Provide a derivation in the operational semantics that shows that the precondition "*True*" is too weak, i.e., that the program **P** (in the **right** variant **without** simultaneous assignments) is **not** partially correct with respect to the precondition "*True*" and the above postcondition. (You may omit the details for expression evaluation.)

Explain why your derivation shows that.

(c) $\boxed{\approx 4\%}$ Identify the weakest precondition for which the program fragment P can be proven partially correct with respect to the above postcondition. **Explain!**

(*True* is "weaker" than every other condition Q, since $Q \Rightarrow True$ holds for every Q.)

(d) $\boxed{\approx 18\%}$ Formally prove that P is partially correct with respect to the precondition you stated in (c) and the above postcondition:

$$Post \quad :\Leftrightarrow \quad 2^i \leq n \quad \wedge \quad n < 2^{i+1}$$

Choose whether you consider ☐ only the version with simultaneous assignments (left), or
☐ only the version without (right).

Include all intermediate steps of the proof, and **show** also the **implications** used.