

Exercise Sheet 6: Axiomatic Semantics

Due date: June 3rd. You can hand in your solutions at the start of the exercise class.

Exercise 1 (Assertion Language)

35%

- (a) [10%] Give an assertion A with a logical variable $i \in \text{LVar}$ which expresses that i is a prime number. More precisely, for every $\sigma \in \Sigma$ and $I \in \text{Int}$, $\sigma \models^I A$ should be valid if and only if i is prime.

Solution: i is prime if and only if $i > 1$ and the only divisors of i are 1 and i which is the case. Hence,

$$i > 1 \wedge \forall j : j|i \Rightarrow (j = 1 \vee j = i).$$

Here, $j|i$ holds if and only if $\exists k : k \geq 1 \wedge i = j * k$. Put together, we obtain

$$i > 1 \wedge \forall j : (\exists k : k \geq 1 \wedge i = j * k) \Rightarrow (j = 1 \vee j = i).$$

- (b) [10%] Give an assertion A with logical variables $i, j, k \in \text{LVar}$ which expresses that k is the least common multiple of i and j .

Solution:

$$i|k \wedge j|k \wedge \forall k' : (i|k' \wedge j|k') \Rightarrow k' \geq k$$

- (c) [15%] Goldbach's conjecture states that every even natural number $n \in \mathbb{N}$ can be written as the sum of two primes $p, q \in \mathbb{N}$. Such a pair (p, q) is called a Goldbach partition of n . Does there exist a partial correctness property $\{A\}c\{B\}$ of a program c that computes the Goldbach partition of any given even natural number? If yes, does the existence of a program c satisfying this property prove Goldbach's conjecture?

Solution: Let $\text{prime}(n)$ be as in (a) and $\text{even}(n) = 2|n$. Then, p, q is a Goldbach partition of n iff $\text{even}(n) \wedge \text{prime}(p) \wedge \text{prime}(q) \wedge n = p + q$. Thus, the required partial correctness property is

$$\{\text{even}(n)\}c\{\text{prime}(p) \wedge \text{prime}(q) \wedge n = p + q\}.$$

Existence of such a c does not imply Goldbach's conjecture, because the postcondition is only to be fulfilled *if* c terminates. A program that does not terminate on an input that cannot be partitioned, is still correct.

Exercise 2 (Axiomatic Semantics of a For-Loop)

35%

- (a) [10%] Develop a proof rule for statements of the form
for $x := a_1$ to a_2 do $\{c\}$ where $x \in \text{var}$, $a_1, a_2 \in \text{AExp}$, and $c \in \text{Cmd}$ (without assuming the presence of a WHILE statement in the programming language).

Solution: Derive from the corresponding *while*-statement:

for $x := a_1$ to a_2 do $\{c\}$ $\equiv$ $x := a_1$; while $(x \leq a_2)$ do $\{(c; x := x + 1)\}$

$$\frac{\frac{\frac{\{A \wedge x \leq a_2\} c \{A[x \mapsto x + 1]\}}{\{A \wedge x \leq a_2\} c; x := x + 1 \{A\}} \quad \{A\} \text{ while } (x \leq a_2) \text{ do } \{(c; x := x + 1)\} \{A \wedge x > a_2\}}{\{A[x \mapsto a_1]\} x := a_1 \{A\}} \quad \{A[x \mapsto a_1]\} x := a_1; \text{ while } (x \leq a_2) \text{ do } \{(c; x := x + 1)\} \{A \wedge x > a_2\}}$$

- (b) [25%] Using this rule (and the known proof system), establish the validity of the following partial correctness property:

$$\{y \geq 0\} z := 0; \text{ for } x := 1 \text{ to } y \text{ do } \{z := z + x\} \left\{ z = \frac{y(y+1)}{2} \right\}$$

Solution: Choose loop invariant:

$$C \equiv (y \geq 0 \wedge x \leq y + 1 \wedge z = \frac{x*(x-1)}{2})$$

For the body of the for statement:

$$\vdash \{C[x \mapsto x + 1][z \mapsto z + x]\} z := z + x \{C[x \mapsto x + 1]\}$$

Precondition and postcondition valid:

$$\begin{aligned} C \wedge x \leq a_2 \\ \equiv (y \geq 0 \wedge x \leq y + 1 \wedge z = \frac{x*(x-1)}{2} \wedge x \leq y) \\ \Leftrightarrow (y \geq 0 \wedge x \leq y \wedge z = \frac{x*(x-1)}{2}) \end{aligned}$$

$$\begin{aligned} C[x \mapsto x + 1][z \mapsto z + x] \\ \equiv (y \geq 0 \wedge x + 1 \leq y + 1 \wedge z + x = \frac{x*(x+1)}{2}) \\ \Leftrightarrow (y \geq 0 \wedge x \leq y \wedge z = \frac{x*(x+1)}{2} - x) \\ \Leftrightarrow (y \geq 0 \wedge x \leq y \wedge z = \frac{x*(x-1)}{2}) \end{aligned}$$

$$\text{Together: } C \wedge x \leq a_2 \Leftrightarrow C[x \mapsto x + 1][z \mapsto z + x]$$

$$\Rightarrow \vdash \{C \wedge x \leq a_2\} z := z + x \{C[x \mapsto x + 1]\} \quad (\text{for the body})$$

$$\Rightarrow \vdash \{C[x \mapsto 1]\} \text{ for } x := 1 \text{ to } a_2 \text{ do } \{z := z + x\} \{C \wedge x > a_2\}$$

$$\Rightarrow \vdash \{C[x \mapsto 1][z \mapsto 0]\} z := 0; \text{ for } x := 1 \text{ to } a_2 \text{ do } \{z := z + x\} \{C \wedge x > a_2\}$$

where

$$\begin{aligned} C[x \mapsto 1][z \mapsto 0] \\ \equiv (y \geq 0 \wedge 1 \leq y + 1 \wedge 0 = \frac{1*(0)}{2}) \end{aligned}$$

$$\Leftrightarrow (y \geq 0) \equiv A$$

and $C \wedge x > a_2$

$$\equiv (y \geq 0 \wedge x \leq y + 1 \wedge z = \frac{x*(x-1)}{2} \wedge x > y)$$

$$\Leftrightarrow (y \geq 0 \wedge x = y + 1 \wedge z = \frac{y*(y+1)}{2})$$

$$\Rightarrow z = \frac{y*(y+1)}{2} \equiv B$$

$$\Rightarrow \vdash \{A\} c \{B\} \quad (\text{the PCP from the exercise})$$

Exercise 3 (Weakest Precondition)

30%

In the lecture, the weakest precondition calculus has been introduced to show relative completeness of Hoare Logic. Informally, the weakest precondition $wp(c, B)$ is the weakest assertion A such that $\{A\}c\{B\}$ holds. Note that termination is not required.

- (a) [15%] Give a formal definition of the weakest precondition $wp(c, B)$ of a **WHILE** program c and an assertion B . For simplicity, the weakest precondition may be infinite (or contain recursion).

Solution:

$$wp(\text{skip}, B) = B$$

$$wp(x := a, B) = B[x \mapsto a]$$

$$wp(c_1; c_2, B) = wp(c_1, wp(c_2, B))$$

$$wp(\text{if } (b) \text{ then } \{c_1\} \text{ else } \{c_2\}, B) = (b \wedge wp(c_1, B)) \vee (\neg b \wedge wp(c_2, B))$$

$$wp(\text{while } (b) \text{ do } \{c\}, B) = (\neg b \Rightarrow B) \wedge (b \Rightarrow wp(c, wp(\text{while } (b) \text{ do } \{c\}, B)))$$

$$= \bigwedge_{i \in \mathbb{N}} F_i(B)$$

Here, $F_0(B) = \text{true}$ and $F_{n+1}(B) = (\neg b \Rightarrow B) \wedge (b \Rightarrow wp(c, F_n(B)))$.

- (b) [15%] Prove that the rules Hoare Logic together with your rules to compute the weakest precondition is relative complete, i.e. show by structural induction that $\vdash \{wp(c, B)\}c\{B\}$ holds for all statements $c \in \mathbf{Cmd}$ and assertions B .

Solution: Proof by structural induction on the syntax of c .

- The cases **skip** and $x := a$ are trivial as they directly correspond to the inference rules.
- For the $c_1; c_2$, the I.H. yields $\vdash \{wp(c_2, B)\}c_2\{B\}$ and $\vdash \{wp(c_1, wp(c_2, B))\}c_1\{wp(c_2, B)\}$. Applying the seq-rule of Hoare logic yields $\vdash \{wp(c_1, wp(c_2, B))\}c_1; c_2\{B\}$. By definition $wp(c_1, wp(c_2, B)) = wp(c_1; c_2, B)$.
- For the case **if** (b) **then** $\{c_1\}$ **else** $\{c_2\}$, the I.H. yields $\vdash \{wp(c_1, B)\}c_1\{B\}$ and $\vdash \{wp(c_2, B)\}c_2\{B\}$. By definition of wp , $b \wedge wp(\text{if}(b) \text{ then } \{c_1\} \text{ else } \{c_2\}, B) \Rightarrow wp(c_1, B)$

and $\neg b \wedge wp(\text{if } (b) \text{ then } \{c_1\} \text{ else } \{c_2\}, B) \Rightarrow wp(c_2, B)$. Hence, the consequence-rule can be applied twice to obtain $\vdash \{b \wedge wp(\text{if } (b) \text{ then } \{c_1\} \text{ else } \{c_2\}, B)\}c_1\{B\}$ and $\vdash \{\neg b \wedge wp(\text{if } (b) \text{ then } \{c_1\} \text{ else } \{c_2\}, B)\}c_2\{B\}$. Now, apply the if-rule to obtain

$$\vdash \{wp(\text{if } (b) \text{ then } \{c_1\} \text{ else } \{c_2\}, B)\}\text{if } (b) \text{ then } \{c_1\} \text{ else } \{c_2\}\{B\}.$$

- For the case **while** (b) **do** $\{c\}$, the I.H. yields $\vdash \{wp(c, A)\}c\{A\}$ for any assertion A . In particular, we may choose $A = wp(\text{while } (b) \text{ do } \{c\}, B)$. Then, by construction of $wp(\text{while } (b) \text{ do } \{c\}, B)$, the implication $A \wedge b \Rightarrow wp(c, A)$ holds. Hence, we can apply the (while) rule with loop invariant A to derive $\vdash \{A\}\text{while } (b) \text{ do } \{c\}\{A \wedge \neg b\}$.

It remains to show that we can derive the required postcondition, i.e. we apply the consequence rule together with the implication $A \wedge \neg b \Rightarrow B$, which is trivially valid by our choice of A and the definition of $wp(\text{while } (b) \text{ do } \{c\}, B)$.

Exercise 4 (Strongest Postcondition (Bonus))

20%

This exercise is a bonus task which considers an alternative to weakest preconditions. Informally, the strongest postcondition $sp(c, A)$ is the strongest assertion B such that $\{A\}c\{B\}$ holds.

- (a) [15%] Give a formal definition of the strongest postcondition $sp(c, A)$ of a **WHILE** program c and an assertion A . Again, it is not required that the strongest postcondition is finite.

Solution:

$$\begin{aligned} sp(\text{skip}, A) &= A \\ sp(x := a, A) &= \exists z : (x = a[z/x] \wedge A[z/x]) \\ sp(c_1; c_2, A) &= sp(c_2, sp(c_1, A)) \\ sp(\text{if } (b) \text{ then } \{c_1\} \text{ else } \{c_2\}, A) &= sp(c_1, A \wedge b) \vee sp(c_2, A \wedge \neg b) \\ sp(\text{while } (b) \text{ do } \{c\}, A) &= sp(\text{while } (b) \text{ do } \{c\}, sp(c, A \wedge b)) \vee (A \wedge \neg b) \\ &= \bigwedge_{i \in \mathbb{N}} F_i \end{aligned}$$

Here, $F_0(A) = \text{true}$ and $F_{n+1}(A) = (\neg b \Rightarrow A) \wedge (b \Rightarrow sp(c, F_n(A)))$.

- (b) [5%] What is an advantage of weakest preconditions in comparison to strongest postconditions when trying to automatically prove Hoare triples?

Solution: The assign rule of strongest postconditions requires an existential quantification. These tend to stack up which can dramatically slow down theorem provers.