

Problem Sheet 1: Axiomatic Semantics

Sample Solutions

Chris Poskitt
ETH Zürich

Starred exercises (*) are more challenging than the others.

1 Partial and Total Correctness

- i. *Partial correctness*: if program P is executed on a state satisfying pre , then if its execution terminates, the state returned will satisfy $post$.
- ii. *Total correctness*: partial correctness (as above), but the termination of P on states satisfying pre is also guaranteed.
- iii. \models, \models_{tot} .
- iv. Neither. The precondition does not express anything about x , and `skip` certainly does not establish its value as 21. It could have any other value, failing the postcondition.
- v. \models, \models_{tot} .
- vi. \models only. If $x > 1$, then the program will terminate and the postcondition will be satisfied. But for $x \in \{0, 1\}$, x will never grow larger than 0 or 1. Since $y > 1$, x will never be larger than y , and the loop will never terminate.
- vii. \models only. Since the loop never terminates, it does not return any program states to check against the postcondition. Hence every program state returned satisfies the postcondition (or in other words, because no program states are returned, the postcondition is never violated).

2 A Hoare Logic for Partial Correctness

- i. The proof rule [cons] allows us to strengthen the precondition and/or weaken the postcondition. It also allows us to replace them with syntactically distinct, but semantically equivalent assertions. Proving the validity of the logical implications proves that the strengthening/weakening holds in all program states, and is necessary for the soundness of the proof rule.
- ii. The assignment axiom [ass] can be interpreted as follows. If an assertion p , with every occurrence of x replaced by e , holds before execution, then surely, after the assignment is executed, the assertion p without this replacement will still hold (since x now has the value of e).

(The axiom, at first, may look a little strange or jarring. But it is much simpler to use than the alternative “forward” axiom—see the later exercise. Reasoning backwards is more efficient!)

- viii. A sound axiom would be $\vdash \{p\} \text{surprise} \{\text{true}\}$. Because we do not know which variable will be changed by **surprise**, we cannot assert anything about program variables in the postcondition. We can however use [cons] to derive postconditions that are true in all program states, e.g. statements about arithmetic such as:

$$\vdash \{p\} \text{surprise} \{\forall x : \mathbb{N}. \exists y : \mathbb{N}. y > x\}.$$

- ix. The following is equivalent to the well-known “backwards” assignment axiom:

$$\vdash \{p\} x := e \{ \exists x^{old}. p[x^{old}/x] \wedge x = e[x^{old}/x] \}$$

where x^{old} is fresh (i.e. it does not occur free in p or e) and is not the same variable as x . The variable x^{old} can be understood as recording the value that x used to have before the assignment. Because x may have changed, the first conjunct replaces each occurrence of it in p with the old value, x^{old} . The second conjunct expresses the value of x after assignment, replacing occurrences of x in the expression with the old value x^{old} .

If the soundness of this axiom is not clear at first, try applying it to an assignment such as $x := x + 5$ with precondition $x > 0$.

While this axiom is sound and equivalent to the backwards axiom, it tends to be used less in practice, in order to avoid the accumulation of existential quantifiers (one per assignment!).

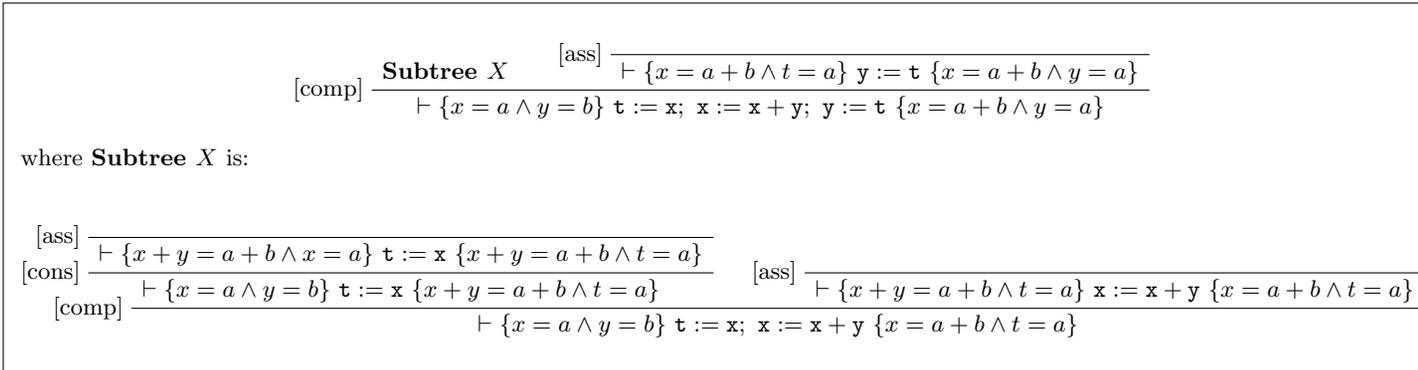


Figure 1: Proof tree for Exercise 2.1-iv

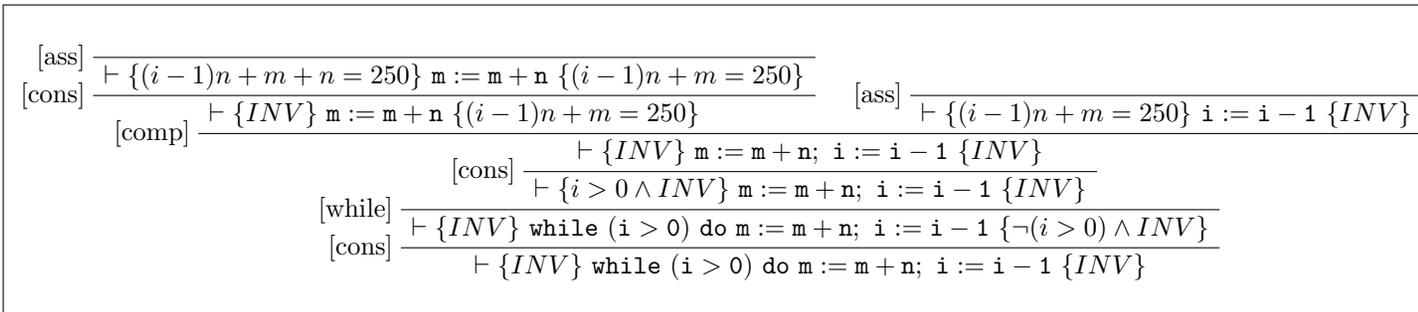


Figure 2: Proof tree for Exercise 2.1-vi