CS611 Midterm Solutions

November 3, 1998

1. *(a) What does it mean when we say that $\{A\}c\{B\}$ is valid?*

   This is a partial correctness assertion. For any state σ, and any interpretation, I, $\sigma \models^I A \rightarrow \mathcal{C}[\![c]\!]\sigma \models^I B$. (i.e. if the command is executed in an state satisfying the precondition, the resultant state (which could be $\perp$) will satisfy the postcondition.)

   (b) What does it mean when we say that $[A]c[B]$ is valid?

   This is a total correctness assertion. It means the same as the above, except that $\mathcal{C}[\![c]\!]\sigma$ must produce a state in Σ. In other words, the command, given any initial state satifying the precondition must terminate (in a state which satisfies the postcondition).

2. *In the proof of $< c, \sigma > \rightarrow \sigma'$ implies $\mathcal{C}[\![c]\!]\sigma = \sigma'$, why was it important to use rule induction as opposed to structural induction on c?*

   The large step operational rule for while commands states:

$$\frac{< \texttt{if}\, b\, \texttt{then}\, c; \texttt{while}\, b\, \texttt{do}\, c\, \texttt{else}\, \texttt{skip}, \sigma > \rightarrow \sigma'}{< \texttt{while}\, b\, \texttt{do}\, c, \sigma > \rightarrow \sigma'}.$$

   In a proof by structural induction, we can only apply the induction hypothesis to commands which are subcommands of the current one. So we would be unable to apply it to the term $\texttt{if}\, b\, \texttt{then}\, c; \texttt{while}\, b\, \texttt{do}\, c\, \texttt{else}\, \texttt{skip}$, as it is in fact a larger command. Rule induction, on the otherhand, allows us to apply the induction hypothesis to all subderivations.

3. *Which of the following are CPOs? If so, is it pointed and why? If it is not a CPO, explain why, and briefly state how to make it a pointed CPO.*

   (a) *natural numbers with $\leq$*

   This is not a CPO, as the ω-chain $\{0, 1, 2, 3...\}$ does not contain a least upper bound. It could be made a CPO with the addition of a greatest element ω, in which case it would pointed, with least element 0.

   (b) *natural numbers with $\geq$*

   This is a CPO, though not pointed. The addition of ω, as before, would provide a least element.

(c) *interval [0,1] of rationals under* $\leq$

This was the trickiest one. It is not a CPO, as an ω-chain of rational numbers with an irrational limit does not have a *least* upper bound within the rationals. For example consider the chain, $\{.1, .14, .145, .1459, ...\}$ which approaches $\pi - 3$. There is no "closest" rational to $\pi - 3$, because, if there were, you could always add one more digit and produce a closer one.

One the other hand the interval [0,1] of reals is a pointed CPO, with bottom element 0.

(d) *A set with one element, a, and the trivial relation* $a \sqsubseteq a$

This is a pointed CPO, with least element a.

(e) *if* $(A, \sqsubseteq_A)$ *and* $(B, \sqsubseteq_B)$ *are pointed CPOs, the sum space* $A + B$*, where the set in this case is the disjoint union of* A *and* B*, and* $\sqsubseteq_{A+B}$ *is defined as:*

$$\begin{cases} \sqsubseteq_A & \textit{if both arguments were originally in } A, \\ \sqsubseteq_B & \textit{if both arguments were originally in } B, \\ \textit{undefined} & \textit{otherwise.} \end{cases}$$

This is a CPO, but it is not pointed, as $\perp_A$ and $\perp_B$ are not comparable. One way to make a sum space pointed is to take the "squash sum", which replaces $\perp_A$ and $\perp_B$ with a single element, $\perp$, defined to be less than any element of A or B.

(f) *if D is a CPO and E a pointed CPO, the continuous function space* $[D \rightarrow E]$

This should have said the function space under the pointwise relation, but everyone figured that out. From your homework you know this is a CPO, and even though D is not pointed, this space is, with least element $f(d) = \perp_E$.

4. *Let D be a pointed cpo (i.e., with a least element* $\perp$*) and let f be a continuous function from D to D. Take* $d = \texttt{lub}\{f^i(\perp) | i >= 0\}$*. That is, d is the least upper bound of the chain* $\{\perp, f(\perp), f(f(\perp)), ...\}$*.*

   (a) Show that d is a fixed point of f.

$$f(d) = f(\bigsqcup_{i \geq 0} f^i(\perp)) \stackrel{a}{=} \bigsqcup_{i \geq 0} f(f^i(\perp)) = \bigsqcup_{i \geq 1} f^i(\perp) \stackrel{b}{=} \bigsqcup_{i \geq 1} f^i(\perp) \sqcup f^0(\perp) = \bigsqcup_{i \geq 0} f^i(\perp) = d.$$

   - a is true by the continuity of f.
   - b is true as $a = a \sqcup \perp$ for any a, and $f^0(\perp) = \perp$

5. *Consider the parallel command,* $c_1 || c_2$*. One way to add it to IMP is by adding the following rules:*

$$\texttt{skip} || c \rightarrow_1 c \qquad c || \texttt{skip} \rightarrow_1 c$$

$$\frac{c_1 \rightarrow_1 c_1'}{c_1 || c_2 \rightarrow_1 c_1' || c_2} \qquad \frac{c_2 \rightarrow_1 c_2'}{c_1 || c_2 \rightarrow_1 c_1 || c_2'}$$

   An alternative way is by adding the rules:

$$\texttt{skip} || c \rightarrow_1 c \qquad c_1 || c_2 \rightarrow_1 c_2 || c_1 \qquad \frac{c_1 \rightarrow_1 c_1'}{c_1 || c_2 \rightarrow_1 c_1' || c_2}$$

Explain briefly why these do not result in the same semantics.

At first glance, these seem to be the same as we can derive $\vdash < c, \sigma > \rightarrow_1^* < \texttt{skip}, \sigma >$ in the first semantics if and only if we can derive it in the second.

However, in the second semantics, we may sometimes derive an infinite computation, where none exists in the first. For example, the command `skip` || `skip` may run forever (by repeated applications of the exchange rule), when this same command will always terminate under the first set of rules.

6. *Suppose we restricted IMP so that it only manipulates 32-bit signed integers and redefined +,-, and * to work like machine arithmetic. Call the resulting language IMP32. Suppose further that we restrict the language of assertions to use only 32-bit integers.*

   Now suppose that I gave you an ML-function:

   ```
   wp : command * assertion -> assertion
   ```

   with the property that `wp (c,B)` *returns the weakest pre-condition of the command c with respect to the assertion B.*

   (a) *Is there a decision procedure to determine whether or not $\models A$ for any IMP32 assertion? Sketch briefly why or why not.*

      Yes there is a decisition procedure. Any IMP32 assertion contains a finite number of locations each of which can have a finite number of values. Plus, every quantifier covers a finite number of possibilities. Therefore, while intractable, a procedure does exist which will return yes or no in a finite amount of time.

   (b) *Suppose you do have a decision procedure, valid_assertion, for determining whether or not $\models A$. How can you combine this with* `wp` *to determine if a partial correctness assertion is valid?*

      Given a partical correctness assertion, $\{A\}c\{B\}$. Run *valid_assertion* on $(A \Rightarrow \texttt{wp}(c, B))$.